NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

FACE RECOGNITION USING NEURAL NETWORKS

Graduation Project
COM - 400

Student: Muhammad Tariq

Supervisor: Assoc. Prof. Dr. Adnan Khashman

Nicosia - 2004

ACKNOWLEDGMENTS

*Praise be to **ALLAH** Most Gracious most Merciful.*

*I would like to thank my family especially my father **Mian Abdul Sattar** for giving me the chance to complete my academic study and support me during the preparation of this project.*

My second thanks goes to my brothers for there valuable information and continuous support in writing this project

*Finally, my special Thanks to my Supervisor **Assoc. Prof. Dr. Adnan Khashman**, for his valuable advice and utmost support in completing this Project.*

ABSTRACT

In recent years considerable progress has been made in the area of face recognition. Through the development of techniques like Eigenfaces and Local feature Analysis computers can now outperform humans in many face recognition tasks, particularly those in which large databases of faces must be searched. Given a digital image of a person's face, face recognition software matches it against a database of other images. If any of the stored images matches closely enough, the system reports the sighting to its owner, and so the efficient way to perform this is to use an Artificial Intelligence system.

The main aim of this project is to discuss the development of the face recognition system. For this purpose the state of art of the face recognition is given. However, many approaches to face recognition involving many applications and there eignfaces to solve the face recognition system problems is given too. For example, the project contains a description of a face recognition system by dynamic link matching which shows a good capability to solve the invariant object recognition problem.

A better approach is to recognize the face in supervised manner using neural network architecture. We collect typical faces from each individual, project them onto the eigenspace or local feature analysis and neural network learns how to classify them with the new face descriptor as input.

TABLE OF CONTENTS

INTRODUCTION

The goal of my project is to show that the face detection problem can be solved efficiently and accurately using a combination of local image sampling and self-organizing map approaches implemented with convolution neural networks. Specifically, I will demonstrate that the use of neural networks in face recognition gives high success rate and detection is faster than the other approaches like eigenfaces and local feature analysis.

Artificial Neural Networks (A.N.N.) is one of the most effective weapons in the world of technology, so the A.N.N. can be found in both fields whether it is peaceful or military fields, the concept behind A.N.N. can identify as an information processing paradigm, implemented in both of hardware's and software's that is modeled after the biological processes of the brain. An A.N.N. is made up of a collection of highly interconnected nodes, called Neurons or Processing Elements.

Chapter one describes the introduction to face recognition. Face recognition is defined as the identification of a person from an image of their face. Face recognition is a very complex problem, as there are numerous factors that influence the appearance of ones facial features.

Chapter two is intended to help the reader to understand what artificial neural networks are? I gave the history of the Artificial Neural Networks and how does it simulate the brain; architecture of the Artificial Neural Networks, and the ways that N.N. can be trained, which are the Supervised and unsupervised Learning Methods.

Chapter three describe details about the techniques that are implemented now a day for recognition. Eigen faces and local feature analysis. Eigenfaces are an excellent basis for face recognition system, providing high recognition accuracy and moderate insensitivity to lighting variations. The key idea is that local features, being manifested by a collection of pixels in a local region, are learnt from the training set instead of arbitrarily defined.

Chapter four is describes a hybrid neural network approach for face recognition. We use a convolutional neural network approach, and compare some results with eigenfaces and local feature methods. As the result we show that the neural networks provide a batter diction rate.

CHAPTER ONE

INTRODUCTION TO FACE RECOGNITION

1.1 Overview

This chapter is intend to help the readers to understand what face recognition is. A detailed historical background is provided. This chapter contain what face recognition is and why we and to use face recognition. Here we also describe the mathematical framework, who to deal with dimensions, commercial system's applications and face- recognition for smart environment. At the end wearable recognition systems and summary of the chapter.

1.2 History of Face Recognition

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

Perhaps the most famous early example of a face recognition system is due to Kohonen [1], who demonstrated that a simple neural net could perform face recognition for aligned and normalized face images. The type of network he employed computed a face description by approximating the eigenvectors of the face image's autocorrelation matrix; these eigenvectors are now known as `eigenfaces.' Destiny is not a matter of chance; it's a matter of choice.

This method functions by projecting a face onto a multi-dimensional feature space that spans the gamut of human faces. A set of basis images is extracted from the database presented to the system by Eigenvalue-Eigenvector decomposition. Any face in the feature space is then characterized by a weight vector obtained by projecting it onto the set of basis images. When a new face is presented to the system, its weight vector is calculated and compared with those of the faces in the database. The nearest neighbor to this weight vector, computed using the Euclidean norm, is determined. If this distance is below a certain threshold (found by experimentation) the input face is adjudged as that face corresponding to the closest weight vector. Otherwise, the input pattern is adjudged as not belonging to the database.

Kohonen's system was not a practical success, however, because of the need for precise alignment and normalization. In following years many researchers tried face

identify a person often with very limited information. Creating a computer system to try and compete with the human visual system is extremely complex and so far unsolved.

The main aim of most commercial face recognition researches is to increase the capability of security and surveillance systems. In theory security systems involving face recognition would be impossible to hack, as the identification process involves unique identification methods, and thus only authorized users will be accepted. The mechanism would be convenient, with no need to remember passwords or personal identification numbers. The system would only require one to be positioned in front of the camera. Another potential commercial use is surveillance.

Face recognition is a very complex problem, as there are numerous factors that influence the appearance of ones facial features. There are two groups of influences, intrinsic and extrinsic factors. Intrinsic factors are independent of the surroundings and are only concerned with the changes in the three dimensional profile of the face. Extrinsic factors are the effect on the appearance of a person's face due to external factors such as lighting conditions. In this dissertation both intrinsic and extrinsic factors have been considered, specifically lighting irregularities, facial occlusions, head orientation and facial expressions. These factors are defined in more detail below.

Figure 1.2 Examples of change in lighting conditions.

Figure 1.3 Examples of facial occlusions.

Figure 1.4 Examples of change in expression.

Lighting irregularities: Highlighting on an individuals face will alter depending on the lighting conditions hindering direct intensity comparisons. Lighting is an extrinsic factor, as it has no influence on the physical structure of the face. Figure (1.2)

Facial occlusions: The obvious examples are facial hair, a scarf or a pair of sunglasses. These will mask important features of the face, hindering detection. Again this is an extrinsic factor. Figure (1.3)

Head orientation: If the head is tilted or rotated direct spatial comparisons are unlikely to work. This is an extrinsic factor, as the face structure remains constant.

Facial expressions: Facial expressions cause parts of the face to 'warp' and move in relation to other features. Speech and emotion are the main reasons for changes in facial expressions. These are intrinsic factors. Figure (1.4)

Other factors do exist which influence face detection such as ageing and camera quality/collaboration (i.e. focus of camera or noise).

1.4 Face Recognition

Smart environments, wearable computers, and ubiquitous computing in general are thought to be the coming `fourth generation' of computing and information technology. Because these devices will be everywhere clothes, home, car, and office, their economic impact and cultural significance are expected to dwarf previous generations of computing. At a minimum, they are among the most exciting and economically important research areas in information technology and computer science.

However, before this new generation of computing can be widely deployed we must invent new methods of interaction that don't require a keyboard or mouse there will be too many small computers to instruct them all individually. To win wide consumer acceptance such interactions must be friendly and personalized (no one likes being treated like just another cog in a machine!), which implies that next-generation interfaces will be aware of the people in their immediate environment and at a minimum know who they are.

The requirement for reliable personal identification in computerized access control has resulted in an increased interest in biometrics. Biometrics being investigated includes fingerprints, speech, signature dynamics, and face recognition. Sales of identity verification products exceed $100 million.

Face recognition has the benefit of being a passive, non-intrusive system for verifying personal identity. The techniques used in the best face recognition systems may depend on the application of the system. We can identify at least two broad categories of face recognition systems:

1. We can find a person within a large database of faces (e.g. in a police database). These systems typically return a list of the most likely people in the database [3]. Often only one image is available per person. It is usually not necessary for recognition to be done in real-time.
2. We can identify particular people in real-time (e.g. in a security monitoring system, location tracking system, etc.), or we can allow access to a group of people and deny access to all others (e.g. access to a building, computer, etc.). Multiple images per person are often available for training and real-time recognition is required.

1.5 Why Face Recognition

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

While appropriate for bank transactions and entry into secure areas, such technologies have the disadvantage that they are intrusive both physically and socially. They require the user to position their body relative to the sensor, and then pause for seconds to `declare' themselves. This `pause and declare' interaction is unlikely to change because of the fine-grain spatial sensing required. Moreover, there is an `oracle-like' aspect to the interaction:

since people can't recognize other people using this sort of data, these types of identification do not have a place in normal human interactions and social structures.

While the `pause and present' interaction and the oracle-like perception are useful in high-security applications (they make the systems look more accurate), they are exactly the opposite of what is required when building a store that recognizes its best customers or an information kiosk that remembers you, or a house that knows the people who live there. Face recognition from video and voice recognition have a natural place in these next-generation smart environments. They are unobtrusive (able to recognize at a distance without requiring a `pause and present' interaction), are usually passive (do not require generating special electro-magnetic illumination), do not restrict user movement, and are now both low-power and inexpensive. Perhaps most important, however, is that humans identify other people by their face and voice, therefore are likely to be comfortable with systems that use face and voice recognition.

1.6 Advantages of Implementing Face Recognition Techniques

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

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

While the `pause and present' interaction and the oracle-like perception are useful in high-security applications (they make the systems look more accurate), they are exactly the

opposite of what is required when building a store that recognizes its best customers, or an information kiosk that remembers you, or a house that knows the people who live there. Face recognition from video and voice recognition have a natural place in these next-generation smart environments -- they are unobtrusive (able to recognize at a distance without requiring a `pause and present' interaction), are usually passive (do not require generating special electro-magnetic illumination), do not restrict user movement, and are now both low-power and inexpensive. Perhaps most important, however, is that humans identify other people by their face and voice, therefore are likely to be comfortable with systems that use face and voice recognition.

1.7 Mathematical Framework

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

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

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

1.7.1 The Typical Representational Framework

The dominant representational approach that has evolved is descriptive rather than generative. Training images are used to characterize the range of 2-D appearances of objects

to be recognized. Although initially very simple modeling methods were used, the dominant method of characterizing appearance has fairly quickly become estimation of the probability density function (PDF) of the image data for the target class.

For instance, given several examples of a target class Ω in a low-dimensional representation of the image data, it is straightforward to model the probability distribution function $P(x|\Omega)$ of its image-level features **x** as a simple parametric function (e.g., a mixture of Gaussians), thus obtaining a low-dimensional, computationally efficient appearance model for the target class.

Once the PDF of the target class has been learned, we can use 'Bayes' rule to perform maximum a posteriori (MAP) detection and recognition. The result is typically a very simple, neural-net-like representation of the target class's appearance, which can be used to detect occurrences of the class, to compactly describe its appearance, and to efficiently compare different examples from the same class. Indeed, this representational framework is so efficient that some of the current face recognition methods can process video data at 30 frames per second, and several can compare an incoming face to a database of thousands of people in fewer than one second and all on a standard PC!

1.8 Dealing with the Curse of Dimensionality

To obtain an `appearance-based' representation, one must first transform the image into a low-dimensional coordinate system that preserves the general perceptual quality of the target object's image. This transformation is necessary in order to address the `curse of dimensionality'. The raw image data has so many degrees of freedom that it would require millions of examples to learn the range of appearances directly.

Typical methods of dimensionality reduction include Karhunen-Loève transform (KLT) (also called Principal Components Analysis (PCA)) or the Ritz approximation (also called `example-based representation'). Other dimensionality reduction methods are sometimes also employed, including sparse filter representations (e.g., Gabor Jets, Wavelet transforms), feature histograms, independent components analysis, and so forth.

These methods have in common the property that they allow efficient characterization of a low-dimensional subspace with the overall space of raw image measurements. Once a low-dimensional representation of the target class (face, eye, hand, etc.) has been obtained, standard statistical parameter estimation methods can be used to learn the range of appearance that the target exhibits in the new, low-dimensional coordinate

system. Because of the lower dimensionality, relatively few examples are required to obtain a useful estimate of either the PDF or the inter-class discriminant function.

An important variation on this methodology is discriminative models, which attempt to model the differences between classes rather than the classes themselves. Such models can often be learned more efficiently and accurately than when directly modeling the PDF. A simple linear example of such a difference feature is the Fisher discriminant. One can also employ discriminant classifiers such as Support Vector Machines (SVM), which attempt to maximize the margin between classes.

1.9 Current State of the Art

By 1993 there were several algorithms claiming to have accurate performance in minimally constrained environments. To better understand the potential of these algorithms, DARPA and the Army Research Laboratory established the FERET program with the goals of both evaluating their performance and encouraging advances in the technology [4].

At the time of this writing, there are three algorithms that have demonstrated the highest level of recognition accuracy on large databases (1196 people or more) under double-blind testing conditions. These are the algorithms from University of Southern California (USC) [5], University of Maryland (UMD) [6], and the MIT Media Lab [7]. All of these are participants in the FERET program. Only two of these algorithms, from USC and MIT, are capable of both minimally constrained detection and recognition; the others require approximate eye locations to operate. A fourth algorithm that was an early contender, developed at Rockefeller University [8], dropped from testing to form a commercial enterprise. The MIT and USC algorithms have also become the basis for commercial systems.

The MIT, Rockefeller, and UMD algorithms all use a version of the eigenface transforms followed by discriminative modeling. The UMD algorithm uses a linear discriminant, while the MIT system, seen in Figure (1.5), employs a quadratic discriminant. The Rockefeller system, seen in Figure (1.6), uses a sparse version of the eigenface transform, followed by a discriminative neural network. The USC system, seen in Figure (1.1), in contrast, uses a very different approach. It begins by computing Gabor `jets' from the image, and then does a `flexible template' comparison between image descriptions using a graph-matching algorithm.

The FERET database testing employs faces with variable position, scale, and lighting in a manner consistent with mugs hot or driver's license photography. On databases of fewer than 200 people and images taken under similar conditions, all four algorithms produce nearly perfect performance. Interestingly, even simple correlation matching can sometimes achieve similar accuracy for databases of only 200 people [4]. This is strong evidence that any new algorithm should be tested with at databases of at least 200 individuals, and should achieve performance over 95% on mugshot-like images before it can be considered potentially competitive.

In the larger FERET testing (with 1166 or more images), the performance of the four algorithms is similar enough that it is difficult or impossible to make meaningful distinctions between them (especially if adjustments for date of testing, etc., are made). On frontal images taken the same day, typical first-choice recognition performance is 95% accuracy. For images taken with a different camera and lighting, typical performance drops to 80% accuracy. And for images taken one year later, the typical accuracy is approximately 50%. Note that even 50% accuracy is 600 times chance performance.

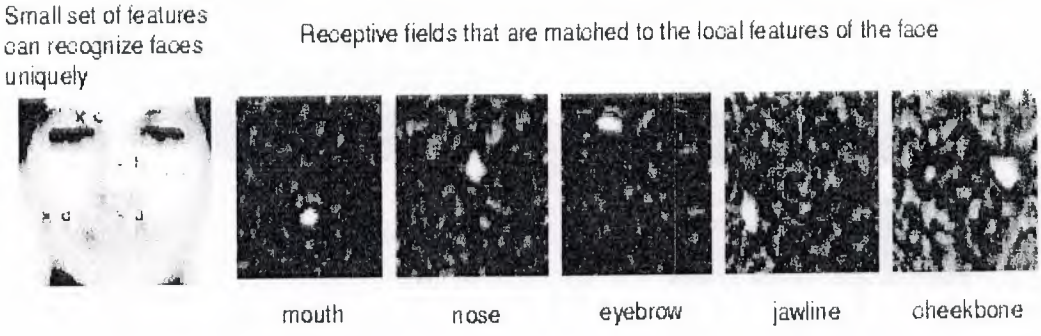

Figure 1.5 Example of face recognition using Local Feature Analysis

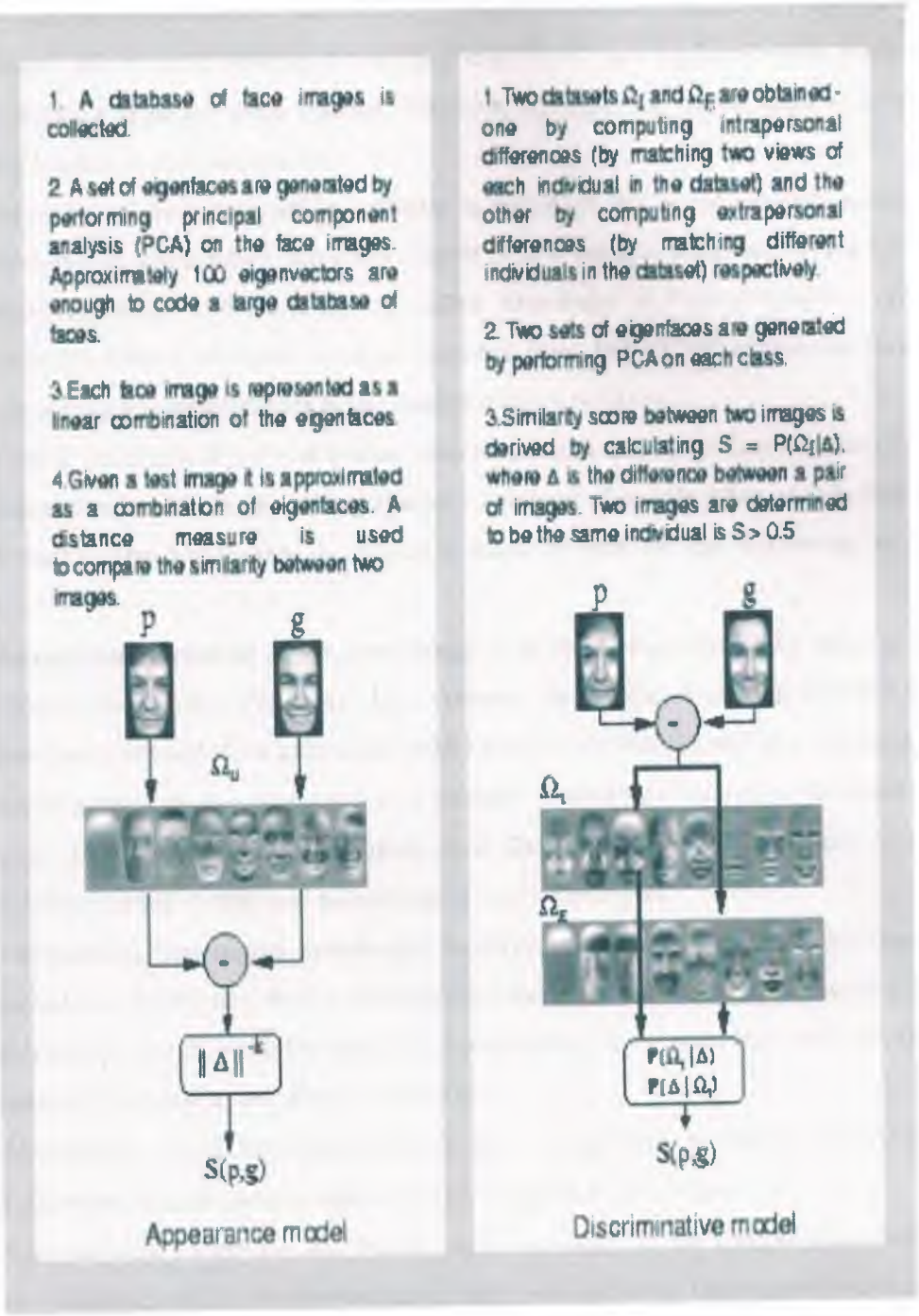

Figure 1.6 Example of face recognition using Eigenfaces.

1.10 Commercial Systems and Applications

Currently, several face-recognition products are commercially available. Algorithms developed by the top contenders of the FERET competition are the basis of some of the available systems; others were developed outside of the FERET testing framework. While it is extremely difficult to judge, three systems Visionics, Viisage, and Miros seem to be the current market leaders in face recognition.

Visionics FaceIt face recognition software is based on the Local Feature Analysis algorithm developed at Rockefeller University. FaceIt is now being incorporated into a Close Circuit Television (CCTV) anti-crime system called 'Mandrake' in United Kingdom. This system searches for known criminals in video acquired from 144 CCTV camera locations. When a match occurs a security officer in the control room is notified.

FaceIt will automatically detect human presence, locate and track faces, extract face images, perform identification by matching against a database of people it has seen before or pre-enrolled users. The technology is typically used in one of the following ways.

- ***Identification (one-to-many searching):*** To determine someone's identity in identification mode, FaceIt quickly computes the degree of overlap between the live face print and those associated with known individuals stored in a database of facial images. It can return a list of possible individuals ordered in diminishing score (yielding resembling images), or it can simply return the identity of the subject (the top match) and an associated confidence level.
- ***Verification (one-to-one matching):*** In verification mode, the face print can be stored on a smart card or in a computerized record. FaceIt simply matches the live print to the stored one if the confidence score exceeds a certain threshold, then the match is successful and identity is verified.
- ***Monitoring:*** Using face detection and face recognition capabilities, FaceIt can follow the presence and position of a person in the field of view.
- ***Surveillance:*** FaceIt can find human faces anywhere in the field of view and at any distance, and it can continuously track them and crop them out of the scene, matching the face against a watch list. Totally hands off, continuously and in real-time.
- ***Limited size storage devices:*** FaceIt can compress a face print into 84 bytes for use in smart cards, bar codes and other limited size storage devices.

Visage, another leading face-recognition company, and uses the eigenface-based recognition algorithm developed at the MIT Media Laboratory. Their system is used in conjunction with identification cards (e.g., driver's licenses and similar government ID cards) in many US states and several developing nations.

Miros uses neural network technology for their TrueFace face recognition software. TrueFace is for checking cash system, and has been deployed at casinos and similar sites in many US states.

1.11 Novel Applications of Face Recognition Systems

Face recognition systems are no longer limited to identity verification and surveillance tasks. Growing numbers of applications are starting to use face-recognition as the initial step towards interpreting human actions, intention, and behavior, as a central part of next-generation smart environments. Many of the actions and behaviors humans' display can only be interpreted if you also know the person's identity, and the identity of the people around them. Examples are a valued repeat customer entering a store, or behavior monitoring in an eldercare or childcare facility, and command-and-control interfaces in a military or industrial setting. In each of these applications identity information is crucial in order to provide machines with the background knowledge needed to interpret measurements and observations of human actions.

1.12 Face Recognition for Smart Environments

Researchers today are actively building smart environments (i.e. visual, audio, and haptic interfaces to environments such as rooms, cars, and office desks). In these applications a key goal is usually to give machines perceptual abilities that allow them to function naturally with people to recognize the people and remember their preferences and peculiarities, to know what they are looking at, and to interpret their words, gestures, and unconscious cues such as vocal prosody and body language. Researchers are using these perceptually aware devices to explore applications in health care, entertainment, and collaborative work.

Recognition of facial expression is an important example of how face recognition interacts with other smart environment capabilities. It is important that a smart system knows whether the user looks impatient because information is being presented too slowly, or confused because it is going too fast facial expressions provide cues for identifying and

distinguishing between these different states. In recent years much effort has been put into the area of recognizing facial expression, a capability that is critical for a variety of human-machine interfaces, with the hope of creating a person-independent expression recognition capability. While there are indeed similarities in expressions across cultures and across people, for anything but the grossest facial expressions analysis must be done relative to the person's normal facial rest state something that definitely isn't the same across people. Consequently, facial expression research has so far been limited to recognition of a few discrete expressions rather than addressing the entire spectrum of expression along with its subtle variations. Before one can achieve a really useful expression analysis capability one must be able to first recognize the person, and tune the parameters of the system to that specific person.

1.13 Wearable Recognition Systems

When we build computers, cameras, microphones and other sensors into a person's clothes, the computer's view moves from a passive third-person to an active first-person vantage point (Figure1.7). These wearable devices are able to adapt to a specific user and to be more intimately and actively involved in the user's activities. The field of wearable computing is rapidly expanding, and just recently became a full-fledged Technical Committee within the IEEE Computer Society. Consequently, we can expect to see rapidly growing interest in the largely unexplored area of first-person image interpretation.

Figure 1.7 Wearable face recognition systems.

Face recognition is an integral part of wearable systems like memory aides, remembrance agents, and context-aware systems. Thus there is a need for many future recognition systems to be integrated with the user's clothing and accessories. For instance, if you build a camera into your eyeglasses, then face recognition software can help you remember the name of the person you are looking at by whispering their name in your ear. Such devices are beginning to be tested by the US Army for use by border guards in Bosnia, and by researchers at the University of Rochester's Center for Future Health for use by Alzheimer's patients.

1.14 Summary

. This chapter provided a general introduction on face recognition. We explained what face recognition is and why we need to use face recognition. In recent years considerable progress has been made in the areas of face recognition. Through the work of people like Alex Pentland computers can now perform outperform humans in many face recognition tasks, particularly those in which large databases of faces must be searched. A system with the ability to detect and recognize faces in a crowd has many potential applications including crowd and airport surveillance, private security and improved human computer interaction.

CHAPTER TWO

NEURAL NETWOREKS

2.1 Overview

This chapter is intended to help the reader to understand what of Artificial Neural Networks are. Also teaching an Artificial Neural Networks? A detailed historical background is provided; definitions and analogy to the biological nervous system. The difference between neural computing and traditional computing and expert systems, also the advantages and the disadvantages of neural networks and summary of the chapter.

2.2 History of Neural Networks

The study of the human brain is thousands of years old. With the advent of modern electronics, it was only natural to try to harness this thinking process. The first step toward artificial neural networks came in 1943[9] when Warren McCulloch, a neurophysiologist, and a young mathematician, Walter Pitts, wrote a paper on how neurons might work. They modeled a simple neural network with electrical circuits.

Reinforcing this concept of neurons and how they work was a book written by Donald Hebb. The Organization of Behavior was written in 1949. It pointed out that neural pathways are strengthened each time that they are used.

As computers advanced into their infancy of the 1950s, it became possible to begin to model the rudiments of these theories concerning human thought. Nathanial Rochester from the IBM research laboratories led the first effort to simulate a neural network. That first attempt failed. But later attempts were successful. It was during this time that traditional computing began to flower and, as it did, the emphasis in computing left the neural research in the background.

Yet, throughout this time, advocates of "thinking machines" continued to argue their cases. In 1956 the Dartmouth Summer Research Project on Artificial Intelligence provided a boost to both artificial intelligence and neural networks. One of the outcomes of this process was to stimulate research in both the intelligent side, AI, as it is known throughout the industry, and in the much lower level neural processing part of the brain.

In the years following the Dartmouth Project, John von Neumann suggested imitating simple neuron functions by using telegraph relays or vacuum tubes. Also, Frank Rosenblatt, a

neuro-biologist of Cornell, began work on the Perceptron. A single-layer perceptron was found to be useful in classifying a continuous-valued set of inputs into one of two classes

In 1959, Bernard Widrow and Marcian Hoff of Stanford developed models they called ADALINE and MADALINE. These models were named for their use of Multiple ADAptive LINear Elements. MADALINE was the first neural network to be applied to a real world problem. It is an adaptive filter which eliminates echoes on phone lines. This neural network is still in commercial use.

In 1982 several events caused a renewed interest. John Hopfield of Caltech presented a paper to the national Academy of Sciences. *Hopfield's approach* was not to simply model brains but to create useful devices. With clarity and mathematical analysis, he showed how such networks could work and what they could do. Yet,

By 1985 the American Institute of Physics began what has become an annual meeting - Neural Networks for Computing. By 1987, the Institute of Electrical and Electronic Engineer's (IEEE) first International conference on neural networks drew more than 1,800 attendees.

Today, neural networks discussions are occurring everywhere. Their promise seems very bright as nature itself is the proof that this kind of thing works. Yet, its future, indeed the very key to the whole technology, lies in hardware development. Currently most neural network development is simply proving that the principal works. This research is developing neural networks that, due to processing limitations, take weeks to learn. To take these prototypes out of the lab and put them into use requires specialized chips. Companies are working on three types of neuro chips-digital, analog, and optical. Some companies are working on creating a "silicon compiler" to generate a neural network Application Specific Integrated Circuit (ASIC). These ASICs and neuron-like digital chips appear to be the wave of the near future. Ultimately, optical chips look very promising. Yet, it may be years before optical chips see the light of day in commercial applications.

2.3 What are Neural Networks?

A neural network is an artificial representation of the human brain that tries to simulate its learning process. The term "artificial" means that neural networks are implemented in computer programs that are able to handle the large number of necessary calculations during the learning process. To show where neural networks have their origin.

2.4 The Biological Model of Human Brain

The human brain consists of a large number (more than a billion) of neural cells that process information's. Each cell works like a simple processor and only the massive interaction between all cells and their parallel processing makes the brain's abilities possible. Below you see a sketch of such a neural cell, called neurons.

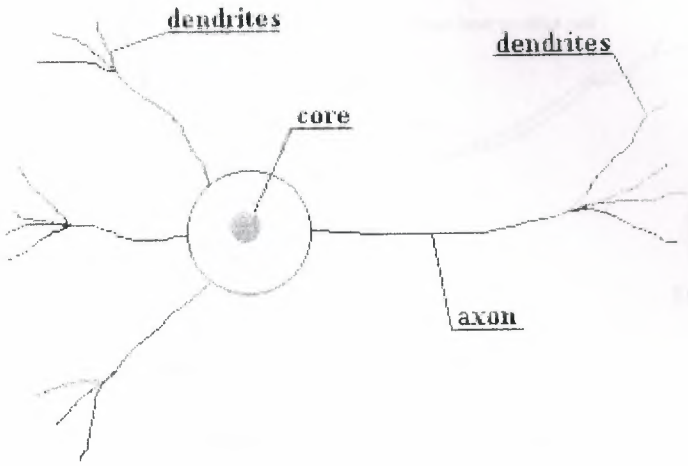

Figure 2.1 Structure of a neural cell in the human brain.

As the figure (2.1) indicates, a neuron consists of a core, dendrites for incoming information and an axon with dendrites for outgoing information that is passed to connected neurons. Information is transported between neurons in form of electrical stimulations along the dendrites. Incoming information's that reach the neuron's dendrites is added up and then delivered along the neuron's axon to the dendrites at its end, where the information is passed to other neurons if the stimulation has exceeded a certain threshold. In this case, the neuron is said to be activated. If the incoming stimulation had been too low, the information will not be transported any further. In this case, the neuron is said to be inhibited.

The connections between the neurons are adaptive, what means that the connection structure is changing dynamically. It is commonly acknowledged that the learning ability of the human brain is based on this adaptation.

2.4.1 Biological Neural Networks

Artificial Neural Networks (A.N.N.s or N.N.s) were inspired by information processing in our brains. The human brain has about 10^{11} neurons and 10^{14} synapses. A neuron consists of a soma (cell body), axons (sends signals), and dendrites (receives signals).

A synapse connects an axon to a dendrite. Given a signal, a synapse might increase (excite) or decrease (inhibit) electrical potential. A neuron fires when its electrical potential reaches a threshold. Learning might occur by changes to synapses and connections.

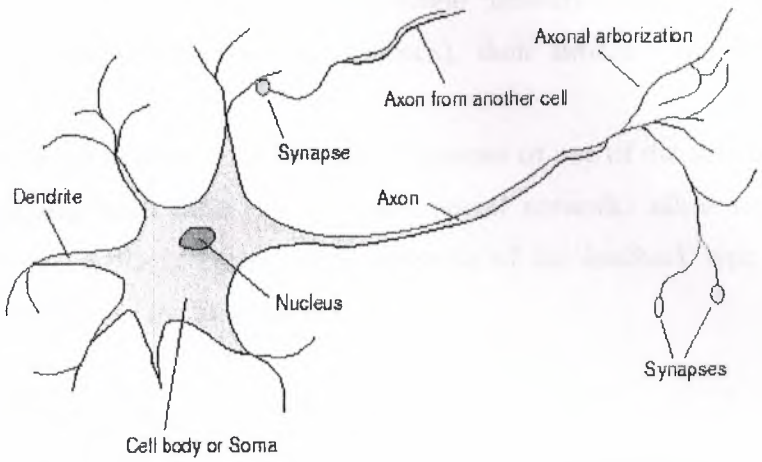

Figure 2.2 Biology of a neuron.

2.4.2 Artificial Neural Networks

An artificial neural network consists of neurons, connections, and weights. An artificial neural network is a model that emulates the biological neural network.

Table 2.1 Artificial and Biological Neural Networks Characteristics.

Biological NN	Artificial NN
Soma	Neurons
Dendrite	Inputs
Axon	Outputs
Synapse	Weight
Potential	Weighted sum
Threshold	Bias weight
Slow speed	Fast speed
Many neurons (1012)	Few neurons (a dozen to hundreds of thousands)

2.5 Types of Neural Networks

We will see how the N.N. has been designed, and how the actions, reactions and the signals travel, also what kind of networks does it have [10].

As mentioned before, several types of neural networks exist. They can be distinguished by their type (feedforward or feedback), their structure and the learning algorithm they use.

The type of a neural network indicates, if the neurons of one of the network's layers may be connected among each other. Feedforward neural networks allow only neuron connections between two different layers, while networks of the feedback type have also connections between neurons of the same layer.

2.5.1 Feed-Forward Networks

The feed-forward, back-propagation architecture was developed in the early 1970's by several independent sources. Feed-forward A.N.Ns (figure 2.3) allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward A.N.Ns tends to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

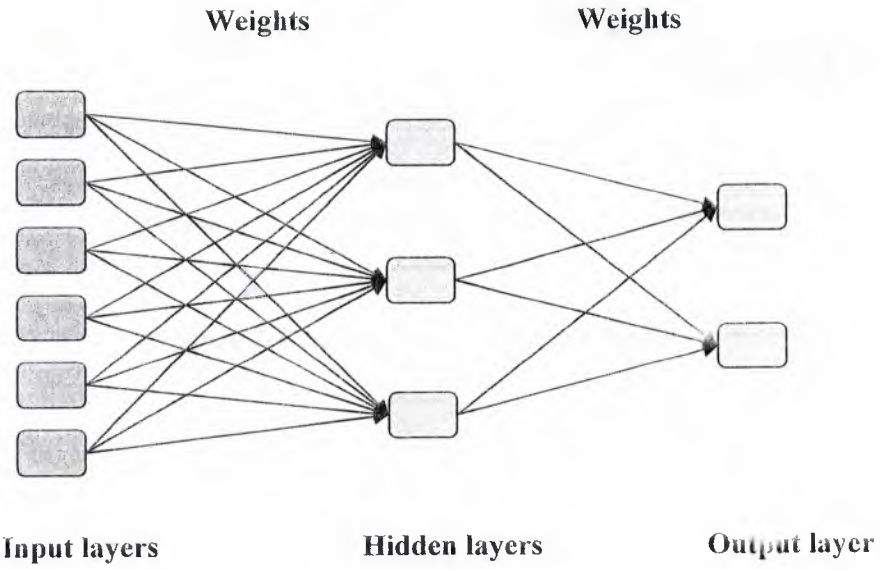

Figure 2.3 An example of a simple feedforward network.

2.5.2 Feedback Networks

Feedback networks (figure 2.4) can have signals traveling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations.

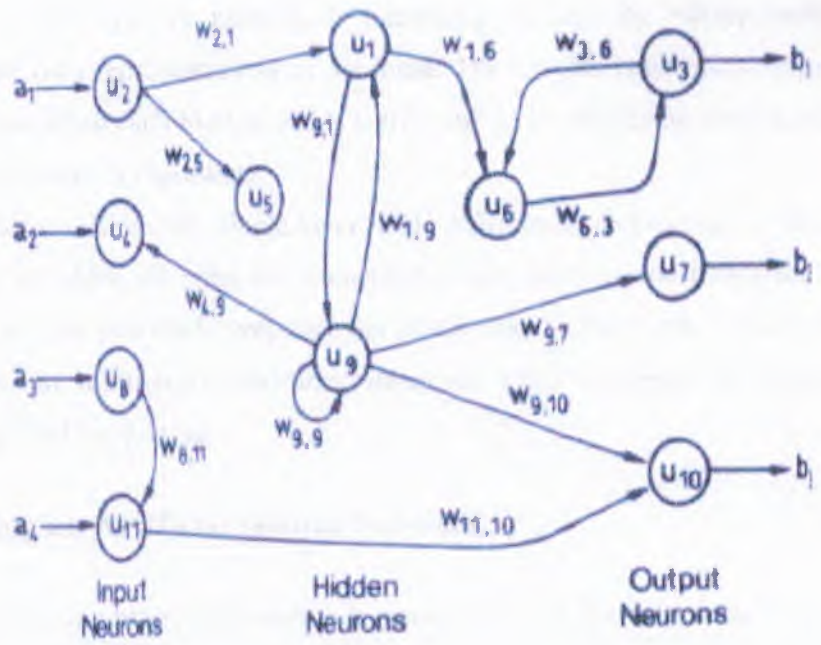

Figure 2.4 An example of a complicated network.

2.5.3 Network Layers

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "**Input**" units is connected to a layer of "**Hidden**" units, which is connected to a layer of "**Output**" units (Figure 2.3) [11].

- The activity of the input units represents the raw information that is fed into the network.
- The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.
- The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

We also distinguish single-layer and multi-layer architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organizations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

2.6 Teaching an Artificial Neural Network

In the human brain, information is passed between the neurons in form of electrical stimulation along the dendrites. If a certain amount of stimulation is received by a neuron, it generates an output to all other connected neurons and so information takes its way to its destination where some reaction will occur. If the incoming stimulation is too low, no output is generated by the neuron and the information's further transport will be blocked.

Explaining how the human brain learns certain things is quite difficult and nobody knows it exactly. It is supposed that during the learning process the connection structure among the neurons is changed, so that certain stimulations are only accepted by certain neurons. This means, there exist firm connections between the neural cells that once have learned a specific fact, enabling the fast recall of this information.

If some related information is acquired later, the same neural cells are stimulated and will adapt their connection structure according to this new information. On the other hand, if specific information isn't recalled for a long time, the established connection structure between the responsible neural cells will get more "weak". This had happened if someone "forgot" a once learned fact or can only remember it vaguely.

As mentioned before, neural networks try to simulate the human brain's ability to learn. That is, the artificial neural network is also made of neurons and dendrites. Unlike the biological model, a neural network has an unchangeable structure, built of a specified number of neurons and a specified number of connections between them (called *"weights"*), which have certain values. What changes during the learning process are the values of those weights? Compared to the original this means: Incoming information "stimulates" (exceeds a specified threshold value) of certain neurons that pass the information to connected neurons or prevent further transportation along the weighted connections. The value of a weight will be increased if information should be transported and decreased if not.

While learning different inputs, the weight values are changed dynamically until their values are *balanced*, so each input will lead to the desired output. The training of a neural networks results in a matrix that holds the weight values between the neurons. Once a neural network had been trained correctly, it will probably be able to find the desired output to a given input that had been learned, by using these matrix values. I said "probably". That is sad but true, for it can't be guaranteed that a neural network will recall the correct results in any case. Very often there is a certain error left after the learning process, so the generated output is only a good approximation to the perfect output in most cases.

All learning methods used for adaptive neural networks can be classified into two major categories

SUPERVISED LEARNING: which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be.

UNSUPERVISED LEARNING: uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties.

2.6.1 Supervised Learning

The vast majority of artificial neural network solutions have been trained with supervision. In this mode, the actual output of a neural network is compared to the desired

output. Weights, which are usually randomly set to begin with, are then adjusted by the network so that the next iteration, or cycle, will produce a closer match between the desired and the actual output. The learning method tries to minimize the current errors of all processing elements. This global error reduction is created over time by continuously modifying the input weights until acceptable network accuracy is reached.

With supervised learning, the artificial neural network must be trained before it becomes useful. Training consists of presenting input and output data to the network. This data is often referred to as the training set. That is, for each input set provided to the system, the corresponding desired output set is provided as well. In most applications, actual data must be used. This training phase can consume a lot of time. In prototype systems, with inadequate processing power, learning can take weeks. This training is considered complete when the neural network reaches a user defined performance level. This level signifies that the network has achieved the desired statistical accuracy as it produces the required outputs for a given sequence of inputs. When no further learning is necessary, the weights are typically frozen for the application. Some network types allow continual training, at a much slower rate, while in operation. This helps a network to adapt to gradually changing conditions.

Training sets need to be fairly large to contain all the needed information if the network is to learn the features and relationships that are important. Not only do the sets have to be large but the training sessions must include a wide variety of data. If the network is trained just one example at a time, all the weights set so meticulously for one fact could be drastically altered in learning the next fact. The previous facts could be forgotten in learning something new. As a result, the system has to learn everything together, finding the best weight settings for the total set of facts. For example, in teaching a system to recognize pixel patterns for the ten digits, if there were twenty examples of each digit, all the examples of the digit seven should not be presented at the same time.

How the input and output data is represented, or encoded, is a major component to successfully instructing a network. Artificial networks only deal with numeric input data. Therefore, the raw data must often be converted from the external environment. Additionally, it is usually necessary to scale the data, or normalize it to the network's paradigm. This pre-processing of real-world stimuli, be they cameras or sensors, into machine readable format is already common for standard computers. Many conditioning techniques which directly apply to artificial neural network implementations are readily available. It is then up to the network

designer to find the best data format and matching network architecture for a given application.

After a supervised network performs well on the training data, then it is important to see what it can do with data it has not seen before. If a system does not give reasonable outputs for this test set, the training period is not over. Indeed, this testing is critical to insure that the network has not simply memorized a given set of data but has learned the general patterns involved within an application. Like these examples the perceptrons, back propagation algorithm, Hopfield algorithm and Hamming algorithm. Here I will explain Perceptrons and Back propagation algorithm.

2.6.1.1 Perceptrons

The most influential work on neural networks in the 60's went under the heading of 'Perceptrons' a term coined by Frank Rosenblatt. The preceptrons (figure 2.5) turns out to be an MCP model (neuron with weighted inputs) with some additional, fixed, preprocessing. Perceptrons mimic the basic idea behind the mammalian visual system. They were mainly used in pattern recognition even though their capabilities extended a lot more.

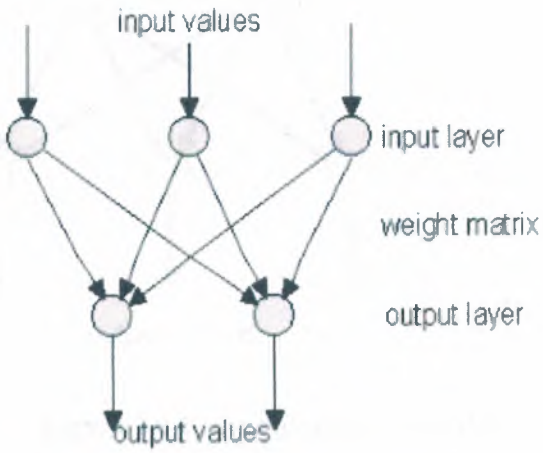

Figure 2.5 The preceptrons.

In 1969 Minsky and Papert wrote a book in which they described the limitations of single layer Perceptrons. The impact that the book had was tremendous and caused a lot of neural network researchers to loose their interest. The book was very well written and showed mathematically that single layer perceptrons could not do some basic pattern recognition operations like determining the parity of a shape or determining whether a shape is connected

or not. What they did not realize, until the 80's, is that given the appropriate training, multilevel perceptrons can do these operations.

2.6.1.2 The Back-Propagation Algorithm

In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (EW). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the EW.

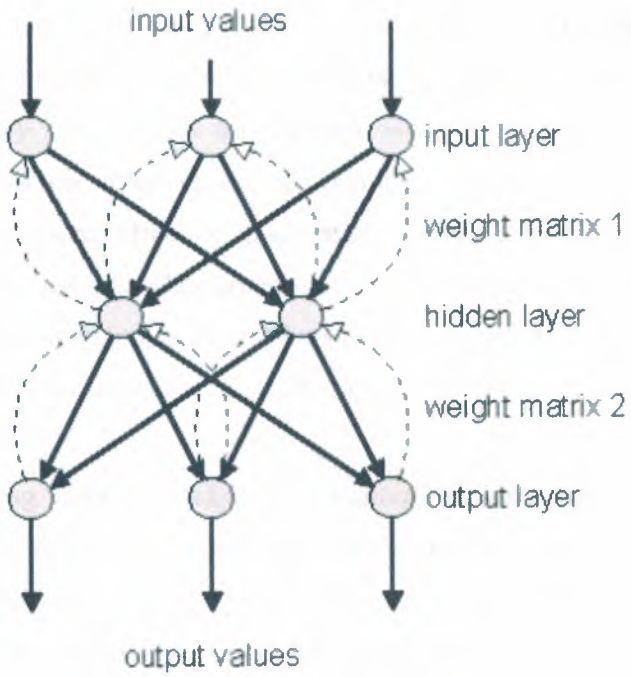

Figure 2.6 Backpropagation Neural Networks.

The back-propagation algorithm is easiest to understand if all the units in the network are linear. The algorithm computes each EW by first computing the EA, the rate at which the error changes as the activity level of a unit is changed. For output units, the EA is simply the difference between the actual and the desired output. To compute the EA for a hidden unit in the layer just before the output layer, we first identify all the weights between that hidden unit and the output units to which it is connected. We then multiply those weights by the EAs of those output units and add the products. This sum equals the EA for the chosen hidden unit. After calculating all the EAs in the hidden layer just before the output layer, we can compute

in like fashion the EAs for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back propagation its name. Once the EA has been computed for a unit, it is straight forward to compute the EW for each incoming connection of the unit. The EW is the product of the EA and the activity through the incoming connection.

Note that for non-linear units, the back-propagation algorithm includes an extra step. Before back-propagating, the EA must be converted into the EI, the rate at which the error changes as the total input received by a unit is changed.

2.6.2 Unsupervised Learning

Unsupervised learning is the great promise of the future. It shouts that computers could someday learn on their own in a true robotic sense. Currently, this learning method is limited to networks known as self-organizing maps. These kinds of networks are not in widespread use. They are basically an academic novelty. Yet, they have shown they can provide a solution in a few instances, proving that their promise is not groundless. They have been proven to be more effective than many algorithmic techniques for numerical aerodynamic flow calculations. They are also being used in the lab where they are split into a front-end network that recognizes short, phoneme-like fragments of speech which are then passed on to a back-end network. The second artificial network recognizes these strings of fragments as words.

This promising field of unsupervised learning is sometimes called self-supervised learning. These networks use no external influences to adjust their weights. Instead, they internally monitor their performance. These networks look for regularities or trends in the input signals, and makes adaptations according to the function of the network. Even without being told whether it's right or wrong, the network still must have some information about how to organize itself. This information is built into the network topology and learning rules.

An unsupervised learning algorithm might emphasize cooperation among clusters of processing elements. In such a scheme, the clusters would work together. If some external input activated any node in the cluster, the cluster's activity as a whole could be increased. Likewise, if external input to nodes in the cluster was decreased, that could have an inhibitory effect on the entire cluster.

Competition between processing elements could also form a basis for learning. Training of competitive clusters could amplify the responses of specific groups to specific

stimuli. As such, it would associate those groups with each other and with a specific appropriate response. Normally, when competition for learning is in effect, only the weights belonging to the winning processing element will be updated.

At the present state of the art, unsupervised learning is not well understood and is still the subject of research. This research is currently of interest to the government because military situations often do not have a data set available to train a network until a conflict arises.

2.6.3 Learning Rates

The rate at which A.N.N.s learn depends upon several controllable factors. In selecting the approach there are many trade-offs to consider. Obviously, a slower rate means a lot more time is spent in accomplishing the off-line learning to produce an adequately trained system. With the faster learning rates, however, the network may not be able to make the fine discriminations possible with a system that learns more slowly. Researchers are working on producing the best of both worlds.

Generally, several factors besides time have to be considered when discussing the off-line training task, which is often described as "tiresome." Network complexity, size, paradigm selection, architecture, type of learning rule or rules employed, and desired accuracy must all be considered. These factors play a significant role in determining how long it will take to train a network. Changing any one of these factors may either extend the training time to an unreasonable length or even result in an unacceptable accuracy.

Most learning functions have some provision for a learning rate, or learning constant. Usually this term is positive and between zero and one. If the learning rate is greater than one, it is easy for the learning algorithm to overshoot in correcting the weights, and the network will oscillate. Small values of the learning rate will not correct the current error as quickly, but if small steps are taken in correcting errors, there is a good chance of arriving at the best minimum convergence [12].

2.6.4 Learning Laws

Many learning laws are in common use. Most of these laws are some sort of variation of the best known and oldest learning law, Hebb's Rule. Research into different learning functions continues as new ideas routinely show up in trade publications. Some researchers have the modeling of biological learning as their main objective. Others are experimenting

with adaptations of their perceptions of how nature handles learning. Either way, man's understanding of how neural processing actually works is very limited. Learning is certainly more complex than the simplifications represented by the learning laws currently developed. A few of the major laws are presented as examples [13].

- ***Hebb's Rule:*** The first, and undoubtedly the best known, learning rule were introduced by Donald Hebb. The description appeared in his book The Organization of Behavior in 1949. His basic rule is: If a neuron receives an input from another neuron and if both are highly active (mathematically have the same sign), the weight between the neurons should be strengthened.
- ***Hopfield Law:*** It is similar to Hebb's rule with the exception that it specifies the magnitude of the strengthening or weakening. It states, "If the desired output and the input are both active and both inactive, increment the connection weight by the learning rate, otherwise decrement the weight by the learning rate.
- ***The Delta Rule:*** This rule is a further variation of Hebb's Rule. It is one of the most commonly used. This rule is based on the simple idea of continuously modifying the strengths of the input connections to reduce the difference (the delta) between the desired output value and the actual output of a processing element. This rule changes the synaptic weights in the way that minimizes the mean squared error of the network. This rule is also referred to as the Windrow-Hoff Learning Rule and the Least Mean Square (LMS) Learning Rule. The way that the Delta Rule works is that the delta error in the output layer is transformed by the derivative of the transfer function and is then used in the previous neural layer to adjust input connection weights. In other words, this error is back-propagated into previous layers one layer at a time. The process of back-propagating the network errors continues until the first layer is reached. The network type called Feedforward; Back-propagation derives its name from this method of computing the error term. When using the delta rule, it is important to ensure that the input data set is well randomized. Well ordered or structured presentation of the training set can lead to a network which can not converge to the desired accuracy. If that happens, then the network is incapable of learning the problem.
- ***The Gradient Descent Rule:*** This rule is similar to the Delta Rule in that the derivative of the transfer function is still used to modify the delta error before it is

applied to the connection weights. Here, however, an additional proportional constant tied to the learning rate is appended to the final modifying factor acting upon the weight. This rule is commonly used, even though it converges to a point of stability very slowly. It has been shown that different learning rates for different layers of a network help the learning process converge faster. In these tests, the learning rates for those layers close to the output were set lower than those layers near the input. This is especially important for applications where the input data is not derived from a strong underlying model.

- ***Kohonen's Learning Law:*** This procedure, developed by Teuvo Kohonen, was inspired by learning in biological systems. In this procedure, the processing elements compete for the opportunity to learn, or update their weights. The processing element with the largest output is declared the winner and has the capability of inhibiting its competitors as well as exciting its neighbors. Only the winner is permitted an output, and only the winner plus its neighbors are allowed to adjust their connection weights. Further, the size of the neighborhood can vary during the training period. The usual paradigm is to start with a larger definition of the neighborhood, and narrow in as the training process proceeds. Because the winning element is defined as the one that has the closest match to the input pattern, Kohonen networks model the distribution of the inputs. This is good for statistical or topological modeling of the data and is sometimes referred to as self-organizing maps or self-organizing topologies.

2.7 The Difference Between Neural Networks Traditional Computing and Expert Systems

Neural networks offer a different way to analyze data, and to recognize patterns within that data, than traditional computing methods. However, they are not a solution for all computing problems. Traditional computing methods work well for problems that can be well characterized. Balancing checkbooks, keeping ledgers, and keeping tabs of inventory are well defined and do not require the special characteristics of neural networks [14].

- ***Traditional computers:*** are ideal for many applications. They can process data, track inventories, network results, and protect equipment. These applications do not need the special characteristics of neural networks.
- ***Expert systems:*** are an extension of traditional computing and are sometimes called the fifth generation of computing. (First generation computing used switches and wires. The second generation occurred because of the development of the transistor. The third generation involved solid-state technology, the use of integrated circuits, and higher level languages like COBOL, FORTRAN, and "C". End user tools, "code generators," are known as the fourth generation.) The fifth generation involves artificial intelligence.

Typically, an expert system consists of two parts, an inference engine and a knowledge base. The inference engine is generic. It handles the user interface, external files, program access, and scheduling. The knowledge base contains the information that is specific to a particular problem. This knowledge base allows an expert to define the rules which govern a process. This expert does not have to understand traditional programming. That person simply has to understand both what he wants a computer to do and how the mechanism of the expert system shell works. It is this shell, part of the inference engine that actually tells the computer how to implement the expert's desires. This implementation occurs by the expert system generating the computer's programming itself; it does that through "programming" of its own. This programming is needed to establish the rules for a particular application. This method of establishing rules is also complex and does require a detail oriented person. Efforts to make expert systems general have run into a number of problems. As the complexity of the system increases, the system simply demands too much computing resources and becomes too slow. Expert systems have been found to be feasible only when narrowly confined.

Artificial neural networks offer a completely different approach to problem solving and they are sometimes called the sixth generation of computing. They try to provide a tool that both programs itself and learns on its own. Neural networks are structured to provide the capability to solve problems without the benefits of an expert and without the need of programming. They can seek patterns in data that no one knows are there.

2.8 Neural Networks in Face Recognition

The requirement for reliable personal identification in computerized access control has resulted in an increased interest in biometrics. Biometrics being investigated includes fingerprints, speech, signature dynamics, and face recognition. Sales of identity verification products exceed $100 million. Face recognition has the benefit of being a passive, non-intrusive system for verifying personal identity. The techniques used in the best face recognition systems may depend on the application of the system. We can identify at least two broad categories of face recognition systems.

1. We want to find a person within a large database of faces (e.g. in police database). These systems typically return a list of the most likely people in the database. Often only one image is available per person. it is usually not necessary for recognition to be done in real-time.
2. We want to identify particular people in real-time (e.g. ia a security monitoring system, location tracking system), or we want to allow access to a group of people and deny access to all others (e.g. access to a building, computer). Multiple images per person are often available for training and real-time recognition is required.

I will go to discuss this part in details in the next chapters.

2.9 Advantage and Disadvantage of Neural Networks

Artificial Neural Networks has several advantages and disadvantages. Because A.N.N. is similar to B.N.N, if parts of the network are damaged, it can still carry on its works. Another advantage is it ability to learn from limited sets of examples. However, unlike traditional program, it parts of the program are damaged, it could no longer function. Furthermore, the same neural network can be used for several programs without any modification.

The speed of the A.N.N. can be both its advantage and disadvantage. Depending on the level of AI required, a network with a larger input, hidden, and output layers may be required. If the computer is not fast enough to process the information, a tremendous amount of time may be required to process a simple question. The complexity of the network is considered to be its disadvantage because you do not know whether the network has "cheated" or not. Because a neural network can memorize and recognize patterns, it is almost impossible to find out how the network comes up with its answers. This is also known as a

black box model. For example, you can provide a neural network with several pictures of a person and ask it to recognize him/her. Due to the problem just described, it is essential test network after its training by introducing it to other inputs that network has never experienced.

2.10 Summary

This chapter presented a historical background on neural networks. We also explained what neural networks are and definitions and analogy to the brain. Also we explained by simple words how the artificial neuron works. The architecture and the structure of the neural network; here the neural networks are able to act in two ways which are feedback and feedforward neural network. Feedforward neural networks means it can only travel in one way (No Looping), and Feedback neural networks has loop which means it can travel in both directions, and the advantages and disadvantages of the neural networks.

CHAPTER THREE

TECHNIQUES USED IN FACE RECOGNITION

3.1 Overview

This chapter describes a face detection approach via learning eigenfaces and local features analysis. The first part of the chapter describes about eigenfaces. Eigenfaces are an excellent basis for face recognition system, providing high recognition accuracy and moderate insensitivity to lighting variations. The second part of the chapter details about local feature analysis. The key idea is that local features, being manifested by a collection of pixels in a local region, are learnt from the training set instead of arbitrarily defined.

3.2 Introduction

Face recognition is a well-studied problem in computer vision. Its current applications include security (ATM's, computer logins, and secure building entrances, criminal photo "mug-shot" databases, and human-computer interfaces.)

One of the more successful techniques of face recognition is Local feature analysis, and specifically eigenfaces [1, 15, 16]. Infrared images (or thermo grams) represent the heat patterns emitted from an object. Since the vein and tissue structure of a face is unique (like a fingerprint), the infrared image should also be unique (given enough resolution, you can actually see the surface veins of the face). At the resolutions used in this study (160 by 120), we only see the averaged result of the vein patterns and tissue structure. However, even at this low resolution, infrared images give good results for face

Recognition the only known usage of infrared images for face recognition is by company Technology Recognition Systems [17]. Their system does not use principle component analysis, but rather simple histogram and template techniques. They do claim to have a very accurate system (which is even capable of telling identical twins apart), but they unfortunately have no published results, which we could use for comparison.

To determine someone's identity

- The computer takes an image of that person.
- Determines the pattern of points that make that individual differ most from other people. Then the system starts creating patterns.
- Either randomly or based on the average eigenface.

- The computer constructs a face image and compares it with the target face to be identified.
- New patterns are created until a facial image that matches with the target can be constructed. When a match is found, the computer looks in its database for a matching pattern of a real person.

3.3 Eigen Faces

Developing a computational model of face recognition is quite difficult, because faces are complex, multidimensional, and meaningful visual stimuli. They are a natural class of objects, and stand in stark contrast to sine wave gratings, the "blocks world", and other artificial stimuli used in human and computer vision research [18]. Thus unlike most early Visual functions, for which we may construct detailed models or retinal or striate activity, face recognition is a very high level task for which computational approaches can currently only suggest broad constraints on the corresponding neural activity.

This chapter is focusing towards developing a sort of early, protective pattern recognition capability that does not depend on having full three-dimensional models or detailed geometry. The aim is to develop a computational model of face recognition, which is fast, reasonably simple, and accurate in constrained environments such as an office or household.

Although face recognition is a high level visual problem, there is a quite a bit of structure imposed on the task. We take advantages of some of this structure by proposing a scheme for recognition which is based on an information theory approach, seeking to encode the most relevant information in a group of faces which will best distinguish them form one another. The approach transform face images into a small set of characteristic feature images, called "eigenfaces", which are the principal components of the initial training set of face images. Recognition is performed by projecting a new image into the subspace spanned by the eigenface ("face space") and then classifying the face by comparing its position in face space with the positions of known individuals.

Automatically learning and later recognizing new faces is practical with this framework. Recognition under reasonably varying conditions is achieved by training on a limited number of characteristic views (e.g. a "straight on" view, a 45° view, and a profile view). The approach has advantage over other face recognition schemes in its speed and simplicity, learning capacity, and relative insensitivity to small or gradual changes in the face.

3.3.1 Eigen Faces for Recognition

Much of the previous work on automated face recognition has ignored the issue of just what aspects of the face stimulus are important for identification, assuming that predefined measurements were relevant and sufficient. This suggested to us that an information theory approach of coding and decoding face images may give insight into the information content of face images, emphasizing the significant local and global "features". Such features may or may not be directly related to our intuitive notion of face features such as the eyes, nose, lips and hair.

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

In mathematical terms, to find the principal components of the distributions of faces or the eigenvectors of the covariance matrix of the set of face images. These eigenvectors can be thought of as a set of features, which together characterize for variation between face images. Each image location contributes more or less to each eigenvector, so that we can display the eigenvector as sort of ghostly face, which we call an eigenface. Some of these faces are shown in figure (3.1).

Each face image in the training set can be represented exactly in terms of a linear combination of the eigenfaces. The number of possible eigenfaces is equal to the number of face images in the training set. However the faces can also be approximated using only the "best" eigenfaces- those that have the largest eigenvalues, and which therefore account for the most variance within the set of face images. The primary reason for using fewer eigenfaces is computational efficiency. The best M' eigenfaces span a M' dimensional subspace "face space" of all possible images. As sinusoids of varying frequency and phase are the basis functions of a Fourier decomposition (and are in fact eigenfunctions of linear systems), the eigenfaces are the basis vectors of the eigenface decomposition.

The idea of using eigenfaces was motivated by a technique developed by Sirovich and Kirby [2] for efficiently representing pictures of faces using principal components analysis.

They argued that a collection of face images can be approximately reconstructed by storing a small collection of weights for each face and a small set of standard pictures.

It occurred that if a multitude of face images can be reconstructed by weighted sums of a small collection of characteristic images, then an efficient way to learn and recognize faces might be to build the characteristic features from known face images and to recognize particular faces by comparing the feature weights needed to (approximately) reconstruct them with the weights associated with the known individuals.

The following steps summarize the recognition process.

1. Initialization, Acquire the training set of face images and calculate the eigenfaces, which define the face space.
2. When a new face image is encountered, calculate a set of weights based on the input image and the M eigenfaces by projecting the input image onto each of the eigenfaces.
3. Determine if the image is a face at all (whether known or unknown) by checking to see if the image is sufficiently close to "face space".
4. If it is a face, classify the weight pattern as either a known person or as unknown.
5. (Optional) If the same unknown face is seen several times, calculate its characteristic weight pattern and incorporate into the known faces (i.e. Learn to recognize it).

A general idea for face recognition is to extract the relevant information in a face image, encode it as efficiently as possible, and compare one face encoding with a database of similarly encoded images. In the eigenfaces technique, we have training and test set of images, and we compute the eigenvectors of the covariance matrix of the training set of Images. These eigenvectors can be thought of as a set of features that together characterize the variation between face images. When the eigenvectors are displayed, they look like a ghostly face, and are termed eigenfaces. The eigenfaces can be linearly combined to reconstruct any image in the training set exactly. In addition, if we use a subset of the eigenfaces, which have the highest corresponding eigenvalue (which accounts for the most variance in the set of training images), we can reconstruct (approximately) any training image with a great deal of accuracy. This idea leads not only to computational efficiency (by reducing the number of eigenfaces we have to work with), but it also makes the recognition more general and robust.

Storage: The face recognition system that we worked on builds a set of orthonormal basis vectors based on the Karhunen-Loève procedure for generation of orthonormal vectors. Using

the best (highest eigenvalue, or most face-like) of these basis vectors, which we call eigenfaces, we map images to "face-space". Using this representation, we can store each image as only a vector of N numbers where N is the number of eigenfaces. These results in huge storage savings as both the MIT group and it was concluded that 50 eigenfaces forms a fairly comprehensive set of eigenvectors for characterizing faces. Thus, 80K images are stored as 50 numbers.

Matching: Use this stored representation of the images, when presented with a new image we can map it to face-space as well and quickly see which vector it most corresponds to or whether it corresponds to any of the vectors at all. By seeing if it corresponds to any of the stored vectors better than a certain threshold we can determine who the person is. If the image does not correspond to any of the stored vectors we conclude that we do not know (or fail to recognize) the person. Also, by taking the image to face-space and then back to image space we can see how good the reconstruction is and by this determine whether the image is in fact a face or not.

Reconstruction: The ability to reconstruct the images from our stored vectors gives us both the ability for face-checking, the determination of whether the image is a face, and also image compression since the 50 values and corresponding set of eigenfaces are enough to reconstruct most any face.

Applications: The face recognition system has a number of uses, which cause apprehension.

- System (key) access based on face/voice recognition.
- Tracking people either spatially with a large network of cameras or temporally by monitoring the same camera over time. (London is currently attempting to do both).
- Locating of people in large images.

The face-key and tracking system both are based on matching faces to other faces stored in a database, while the people locating system is based on 'face-ness'. For the location task, an image is scanned and each region is converted to face-space and back to check to see if it is a face. This scanning task can be used to find everything from license plates (using eigen-license-plates) to Waldo (using eigen-Waldos).

3.4 Constructing Eigenfaces

This procedure is a form of principle component analysis. First, the conceptually simple version;

- Collect a bunch (call this number N) of images and crop them so that the eyes and chin are included, but not much else.
- Convert each image (which is x by y pixels) into a vector of length xy.
- Pack these vectors as columns of a large matrix.
- Add xy - N zero vectors so that the matrix will be square (xy by xy).
- Compute the eigenvectors of this matrix and sort them according to the corresponding eigenvalues. These vectors are your eigenfaces. Keep the M eigenfaces with the largest associated eigenvalues.

Unfortunately, this procedure relies on computing eigenvectors of an extremely large matrix. Our images are 250×300, so the matrix would be 75000 by 75000 (5.6 billion entries!). On the bright side, there's another way (the Karhunen-Loève expansion).

Collect the N images, crop them, and convert them to vectors. Compute the N by N outer product matrix (call it L) of these images. The entry L_{ij} of this matrix is the inner product of image vectors number i and j. As a result, L will be symmetric and non-negative. Compute the eigenvectors of L. This will produce N - 1 vectors of length N. Use the eigenvectors of L to construct the eigenfaces as follows: for each eigenvector v, multiply each element with the corresponding image and add those up. The result is an eigenface, one of the basis elements for face space. Use the same sorting and selecting process described above to cut it down to M eigenfaces.

Transforming an Image to Face Space

This procedure is exactly what had expected for the usual Hilbert space change of basis. Take inner products between the image and each of the eigenfaces and pack these into a vector of length M.

The Inverse Face Space Transforms

- Multiply each of the elements of the face space vector with the corresponding eigenfaces, and add up the result.
- Transform it to face space.
- Record the resulting vector (which will be much smaller than the image).

Recognizing a known Face

- Transform the image presented for recognition to face space.
- Take inner products with each of the learned face space vectors (think Cauchy-Schwartz).

- If one of these inner products is above the threshold, take the largest one and return that its owner also owns the new face.
- Otherwise, it's an unknown face. Optionally add it to the collection of known faces as "Unknown Person #1".

Evaluating "Face-ness" of an Image

If unsure whether an image is a face or not, transform it to face space, then do the inverse transform to get a new image back. Use mean-squared-error to compare these two images. If the error is too high, it isn't a face at all. Note that this process does not rely on knowing any faces, just having a set of eigenfaces.

The Face recognition is an important task for computer vision systems, and it remains an open and active area of research. To implement and experiment with a promising approach to this problem: eigenfaces.

Think of an image of a face (grayscale) as an N by N matrix - this can be rearranged to form a vector of length N^2, which is just a point in R^{N2}. That's a very high dimensional space, but picture of faces only occupy a relatively small part of it. By doing some straightforward principal component analysis (discussion of this part to be added later), a smaller set of M "eigenfaces" can be chosen (M is a design parameter), and the faces to be remembered can be expressed as a linear combination of these M eigenfaces. In other words the faces have been transformed from the image domain (where they take up lots of storage space: $\sim N^2$) to the face domain (where they require much less, $\sim M$). This will necessarily be an approximation, but it turns out to be a pretty good one in practice. To recognize a new image of a face, simply transform it to the face domain and take an inner product with each of the known faces to see if we have a match. Faces presented for recognition will be scaled, rotated, and shifted the same as they were first seen. However, changes in lighting, facial expression, etc are fair game. No hats or heavy make-up or anything silly like that.

The general implementation plan is:

1. Take some pictures with a handy digital camera (got one).
2. Scale, rotate, crop, etc the images by hand using image-editing software.
3. Construct the eigenfaces.
4. Compute and store face domain versions of each person's face.
5. Grab and fix up some more images - some of known people and some of unknown people.
6. Test the recognizer!

Most likely the actual implementation stuff will be done in some combination of Python and Matlab, unless we get crazy and decide to try this in real-time (it should be feasible these are efficient algorithms), in which case, some C will be necessary. Procedure could also serve for searching for faces in a larger image.

3.5 Computing Eigenfaces

Consider a black and white image of size N×N $I(x,y)$.$I(x,y)$ is simply a matrix of 8-bit values with each element representing the intensity at that particular pixel. These images can be thought of as a vector of dimension N^2, or a point in N^2 dimensional space. A set of images therefore corresponds to a set of points in this high dimensional space. Since facial images are similar in structure, these points will not be randomly distributed, and therefore can be described by a lower dimensional subspace. Principal component analysis gives the basis vectors for this subspace (which is called the "facespace"). Each basis vector is of length N^2, and is the eigenvector of the covariance matrix corresponding to the original face images.

So 128×128 pixel image can be represented as a point in a 16,384 dimensional space facial images in general will occupy only a small sub-region of this high dimensional "image space" and thus are not optimally represented in this coordinate system.

The eigenfaces technique works on the assumption that facial images from a simply connected sub-region of this image space. Thus it is possible, through principal components analysis (PCA) to work out an optimal co-ordinate system for facial images. Here an optimal coordinate system refers to one along which the variance of the facial images is maximized.

This becomes obvious when we consider the underlying ideas of PCA. PCA aims to catch the total variation in a set of facial images, and to explain this variation by as few variables as possible. This not only decreases the computational complexity of face recognition, but also scales each variable according to its relative importance in explaining the observation.

Let $T_1, T_2, \ldots\ldots, T_M$ be the training set of face images. The average face is defined by

$$\phi = \frac{1}{M}\sum_{i=1}^{M} T_i \tag{3.1}$$

Each face differs from the average face by the vector $\phi = T_i - \Psi$. The covariance matrix

$$C = \frac{1}{M}\sum_{i=1}^{M}\phi_i\phi_i^T \tag{3.2}$$

Has a dimension of N^2 by N^2. Determining the eigenvectors of C for typical sizes of N is intractable. We are determining the eigenvectors by solving a M by M matrix instead.

3.5.1 Classification

The eigenfaces span a M' dimensional subspace of the original N^2 image space. The M' significant eigenvectors are chosen as those with the largest corresponding eigenvalues. A test face image I' is projected into face space by the following operation $w_i = u_i^T(T - \Psi)$, for $i = 1,.....,M'$, where u_i are the eigenvectors for C. The weights w_i form a vector $\Omega^T = [w_1, w_2,...., w_{M'}]$, which describes the contribution of each eigenface in representing the input face image. This vector can then be used to fit the test image to a predefined face class. A simple technique is to use the Euclidian distance $\varepsilon_i = \|\Omega - \Omega_i\|$, where Ω_i describes the i^{th} face class. A test image is in class i when $\varepsilon_i < \theta_i$, where θ_i a user is specified threshold.

Given a vector C the eigenvectors u and Eigen values λ of C satisfy

$$\mathrm{Cu} = \lambda\,\mathrm{u} \tag{3.3}$$

The eigenvectors are orthogonal and normalized hence

$$u_i^T u_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{3.4}$$

Let T_k represent the column vector of face k obtained through lexographical ordering of $I_k(x, y)$. Now let us define ϕ_k as the mean normalized column vector for face k. this means that

$$\phi_K = T_K - \Psi \tag{3.5}$$

Where

$$\Psi = \frac{1}{M}\sum_{i=1}^{M} T_i \tag{3.6}$$

Now let C be the covariance matrix of the mean normalized faces.

$$C = \frac{1}{M}\sum_{K=1}^{M}\phi_K\phi_K^T \tag{3.7}$$

M is the number of facial images in our representation set. These facial images help to characterize the sub-space formed by faces within image space. This sub-space will henceforth be referred to as 'face-space'.

$$\begin{aligned} Cu_i &= \lambda_i u_i \\ u_i^T Cu_i &= u_i^T \lambda_i u_i \\ &= \lambda_i u_i^T u_i \end{aligned} \tag{3.8}$$

Now since $u_i^T u_j = 1$

$$\begin{aligned} u_i^T Cu_i &= \lambda_i \\ \lambda_i &= \frac{i}{M} u_i^T \sum_{k=1}^{M} \phi_k \phi_k^T u_i \\ &= \frac{1}{M} \sum_{k=1}^{M} u_i^T \phi_k \phi_k^T u_i \\ &= \frac{1}{M} \sum_{k=1}^{M} \left(u_i \phi_k^T\right)^T \left(u_i \phi_k^T\right) \\ &= \frac{1}{M} \sum_{k=1}^{M} \left(u_i \phi_k^T\right)^2 \\ &= \frac{1}{M} \sum_{k=1}^{M} \left(u_i \Gamma_k^T - mean\left(u_i \Gamma_k^T\right)\right)^2 \\ &= \frac{1}{M} \sum_{k=1}^{M} \mathrm{var}\left(u_i \Gamma_k^T\right) \end{aligned} \tag{3.9}$$

Thus eigenvalue i represent the variance of the representation facial image set along the axis describes by eigenvector i.

So by selecting the eigenvector with the largest eigenvalues as our basis, we are selecting the dimensions, which can express the greatest variance in facial images or the dominant modes of face-space. Using this coordinate system a face can be accurately reconstructed with as few a 6 coordinates. This means that a face, which previously took 16,384 bytes to represent in image space, now requires only 6 bytes. Once again, this reduction in dimensionality makes the problem of face recognition much simpler since we concern only with the attributes of the face.

3.5.2 Equipment and Procedures

The infrared camera used is a Cincinnati Electronics IRC-160. This camera has a resolution of 160 by 120 pixels, 12 bit planes, and is sensitive over the 2.5 to 5.5 nm infrared range. The IRC has a digital interface, which was connected to a Sparc 20 with an EDT SDV board.

The subjects were at a fixed distance from the camera (6.5'); a 50 mm lens was used on the IRC. Three views points were used in this study (frontal, 45^{O}, profile). In addition, for each view the subject made two expressions (normal and smile). For each expression, two images were captured 4 seconds apart. Thus a total of 12 images were captured for each subject, giving a grand total of 288 images in the database.

The faces were aligned (by hand) to improve the performance of the eigenface technique. Specifically, frontal images were aligned using the midpoint of the subject's eyes; 45^{O} 45-images were aligned on the subject's right eye; and profile images were aligned using the tip of the subject's nose. The images were not scaled in any way. The subjects did not have glasses on during the imaging, as most glasses appear completely opaque in infrared. While this may be reasonable for security applications, it isn't for most others.

3.5.3 Results

For each of the three views, 24 normal-expression images were used as the training set, and 24 smiling-expression images were used as the test set. For the frontal and 45^{O} views only one person was incorrectly classified; the profile view classified all 24 people correctly. A separate face space was used for each test. Figure (3.1) for an example of the training images, and Figure (3.2) for an example of the eigenfaces generated from this training set.

3.6 Face Recognition using Eigenfaces

Once the optimal coordinate system has been calculated any facial image can be projected into face-space by calculating its projecting onto each axis.

Thus for some test images T_a we can find its projection onto axis i w_k by

$$\varpi_k = u_k^T(\Gamma_a - \psi) \tag{3.10}$$

Now let us define the vector Ω_a, which contains the projections of T_a onto each of the dominant eigenvectors.

Figure 3.1 Training set example.

Figure 3.2 Eigen faces created form the training set.

$$\Omega_a = [\varpi_1, \varpi_2, \cdots, \varpi_{M'}] \tag{3.11}$$

Where M' is the number of dominant eigenvectors $M' << 16384$

Given a set of photographs of 30 people $T_1 - T_{30}$ we can then determine the identity of an unknown face T_a by finding which photograph it is most closely positioned to in face-space. A simplistic way to achieve this would be to determine the Euclidean distance.

$$C_n = \|\Omega_a - \Omega_n\| \tag{3.12}$$

Where Ω_n is the projection of T_n into face-space?

Statistically, the Euclidean distance can be used to model the probability that T_a and T_n are the same person through the use of a high dimensional Gaussian distribution.

This distribution will have uniform variance in each of the eigen-dimensions since we give equal weighting to each of the projection errors when we calculate the Euclidean

distance. Here the projection error is simply the difference between Ω_a and Ω_n for eigen-dimension i.

The purpose of this model is to convert the distance measure into a probability. Assuming that the data follows Gaussian distributions the relationship is as follows.

$$\begin{aligned} P\left(\left\langle T_a \middle| \Omega_n \right\rangle \middle| \left\langle T_a \middle| \Omega \right\rangle\right) &= e^{-\sum_{i=1}^{N} \frac{\Delta w_i^2}{2\alpha_i}} \\ &= e^{-\sum_{i=1}^{N} \frac{\Delta w_i^2}{2\lambda_i}} \\ &= e^{-\sum_{i=1}^{M} \frac{\Delta w_i^2}{2\lambda_i}} \cdot e^{-\sum_{i=M+1}^{N} \frac{\Delta w_i^2}{2\lambda_i}} \end{aligned} \tag{3.13}$$

Now if we consider the minor principal components to be insignificant.

$$\begin{aligned} P\left(\left\langle T_a \middle| \Omega_n \right\rangle \middle| \left\langle T_a \middle| \Omega \right\rangle\right) &\cong e^{-\sum_{i=1}^{M} \frac{\Delta w_i^2}{2P}} \\ &\cong e^{-\frac{a^2}{2P}} \end{aligned}$$

$$d = \sum_{i=1}^{N} \frac{\Delta w_i^2}{\lambda_i}$$

$$P\left(\left\langle T_a \middle| \Omega_n \right\rangle \middle| \left\langle T_a \middle| \Omega \right\rangle\right) = e^{-\frac{d}{2}} \tag{3.14}$$

Furthermore, it can be shown that the optimal value for p is simply the average of the eigenvalues for the first M principal components. Whilst this method has been shown to work, it ignores the fact that each of the eigen-dimensions exhibits a different variance. A measure, which takes this into account by normalizing each of the eigen-dimensions for unity variance, is the Mahalanobis distance. The Mahalanobis distance is defined as:

$$d = \sum_{i=1}^{N} \frac{\Delta w_i^2}{\lambda_i} \tag{3.15}$$

This distance measure will result in high-dimensional Gaussian distributions with different variances in each of the eigen-dimensions. This is illustrated below for the simplistic case of M = 2. The circular cross-section of the Euclidean distance probability model and ellipsoidal cross-section of the Mahalanobis distance probability model and the ellipsoidal cross-section of the Mahalanobis distance probability model. Here the height of the graph represents the probability that T_a and T_n are the same person, whilst the two horizontal dimensions correspond the projection error in the first and second principal components.

By a method similar to that above, the relationship between probability and the Mahalanobis distance can be found to be:

$$P\left(\left\langle T_a \middle| \Omega_n \right\rangle \middle| \left\langle T_a \middle| \Omega \right\rangle\right) = e^{-\frac{d}{2}} \tag{3.16}$$

3.7 Local Feature Analysis

Local feature analysis is derived from the eigenface method but overcomes some of its problems by not being sensitive to deformations in the face and changes in poses and lighting. Local feature analysis considers individual features instead of relying on only a global representation of the face. The system selects a series of blocks that best define an individual face. These features are the building blocks from which all facial images can be constructed.

The procedure starts by collecting a database of photographs and extracting eigenfaces from them. "Applying local feature analysis, the system selects the subset of building blocks, or features, in each face that differ most from other faces. Any given face can be identified with as few as 32 to 50 of those blocks. The most characteristic points as shown to the right are the nose, eyebrows, mouth and the areas where the curvature of the bones changes.

The patterns have to be elastic to describe possible movements or changes of expression. The computer knows that those points, like the branches of a tree on a windy day, can move slightly across the face in combination with the others without losing the basic structure that defines that face.

3.7.1 Learning Representative Features for Face Recognition

There is psychological [19] and physiological [20,21] evidence for parts based representations in the brain. Some face detection algorithms also rely on such representations. However, the spatial shape of their local features is often subjectively defined instead of being learnt from the training data set.

Yang *et al.* [22] describe a method for frontal face detection on 20x20 regions. They assign a weight to every possible pixel value at every possible location within the region.

The weights are determined by an iterative training procedure using the Winnow update rule. Once they have determined the weights they can classify any region by looking up and summing the weights corresponding to each pixel value. Thus each of their local features relies on only one pixel.

Colmenarez and Huang [23] used first order Markov Chain model over 11x11 input region to model face and non-face class conditional probabilities. To build the model, they calculate 1st order conditional probabilities for all pixels pairs, indicating that each of their local feature involve two pixels. The training procedure finds the mapping from the region

into a 1 dimensional array with maximum sum of the corresponding 1st order conditional probabilities according to the training set. Any region can then be classified as face or non-face by looking up and summing the probabilities corresponding to the intensity values of each selected pixel pair.

Schneiderman and Kanade [24] argued that local features, which are too small one pixel at the extreme, would not be powerful enough to describe anything distinctive about the object. They use multiple appearance-based detectors that span a range of the object's orientation. Each detector uses a statistical model to represent object's appearances over a small range of views, to capture variation that cannot be modeled explicitly. They use rectangular sub regions at multi-scales as local features in the statistical model. Size of those rectangles is pre-defined.

Burl and Perona [25] detected 5 types of features on the face. The left eye, right eye, nose/lip junction, left nostril, and right nostril. They assume that the feature detectors for each feature are fallible. Since they assume only one face is present in each image, at most one feature response is correct for each type of detector. Such handpicked local features can also be found in Pentland's method [25].

Rowley *et al.* [26] used a multiplayer perceptron neural network system for classification. A 20x20 input region is divided into blocks of 5x5, 10x10, or 20x5. Each hidden unit has one block as its receptive field. In their experiments with modular systems, they separately trained two or three of the above networks and then applied various methods for merging their results. Since the hidden units have only local support, we can infer that this particular network topology emphasizes local features over global one.

Viola and Jones [27] argued that the most common reason for using features rather than the pixels directly is that features can act to encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data. Given a 24x24 region, they use an exhaustive set of three kinds of Harr like rectangular features. A following AdaBoost procedure is applied to learn important features from the over complete feature set. In contrast to their method, Papageorgiou et al. [28] use a over complete set of Quadruple density 2D Harr basis at scales 4×4 and 2×2 pixels since they think the dimensions correspond to typical facial features for their 19×19 face images. They average the normalized coefficients over the entire set of example to identify the important Harr basis.

From the methods above it had been conclude that there are two main steps for learning local features. The first step determines various characteristics of the local feature, including size, shape, location and calculations over the corresponding pixels, etc. Generally

an over complete feature set is required for further selection of the features. The second step aims to find out the important features among the over complete set with the knowledge contained in the training data. Most previous face detection algorithms put learning procedure in the second step while little or no attention was put in the much, if not more, important first step. Instead, they define the spatial shape and other properties of their local features manually and intuitively.

Several existing algorithms can be applied to learn parts based representation from examples. Local feature analysis (LFA) [29] is a method for extracting local topographic representation in terms of local features. The extraction is from the global PCA basis, also based on second order statistics. The LFA representation enables use of specific local features for identification instead of a global representation.

Independent component analysis [30,31] is a linear nonorthogonal transform, which makes unknown linear mixtures of multi-dimensional random variables as statistically independent as possible. It not only decor relates the second order statistics but also reduces higher-order statistical dependencies. It extracts independent components even if their magnitudes are small whereas PCA extracts components having largest magnitudes. It is found that independent component of natural scenes are localized edge like filters [32].

The projection coefficients for the linear combinations in the above methods can be either positive or negative, and such linear combinations generally involve complex cancellations between positive and negative numbers. Therefore, these representations lack the intuitive explanation from the relationship between parts and the whole.

Non-negative matrix factorization (NMF) [33] imposes the non-negativity constraints in learning basis images. The pixel values of resulting basis images, as well as coefficients for reconstruction, are all non-negative. By this way, only non-subtractive (or additive) combinations are allowed. This ensures that the components are combined to form a whole in an accumulative means. For this reason, NMF is considered as a procedure for learning a parts based representation [33]. However, Li et al. [34] found that the non-negative basis components learned by NMF are not necessarily as localized as describe in the original NMF paper, at least for the ORL face database; moreover, the original NMF representation yields low recognition accuracy lower than can be obtained by using the standard PCA method. Motivated by these observations, they proposed a local non-negative matrix factorization (LNMF) algorithm, which optimizes the objective to learn truly localized, parts-based components. Their experimental results demonstrate that LNMF basis leads to much more

stable recognition results when there are occlusions, better than the standard NMF and PCA methods.

LNMF is employed to learn parts-based components. It has been applied on the input region (I) and (1-I) to get both bright local components and dark local components, suppose the input region (I) has the pixel value in the range of [0, 1]. Each local feature is calculated from a bright component and a dark one. We can then construct a face detector by selecting a small number of important features using AdaBoost from the over complete local feature set.

3.7.2 NMF and Constrained NMF

Given a set of NT training images represented as an n×NT matrix $X = [x_{ij}]$, each column of which contains n nonnegative pixel values. Denote a set of m≤n basis images by a $n \times m$ m matrix W. Each image can be represented as a linear combination of the basis images (eigenvectors of unit length), and hence the (approximate) factorization $X \approx WH$.

Where H is the matrix of $m \times NT$ coefficients or weights, dimension reduction is achieved when $m < n$.

The PCA factorization requires that the basis images (columns of W be orthonormal and the rows of H be mutually orthogonal. It imposes no other constraints than the orthogonally, and hence allows the entries of W and H to be of arbitrary sign. The NMF and LNMF, however, allow only positive coefficients and thus additive combinations of basis components.

3.7.3 NMF

NMF imposes the non-negativity constraints instead of the orthogonality. As the result, the entries of w and h are all non-negative. This way, only additive combinations are allowed, and no subtractions can occur. This is believed to be compatible to the intuitive notion of combining parts to form a whole, and is how NMF learns a parts-based representation [36]. It is also consistent with the physiological fact that the firing rate is non-negative. NMF uses the divergence of X from Y =WH, defined as

$$D(X \| Y) = \sum_{i,j} \left[x_{ij} \log \frac{x_{ij}}{y_{ij}} - x_{ij} + y_{ij} \right] \tag{3.17}$$

As the measure of cost for factorizing X into WH, An NMF factorization is defined as a solution to the following constrained optimization problem

$$\min_{W,H} D(X \| WH) \qquad s.t \quad WH \geq 0, \sum_i w_{ij} = 1 \ \forall j \tag{3.18}$$

Where W, H≥ 0 means that all entries of W and H are nonnegative.

3.7.4 Constrained NMF

The NMF model defined by (3) does not impose any constraints on the spatial locality. Therefore, minimizing the objective function can hardly yield a factorization, which reveals local features in the data X. LNMF is aimed to improve the locality of the learned features by imposing additional constraints. Let $(W^T W) = U = [u_{ij}], (HH^T) = V = [v_{ij}]$. The following three additional constraints are imposed on the NMF basis:

1. The number of basis components, which is required to represent X, should be minimized. This requires that a basis component should not be further decomposed into more components. Let w_j be a basis vector. Given the existing constraints $\sum w_{ij} = 1$ for all j, the value $\sum_i w_{ij}^2$ 2 should be as small as possible so that w_j contains as many non-zero elements as possible. This constraint can be formulated as minimizing $\sum i \ u_{ii}$.
2. To minimize redundancy between different bases, different bases should be as orthogonal as possible. This can be imposed by minimizing $\sum_{i \neq j} u_{ij}$
3. Only basis containing most important information need to be retained. Given that every image in X is normalized into a range such as in [0, 1], the total "activity" on each component, i.e. the total squared projection coefficients summed over all training images, should be maximized. This is imposed by $\sum i \ vii = \max$.

Incorporating the above constraints into the original NMF formulation, the new objective function for LNMF is:

$$D(X \| WH) = \sum_{i,j} \left[x_{ij} \log \frac{x_{ij}}{y_{ij}} - x_{ij} + y_{ij} \right] + \alpha \sum_{i,j} u_{ij} - \beta \sum_i v_{ii} \tag{3.19}$$

Where $\alpha, \beta > 0$ are some constants

A comparison shown in Figure (3.4) gives the different factorization results (image basis) of NMF and LNMF in our face database. LNMF basis are obviously more localized than NMF basis. One should note that because of the orthogonality constraint , the coefficient

matrix H is no longer sparse in LNMF as it is in NMF. But this takes no effect since only image basis has been used.

Figure 3.3 Factorization result of 49 bases on face database.

Figure 3.4 Constrained NMF Obviously LNMF has more localized basis.

3.7.5 Getting Local Features

Investigating the Harr-like features used in Viola's [27] and Papageorgiou's [28] systems, we notice that differential operator is robust to varying lighting. Inspired by

this, we desire to get local components that contain both bright and dark parts of the faces, and then put differential operator on bright and dark components to get the final value.

To achieve this, each sample (I) in X is mapped into X' as (1-I), suppose (I) have its pixel value in range [0, 1]. Then apply LNMF on both the sample set X and X' to get two sets of basis, W and W', which could be used as bright and dark components, respectively.

This can be explained as below. Recall that in last section, the matrix $V = (H H^T)$ indicates the energy relationship between the basis (include each basis itself). From the experiment we find that the values of the entries of V matrix are much closed to each other, implying that each basis contribute roughly the same to the whole data set. Thus we cannot say individually which component is more "bright" than others. That is why LNMF is performed on the other sample set X.

Given the two basis sets W and W', for each input region we can get two coefficient vector h and h'. The local feature set corresponding to the basis sets could be $\{hi - h_j^{'}\} \forall i, j$. In practice, several local feature sets, correspond to different basis sets, are combined together to form an over complete feature set. In next section, AdaBoost is applied on the set to select important features and construct the classifier at the same time.

3.7.6 AdaBoost for Feature Selection

After the process described in previous section, an over-complete set of local features has been obtained. Using the entire feature set is obviously infeasible in practice. Oppositely, we seek for an approach to select those most discriminating features. Viola [27] uses a variant of AdaBoost to select features from an overcomeplete Harr-like feature set and train the classifier. The similar method is being used in this project.

The AdaBoost algorithm was first introduced in 1995 by Freund and Schapire [35]. In its original forms, the goal of AdaBoost is to improve the performance of any given classification algorithms via combining a collection of classification functions to form a stronger classifier. These classification functions, in the language of boosting, are usually called weak learners. The major idea of AdaBoost is to enforce the weak learners to focus on the examples misclassified by previous classifiers. It does this by adjusting the weight of each training sample. In the initial state, all weights are set equally but on each round of training, the weights of misclassified samples will be increased in the proportion of previous classification errors.

Viola et al. adapted this greedy boosting procedure to feature selection. The weak learner is restricted to a set of classification functions while each of which depends on only one single feature. For each feature, the weak learner determines an optimal threshold classification function, such that the number of misclassified examples is minimized.

The procedure of applying AdaBoost to feature selection [27] can be formulated as follows. Given a set of training examples $(x_1, y_1),(x_2, y_2),\cdots,(x_m, y_m)$ where x_1 represents 20×20 image patterns and $y_1 = 0,1$ for faces and nonfaces examples respectively, assign a weight value w_i to each. Example (x_i, y_i). Before the training all w_i are equal and the sums of all eights are normalized to unit. For each feature, we train a simple Bayesian classifier which is restricted to this single feature. The classification error is evaluated with respect to $w_i, \varepsilon_j = \sum w_i \left| h_j(x_i) - y_i \right|$. The classifier, h_i with the lowest error ε_j chosen as one component of the final strong classifier and its importance in final classification function is determined by classification rate. Subsequently all weights are updated in terms of the training error before next round of training.

Besides Viola's successful experience, the formal guarantees provided by the AdaBoost learning procedure are quite strong. Freund and Schapire [35] proved that the training error of the strong classifier approaches zero exponentially in the number of rounds. More importantly a number of results were later proved about generalization performance. The key insight is that generalization performance is related to the margin of the examples, and that AdaBoost achieves large margins rapidly.

Using the same learning framework, we can now compare our learnt local features with Viola's Harr like features. This comparison is done on the training set which contains 5,000 face samples and 10,000 non-face samples. LNMF representations of dimensions 25, 36, 45, 47, 49, 51, 53, 55, 64, 81, 100 are computed from the training set to form a feature set with 37648 local features. From each features set, we select 200 features. The error ε_i of first 20 features is shown in Figure (3.5) Figure (3.6) shows the ROC curves of the two classifiers on our testing set which contains 2000 face samples and 5000 non-face samples.

3.7.7 Experimental Results

This section describes the final face detection system, including training data preparation, training procedure, and the performance comparison with state-of-the-art face detection system.

2.7.8 Training Data Set

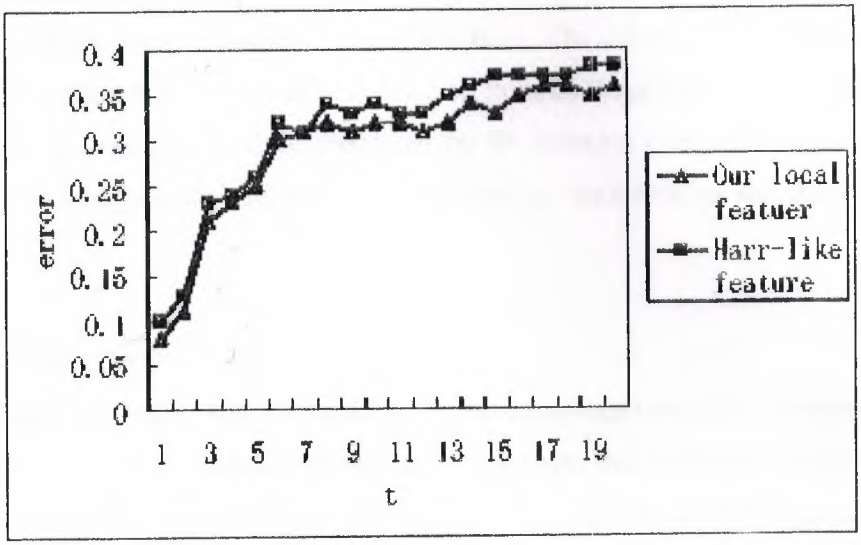

Figure 3.5 Comparison of local feature set with Viola's Harr-like feature set using the first 20 features selected by AdaBoost.

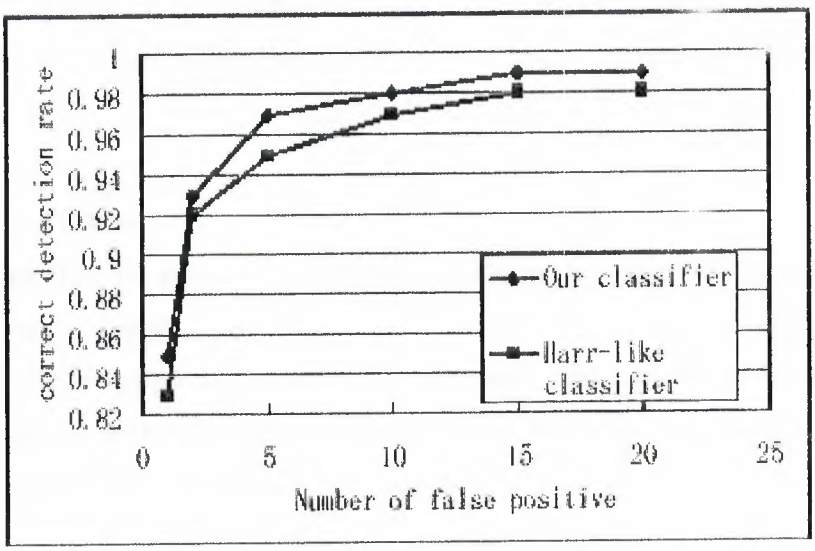

Figure 3.6 ROC curves of the two-classifier using 200 features selected from local feature set and Harr-like feature set.

The frontal face images are collected from databases of CMU, Rockefeller, Umist, Corel and our own database. There are more than 7,000 faces in total. 5,000 of them are selected as positive training samples and 2,000 as testing samples. Each face image is resized

into 20x20 and aligned by the center point of the two eyes and the horizontal distance between the two eyes.

For non-face training set, an initial 10,000 non-face samples were selected randomly from 15,000 large images, which contain no face. The 5,000 testing non-face samples mentioned in section 4 are also randomly selected from the large images. All samples, both in training set and in testing set, are processed by illumination compensation and histogram equalization to minimize the effect of different lighting conditions, as was done in Rowley's method.

3.7.9 Training Phase

The similar feature selection framework has been used with Viola's method [27]. The final detector is a 29-layer cascade of classifier. 2 features had been used in the first layer, 5 features in the second layer, and 20 features in three layers. In the fifteenth layers 200 features are used for training the classifier.

The initial 10,000 non-face samples are used to train the first three layers. In subsequent layers, scanning the partial cascade across large non-face images and collecting false positive samples obtain non-face samples. Different sets of nonface sub-windows are used in training the different classifiers to ensure that they are somewhat independent and use different features.

3.7.10 Testing Phase

The face detector is tested on the images collected from the MIT+CMU test set [37]. For an input image, scanning each 20 x 20 sub-window exhaustively in both spatial and scale space, as was done is Rowley's system [26]. The starting scale is 1, the scale step is 1.25 and the spatial step is 1 pixel at each scale level. Results from different scale levels or spatial locations are merged to get the final result.

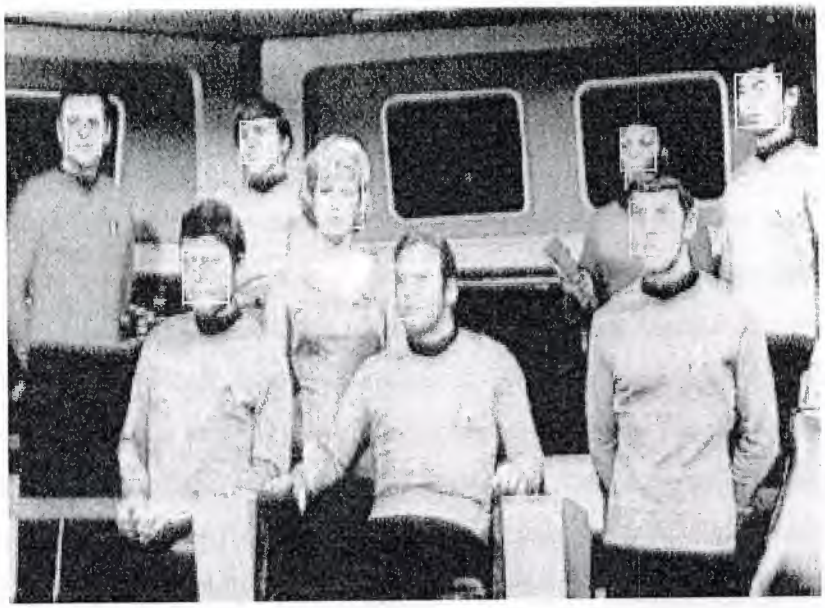

Figure 3.7 An example image of Output by face detector.

3.8 Face Modeling for Recognition

Current trend in face recognition is to use 3D face model explicitly. As an object-centered representation of human faces, 3D face models are used to overcome the large amount of variations present in human face images. These variations, which include extra-subject variations (individual appearance) and intra-subject variations (3D head pose movement, facial expression, lighting, and aging) are still the primary challenges in face recognition. However, the three major recognition algorithms [38] merely use viewer-centered representations of human faces: **(i)** a PCA-based algorithm; **(ii)** An LFA-based (local feature analysis) algorithm; and **(iii)** a dynamic-link-architecture-based paradigm.

Researchers in computer graphics have been interested in modeling human faces/heads for facial animation. We briefly review three major approaches to modeling human faces. DeCarlo et al. [39] use the anthropometric measurements to generate a general face model. This approach starts with manually constructed B-spline surfaces and then applies surface fitting and constraint optimization to these surfaces.

In the second approach, facial measurements are directly acquired from 3D digitizers or structured light range sensors. Water's [40] face model is a well-known instance. A morphable model [41] was constructed from a linear combination of eigenshapes and a linear combination of eigentextures, based on laser scans of 200 subjects. The third approach, in which models are reconstructed from photographs, only requires low-cost and passive input

devices (video cameras). For instance, Chen and Medioni [42] build face models from a pair of stereo images. However, currently it is still difficult to extract sufficient information about the facial geometry only from 2D images. This difficulty is the reason why Guenter et al. [43] utilize a large number of fiducially points to capture 3D face geometry for photo realistic animation. Even though we can obtain dense 3D measurements from high-cost 3D digitizers, it still takes too much time to scan every face. Hence, advanced modeling methods, which incorporate some prior knowledge of facial geometry, are needed. Reinders et al. [44] use a fairly coarse wire-frame model, compared to Water's model, to do model adaptation for image coding. Lee et al. [45] modify a generic model from two orthogonal pictures (frontal and side views), or from range data for animation. Lengagne et al. [46] and Fua [47] fit a range animation model to uncalibrated videos using bundle-adjustment and least squares fitting, given five manually selected features points and initial camera positions. Zhang [48] deforms a generic mesh model to an individual's face based on two images, each of which contains five manually picked markers.

A face modeling method is proposed, which adapts an existing generic face model (a priori knowledge of human face) to an individual's facial measurements. Our goal is to employ the learned 3D model to verify the presence of an individual in a face image database/video, based on the estimates of head pose and illumination.

3.8.1 Face Modeling

An individual face model is starting with a generic face model, instead of extracting isosurfaces directly from facial measurements (range data or disparity maps), which are often noisy (e.g., near ears and nose) as well as time consuming, and usually generates equal-size triangles. Our modeling process aligns the generic model using facial measurements in a global-to-local way so that feature points/ regions that are crucial for recognition are fitted to the individual's facial geometry.

3.8.2 Generic Face Model

The Water's animation model has been chosen [49], which contains 256 vertices and 441 facets for one half of the face. The use of triangular meshes is suitable for the free-form shapes like faces and the model captures most of the facial features that are needed for face recognition. Figure (3.8) shows the frontal and a side view of the model, and features such as eyes, nose, mouth, face border, and chin. There are openings at both the eyes and the mouth.

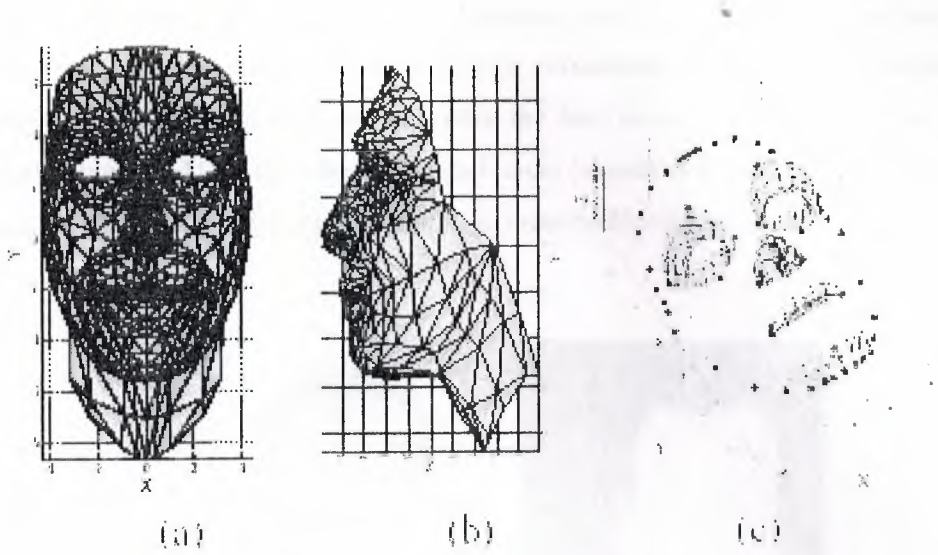

Figure 3.8 3D triangle-mesh model and its feature components ;(a) Frontal view;(b) Slide view;(c) feature components.

3.8.3 Facial Measurements

Facial measurements include information about face shape and face texture. 3D shape information can be derived from a stereo pair combined with shape from shading, a sequence of frames in a video, or obtained directly from range data. The range database of human faces used here [50] was acquired using a Minolta Vivid 700 digitizer. It generates a registered200× 200 range map and a 400× 400color image. Figure (2.9)(a,b) shows a range map and a color image of a frontal view, and the texture-mapped appearance. The locations of face and facial features such as eyes and mouth in the black and white image can be obtained by our face detection algorithm [51]. The corners of eyes, mouth, and nose can be easily obtained based on the locations of detected eyes and mouth. Figure (2.10)(c,d,e) shows the detected feature points.

3.8.4 Model Construction

Our face modeling process consists of global alignment and local adaptation. Global alignment brings the generic model and facial measurements into the same coordinate system. Based on the head pose and face size, the generic model is translated and scaled to fit the facial measurements. Figure (3.10) shows the global alignment results in two different modes. Local adaptation consists of local alignment and Local feature refinement. Local alignment

involves translating and scaling several model features, such as eyes, nose, mouth, and chin to fit the extracted facial features. Local feature refinement makes use of displacement propagation and 2.5D active contours smoothen the face model and to refine local features. Both the alignment and the refinement of each feature (shown in Figure 3.8(c)) is followed by displacement (of model vertices) propagation, in order to blend features in the face model.

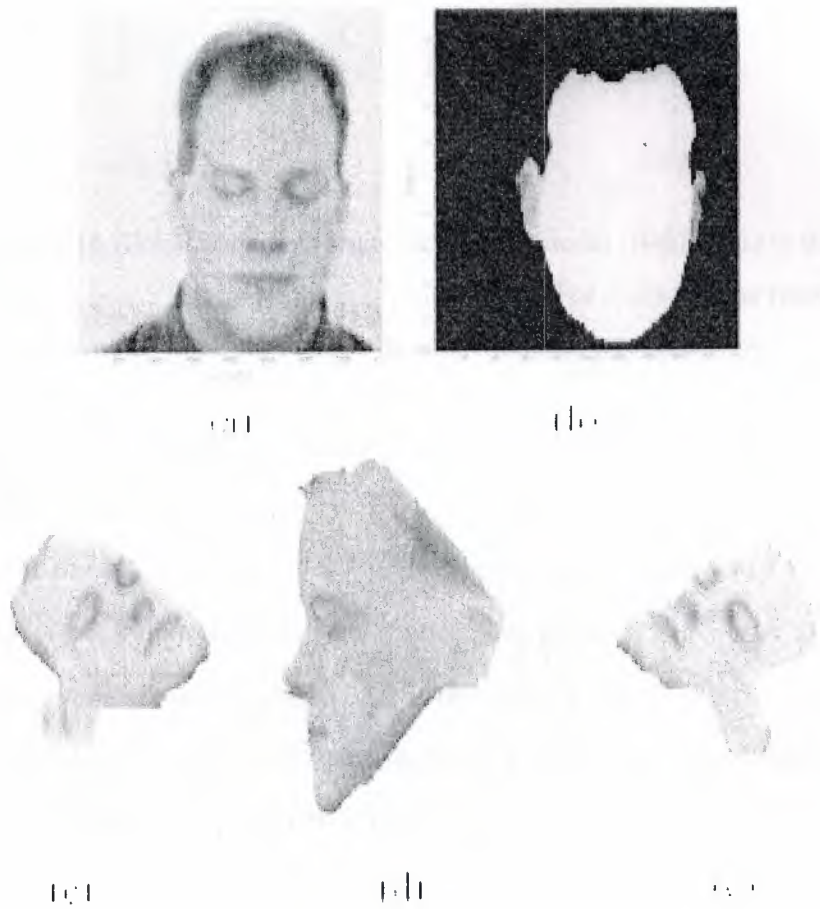

Figure 3.9 Range data of a face (a) Color texture; (b) Range map; and with texture mapping of (c) A left view; (d) A profile view; (e) A right view.

Figure 3.10 Global alignment from the generic model (Bold lines) to the facial measurements (gray lines): the target mesh is plotted in (a) For a hidden line removal mode for a frontal view; (b) For see-through mode for a profile view.

Displacement propagation inside a triangular mesh mimics the transmission of message packets in computer networks. Let N_i be the number of vertices that are connected to a vertex i, J_i be the set of all the indices of vertices that are connected to a vertex i, w_i be the sum of weights from all vertices that are connected to vertex i, and d_{ij} be the Euclidean distance between a vertex V_i and a vertex V_j ΔV_j is the displacement of vertex V_j, and $\alpha\_$ is a decay factor, which can be determined by the face size and the size of active facial feature in each coordinate. Equation (2.20) computes the contribution of vertex V_j to the displacement of vertex V_i

$$\Delta V_{ij} = \begin{cases} \Delta V_i & \dfrac{w_i - d_{ij}}{w_i \cdot [N_i - 1]} \cdot e^{-\Delta d_{ij}} \quad Where \quad N_i > 1, w_i = \sum_{j \in J_i} d_{ij} \\ \Delta V_j & e^{-\Delta d_{ij}} \quad Where \quad N_i = 1, j \in J_i \end{cases} \tag{3.20}$$

The total displacement ΔV_i of V_i can be obtained by summing up all the displacement contributed from its neighbor vertices. The displacement will decay during propagation and it continues for a few iterations, which is determined by the number of edge connections from the current feature to the nearest neighbor feature. In the future implementation, we will include symmetric property of a face and facial topology in computing this displacement. Figure (3.11) shows the results of local alignment for a frontal view.

Local feature refinement follows local alignment to further adapt the results of alignment to an individual face by using 2.5D active contours (snakes). We modify Amini et al.'s [52] 2D snakes for our 3D active contours on boundaries of facial features.

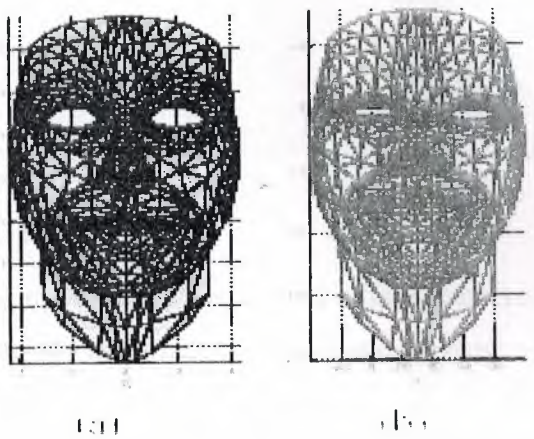

Figure 3.11 Local Feature alignment and displacement and displacement propagation shown for frontal views: (a) The generic model ;(b) The model adapted to eyes, nose, mouth, and chin.

Hence, the crucial initial contours for fitting the snakes are known in our generic face model. Another important point for fitting snakes is to find appropriate external energy maps that contain local maximum/minimum at the boundaries of facial features. For the face and the nose, the external energy is computed by the maximum magnitude of vertical and horizontal gradients from range measurements. These two facial features have steeper borders than others. For features such as eyes and the mouth, the product of the magnitude obtains the external energy of the luminance gradient and the squared luminance. Figure (3.12) shows the results of local refinement for the left eye and nose.

Although our displacement propagation smoothes nonfeature skin regions in local adaptation, they can be further updated if a dense range map is available. Figure (3.13) shows the overlay of the final adapted face model in red and the target facial measurements in blue, for a comparison with.

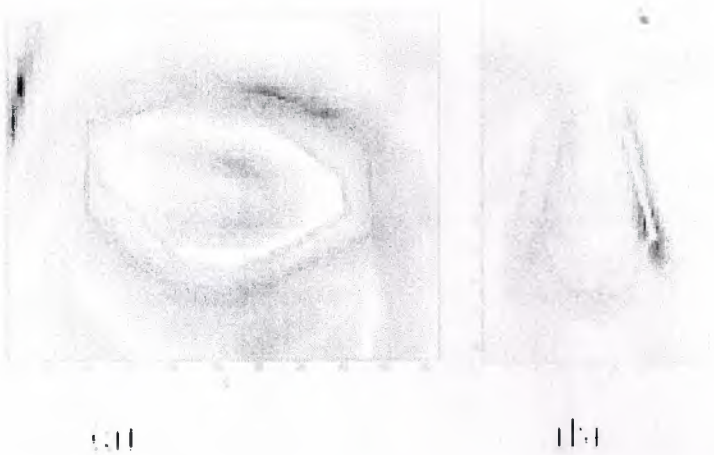

Figure 3.12 Boundary alignment: initial (inner) and refined (outer) contours overlaid on the energy maps for (a) Left eye and (b) Nose.

Figure (3.9) shows the texture-mapped face model. Further face recognition algorithm [53] is used to demonstrate the use of 3D model. The training database contains504 image from 28 subject's and 15_images generated from our 3D face model, shown in Figure (3.14). All the 10_test images were correctly matched.

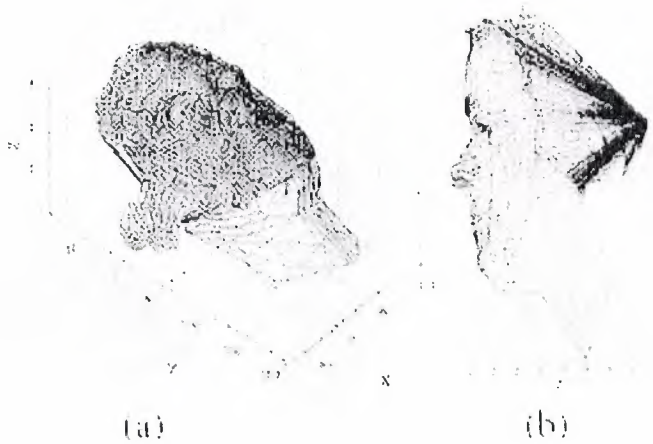

Figure 3.13 The adapted model (gray lines) overlapping the target measurement(Dark lines):The adapted model plotted (a) in 3D;(b) With colored facets at a profit view.

3.8.5 Future Work

Face modeling plays a crucial role in face recognition systems. A method had been adapted for generic face.

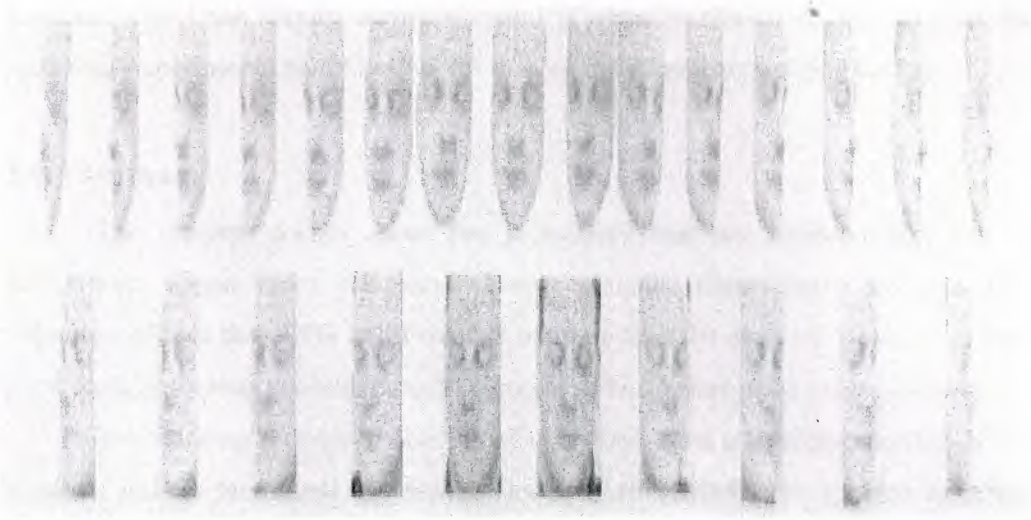

Figure 3.14 Face matching: The first row shows the 15 training images from the 3D model; the second shows 10 test images captured from a CCD camera.

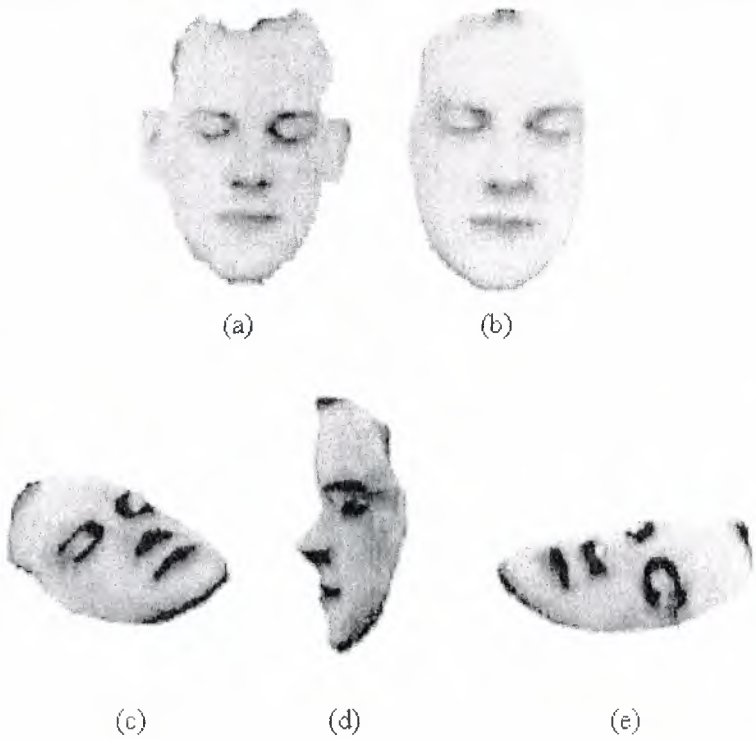

Figure 3.15 The texture-mapped (a) Input range image; adapted mesh model (b) From a frontal view ;(d) from a left view; (e) from a profile view; (f) Form a right view.

Model to input facial features in a global-to-local fashion. The model adaptation first aligns the generic model globally, and then aligns and refines each facial feature locally using displacement (of model vertices) propagation and active contours associated with facial

features. The final texture mapped model is visually similar to the original face. Initial matching experiments based on the 3D face model show encouraging results.

3.9 Summary

The chapter details about the techniques that are implemented now a day for recognition. Eigen faces and local feature analysis. Eigen faces are sensitive to scale reduction of less than 88% and rotations of more than 10 degrees. Hence it is essential that good scale and rotation normalization algorithms be applied prior to recognition.

The learning procedure consists of two steps. First a modified version of NMF (Non-negative matrix factorization), namely local NMF (LNMF), is applied to select a small number of local features. Second, a learning algorithm based on AdaBoost is used to select a small number of local features and yields extremely efficient classifiers. Experiments are presented which show the face detection performance is comparable to the state-of-art face recognition systems.

CHAPTER FOUR

FACE RECOGNITION USING NEURAL NETWORKS

4.1 Overview

This chapter describes a face recognition approach via learning hybrid neural network. In first part of this chapter describes about some related work to this technique and an introduction to the technique. The second part of the chapter details about face recognition using hybrid neural net work technique. Finally some experimental results are shown, at the end summary of this chapter.

4.2 Introduction

The requirement for reliable personal identification in computerized access control has resulted in an increased interest in biometrics. Face recognition has the benefit of being a passive, non-intrusive system for verifying personal identity. The techniques used in the best face recognition systems may depend on the application of the system. We can identify at least two broad categories of face recognition systems.

1. We want to find a person within a large database of faces (e.g. in a police database). These systems typically return a list of the most likely people in the database [4]. Often only one image is available per person. It is usually not necessary for recognition to be done in real-time.
2. We want to identify particular people in real-time (e.g. in a security monitoring system, location tracking system, etc.), or we want to allow access to a group of people and deny access to all others (e.g. access to a building, computer, etc.). Multiple images per person are often available for training and real-time recognition is required. In this paper, we are primarily interested in the second case. We are interested in recognition with varying facial detail, expression, pose, etc. We do not consider invariance to high degrees of rotation or scaling we assume that a minimal preprocessing stage is available if required. We are interested in rapid classification and hence we do not assume that time is available for extensive preprocessing and normalization.
3. The ORL database has been used which contains a set of faces taken between April 1992 and April 1994 at the Olivetti Research Laboratory in CambridgeUK^3. There are 10 different images of 40 distinct subjects. For some of the subjects, the images were taken at different times. There are variations in facial expression

(open/closed eyes, smiling/non-smiling), and facial details (glasses/no glasses). All the images were taken against a dark homogeneous background with the subjects in an up-right, frontal position, with tolerance for some tilting and rotation of up to about 20 degrees. There is some variation in scale of up to about 10%. Thumbnails of all of the images are shown in figure (4.1) and a larger set of images for one subject is shown in figure (4.2). The images are grayscale with a resolution of 92×112.

4.3 Related Work

4.3.1 Geometrical Features

Many people have explored geometrical feature based methods for face recognition. Kanade [55] presented an automatic feature extraction method based on ratios of distances and reported a recognition rate of between 45-75% with a database of 20 people. Brunelli and Poggio [54] compute a set of geometrical features such as nose width and length, mouth position, and chin shape. They report a 90% recognition rate on a database of 47 people. However, they show that a simple template matching scheme provides 100% recognition for the same database. Cox et al. [56] have recently introduced a mixture-distance technique which achieves a recognition rate of 95% using a query database of 95 images from a total of 685 individuals. Each face is represented by 30 manually extracted distances. Systems, which employ precisely measured distances between features, may be most useful for finding possible matches in a large mugshot database. For other applications, automatic identification of these points would be required, and the resulting system would be dependent on the accuracy of the feature location algorithm. Current algorithms for automatic location of feature points do not provide a high degree of accuracy and require considerable computational capacity.

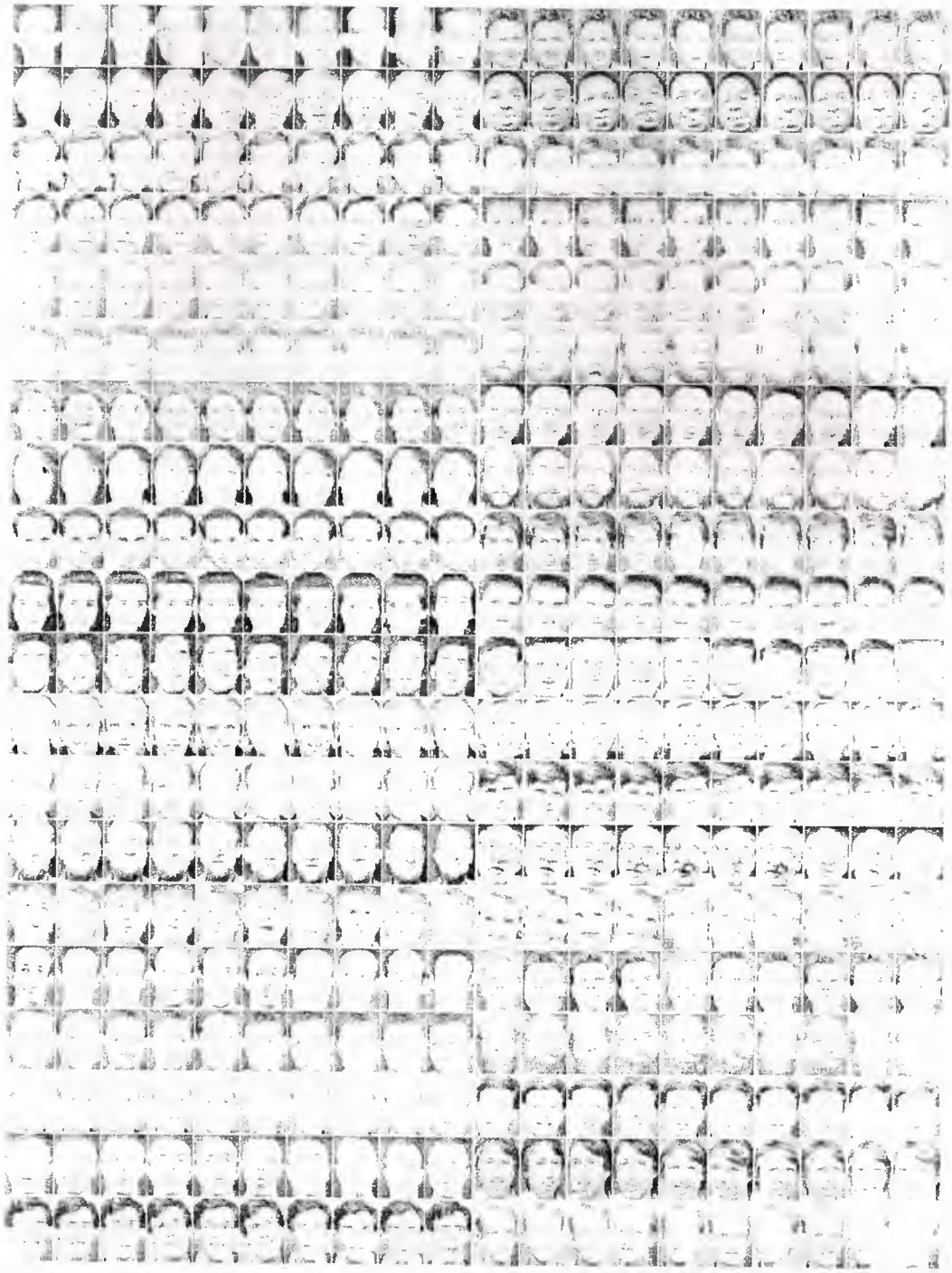

Figure 4.1 The ORL face database. There are 10 images each of the 40 subjects.

Figure 4.2 The set of 10 images for one subject. Considerable variation can be seen.

4.3.2 Eigenfaces

High-level recognition tasks are typically modeled with many stages of processing as in the Marr paradigm of progressing from images to surfaces to three-dimensional models to matched models. However, Turk and Pentland [1] argue that it is likely that there is also a recognition process based on low-level, two dimensional image processing. Their argument is based on the early development and extreme rapidity of face recognition in humans, and on physiological experiments in monkey cortex which claim to have isolated neurons that respond selectively to faces. However, it is not clear that these experiments exclude the sole operation of the Marr paradigm.

Turk and Pentland [1] present a face recognition scheme in which face images are projected onto the principal components of the original set of training images. The resulting *eigenfaces* are classified by comparison with known individuals. Turk and Pentland present results on a database of 16 subjects with various head orientation, scaling, and lighting. Their images appear identical otherwise with little variation in facial expression, facial details, pose, etc. For lighting, orientation, and scale variation their system achieves 96%, 85% and 64% correct classification respectively. Scale is renormalized to the eigenface size based on an estimate of the head size. The middle of the faces is accentuated, reducing any negative affect of changing hairstyle and backgrounds.

In Pentland et al. [15] good results are reported on a large database (95% recognition of 200 people from a database of 3,000). It is difficult to draw broad conclusions as many of the images of the same people look very similar, and the database has accurate registration and alignment [5]. In Moghaddam and Pentland [16], very good results are reported with the

FERET database only one mistake was made in classifying 150 frontal view images. The system used extensive preprocessing for head location, feature detection, and normalization for the geometry of the face, translation, lighting, contrast, rotation, and scale.

4.3.3 Template Matching

Template matching methods such as [57] operate by performing direct correlation of image segments. Template matching is only effective when the query images have the same scale, orientation, and illumination as the training images [4].

4.3.4 Graph Matching

Another approach to face recognition is the well known method of Graph Matching. A Dynamic Link Architecture for distortion invariant object recognition which employs elastic graph matching to find the closest stored graph. Objects are represented with sparse graphs whose vertices are labeled with a multi-resolution description in terms of a local power spectrum, and whose edges are labeled with geometrical distances. They present good results with a database of 87 people and test images composed of different expressions and faces turned 15 degrees. The matching process is computationally expensive, taking roughly 25 seconds to compare an image with 87 stored objects when using a parallel machine with 23 transporters. An updated version of the technique is then used which compares 300 faces against 300 different faces of the same people taken from the FERET database. They report a recognition rate of 97.3%. The recognition time for this system was not given.

4.3.5 A Hybrid Neural Network Approach

Much of the present literature on face recognition with neural networks presents results with only a small number of classes. In the first 50 principal components of the images are extracted and reduced to 5 dimensions using an auto associative neural network. The resulting representation is classified using a standard multi-layer perceptron. Good results are reported but the database is quite simple: the pictures are manually aligned and there is no lighting variation, rotation, or tilting. There are 20 people in the database.

A hierarchical neural network which is grown automatically and not trained with gradient-descent was used for face recognition by Weng and Huang [53]. They report good results for discrimination of ten distinctive subjects.

4.3.6 The ORL Database

In [58] a HMM-based approach is used for classification of the ORL database images. The best model resulted in a 13% error rate. Samaria also performed extensive tests using the popular eigenfaces algorithm [5] on the ORL database and reported a best error rate of around 10% when the number of eigenfaces was between 175 and 199. We implemented the eigenfaces algorithm and also observed around 10% error. In [58] Samaria extends the top-down HMM of [58] with pseudo two-dimensional HMMs. The error rate reduces to 5% at the expense of high computational complexity – a single classification takes four minutes on a Sun Sparc II. Samaria notes that although an increased recognition rate was achieved the segmentation obtained with the pseudo two-dimensional HMMs appeared quite erratic. Samaria uses the same training and test set sizes as we do (200 training images and 200 test images with no overlap between the two sets). The 5% error rate is the best error rate previously reported for the ORL database that we are aware of.

4.4 System Components

4.4.1 Local Image Sampling

Two different methods of representing local image samples have been evaluated. In each method a window is scanned over the image as shown in figure (4.3).

1 The first method simply creates a vector from a local window on the image using the intensity values at each point in the window. Let x_{ij} be the intensity at the i^{th} column, and the j^{th} row of the given image. If the local window is a square of sides long, centered on, then the vector associated with this window is simply

$$[x_i -, j-w, x_i - w, j-w+1, \ldots, x_{ij}, \ldots, x_i + w, j+w-1, x_i + w, j+w]$$

2 The second method creates a representation of the local sample by forming a vector out of a) the intensity of the center pixel x_{ij} and b) the difference in intensity between the center pixel and all other pixels within the square window. The vector is given by

$$[x_{ij} - x_i - w, j-w, x_{ij} - x_i - w, j-w+1, \cdots, w_{ij}, x_{ij}, \cdots, x_{ij} - x_i + w, j+w-1, x_{ij} - x_i + w, j+w]$$

The resulting representation becomes partially invariant to variations in intensity of the complete sample. The degree of invariance can be modified by adjusting the weight w_{ij} connected to the central intensity component.

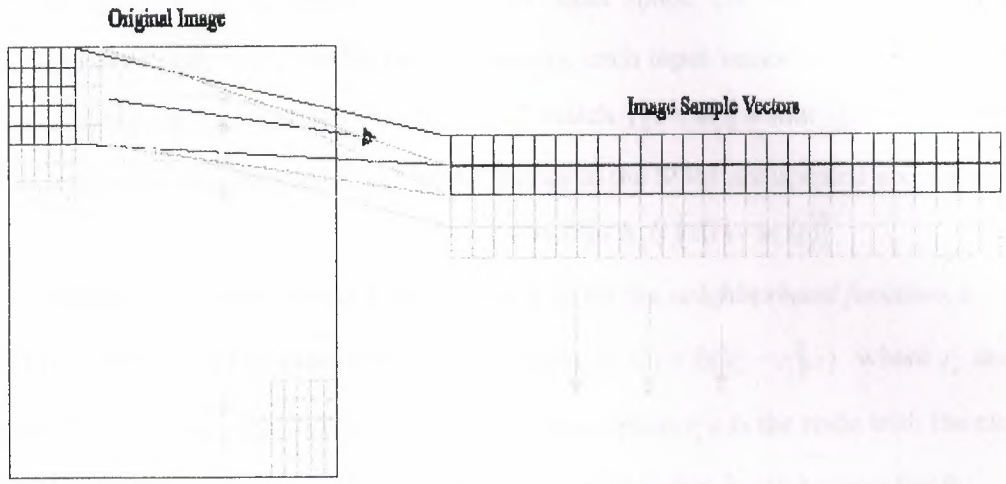

Figure 4.3 A depiction of the local image sampling. A window is stepped over the image and a vector is created at each location.

4.5 The Self-Organizing Map

Maps are an important part of both natural and artificial neural information processing systems. Examples of maps in the nervous system are retinotopic maps in the visual cortex, tonotopic maps in the auditory cortex, and maps from the skin onto the somato sensoric cortex. The self-organizing map, or SOM, introduced by Teuvo Kohonen [1] is an unsupervised learning process which learns the distribution of a set of patterns without any class information. A pattern is projected from an input space to a position in the map information is coded as the location of an activated node. The SOM is unlike most classification or clustering techniques in that it provides a topological ordering of the classes. Similarity in input patterns is preserved in the output of the process. The topological preservation of the SOM process makes it especially useful in the classification of data which includes a large number of classes. In the local image sample classification, for example, there may be a very large number of classes in which the transition from one class to the next is practically continuous (making it difficult to define hard class boundaries).

4.5.1 Algorithm

The SOM defines a mapping from an input space R^n onto a topologically ordered set of nodes, usually in a lower dimensional space. An example of a two-dimensional SOM is shown in figure 4. A reference vector in the input space, $,m_i = [\mu_{i1}, \mu_{i2}, ..., \mu_{im}]^T \varepsilon R^n$, is assigned to each node in the SOM. During training, each input vector, x, is compared to all of the m_i , obtaining the location of the closest match ($\|x - m_c\| = \min_i \{\|x - m_i\|$). The input point is mapped to this location in the SOM. Nodes in the SOM are updated according to.

$$m_i(i+1) = m_i(i) + h_{ci}(t)[x(t) - m_i(t)] \tag{4.1}$$

Where t is the time during learning and $h_{ci}(t)$ is the *neighborhood function*, a smoothing kernel which is maximum at m_c . Usually, $h_{ci}(t) = h(\|r_c - r_i\|, t)$ where r_c and r_i represent the location of the nodes in the SOM output space. r_c c is the node with the closest weight vector to the input sample and r_i i ranges over all nodes. $h_{ci}(t)$ Approaches 0 as $\|r_c - r_i\|$ increases and also as t approaches ∞ a widely applied neighborhood function is

$$h_{ci} = \alpha(t)\exp\left(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}\right) \tag{4.2}$$

Where $\alpha(t)$ a scalar is valued learning rate and $\sigma(t)$ defines the width of the kernel. They are generally both monotonically decreasing with time. The use of the neighborhood function means that nodes which are topographically close in the SOM structure activate each other to learn something from the same input x, a relaxation or smoothing effect results which *leads to a global ordering of the map. Note that* $\sigma(t)$ *should not be reduced too far as the map* will lose its topographical order if neighboring nodes are not updated along with the closest node. The SOM can be considered a non-linear projection of the probability density p(x).

4.5.2 Improving the Basic SOM

The original self-organizing map is computationally expensive due to.

1 In the early stages of learning, many nodes are adjusted in a correlated manner. Luttrel [2] proposed a method which we use that starts by learning in a small network, and doubles the size of the network periodically during training. When doubling, new nodes are inserted between the current nodes. The weights of the

new nodes are set equal to the average of the weights of the immediately neighboring nodes.

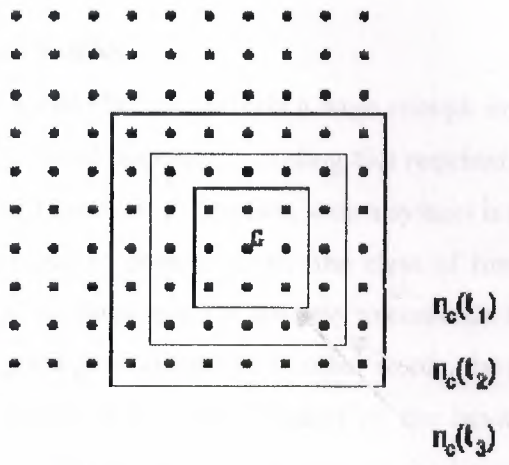

Figure 4.4 A two-dimensional SOM showing, a square neighborhood function which starts as $n_c(t_1)$ and reduces in size to $n_c(t_3)$ over time.

2 Each learning pass requires computation of the distance of the current sample to all nodes in the network, which is O(N).However, this may be reduced to O(logN) using a hierarchy of networks which is created from the above node doubling strategy.

4.6 Karhunen-Lo`eve Transform

The optimal linear method6 for reducing redundancy in a dataset is the Karhunen-Lo`eve (KL) transform or eigenvector expansion via Principle Components Analysis (PCA) [6] . PCA generates a set of orthogonal axes of projections known as the principal components, or the eigenvectors, of the input data distribution in the order of decreasing variance. The KL transform is a well known statistical method for feature extraction and multivariate data projection and has been used widely in pattern recognition, signal processing, image processing, and data analysis. Points in an n-dimensional input space are projected into an m-dimensional space, m≤ n. The KL transform has been used for comparison with the SOM in the dimensionality reduction of the local image samples. The use of the KL transform here is not the same as in the eigenfaces approach because we operate on small local image samples as opposed to the entire images. The KL technique is

fundamentally different to the SOM method, as it assumes the images are sufficiently described by second order statistics, while the SOM is an attempt to approximate the probability density as shown in Kohonen [1].

4.7 Convolutional Networks

Theoretically, we should be able to train a large enough multi-layer perceptron neural network to perform any required mapping, including that required to perfectly distinguish the classes in face recognition. However, in practice, such a system is unable to form the required features in order to generalize to unseen inputs (the class of functions which can perfectly classify the training data is too large and it is not easy to constrain the solution to the subset of this class which exhibits good generalization). In other words, the problem is ill-posed - there are not enough training points in the space created by the input images in order to allow accurate approximation of class probabilities throughout the input space. Additionally, there is no invariance to translation or local deformation of the images [35]. Convolutional networks (CN) incorporate constraints and achieve some degree of shift and deformation invariance using three ideas: local receptive fields, shared weights, and spatial sub sampling. The use of shared weights also reduces the number of parameters in the system aiding generalization. .

A typical convolutional network for recognizing characters is shown in figure (4.5). The network consists of a set of layers each of which contains one or more planes. Approximately centered and normalized images enter at the input layer. Each unit in a plane receives input from a small neighborhood in the planes of the previous layer. The idea of connecting units to local receptive fields dates back to the 1960s with the perceptron and Hubel and Wiesel's [30] discovery of locally sensitive, orientation-selective neurons in the cat's visual system. The weights forming the receptive field for a plane are forced to be equal at all points in the plane. Each plane can be considered as a feature map which has a fixed feature detector that is convolved with a local window which is scanned over the planes in the previous layer. Multiple planes are usually used in each layer so that multiple features can be detected. These layers are called convolutional layers. Once a feature has been detected, its exact location is less important. Hence, the convolutional layers are typically followed by another layer which does a local averaging and sub sampling operation (e.g. for a sub sampling factor of 2) $y_{ij} = \left(x_{2i,2j} + x_{2i+1,2j} + x_{2i,2j+1} + x_{2i+1,2j+1}\right)/4$ where y_{ij} is the output of a

sub sampling plane at position i, j and x_{ij} is the output of the same plane in the previous layer). The network is trained with the usual back propagation gradient-descent procedure.

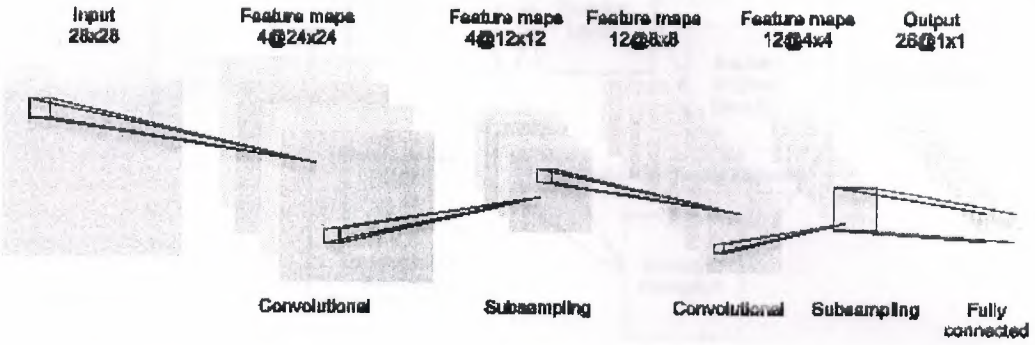

Figure 4.5 A typical convolutional network for recognizing characters.

4.8 System Details

The system we have used for face recognition is a combination of the preceding parts a high-level block diagram is shown in figure (4.6) and figure (4.7) shows a breakdown of the various subsystems.

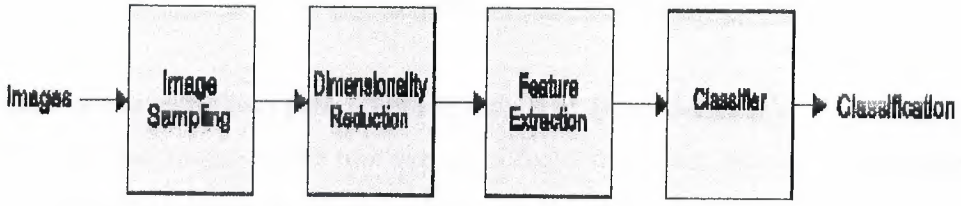

Figure 4.6 A high-level block diagram of the system used for face recognition.

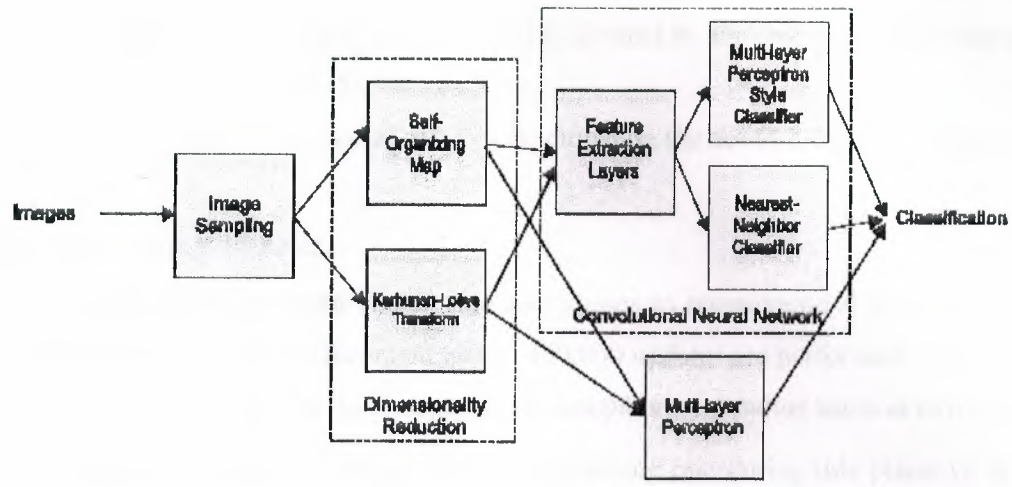

Figure 4.7 A diagram of the system used for face recognition showing alternative methods which had been considered in this chapter. The results are presented with either a self-organizing map or the Karhunen-Lo`eve transform used for dimensionality reduction, and either a convolutional neural network or a multi-layer perceptron for classification. The e possibility of replacing the final classification stage in the convolutional neural network with a nearest neighbor or related classifier had been considered. A complete recognizer consists of only one path through the diagram.

The system works as follows (The complete details of dimensions etc.).

1. For the images in the training set, a fixed size window (e.g. 5x5) is stepped over the entire image as shown in figure (4.3) and local image samples are extracted at each step. At each step the window is moved by 4 pixels.
2. A self-organizing map (e.g. with three dimensions and five nodes per dimension, $5^3 = 125$ total nodes) is trained on the vectors from the previous stage. The SOM quantizes the 25-dimensional input vectors into 125 topologically ordered values. The three dimensions of the SOM can be thought of as three features. We also experimented with replacing the SOM with the Karhunen-Lo`eve transform. In this case, the KL transform projects the vectors in the 25-dimensional space into a 3-dimensional space.
3. The same window as in the first step is stepped over all of the images in the training and test sets. The local image samples are passed through the SOM at each step, thereby creating new training and test sets in the output space created by the self-organizing map. (Each input image is now represented by 3 maps, each of

which corresponds to a dimension in the SOM. The size of these maps is equal to the size of the input image (92x112) divided by the step size (for a step size of 4, the maps are 23x28).

4. A convolutional neural network is trained on the newly created training set.

4.8.1 Simulation Details

For the SOM, training is split into two phases as recommended by Kohonen [33] an ordering phase, and a fine-adjustment phase. 100,000 updates are performed in the first phase, and 50,000 in the second. In the first phase, the neighborhood radius starts at two-thirds of the size of the map and reduces linearly to 1. The learning rate during this phase is: $0.7 \times \left(1 - n/N\right)$ where n is the current update number and N is the total number of updates. In the second phase, the neighborhood radius starts at 2 and is reduced to 1. The learning rate during this phase is $0.02 \times \left(1 - n/N\right)$.

The convolutional network contained five layers excluding the input layer. A confidence measure was calculated for each classification: $y_m(y_m - y_{2m})$ where y_m ym is the maximum output, and y_{2m} is the second maximum output (for outputs which have been transformed using the *softmax* transformation $y_i = \frac{\exp(u_i)}{\sum_{j=1}^{k} \exp(u_j)}$ where u_i are the original outputs, y_i are the transformed outputs, and k is the number of outputs). The number of planes in each layer, the dimensions of the planes, and the dimensions of the receptive fields are shown in table (4.1). The network was trained with back propagation [25] for a total of 20,000 updates. Weights in the network were updated after each pattern presentation, as opposed to batch update where weights are only updated once per pass through the training set. All inputs were normalized to lie in the range minus one to one. All nodes included a bias input which was part of the optimization process. The best of 10 random weight sets was chosen for the initial parameters of the network by evaluating the performance on the training set. Weights were initialized on a node by node basis as uniformly distributed random numbers in the range $\left(-2.4/F_i, 2.4/F_i\right)$ *where* F_i is the fan-in of neuron i. Target outputs were -0.8 and 0.8 using the tanh output activation function.

4.9 Experimental Results

All experiments were performed with 5 training images and 5 test images per person for a total of 200 training images and 200 test images. There was no overlap between the training and test sets. A system which guesses the correct answer would be right one out of forty times, giving an error rate of 97.5%. For the following sets of experiments, vary only one parameter in each case. The error bars shown in the graphs represent plus or minus one standard deviation of the distribution of results from a number of simulations9. The constants used in each set of experiments were: number of classes: 40, dimensionality reduction method: SOM, dimensions in the SOM: 3, number of nodes per SOM dimension: 5, texture extraction: original intensity values, training images per class: 5. The constants in each set of experiments may not give the best possible performance as the current best performing system was only obtained as a result of these experiments. The experiments are as follows:

1. ***Variation of the number of output classes:*** Table (4.2) and figure (4.9) show the error rate of the system as the number of classes is varied from 10 to 20 to 40. No attempt has been made to optimize the system for the smaller numbers of classes. Performance improves with fewer classes to discriminate between (if we continue to add new classes then the chance of a new class being very similar to an existing class increases).
2. ***Variation of the dimensionality of the SOM:*** Table (4.3) and figure (4.10) show the error rate of the system as the dimension of the self-organizing map is varied from 1 to 4. The best performing value is three dimensions.
3. ***Variation of the quantization level of the SOM:*** Table (4.4) and figure (4.11) show the error rate of the system as the size of the self-organizing map is varied from 4 to 8 nodes per dimension. The SOM has three dimensions in each case. The best error rate occurs for 8 nodes per dimension. This is also the best error rate of all experiments.

Table 4.1 Dimensions for the convolutional network. The connection percentage refers to the percentage of nodes in the previous layer which each node in the current layer is connected to - a value less than 100% reduces the total number of weights in the network and may improve generalization. The connection strategy used here is similar to that used by Le Cun et al. for character recognition. As an example of how the precise connections can be determined from the table - the size of the first layer planes (21x26) is equal to the total number of ways of positioning a 3x3 receptive field on the input layer planes (23x28).

Layer	Type	Units	x	y	Receptive field x	Receptive field y	Connection Percentage
1	Convolutional	20	21	26	3	3	100
2	Subsampling	20	9	11	2	2	-
3	Convolutional	25	9	11	3	3	30
4	Subsampling	25	5	6	2	2	-
5	Fully connected	40	1	1	5	6	100

Table 4.2 Error rate of the face recognition system with varying number of classes (subjects) each result is the average of three simulations.

Number of classes	10	20	40
Error rate	1.33%	4.33%	5.75%

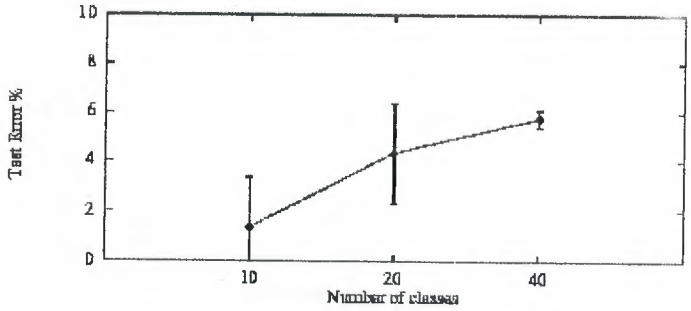

Figure 4.8 The error rate as a function of the number of classes. We did not modify the network from that used for the 40 class case.

Table 4.3 Error rate of the face recognition system with varying number of dimensions in the self-organizing map each result given is the average of three simulations.

SOM Dimension	1	2	3	4
Error rate	8.25%	6.75%	5.75%	5.83%

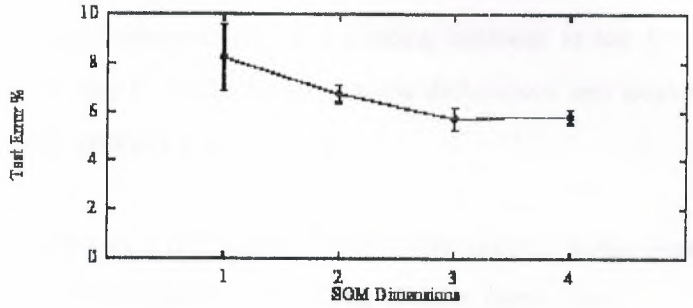

Figure 4.9 The error rate as a function of the number of dimensions in the SOM.

Table 4.4 Error rate of the face recognition system with varying number of nodes per dimension in the self-organizing map each result given is the average of three simulations.

SOM Size	4	5	6	7	8
Error rate	8.5%	5.75%	6.0%	5.75%	3.83%

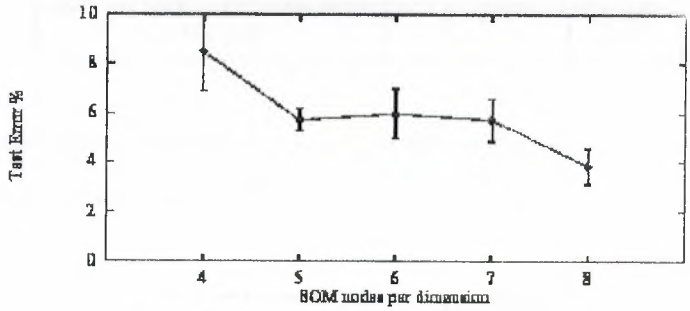

Figure 4.10 The error rate as a function of the number of nodes per dimension in the SOM.

1. ***Variation of the texture extraction algorithm:*** Table (4.5) shows the result of using the two local image sample representations described earlier. We found that using the original intensity values gave the best performance. We tried altering the weight

assigned to the central intensity value in the alternative representation but were unable to improve the results.

2. ***Substituting the SOM with the KL transform***: Table (4.6) shows the results of replacing the self-organizing map with the Karhunen-Lo`eve transform. We tried using the first one, two, or three eigenvectors for projection. Surprisingly, the system performed best with only 1 eigenvector. The best SOM parameters we tried produced slightly better performance. The quantization inherent in the SOM could provide a degree of invariance to minor image sample differences and quantization of the PCA projections may improve performance.

Table 4.5 Error rate of the face recognition system with varying image sample representation, each result is the average of three simulations.

Input type	Pixel intensities	Differences w base intensity
Error rate	5.75%	7.17%

Table 4.6 Error rate of the face recognition system with linear PCA and SOM feature extraction mechanisms. Each result is the average of three simulations.

Dimensionality reduction	Linear PCA	SOM
Error rate	5.33%	3.83%

Table 4.7 Error rate comparison of the various feature extraction and classification methods. Each result is the average of three simulations.

	Linear PCA	SOM
MLP	41.2%	39.83%
CN	5.33%	3.83%

1. ***Replacing the CN with an MLP:*** Table (4.7) shows the results of replacing the convolutional network with a multi-layer perceptron. Performance is very poor, as we expect due to the loss of shift and deformation invariance. We tried a number

of different hidden layer sizes for the multi-layer perceptron in the range 20 to 100. Note that the best performing KL parameters were used while the best performing SOM parameters were not.

2. ***The tradeoff between rejection threshold and recognition accuracy***: Figure 4(.11) shows a histogram of the recognizer's confidence for the cases when the classifier is correct and when it is wrong for one of the best performing systems. From this graph we expect that classification performance will increase significantly if we reject cases below a certain confidence threshold. Figure (4.12) shows the system performance as the rejection threshold is increased. We can see that by rejecting examples with low confidence we can significantly increase the classification performance of the system. If we consider a system which used a video camera to take a number of pictures over a short period, we could expect that a high performance would be attainable with an appropriate rejection threshold.

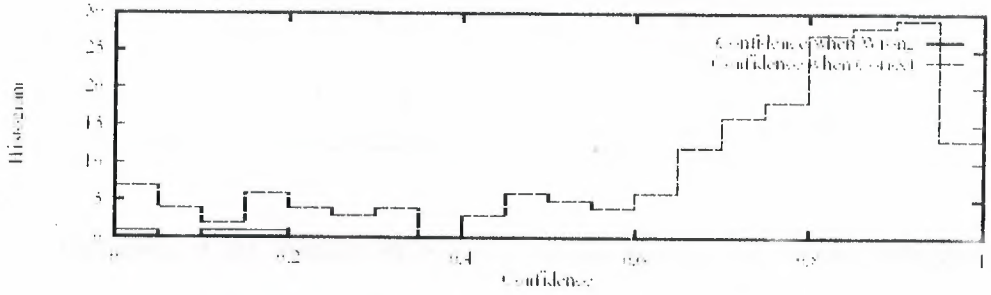

Figure 4.11 A histogram depicting the confidence of the classifier when it turns out to be correct, and the confidence when it is wrong. The graph suggests that we can improve classification performance considerably by rejecting cases where the classifier has a low confidence.

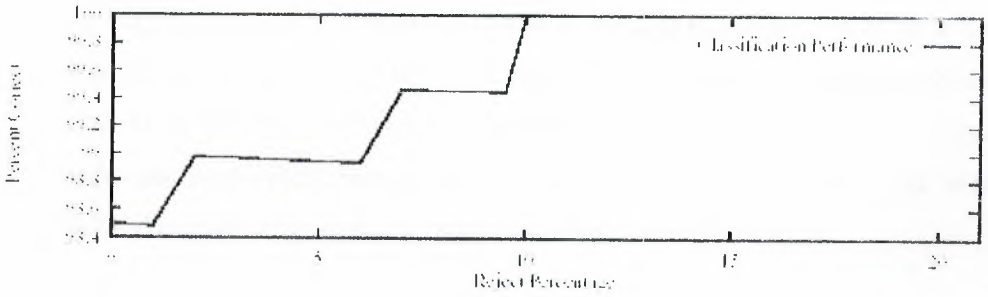

Figure 4.12 The test set classification performance as a function of the percentage of samples rejected. Classification performance can be improved significantly by rejecting cases with low confidence.

1. ***Comparison with other known results on the same database:*** Table (4.8) shows a summary of the performance of the systems for which we have results using the ORL database. In this case, we used a SOM quantization level of 8. Our system is the best performing system10 and performs recognition roughly 500 times faster than the second best performing system the pseudo 2D-HMMs of Samaria. Figure 3.13 shows the images which were incorrectly classified for one of the best performing systems.

Table 4.8 Error rate of the various systems. On a Sun Sparc II. On an SGI Indy MIPS R4600 100 MHz system.

System	Error rate	Classification time
Top-down HMM	13%	n.a
Eigenfaces	10.5%	n.a
Pseudo 2D-HMM	5%	240 seconds[1]
SOM+CN	3.8%	<0.5 seconds[2]

2. ***Variation of the number of training images per person:*** Table (4.9) shows the results of varying the number of images per class used in the training set from 1 to 5 for PCA+CN, SOM+CN and also for the eigenfaces algorithm. Two versions of the eigenfaces algorithm are implemented - the first version creates vectors for each class in the training set by averaging the results of the eigenface representation over all images for the same person. This corresponds to the algorithm as described by Turk and Pentland [42]. However, that using separate training vectors for each training image resulted in better performance. It has been found that using between 40 to 100 eigenfaces resulted in similar performance. The PCA+CN and SOM+CN methods are both superior to the eigenfaces technique even when there is only one training image per person. The SOM+CN method consistently performs better than the PCA+CN method.

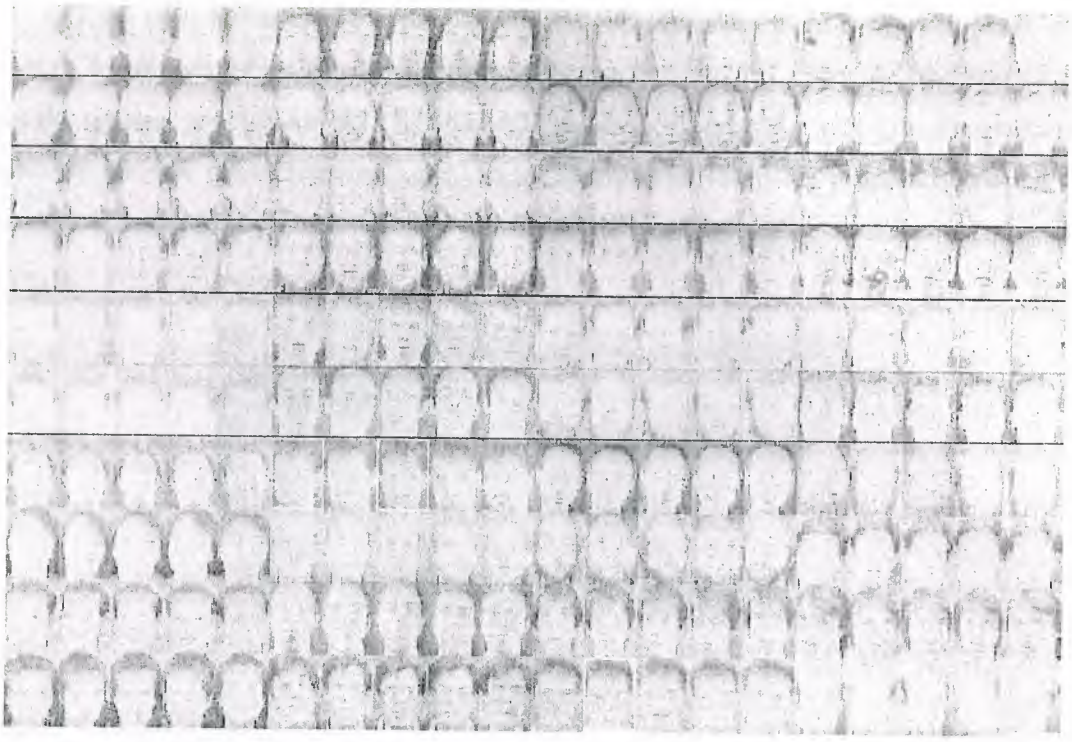

Figure 4.13 Test images. The images with a thick white border were incorrectly classified by one of the best performing systems.

Table 4.9 Error rate for the eigenfaces algorithm and the SOM+CN as the size of the training set is varied from 1 to 5 images per person. Averaged over two different selections of the training and test sets.

Images per person	1	2	3	4	5
Eigenfaces - average per class	38.6	28.8	28.9	27.1	26
Eigenfaces - one per image	38.6	20.9	18.2	15.4	10.5
PCA+CN	34.2	17.2	13.2	12.1	7.5
SOM+CN	30.0	17.0	11.8	7.1	3.5

Figure (4.14) shows the randomly chosen initial local image samples corresponding to each node in a two dimensional SOM, and the final samples which the SOM converges to. Scanning across the rows and columns we can see that the quantized samples represent smoothly changing shading patterns. This is the initial representation from which successively higher level features are extracted using the convolutional network. Figure (4.15) shows the activation of the nodes in a sample convolutional network for a particular test image.

Using both fixed feature extraction (the representation of local image samples), and a trainable feature extractor (the convolutional network). Can this trainable feature extractor form the optimal set of features? The answer is negative - it is unlikely that the network could extract an optimal set of features for all images. Although the exact process of human face recognition is unknown, there are many features which humans

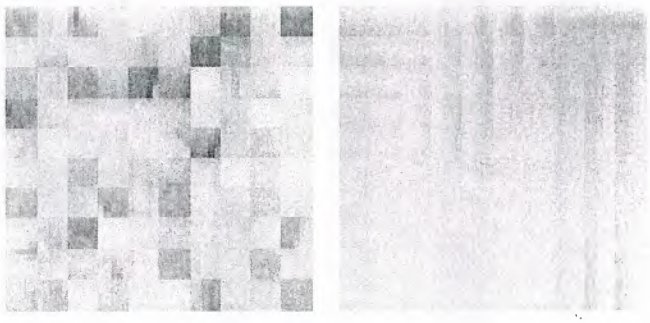

Figure 4.14 SOM image samples before training (a random set of image samples) and after training.

may use but our system is unlikely to discover optimally e.g. a) knowledge of the three-dimensional structure of the face, b) knowledge of the nose, eyes, mouth, etc., c) generalization to glasses/no glasses, different hair growth, etc., and d) knowledge of facial expressions.

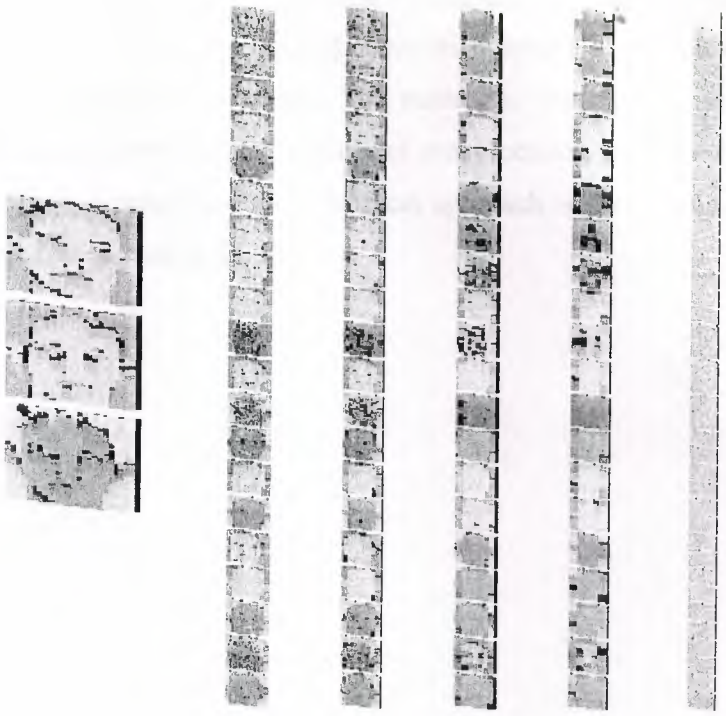

Figure 4.15 A depiction of the node maps in a sample convolutional network showing the activation values for a particular test image. In this case the image is correctly classified with only one activated output node (the top node). From left to right, the layers are: the input layer, convolutional layer 1, and sub sampling layer 1, convolutional layer 2, sub sampling layer 2, and the output layer.

4.10 Summary

This chapter details about the face recognition using the hybrid neural network technique. First we give an introduction, detail information to the related work. A fast, automatic system for face recognition in presented which is a combination of a local image representation, a self organizing map network, and a convolutional network. This provides very successful results. After substitution of the Karhunen Lo`eve transform for the self-organizing map produced similar but slightly worse results. This also produces batter classification and performance than the eigenfaces approach

A fast, automatic system for face recognition is presented which is a combination of a local image sample representation, a self-organizing map network, and a convolutional network. The self-organizing map provides a quantization of the image samples into a topological space where inputs that are nearby in the original space are also nearby in the output space, which results in invariance to minor changes in the image samples, and the

convolutional neural network provides for partial invariance to translation, rotation, scale, and deformation. Substitution of the Karhunen-Lo`eve transform for the self-organizing map produced similar but slightly worse results. The method is capable of rapid classification, requires only fast, approximate normalization and preprocessing, and consistently exhibits better classification performance than the eigenfaces approach on the database considered as the number of images per person in the training.

CONCLUSION

Face recognition is one of the several approaches for recognizing people. There are several methods that can be used for that purpose. Some of the most common are using Local features or Eigenfaces. Though there are other new techniques simpler to understand use and implement but also with very good performance.

Face recognition technology has come a long way in the last twenty years. Today, machines are able to automatically verify identify information for secure transactions, for surveillance and security tasks, and for access control to buildings. These applications usually work in controlled environments and recognition algorithms that can take advantage of the environmental constraints to obtain high recognition accuracy. However, next generation face recognition systems are going to have wide spread applications in smart environments, where computers and machines are more like helpful assistants. A major factor of that evolution is the use of neural networks in face recognition. A different field of science that also is very fast becoming more and more efficient, popular and helpful to other applications.

The combination of these two fields of science manage to achieve the goal of computers to be able to reliably identify nearby people in a manner that fits naturally within the pattern of normal human interactions. They must not require special interactions and must conform to human intuitions about when recognition is likely. This implies that future smart environments should use the same modalities as humans, and have approximately the same limitations. These goals now appear in reach however, substantial research remains to be done in making person recognition technology work reliably, in widely varying conditions using information from single or multiple modalities.

The importance of face recognition is shown with many applications in which the face recognition is approached, using eigenfaces and local feature analysis I described the work for an automatic system detection, recognition and classification. Also I described how we can perform face recognition by neural network approach which involves covolutional network and related work and also the system components and system details.

References

[1] T. Kohonen, *Self-organization and Associative Memory*, Springer-Verlag, Berlin, 1989. Turk. M., and Pentland. A., (1991) Eigenfaces for Recognition, Journal of Cognitive Neuroscience, Vol. 3,No. 1, pp. 71-86.

[2] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America*, March 1987

[3] R. Chellappa, C. Wilson, and S. Sirohev, ``Human and machine recognition of faces: A survey," in *Proceedings of IEEE*, May 1995, vol. 83, pp. 705-740.

[4] P. Phillips, H. Wechsler, J. Huang, and P. Rauss, ``The FERET database and evaluation procedure for face recognition algorithms," *Image and Vision Computing*, vol. 16, no. 5, pp. 295-306, 1998.

[5] L. Wiskott, J-M. Fellous, N. Kruger, and C. von der Malsburg, ``Face recognition by elastic bunch graph matching," *IEEE Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775-779, 1997.

[6] K. Etemad and R. Chellappa, ``Discriminant analysis for recognition of human face images," *Journal of the Optical Society of America*, vol. 14, pp. 1724-1733, 1997.

[7] B. Moghaddam and A. Pentland, ``Probabalistic visual recognition for object recognition," *IEEE Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 696 710, 1997.

[8] J. Atick and Penev P. ``Local feature analysis: A general statistical theory for object representation," *Network: Computation in Neural Systems*, vol. 7, pp. 477-500, 1996. [9] http://www.dacs.dtic.mil/techs/neural/neural_ToC.html

[10] http://www.dacs.dtic.mil/techs/neural6.html

[11] http://www.pnl.gov/

[12] http://www.dacs.dtic.mil/techs/neural6.html

[13] http://www.singnet.com.sg/~midaz Hutchison & Stephens, 1987

[14] ftp://fip.cs.cmu.edu/user/ai/pubs/news/comp.ai.neural-nets

[15] Et. Al. Pentland, (1991) Experiments with Eigenfaces, MIT VISMOD TR-194.

[16] A. Pentland and Moghaddam. B, (1994) Face Recognition using View-Based and Modular Eigenspaces,Automatic Systems for the Identification and Inspection of Humans, SPIE Vol. 2277.

[17] U.S Patent 5,163,094, Technology Recognition Systems.

[18] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[19] S.E. Palmer, "Hierachical structure in perceptual representation", *Cogn. Psychol.* 9, 441-474, 1977.

[20] N.K. Logothetis; D.L. Sheinberg, "Visual object recognition", *Annu. Rev. Neurosci.* 19, 577-621, 1996.

[21] E. Wachsmuth; M.W. Oram; D.I. Perrett, "Recognition of objects and their component parts: responses of single units in the temporal cortex of the macaque", *Cereb. Cortex* 4, 509-522, 1994.

[22] S.E. Palmer, "Hierachical structure in perceptual representation", *Cogn. Psychol.* 9, 441-474, 1977.

[23] A.J. Colmenarez and T.S. Huang, "Face detection with information-based maximum discrimination", *IEEE CVPR,* 782-787, 1997.

[24] H. Schneiderman, "A statistical approach to 3D object detection applied to faces and cars", Ph.D. thesis, CMURI- TR-00-06, May 2000.

[25] M.C. Burl and P. Perona, "recognition of planar object classes." pp.223-230, CVPR'96.

[26] H. Rowley; S. Baluja; and T. Kanade, "Neural Network- based face detection", *IEEE PAMI.,* 20(1), 23-38, 1998.

[27] P. Viola and M.J. Jones, "Robust real-time object detection", *Technical Report Series, Compaq Cambridge Research Laboratory, CRL* 2001/01, Feb. 2001.

[28] C.P. Papageorgiou; M. Oren and T. Poggio, "A general framework for object detection", ICCV '98.

[29] P. Penev and J. Atick, "Local feature analysis: A general statistical theory for object representation", *Neural Systems,* vol.7, no.3, 477-500, 1996.

[30] P. Comon, Hubel and Wiesel's "Independent component analysis – a new concept?", *Signal Processing*, vol.36, pp.287-314, 1994.

[31] C. Jutten and J. Herault, "Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture", *Signal Processing,* vol.24, pp.1-10, 1991.

[32] A.J. Bell and T.J. Sejnowski, "The 'independent components' of natural scenes are edge filters", *Vision Research,* vol.37, pp.3327-3338, 1997.

[33] D. Lee; H.S. Seung, "Learning the parts of objects by non-negative matrix factorization", *Nature* V.401, 21, 788-791, Oct. 1999.

[34] S.Z. Li, X.W. Hou and H.J. Zhang, "Learning Spatially Localized, Parts-Based Representation", *IEEE CVPR*, 2001.

[35] Y. Freund and R.E. Schapire. "A decision-theoretic generalization of no line learning and an application to boosting", *Computational-learning theory: Eurocolt'95* pp23-37, 1995.

[36] N.K. Logothetis; D.L. Sheinberg, "Visual object recognition", *Annu. Rev. Neurosci.* 19, 577-621, 1996.

[37] http://www.vasc.ri.cmu.edu/idb/html/face/frontal_images/index.html, MIT+CMU frontal face database.

[38] P. Phillips, H. Wechsler, J. Huang, and P. Rauss, "The FERET database and evaluation procedure for face recognition algorithms," *IVC*, vol. 16, no. 5, pp. 295-306, Mar.1998.

[39] D. DeCarlo, D. Metaxas, and M. Stone, "An anthropometric face model using variation techniques," *SIGGRAPH Conf. Proc.*, pp. 67-74, Jul. 1998.

[40] F.I. Parke and K. Waters, "Appendix 1: Three-dimensional muscle model facial animation," *Computer Facial Animation*,A.K. Peters, Sept. 1996.

[41] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," *SIGGRAPH Conf. Prof.*, pp. 187-194, 1999.

[42] Q. Chen and G. Medioni, "Building human face models from two images," *IEEE 2ndWorkshop Multimedia Signal Processing*, pp. 117-122, Dec. 1998.

[43] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin, "Making faces," *SIGGRAPH Conf.*, pp. 55-66, Jul. 1998.

[44] M.J.T. Reinders, P.J.L. van Beek, B. Sankur, and J.C.A. vander Lubbe, "Facial feature localization and adaptation of ageneric face model for model-based coding," *Signal Processing: Image Comm.*, vol. 7, no. 1, pp. 57-74, Mar. 1995.

[45] W. Lee and N. Magnenat-Thalmann, "Fast head modeling for animation," *Image and Vision Computing (IVC)*, pp. 355-364, vol. 18, no. 4, Mar. 2000.

[46] R. Lengagne, P. Fua, and O. Monga, "3D stereo reconstruction of human faces driven by differential constraints," *IVC*, vol. 18, no.4, pp. 337-343, Mar. 2000.

[47] P. Fua, "Using model-driven bundle-adjustment to model heads from raw video sequences," *Int'l Conf. Computer Vision*, pp. 46-53, Sept. 1999.

[48] Z. Zhang, "Image-based modeling of objects and human faces," *Proc. SPIE*, vol. 4309, Jan. 2001.

[49] F.I. Parke and K. Waters, "Appendix 1: Three-dimensional muscle model facial animation," *Computer Facial Animation*,A.K. Peters, Sept. 1996.

[50] Range databases: _http://sampl.eng.ohio-state.edu/_sampl/data/3DDB/RID/minolta/faceimages.0300/_

[51] R.-L. Hsu, M. Abdel-Mottaleb, and A.K. Jain, "Face detection in color images," *Tech. Report MSU-CSE-01-7*, Michigan State Univ., Mar. 2001.

[52] A.A. Amini, T.E. Weymouth, and R.C. Jain, "Using dynamic programming for solving variational problems in vision," *IEEE Trans. PAMI*, vol. 12, pp. 855-867, Sept. 1990.

[53] W.-.S.Hwang and J.Weng, "Hierarchical discriminant regression," *IEEE Trans. PAMI*, vol. 22, pp. 1277-1293, Nov. 2000.

[54] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, October 1993.

[55] T. Kanade. *Picture Processing by Computer Complex and Recognition of Human Faces*. PhD thesis, Kyoto University, 1973. [19] Hajime Kita and Yoshikazu Nishikawa. Neural network model of tonotopic map formation based on the temporal theory of pages 413–418, Hillsdale, NJ, 1993. Lawrence Erlbaum.

[56] J. Ingemar Cox, Joumana Ghosn, and Peter N. Yianilos. Feature-based face recognition using mixture-distance. Technical report, NEC Research Institute, Princeton, NJ, October 1995.

[57] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, October 1993.

[58] F. Silvestro Samaria. *Face Recognition using Hidden Markov Models*. PhD thesis, Trinity College, University of Cambridge, Cambridge, 1994.