Visvesvaraya Technological University, Belgaum

Project Work (EC85)
on

Face Recognition Using Gabor Wavelets

Submitted in partial fulfillment of the requirement of VIII semester

Electronics & Communication Engineering by

Vinay Kumar B

USN: 1GA02EC053

Under the guidance of

Dr. Ganesh Murthy C N S

Senior Scientist
DaimlerChrysler
Research and Technology
Bangalore, India

Department of Electronics and Communication Engineering
Global Academy of Technology

2006

ACKNOWLEDGEMENTS

I extend my heartfelt thanks to all the people who directly or indirectly helped me in the completion of my project.

From the beginning to the end, Dr. C.N.S Ganesh Murthy, my guide has shown immense patience and support, besides providing an incredible amount of guidance. Thank you, sir.

I am highly indebted to Prof. Ramachandra Rao, Head of the Department, Electronics & Communication Engineering, Global Academy of Technology and Prof. Col. V G Rajan, Head of the Department, Electrical & Electronics Engineering, Global Academy of Technology for their constant motivation and encouragement.

There have been many more people involved in this venture, whose names may not have been mentioned above, but their help is undoubtedly acknowledged.

The images have been derived from the Yale face database and the Olivetti face database.

ABSTRACT

The choice of the object representation is crucial for an effective performance of cognitive tasks such as object recognition, fixation, etc. Face recognition is an example of advanced object recognition. In our project we demonstrate the use of Gabor wavelets for efficient face representation. Face recognition is influenced by several factors such as shape, reflectance, pose, occlusion and illumination which make it even more difficult. Today there exist many well known techniques to try to recognize a face. We want to introduce the Gabor wavelets for an efficient face recognition system simulating human perception of objects and faces. A face recognition system could greatly aid in the process of searching and classifying a face database and at a higher level help in identification of possible threats to security. The purpose of this study is to demonstrate that it is technically feasible to scan pictures of human faces and compare them with ID photos hosted in a centralized database using Gabor wavelets.

CONTENTS

1. PREAMBLE

1.1. INTRODUCTION:

The modern information age confronts humanity with various challenges that did not exist to the same extent in earlier times. Two such challenges are the organization of society and its security. The ever increasing human population and its mobility in all its facets caused an increasing demand for enhanced ways of transferring data and sharing information. In this context of increased mobility and world population, security and organization have become important social issues. Within this environment of increased importance for security and organization, identification and authentication methods have developed into a key technology in various areas.

Still most of these methods, with all their legitimate applications in an expanding society, have a bothersome drawback. Except for human voice recognition, these methods require the user to remember a password, to enter a PIN code, to carry a badge, or in general, require a human action in the course of identification or authentication. In addition, the above mentioned means (keys, badges, passwords, PIN codes) are prone to being lost or forgotten, while fingerprints and retina scans suffer from low user acceptance.

Thus we need to investigate novel methods of authentication that find mass appeal. This brings us to identification and authentication using FACE RECOGNITION SYSTEMS. Face Recognition systems have the advantage of being non-intrusive. This opens up a new realm of interesting possibilities. Over the last ten years or so, face recognition has become a popular area of research in computer vision and one of the most successful applications of image analysis and understanding. Because of the nature of the problem, not only computer science researchers are interested in it, but neuroscientists and psychologists also. It is the general opinion that advances in computer vision research will provide useful insights to neuroscientists and psychologists into how human brain works, and vice versa.

Face recognition could be employed in intelligent PCs, for instance to automatically bring up a user's individual desktop environment. Even more advanced, when employed in a gesture and facial expression recognition environment, it could turn “personal” computers into “personalized” computers able to interact with the user on a level higher than just mouse clicks and key strokes. Such recognition would make possible intelligent man-machine interfaces and, in the future, intelligent interaction with robots.

With the background of such an abundant field of research, our project shall restrict itself to development of a Face Recognition system that can be used for sorting, searching, classification and possibly even for security purposes, with some remarks to related research topics. We will demonstrate that it is technically feasible to scan pictures of human faces and compare them with ID photos hosted in a centralized database.

The principle aim of this research project is to investigate alternative methods to be used for face recognition, in particular the use of wavelets. Wavelets are mathematical functions that cut up data into different frequency components, and then study each component with a resolution matched to its scale. They have advantages over traditional Fourier methods in analyzing physical situations when the signal contains discontinuities and sharp spikes. Wavelets were developed independently in the fields of mathematics, quantum physics, electrical engineering and seismic geology. Interchanges between these fields during the last ten years have led to many new wavelets. We will use Gabor wavelets to implement our face recognition system.

The use Gabor wavelets for face recognition has several advantages such as invariance to some degree with respect to translation, rotation and dilation. Furthermore, it has the ability to generalize and to abstract from the training data and to assure, for a given network size, that a maximum of object information is coded. Further advantages include - Saves neighborhood relationships between pixels, Robust against illumination when face is correctly normalized, Robust against noise, easy to update, fast recognition and low computational cost. The significance of this research project would be to provide a basis for future development of many applications, such as aids to security, surveillance, digital personal assistant and camera-equipped cell-phones, which require personalized face recognition systems.

1.2. PROBLEM DEFINITION:

Certain factors have to be considered while developing a face recognition system that is not of much importance in other object recognition problems. Illumination, facial hair, makeup, head pose and facial expression have to be kept in mind while developing the face recognition system. This requires an efficient method to represent the face images. The choices that we have for representation are “template based” and “feature based” representations. The problem consists of choosing the best method that gives efficient image representation and in turn good recognition.

1.3. OBJECTIVES OF THE STUDY:

- To develop a method of biometrics that is efficient and has high user acceptance.
- To demonstrate that it is feasible to develop a face recognition system using image processing that satisfies the above mentioned criterion.
- To demonstrate that “feature based” representation of an image is better than “template based” representation.
- To show that Gabor wavelets can be used for feature based representation.
- To demonstrate that face recognition using Gabor wavelets is robust against illumination to a certain extent.
- To demonstrate that our system can be used for both online and offline applications depending upon the computing power available.
- To show that our system can be used for various applications by making suitable minor modifications.
- To demonstrate that our system is easy to update and has low computational cost.

1.4. SCOPE OF THE STUDY:

This project is limited to be a reference model, designed in C language on Cygwin, which provides a complete UNIX environment on Windows. Actual implementation of these modules will be done on a DSP processor, by porting and optimizing the C code written here. Porting on the processor and optimizing the code for the same would be extremely time consuming and completion of that task within the stipulated time would be practically impossible.

Thus, this project is restricted to being a reference model. However, simulations can be carried out on various images using this code. The results of some of these simulations have been shown in a later section of this report.

1.5. REVIEW OF LITERATURE:

Use of biometrics in the field of security and organization of society is fast becoming a norm in this modern information age. In his paper “Face recognition 101: A brief primer”, 2003, D. M. Blackburn gives an introduction to biometrics; highlighting the advantages of using face recognition over other biometric methods. He also gives an insight into a generic biometric system operation. Having established face recognition as the most efficient biometric tool, the different types of face recognition is examined in the paper “A survey of face recognition”, *MML Technical Report No. 97.01*, Department of Computer Science, University of Zurich, 1997 by T. Fromherz, P. Stucki, M. Bichsel. This paper gives an idea about the various types of Face recognition algorithms that exist: Frontal, Profile and View tolerant recognition algorithms. It also gives a brief overview about face recognition, identification and authentication. Different techniques to implement Frontal face recognition are analyzed by Atle Nes in his M.Sc thesis paper “Hybrid systems for face recognition”, Norwegian University of Science and Technology, Department of Computer and Information Science, Division of Intelligent Systems. This paper gives an investigation into individual strengths and weaknesses of the most common techniques used for face recognition.

Use of Gabor wavelets approach to face recognition requires us to be acquainted with the wavelets. Dr. Robi Polikar in his simple and yet complete online tutorial, appropriately named “The wavelet tutorial” gives an in-depth look into the fundamental concepts and an overview of the wavelet theory. It is neatly structured into four parts that give details on “Why wavelet transform ?”, “The Fourier transform, The short term Fourier transform, Resolution problem", "The continuous wavelet transform" and "The discrete wavelet transform" respectively. This tutorial does not assume any prior knowledge about wavelets on the part of the reader. The theory and principles behind wavelets is given by C. Valens in his paper "A really friendly guide to wavelets". This is a wavelet tutorial aimed at engineers rather than mathematicians. It gives a thorough description of wavelet transforms, wavelet properties and discrete wavelets. Paul Addison in his article for the "Physics World" titled "The little wave with the big future" gives a very basic approach to understanding wavelets and their properties. He also lists numerous practical applications of wavelets and wavelet transforms especially in the field of medicine. Amara Graps in her paper "An introduction to wavelets", *IEEE Computational Science and Engineering*, vol. 02(2), pp. 50-61, 1995 gives a thorough analysis of wavelets, Fourier transforms, wavelet transforms and wavelet analysis along with the mathematics behind them. She also digresses on the history of wavelets.

The usage of Gabor jets for image representation is presented in-depth by L. Wiskott, J. M. Fellous, N. Kruger, and C. v. d. Malsburg in their paper "Face recognition by elastic bunch graph matching", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19(7), pp.775-779, July, 1997. Michal Spanel, Dr. Ing, Pavel Zemcik give a very good description about representing images using GWN (Gabor Wavelet Networks) in their research paper "Face representation and tracking using GWN", Computer graphics and Multimedia, Department, Brno University of Technology. Gabor Wavelet Networks as an effective and efficient method for object representation is introduced by V. Kruger, G. Sommer in "Gabor wavelet networks for object representation", Proceedings of the 10th International Workshop on Theoretical Foundations of Computer Vision: Multi-Image Analysis, Lecture Notes in Computer Science, vol. 2032, pp.115-128, 2000. This paper gives an enormous amount of information regarding related work, Gabor wavelet transforms, Wavelet networks, Gabor filters and Face recognition using Gabor wavelet networks. We would sincerely like to thank Dr. Radu Balan from Siemens Research at Princeton for helping us with this particular literature survey (Dr. Volker Krueger's technical report TR4245 that is available for free download from his website www.cfar.umd.edu/~vok/TR4245-krueger.pdf).

In our paper we examine the use of Gabor wavelets for face representation and present a reference model of "Face Recognition Using Gabor Wavelets" that can be used for sorting, searching, classification and possibly even for security purposes.

1.6. LIMITATIONS:

- Not so robust against extreme variations in expression.
- Cannot be used for faces with lateral head rotation.
- Face recognition alone as of now, cannot be successfully used for authentication but can be used for identification and classification.
- The computing power that is in general available makes face recognition suitable for offline applications as online applications require higher computing power.
- Identical twins cannot be distinguished using face recognition.

1.7. METHODOLOGY:

The face recognition system is structured into three sections namely

- The Image Acquisition and Analysis Section
- The Feature Extraction Section
- The Neural Network Classifier

THE IMAGE ACQUISITION AND ANALYSIS SECTION:

The photograph of a person's face is captured such that the head pose is aligned normally. The image is pre-processed using digital image enhancement techniques, if necessary. Since this image is usually in the JPEG (Joint Photographic Expert Group) format, it is converted to PGM (Portable Gray Map) format.

The wavelets are placed in the region of importance, i.e. the inner face region. The image is processed with the Gabor wavelet kernels with an optimal number of frequencies and orientations and a representation is obtained.

THE FEATURE EXTRACTION SECTION:

Features that are of importance are extracted from the complete list of features obtained from the above mentioned section. These features are a vector of values that are automatically saved to a parameters file.

THE NEURAL NETWORK CLASSIFIER:

The features saved into the parameters file are fed to the neural network. The neural network can be configured to be in either “Training/Learning mode” or in “Recognition mode”. Depending upon its mode, the neural network undergoes training or outputs a recognition label.

2. THEORY

2.1. MATHEMATICAL FOUNDATION:

2.1.1. VECTORS:

This wonderful branch of mathematics is both beautiful and useful. It is the cornerstone upon which signal and image processing is built. This short chapter cannot be a comprehensive survey of linear algebra; it is meant only as a brief introduction and review. The ideas and presentation order are modeled after Strang's Linear Algebra and its Applications [1].

At the heart of linear algebra is machinery for solving *linear equations*. In the simplest case, the number of unknowns equals the number of equations. For example, here are two equations with two unknowns:

$$2x - y = 1$$
$$x + y = 5$$

There are at least two ways in which we can think of solving these equations for *x* and *y*. The first is to consider each equation as describing a line, with the solution being at the intersection of the lines: in this case the point (2; 3) in the figure 2.1.

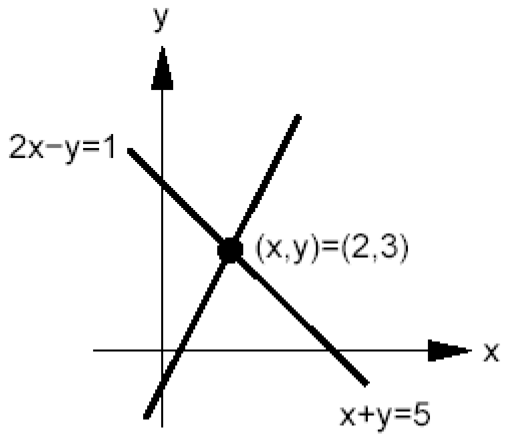


Figure 2.1: "Row" solution

This solution is termed a "row" solution because the equations are considered in isolation of one another. This is in contrast to a "column" solution in which the equations are rewritten in *vector* form:

$$\begin{pmatrix} 2 \\ 1 \end{pmatrix} x + \begin{pmatrix} -1 \\ 1 \end{pmatrix} y = \begin{pmatrix} 1 \\ 5 \end{pmatrix}$$

The solution reduces to finding values for *x*, *y* that *scale* the vectors (2, 1) and (1, 1) so that their *sum* is equal to the vector (1, 5) as shown in figure 2.2. Of course the solution is again $x = 2$ and $y = 3$.

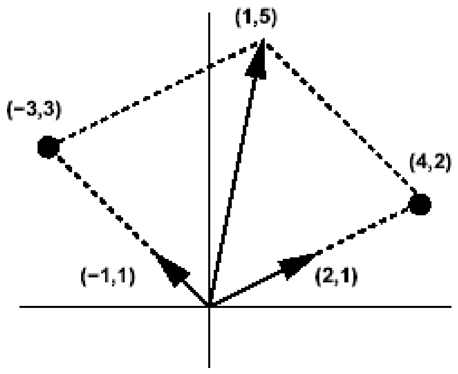


Figure 2.2: "Column" solution

These solutions generalize to higher dimensions. Here is an example with $n = 3$ unknowns and equations:

$$2u + v + w = 5$$
$$4u - 6v + 0w = -2$$
$$-2u + 7v + 2w = 9$$

Each equation now corresponds to a plane, and the row solution corresponds to the intersection of the planes (i.e., the intersection of two planes is a line, and that line intersects the third plane at a point: in this case, the point $u = 1$, $v = 1$, $w = 2$). In vector form, the equations take the form:

$$\begin{pmatrix} 2 \\ 4 \\ -2 \end{pmatrix} u + \begin{pmatrix} 1 \\ -6 \\ 7 \end{pmatrix} v + \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix} w = \begin{pmatrix} 5 \\ -2 \\ 9 \end{pmatrix}$$

The solution again amounts to finding values for *u*, *v*, and *w* that scale the vectors on the left so that their sum is equal to the vector on the right, as shown in the figure 2.3:

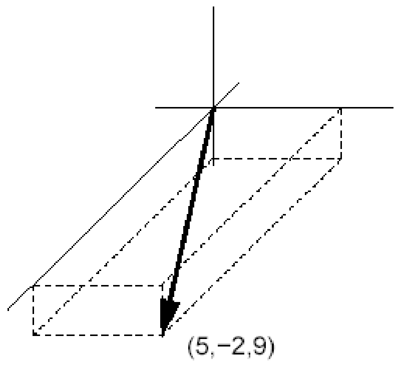


Figure 2.3: "Column" solution

In the context of solving linear equations we have introduced the notion of a *vector*, *scalar multiplication* of a vector, and *vector sum*. In its most general form, a *n*-dimensional *column vector* is represented as:

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix},$$

and a *n*-dimensional *row vector* as:

$$\vec{y} = (y_1 y_2 \ldots\ldots y_n)$$

Scalar multiplication of a vector $\vec{x}$ by a scalar value *c* scales the length of the vector by an amount *c* is given by:

$$c\vec{v} = \begin{pmatrix} cv_1 \\ \vdots \\ cv_n \end{pmatrix}$$

The *vector sum* $\vec{w} = \vec{x} + \vec{y}$ is computed via the parallelogram construction or by "stacking" the vectors head to tail and is computed by a pair wise addition of the individual vector components:

$$\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{pmatrix}$$

The *linear combination* of vectors by vector addition and scalar multiplication is one of the central ideas in linear algebra (more on this later).

2.1.2. MATRICES:

In solving *n* linear equations in *n* unknowns there are three quantities to consider. For example consider again the following set of equations:

$$2u + v + w = 5$$

$$4u - 6v + 0w = -2$$

$$-2u + 7v + 2w = 9$$

On the right is the column vector:

$$\begin{pmatrix} 5 \\ -2 \\ 9 \end{pmatrix}$$

and on the left are the three unknowns that can also be written as a column vector:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

The set of nine coefficients (3 rows, 3 columns) can be written in *matrix* form:

$$\begin{pmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{pmatrix}$$

Matrices, like vectors, can be added and scalar multiplied. Not surprising, since we may think of a vector as a skinny matrix: a matrix with only one column. Consider the following 3 X 3 matrix:

$$A = \begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix}$$

The matrix ***cA***, where *c* is a scalar value, is given by:

$$cA = \begin{pmatrix} ca_1 & ca_2 & ca_3 \\ ca_4 & ca_5 & ca_6 \\ ca_7 & ca_8 & ca_9 \end{pmatrix}$$

And the sum of two matrices, ***A = B + C***, is given by:

$$\begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix} = \begin{pmatrix} b_1+c_1 & b_2+c_2 & b_3+c_3 \\ b_4+c_4 & b_5+c_5 & b_6+c_6 \\ b_7+c_7 & b_8+c_8 & b_9+c_9 \end{pmatrix}$$

With the vector and matrix notation we can rewrite the three equations in the more compact form of $\boldsymbol{A}\vec{x} = \vec{b}$:

$$\begin{pmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{pmatrix}\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} 5 \\ -2 \\ 9 \end{pmatrix}$$

where the multiplication of the matrix $\boldsymbol{A}$ with vector $\vec{x}$ must be such that the three original equations are reproduced. The first component of the product comes from "multiplying" the first row of $\boldsymbol{A}$ (a row vector) with the column vector $\vec{x}$ as follows:

$$\begin{pmatrix} 2 & 1 & 1 \end{pmatrix}\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} 2u + 1v + 1w \end{pmatrix}$$

This quantity is equal to 5, the first component of $\vec{b}$, and is simply the first of the three original equations. The full product is computed by multiplying each row of the matrix $\boldsymbol{A}$ with the vector $\vec{x}$ as follows:

$$\begin{pmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{pmatrix}\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} 2u + 1v + 1w \\ 4u - 6v + 0w \\ -2u + 7v + 2w \end{pmatrix} = \begin{pmatrix} 5 \\ -2 \\ 9 \end{pmatrix}$$

In its most general form the product of a *m* X *n* matrix with a *n* dimensional column vector is a *m* dimensional column vector whose *i*th component is:

$$\sum_{j=1}^{n} a_{ij} x_j$$

where a_{ij} is the matrix component in the *i*th row and *j*th column. The sum along the *i*th row of the matrix is referred to as the *inner product* or *dot product* between the matrix row (itself a vector) and the column vector $\vec{x}$. Inner products are another central idea in linear algebra (more on this later). The computation for multiplying two matrices extends naturally from that of multiplying a matrix and a vector. Consider for example the following 3 X 4 and 4 X 2 matrices:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix} \text{ and } B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \end{pmatrix}$$

The product ***C*** = ***AB*** is a 3 X 2 matrix given by:

$$\begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} + a_{14}b_{42} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} + a_{24}b_{41} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} + a_{34}b_{41} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} + a_{34}b_{42} \end{pmatrix}$$

that is, the *i, j* component of the product ***C*** is computed from an inner product of the i*th* row of matrix ***A*** and the *jth* column of matrix ***B***. Notice that this definition is completely consistent with the product of a matrix and vector. In order to multiply two matrices ***A*** and ***B*** (or a matrix and a vector), the column dimension of ***A*** must equal the row dimension of ***B***. In other words if ***A*** is of size *m* X *n*, then **B** must be of size *n* X *p* (the product is of size *m* X *p*). This constraint immediately suggests that matrix multiplication is not commutative: usually ***AB*** = ***BA***. However matrix multiplication is both associative (***AB***)***C*** = ***A***(***BC***) and distributive ***A***(***B*** + ***C***) = ***AB*** + ***AC***.

The *identity matrix* ***I*** is a special matrix with 1 on the diagonal and zero elsewhere:

$$I = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

Given the definition of matrix multiplication, it is easily seen that for any vector $\vec{x}$, $\boldsymbol{I}\vec{x} = \vec{x}$, and for any suitably sized matrix, ***IA*** = ***A*** and ***BI*** = ***B***. In the context of solving linear equations we have introduced the notion of a *vector* and a *matrix*. The result is a compact notation for representing linear equations, $\boldsymbol{A}\vec{x} = \vec{b}$. Multiplying both sides by the *matrix inverse* A^{-1} yields the desired solution to the linear equations:

$$A^{-1}A\vec{x} = A^{-1}\vec{b}$$

$$I\vec{x} = A^{-1}\vec{b}$$

$$\vec{x} = A^{-1}\vec{b}$$

A matrix ***A*** is invertible if there exists a matrix ***B*** such that ***BA*** = ***I*** and ***AB*** = ***I***, where ***I*** is the identity matrix. The matrix ***B*** is the inverse of ***A*** and is denoted as $\boldsymbol{A}^{-1}$. Note that this commutative property limits the discussion of matrix inverses to square matrices.

Not all matrices have inverses. Let's consider some simple examples. The inverse of a 1 X 1 matrix ***A*** = (a) is $\boldsymbol{A}^{-1}$ = (1/a); but the inverse does not exist when a = 0. The inverse of a 2 X 2 matrix can be calculated as:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

but does not exist when *ad* - *bc* = 0. Any diagonal matrix is invertible:

$$A = \begin{pmatrix} a_1 & & \\ & \ddots & \\ & & a_n \end{pmatrix} \text{ and } A^{-1} = \begin{pmatrix} 1/a_1 & & \\ & \ddots & \\ & & 1/a_n \end{pmatrix}$$

as long as all the diagonal components are non-zero. The inverse of a product of matrices ***AB*** is $(\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$. This is easily proved using the associativity of matrix multiplication. The inverse of an arbitrary matrix, if it exists, can itself be calculated by solving a collection of linear equations. Consider for example a 3 X 3 matrix ***A*** whose inverse we know must satisfy the constraint that $\boldsymbol{AA}^{-1} = \boldsymbol{I}$:

$$\begin{pmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{pmatrix}\begin{pmatrix} \vec{x}_1 & \vec{x}_2 & \vec{x}_3 \end{pmatrix} = \begin{pmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

This matrix equation can be considered "a column at a time" yielding a system of three equations $\boldsymbol{A}\vec{x}_1 = \vec{e}_1$, $\boldsymbol{A}\vec{x}_2 = \vec{e}_2$ and $\boldsymbol{A}\vec{x}_3 = \vec{e}_3$. These can be solved independently for the columns of the inverse matrix, or simultaneously using the *Gauss-Jordan method.*

A system of linear equations $\boldsymbol{A}\vec{x} = \vec{b}$ can be solved by simply left multiplying with the matrix inverse A^{-1} (if it exists). We must naturally wonder the fate of our solution if the matrix is not invertible. The answer to this question is explored in the next section. But before moving forward we need one last definition.

The *transpose* of a matrix $\boldsymbol{A}$, denoted as $\boldsymbol{A}^{\mathbf{t}}$, is constructed by placing the i^{th} row of $\boldsymbol{A}$ into the i^{th} column of $\boldsymbol{A}^{\mathbf{t}}$. For example:

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 4 & -6 & 0 \end{pmatrix} \text{ and } A^t = \begin{pmatrix} 1 & 4 \\ 2 & -6 \\ 1 & 0 \end{pmatrix}$$

In general, the transpose of a m X n matrix is a n X m matrix with $(\boldsymbol{A}^{\mathbf{t}})_{\mathbf{i\,j}} = \boldsymbol{A}_{\mathbf{j\,i}}$. The transpose of a sum of two matrices is the sum of the transposes: $(\boldsymbol{A} + \boldsymbol{B})^{\mathbf{t}} = \boldsymbol{A}^{\mathbf{t}} + \boldsymbol{B}^{\mathbf{t}}$. The transpose of a product of two matrices has the familiar form $(\boldsymbol{AB})^{\mathbf{t}} = \boldsymbol{B}^{\mathbf{t}}\boldsymbol{A}^{t}$. And the transpose of the inverse is the inverse of the transpose: $(\boldsymbol{A}^{-1})^{\mathbf{t}} = (\boldsymbol{A}^{\mathbf{t}})^{-1}$. Of particular interest will be the class of symmetric matrices that are equal to their own transpose $\boldsymbol{A}^{\mathbf{t}} = \boldsymbol{A}$. Symmetric matrices are necessarily square; here is a 3 X 3 symmetric matrix:

$$A = \begin{pmatrix} 2 & 1 & 4 \\ 1 & -6 & 0 \\ 4 & 0 & 3 \end{pmatrix}$$

notice that, by definition, $\boldsymbol{a}_{\mathbf{ij}} = \boldsymbol{a}_{\mathbf{ji}}$.

2.1.3. VECTOR SPACES:

The most common *vector space* is that defined over the reals, denoted as $\boldsymbol{R}^{\mathbf{n}}$. This space consists of all column vectors with n real-valued components, with rules for vector addition and scalar multiplication. A vector space has the property that the addition and multiplication of vectors always produces vectors that lie within the vector space. In addition, a vector space must satisfy the following properties, for any vectors $\vec{x}$, $\vec{y}$, $\vec{z}$ and scalar c:

1. $\vec{x} + \vec{y} = \vec{y} + \vec{x}$
2. $(\vec{x} + \vec{y}) + \vec{z} = \vec{x} + (\vec{y} + \vec{z})$
3. There exists a unique "zero" vector $\vec{0}$ such that $\vec{x} + \vec{0} = \vec{x}$
4. There exists a unique "inverse" vector $-\vec{x}$ such that $\vec{x} + (-\vec{x}) = \vec{0}$
5. $1\vec{x} = \vec{x}$
6. $(c_1 + c_2)\ \vec{x} = c_1(c_2\vec{x})$
7. $c(\vec{x} + \vec{y}) = c\vec{x} + c\vec{y}$
8. $(c_1 + c_2)\ \vec{x} = c_1\vec{x} + c_2\vec{x}$

Vector spaces need not be finite dimensional, R^{∞} is a vector space. Matrices can also make up a vector space. For example the space of 3 X 3 matrices can be thought of as R^9 (imagine stringing out the nine components of the matrix into a column vector).

A *subspace* of a vector space is a non-empty subset of vectors that is closed under vector addition and scalar multiplication. That is, the following constraints are satisfied:

1. The sum of any two vectors in the subspace remains in the subspace
2. Multiplication of any vector by a scalar yields a vector in the subspace.

With the closure property verified, the eight properties of a vector space automatically hold for the subspace.

Vector subspaces play a critical role in understanding systems of linear equations of the form $\boldsymbol{A}\vec{x} = \vec{b}$. Consider for example the following system:

$$\begin{pmatrix} u_1 & v_1 \\ u_2 & v_2 \\ u_3 & v_3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Unlike the earlier system of equations, this system is *over-constrained*; there are more equations (three) than unknowns (two). A solution to this system exists if the vector $\vec{b}$ lies in the subspace of the columns of matrix ***A***. To see why this is so, we rewrite the above system according to the rules of matrix multiplication yielding an equivalent form:

$$x_1 \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} + x_2 \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

In this form, we see that a solution exists when the scaled columns of the matrix sum to the vector $\vec{b}$. This is simply the closure property necessary for a vector subspace.

The vector subspace spanned by the columns of the matrix ***A*** is called the *column space* of ***A***. It is said that a solution to $\boldsymbol{A}\vec{x} = \vec{b}$ exists if and only if the vector $\vec{b}$ lies in the column space of ***A***.

At this point we have seen three seemingly different classes of linear equations of the form $\boldsymbol{A}\vec{x} = \vec{b}$, where the matrix $\boldsymbol{A}$ is either:

1. Square and invertible (*non-singular*)
2. Square but not invertible (*singular*)
3. Over-constrained.

In each case solutions to the system exist if the vector $\vec{b}$ lies in the column space of the matrix $\boldsymbol{A}$. At one extreme is the invertible n X n square matrix whose solutions may be any vector in the whole of $\boldsymbol{R}^{\mathbf{n}}$. At the other extreme is the zero matrix $\boldsymbol{A} = 0$ with only the zero vector in its column space, and hence the only possible solution. In between are the singular and over-constrained cases, where solutions lie in a subspace of the full vector space.

The second important vector space is the *nullspace* of a matrix. The vectors that lie in the nullspace of a matrix consist of all solutions to the system $\boldsymbol{A}\vec{x} = \vec{0}$. The zero vector is always in the nullspace.

2.1.4. BASIS:

Recall that if the matrix $\boldsymbol{A}$ in the system $\boldsymbol{A}\vec{x} = \vec{b}$ is invertible, then left multiplying with $\boldsymbol{A}^{-1}$ yields the desired solution: $\vec{x} = \boldsymbol{A}^{-1}\vec{b}$. In general it is said that a n X n matrix is invertible if it has *rank n* or is *full rank*, where the rank of a matrix is the number of *linearly independent* rows in the matrix. Formally, a set of vectors $\vec{u}_1$; $\vec{u}_2$; …, $\vec{u}_n$ are *linearly independent* if:

$$c_1\vec{u}_1 + c_2\vec{u}_2 + \ldots + c_n\vec{u}_n = \vec{0}$$

is true only when $c_1 = c_2 = \ldots = c_n = 0$. Otherwise, the vectors are *linearly dependent*. In other words, a set of vectors are linearly dependent if at least one of the vectors can be expressed as a sum of scaled copies of the remaining vectors.

Linear independence is easy to visualize in lower-dimensional sub-spaces. In 2-D, two vectors are linearly dependent if they lie along a line, as shown in the figures 2.4.

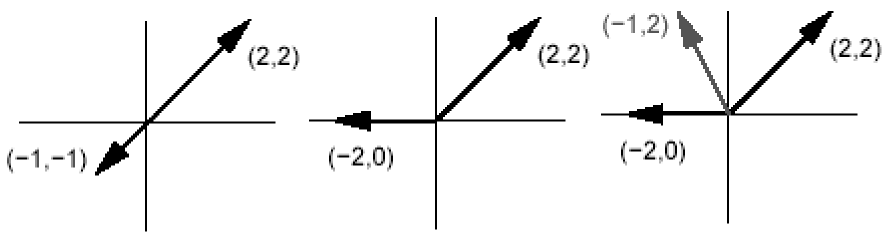


Figure 2.4: The first and third are linearly dependent, second one linearly independent

That is, there is a non-trivial combination of the vectors that yields the zero vector. In 2-D, any three vectors are guaranteed to be linearly dependent. For example, in above figure, the vector (-1, 2) can be expressed as a sum of the remaining linearly independent vectors: $\frac{3}{2}$(-2, 0) + (2, 2).

Linear independence is directly related to the nullspace of a matrix. Specifically, the columns of a matrix are linearly independent (i.e., the matrix is full rank) if the matrix nullspace contains only the zero vector. For example, consider the following system of linear equations:

$$\begin{pmatrix} u_1 & v_1 & w_1 \\ u_2 & v_2 & w_2 \\ u_3 & v_3 & w_3 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Recall that the nullspace contains all vectors $\vec{x}$ such that $\vec{x}_1\vec{u} + \vec{x}_2\vec{v} + \vec{x}_3\vec{w} = 0$. Notice that this is also the condition for linear independence. If the only solution is the zero vector then the vectors are linearly independent and the matrix is full rank and invertible.

Linear independence is also related to the column space of a matrix. If the column space of a *n* X *n* matrix is all of $\boldsymbol{R}^n$, then the matrix is full rank. For example, consider the following system of linear equations:

$$\begin{pmatrix} u_1 & v_1 & w_1 \\ u_2 & v_2 & w_2 \\ u_3 & v_3 & w_3 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

If the matrix is full rank, then the solution can be any vector in $\boldsymbol{R}^3$. In such cases, the vectors $\vec{u}$, $\vec{v}$, $\vec{w}$ are said to *span* the space.

Now, a *linear basis* of a vector space is a set of linearly independent vectors that span the space. Both conditions are important. Given an *n* dimensional vector space with *n* basis vectors $\vec{v}_1 \ldots, \vec{v}_n$, any vector $\vec{u}$ in the space can be written as a linear combination of these *n* vectors:

$$\vec{u} = a_1\vec{v}_1 + \ldots + a_n\vec{v}_n$$

In addition, the linear independence guarantees that this linear combination is unique. If there is another combination such that:

$$\vec{u} = b_1\vec{v}_1 + \ldots + b_n\vec{v}_n$$

then the difference of these two representations yields

$$\begin{aligned}\vec{0} &= (a_1 - b_1)\vec{v}_1 + \ldots + (a_n - b_n) + \vec{v}_n \\ &= c_1\ \vec{v}_1 + \ldots + c_n\ \vec{v}_n\end{aligned}$$

which would violate the linear independence condition. While the representation is unique, the basis is not. A vector space has infinitely many different bases. For example in $\boldsymbol{R}^2$ any two vectors that do not lie on a line form a basis, and in $\boldsymbol{R}^3$ any three vectors that do not lie in a common plane or line form a basis.

2.1.5. INNER PRODUCTS AND PROJECTIONS:

The inner product is defined mathematically as:

$$\begin{aligned}\langle x, y\rangle &= y^T x \\ &= \begin{pmatrix} y_0 & y_1 & \cdots & y_{n-1}\end{pmatrix}\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1}\end{pmatrix} \\ &= \sum_{i=0}^{n-1}(x_i y_i)\end{aligned}$$

INNER PRODUCT IN 2D:

If we have $\mathbf{x} \in \mathbb{R}^{\mathbf{2}}$ and $\mathbf{y} \in \mathbb{R}^{\mathbf{2}}$, then we can write the inner product as:

$$\langle x, y\rangle = \|x\|\|y\|\cos(\theta)$$

where θ is the angle between x and y. Geometrically, the inner product tells us about the strength of x in the direction of y.

The following characteristics are revealed by the inner product:

- $< x, y >$ measures the length of the **projection** of y onto x.
- $< x, y >$ is maximum (for given $\| x \|, \| y \|$) when x and y are in the same direction ($\theta = 0$ implies $\cos(\theta) = 1$).
- $< x, y >$ is zero when $\cos(\theta) = 0$ implies $\theta = 90^o$, i.e. x and y are **orthogonal.**

INNER PRODUCT RULES:

In general, an inner product on a complex vector space is just a function (taking two vectors and returning a complex number) that satisfies certain rules:

- Conjugate Symmetry: $< x, y > = \overline{< x, y >}$
- Scaling: $< \alpha x, y > = \alpha < x, y >$
- Additivity: $< x + y, z > = < x, z > + < y, z >$
- Positivity: $\forall x, x \neq 0 : < x, x > > 0$

DEFINITION OF ORTHOGONALITY:

We say that x and y are orthogonal if:

$$< x, y > = 0$$

PROJECTIONS:

One important use of dot products is in projections. The scalar projection of ***b*** onto ***a*** is the *length* of the segment AB shown in the figure 2.5. The vector projection of ***b*** onto ***a*** is the *vector* with this length that begins at the point A points in the same direction (or opposite direction if the scalar projection is negative) as ***a***.

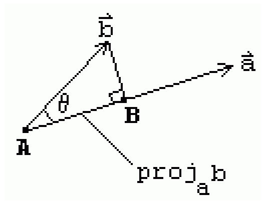


Figure 2.5: Scalar projection of ***b*** onto ***a***

Thus, mathematically, the scalar projection of ***b*** onto ***a*** is |**b**| cos (θ) (where θ is the angle between ***a*** and ***b***) which from is given by

$$\mathrm{comp}_{\vec{a}}\vec{b} = \frac{\vec{a}.\vec{b}}{|\vec{a}|}$$

This quantity is also called the component of ***b*** in the ***a*** direction (hence the notation comp). And, the vector projection is merely the unit vector **a**/|**a**| times the scalar projection of ***b*** onto ***a***:

$$\mathrm{proj}_{\vec{a}}\vec{b} = \frac{\vec{a}.\vec{b}}{|\vec{a}|}\frac{\vec{a}}{|\vec{a}|} = \frac{\vec{a}.\vec{b}}{|\vec{a}|^2}\vec{a}$$

Thus, scalar projection of ***b*** onto ***a*** is the magnitude of the vector projection of ***b*** onto ***a***.

2.2. FUNDAMENTALS OF DIGITAL IMAGE PROCESSING:

Modern digital technology has made it possible to process multi-dimensional signals with systems that range from simple digital circuits to advanced parallel computers. The goal of this processing can be divided into three categories:

- Image Processing: *image in* → *image out*, Example: Gamma Correction.
- Image Analysis: *image in* → *measurements out*, Example: Histogram.
- Image Understanding: *image in* → *high-level description out*, Example: Detecting the number of faces in an image.

We will focus on the fundamental concepts of *image processing*. Further, we will restrict ourselves to two-dimensional (2D) image processing although most of the concepts and techniques that are to be described can be extended easily to three or more dimensions. A digital image $a[m,n]$ described in a 2D discrete space is derived from an analog image $a(x,y)$ in a 2D continuous space through a *sampling* process that is frequently referred to as digitization.

The 2D continuous image $a(x,y)$ is divided into N *rows* and M *columns*. The intersection of a row and a column is termed a *pixel*. The value assigned to the integer coordinates $[m,n]$ with $\{m=0,1,2,...,M-1\}$ and $\{n=0,1,2,...,N-1\}$ is $a[m,n]$. In fact, in most cases $a(x,y)$ which we might consider to be the physical signal that impinges on the face of a 2D sensor, is actually a function of many variables including depth (z), color (λ), and time (t).

The image shown in Figure 2.6 has been divided into N = 16 rows and M = 16 columns. The value assigned to every pixel is the average brightness in the pixel rounded to the nearest integer value. The process of representing the amplitude of the 2D signal at a given coordinate as an integer value with L different gray levels is usually referred to as amplitude quantization or simply *quantization*

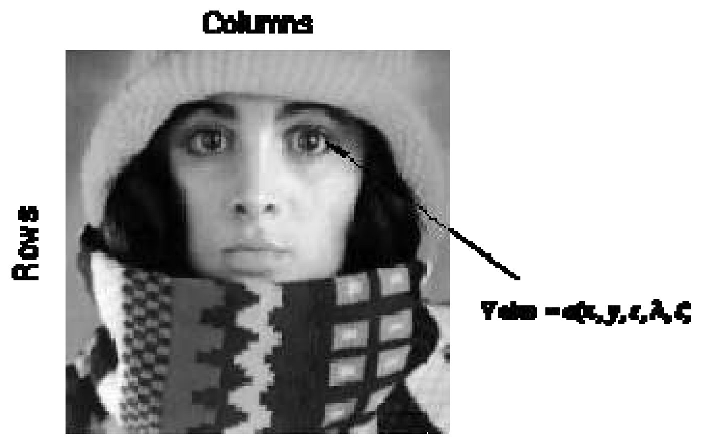


Figure 2.6: Digitization of a continuous image.

There are standard values for the various parameters encountered in digital image processing. These values can be caused by video standards, by algorithmic requirements, or by the desire to keep digital circuitry simple. Table 2.1 gives some commonly encountered values.

Parameter	*Symbol*	*Typical values*
Rows	N	256,512,525,625,1024,1035
Columns	M	256,512,768,1024,1320
Gray Levels	L	2,64,256,1024,4096,16384

Table 2.1: Common values of digital image parameters

Quite frequently we see cases of $M=N=2^K$ where $\{K = 8,9,10\}$. This can be motivated by digital circuitry or by the use of certain algorithms such as the (fast) Fourier transform (see Section 3.3). The number of distinct gray levels is usually a power of 2, that is, $L=2^B$ where B is the number of bits in the binary representation of the brightness levels. When $B>1$ we speak of a *gray-level image*; when $B=1$ we speak of a *binary image*. In a binary image there are just two gray levels which can be referred to, for example, as “black” and “white” or “0” and “1”. In our project we use only PGM (Portable Gray Map) images that have a value of B=8. Thus the gray values range from 0 to 255.

2.3. PATTERN RECOGNITION:

Pattern recognition is a field within the area of machine learning. Alternatively, it can be defined as “the act of taking in raw data and taking an action based on the category of the data”. As such, it is a collection of methods for supervised learning.

Pattern recognition aims to classify data (patterns) based on either a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space.

A complete pattern recognition system consists of a sensor that gathers the observations to be classified or described; a feature extraction mechanism that computes numeric or symbolic information from the observations; and a classification or description scheme that does the actual job of classifying or describing observations, relying on the extracted features.

The classification or description scheme is usually based on the availability of a set of patterns that have already been classified or described. This set of patterns is termed the training set and the resulting learning strategy is characterised as supervised learning. Learning can also be unsupervised, in the sense that the system is not given an a priori labelling of patterns, instead it establishes the classes itself based on the statistical regularities of the patterns.

The classification or description scheme usually uses one of the following approaches: statistical (or decision theoretic), syntactic (or structural). Statistical pattern recognition is based on statistical characterisations of patterns, assuming that the patterns are generated by a probabilistic system. Structural pattern recognition is based on the structural interrelationships of features. A wide range of algorithms can be applied for pattern recognition, from very simple Bayesian classifiers to much more powerful neural networks.

Typical applications are automatic speech recognition, classification of text into several categories (e.g. spam/non-spam email messages), the automatic recognition of handwritten postal codes on postal envelopes, or the automatic recognition of images of human faces. The last two examples form the subtopic image analysis of pattern recognition that deals with digital images as input to pattern recognition systems.

Pattern recognition is more complex when templates are used to generate variants. For example, in English, sentences often follow the "N-VP" (noun - verb phrase) pattern, but some knowledge of the English language is required to detect the pattern. Pattern recognition is studied in many fields, including psychology, ethology, and computer science.

2.4. OBJECT RECOGNITION:

We will give a very brief description on object recognition since face recognition is a very important application of object recognition. Object recognition is detecting the presence of known objects or living beings in an image, possibly together with estimating the alignment of these objects. Every day we recognize a multitude of familiar and novel objects. We do this with little effort, despite the fact that these objects may vary somewhat in form, color, texture, etc. Objects are recognized from many different vantage points (from the front, side, or back), in many different places, and in different sizes. Objects can even be recognized when they are partially obstructed from view.

Example:

- Searching in digital images for specific content (content-based image retrieval).

- Recognizing human faces and their location in images.
- Estimation of the three-dimensional pose of humans and their limbs.
- Detection of objects which are passing through a manufacturing process, e.g., on a conveyor belt, and estimation of their pose so that a robot arm can pick up the objects from the belt.

Recognition-by-components (RBC) [2] is a theory of object recognition in humans that accounts for the successful identification of objects despite changes in the size or orientation of the image. Moreover, RBC explains how moderately occluded or degraded images, as well as novel examples of objects, are successfully recognized by the visual system. RBC was developed to account for primal recognition of objects; primal recognition is fast-acting and does not utilize higher-level cognitive processes. Higher-level processing may involve the use of shading, texture, or color in finer discriminations of objects. Additional top-down processing may also occur when environmental cues such as context are used to identify particularly difficult instances of objects (e.g., a pencil would be easier to recognize if it was partially occluded by a stack of papers on a desk than a pile of leaves in the yard).

The *representation* of the object model is extremely important. Clearly, it is impossible to keep a database that has examples of every view of an object under every possible lighting condition. The next section concentrates on face recognition: image representation and related work in face recognition.

2.5. AN INTRODUCTION TO FACE RECOGNITION:

Face recognition is an example of advanced object recognition. The process is influenced by several factors such as shape, reflectance, pose, occlusion and illumination. A human face is an extremely complex object with features that can vary over time, sometimes very rapidly. It is covered with skin, a non-uniformly textured material that is difficult to model. Skin can change colour quickly when one is embarrassed or becomes warm or cold and the reflectance properties of the skin change as the perspiration level changes. The face is a highly deformable object, and facial expressions come in a wide variety of possible configurations.

Time-varying changes include growth and removal of facial hair, wrinkles and sagging of the skin caused by aging and change in skin colour because of exposure to sunlight. Artifact related changes include cuts, scrapes and bandages from injuries and fashion-related issues like makeup, jewellery and piercing. It should be quite clear that the human face is much more difficult to model and recognize than most industrial parts. This hard challenge is one of the reasons why computer vision research community has been devoted to face recognition for quite some time.

Access control by face recognition has the following advantages in comparison with other biometric systems. There are no requirements for expensive or specialized equipment. A system can be built using a simple video camera and a personal computer. Another advantage is that it is a passive system. There is no need for individuals to touch something by fingers or palm, no need to say any word or lean eye to a detector. Any person can just walk or stay before the camera and the system performs recognition. It is especially useful in everyday usage. Also it has advantages in different extreme and non-standard situations, for example when catching criminals.

2.5.1. IMAGE REPRESENTATION:

It is a crucial question how object information, or image information in general, should be represented for cognitive systems to perform effectively and efficiently. A good representation is a hallmark for robust and successful performance and the choice of representation has far ranging consequences for the entire system that relies upon it.

The reasons for this are manifold:

1. The representation implies the distance and similarity measurements. This is important for, e.g., recognition tasks.

2. When dealing with digital images, the representation that encodes the image information usually results in data reduction, and it is again the type of information representation that determines which image information is relevant, i.e. which is encoded, and which is not.

3. Other important properties are invariance properties with respect to perceived object sizes, geometric deformations, and especially illumination (color constancy [3], [4]).

4. The abstraction capability of the representation has to be mentioned. On one hand, the representation can take information literally, i.e. it can be data-driven or appearance based. This may be useful in some situations; in other situations it should be avoided: well known are the amusing translations that derive from modern language translation programs. On the other hand, the representation can be abstract, i.e. model-driven.

5. Also, the representation determines whether geometric information is represented or discarded. How important geometric information is, was demonstrated e.g. by S. Zeki [5], who reports absurdities that happened to humans whose brains lost their ability to represent geometric information.

6. A further aspect is the efficiency of the representation. Low reaction time is of vital importance to many cognitive systems. However, the reaction speed depends on the data that needs to be evaluated and on the number of filters that need to be applied. The possibility of controlling computation speed by controlling the complexity of the data representation should be of great use for the construction of active vision systems.

The above points are only a selection from a variety of points that reveal the role of information representation in cognitive systems. Various image information representations for artificial cognitive systems have been developed. We want to restrict our consideration to 2-D object representations only, and leave out representations other than for objects and representations of higher dimensionality [6]. Therefore, when we use the term “object representation” or “object information”, we always refer to information that is derived from 2-D image data.

Also, it appears that the object representation approaches that have been used in the context of face detection/recognition have been evaluated most thoroughly. Consequently, most of the object representation approaches that we will consider here were used in the context of face recognition.

Generally, two main types of 2-D object representations appear to exist: Feature-based representations and template-based representations [7]. The feature-based approaches [8], [9], [10], [11], [12], [13], [14] describe objects through abstraction:

An object is represented as a selected collection of abstract features. Simple abstract features are e.g. edges, lines, line segments and points. More complex features may be composed from several simple ones. Also local gray-value patterns can be used as features. This type of representation leads to an abstraction from the image pixel values. In most feature-based approaches, the selection of features as well as their description is given a priori through heuristics.

On the other hand, template-based representations [15], [16], [17], [18], [19], [20], [21] are completely data-driven. The template-based representation uses in its simplest version a gray-value template of the object. But sophisticated variations like PCA-related approaches [22], [23] also exist. In contrast to feature-based representations, a template-based representation is a holistic representation, where the object is treated as a whole. Prior knowledge is needed here only for segmentation of the object from the background. A rudimentary segmentation may result in significant instabilities with respect to background variations.

The major known drawbacks of template-based representations relative to abstract representations are the following:

- Geometrical deformations of abstract object information are relatively easy to handle, but the problem of aligning template and image into a common coordinate system appears to be a major problem for template-based approaches.
- Abstract representations are mostly robust with respect geometric deformations or illumination, contrast and background changes, but these changes lead to great instabilities in template-based approaches.

- On the other hand, the feature-based approach can adapt the number of features used to the needs of the problem, but the holistic approach prohibits this to a certain degree.

To summarize, abstraction from pixel gray-values introduces valuable robustness with respect to illumination variations, etc., but valuable image information is lost. On the other hand, relying on the pixel information only while preserving the image data leads to serious instabilities.

2.5.2. RELATED WORK:

This section will give in-depth information about and evaluation of the most important computer vision face recognition methods that exist today. The focus here is on trying to find advantages and disadvantages with the different approaches. First we start by looking at some of the earliest approaches using simple feature based methods. Then we take a look at some more sophisticated statistical and holistic methods like PCA (eigenfaces), LDA (fisherfaces) and ICA. We will also look at Hidden Markhov Models (HMM).

The earliest approaches to face recognition were focused on detecting individual features such as eyes, ears, head outline and mouth, and measuring different properties such as size, distance and angles between features. This data was used to build models of faces and made it possible to distinguish between different identities. This kind of system was proposed by Kanade [24] and was one of the first approaches to automated face recognition. Later work by Yuille, Cohen and Hallinan describes a method for feature extraction using deformable templates [25].

I) GEOMETRIC METHODS AND TEMPLATE MATCHING:

Brunelli and Poggio developed two simple algorithms for face recognition [7]. The first one is based on the computation of a set of geometrical features, such as nose width and length, mouth position and chin shape. One motivation for using geometric methods is that in an image with sufficiently low resolution it is impossible to distinguish the fine details of a face, but often possible for a human to recognize the person. The remaining information in the low resolution image is almost pure geometrical and implies that these properties of face features are sufficient enough for face recognition. The configuration of the features can be described by a vector of numerical data representing the position and size of the main facial features, eyes and eyebrows, nose and mouth. This information can be supplemented by the shape of the face outline.

The other algorithm proposed by Brunelli and Poggio is based on template matching. In the simplest version of template matching the image, represented by an array of intensity values, is compared using a suitable metric (usually Euclidean distance) to a single template representing the whole face. More sophisticated methods can use several templates per face to take into account the recognition from different viewpoints. First the image is normalized using the same technique described in the previous section. Each person is stored in the database associated with template masks representing digital images of eyes, nose, mouth etc. Recognition of an unclassified image is done by comparing parts of it with all the templates stored in the database returning a matching score for each individual. The unknown individual is then classified as the one giving the highest cumulative comparison score.

EVALUATION: The use of feature vectors seems very unstable and limited because the variation of the data from different pictures of the same face was in the same order of magnitude as the variation between different faces. The method is sensitive to inaccurate detection of features and to all sorts of disturbance such as facial expressions or varying pose. Template-based approaches outperform geometrical methods. Templates seem to offer satisfactory results for recognition from frontal views. A more difficult problem is how to deal with non-frontal views. It should be possible to use almost the same scheme for different viewpoints at the expense of considerably greater computational complexity.

II) PRINCIPAL COMPONENT ANALYSIS (PCA):

PCA also known as Karhunen-Loeve (KL) transformation or eigenspace projection, a frequently used statistical technique for optimal lossy compression of data under least square sense, provides an orthogonal basis vector-space to represent original data. The first introduction of a low-dimensional characterization of faces was developed at Brown University by Kirby and Sirovich in 1987 [16]. This was later extended to eigenspace projection for face recognition by Pentland, Turk, Moghaddam and Starner at the Vision and Modeling Group of MIT in 1991 [18], [19].

The eigenspace is a subspace of the image space spanned up by a set of eigenvectors of the covariance matrix of the trained images. These eigenvectors are also called eigenfaces because of their face-like appearance. The projection of an image into eigenspace will transform the image into a representation of a lower dimension which aims to hold the most important features of the face and make the comparison of images easier.

There are two main approaches of recognizing faces by using eigenfaces. In the appearance model each face in the database is represented as a linear combination of eigenfaces. The recognition process is done by projecting a test image to be identified into the same eigenspace. The resulting vector will be a point in eigenspace and comparison with the training images is normally done by using a distance measure between the points in eigenspace. The other recognition scheme is called the discriminative model. Two datasets are obtained by computing intrapersonal differences (matching two different images of the same individual in the dataset) and extra personal differences (matching different individuals in the dataset). Two datasets of eigenfaces are generated by performing PCA on each class and a similarity score between two images is derived by calculating a Bayesian probability measure. Although the recognition performance of the appearance model is lower than the discriminative model, the substantial reduction in computational complexity makes this recognition scheme very attractive.

EVALUATION: Results show that eigenfaces methods are robust over a wide range of parameters and produce good recognition rates on various databases. However outside this parameter range the algorithm can breakdown sharply. Results show that eigenfaces are very robust to low resolution images as long as the preprocessing step can extract sufficient features for normalization. They also handle high resolution images very efficiently. But significant variation in scale, orientation, translation and lightning will cause it to fail.

III) LINEAR DISCRIMINANT ANALYSIS (LDA):

R. A. Fisher developed Fisher’s Linear Discriminant (FLD) in the 1930’s but not until recently have Fisher discriminants been utilized for object recognition. Swets and Weng used

FLD to cluster images for the purpose of identification [26]. Also in 1997, Belhumeur, Hespanha and Kriegman of Yale University used FLD to identify faces, by training and testing with several faces under different lighting [27].

Fisher Linear Discriminant (FLD) analysis, also called Linear Discriminant Analysis (LDA) finds the line that best separates the points. For example, consider two sets of points, coloured green and blue, in two-dimensional space being projected onto a single line. Depending on the direction of the line, the points can either be mixed together or be separated. In terms of face recognition this means grouping images of the same class and separate images of different classes. Images are projected from a N-dimensional space, where N is the number of pixels in the image, to a M-1 dimensional space, where M is the number of classes of images

The LDA method, which creates an optimal projection of the dataset, maximizes the ratio of the determinant of the between-class scatter matrix of the projected samples to the determinant of the within-class scatter matrix of the projected samples. The within-class scatter matrix, also called intra-personal, represents variations in appearance of the same individual due to different lighting and face expression, while the between-class scatter matrix, also called the extra-personal, represents variations in appearance due to a difference in identity. In this way fisherfaces can project away some variation in lighting and facial expression while maintaining discriminability.

EVALUATION: The fisherface method is very similar to the eigenface method, but with improvement in better classification of face images by using interclass and intraclass relationships to separate them. With LDA it is possible to classify the training set to deal with different people and different facial expressions. The accuracy for handling facial expressions has shown to be better than the eigenfaces method. The fisherfaces method is quite insensitive to large variations in lighting direction and facial expression. Compared to the eigenface method this algorithm is more complex, something which increases the computational requirements, but show lower error rates. Besides, due to the need of better classification, the dimension of the projection in face space is not as compact as in the eigenfaces approach. This results in larger storage of the faces and more processing time in recognition. Another drawback comes from the fact that the fisherface method uses particular class information and therefore is recommended to have many images per class in the training process. On the other hand, having many images belonging to the same class can make the recognition system suffer from a lack of generalization resulting in a lower recognition rate. In general the algorithm is performing very well, but cannot always work. In general it fails when the between class scatter is inherently greater than the within class scatter.

IV) INDEPENDENT COMPONENT ANALYSIS (ICA):

Independent Component Analysis (ICA) is a technique for extracting statistically independent variables from a mixture of them. The technique is quite new and has originated from the world of signal processing. A classical example demonstrating the original problem is the cocktail-party problem where two people being in the same room speak simultaneously. Two microphones are placed at different locations recording the mixed conversations. It would be very useful if one could estimate the two original speech signals from the two mixed recordings. Surprisingly it turns out that it is enough to assume that the two speech signals are statistically independent. This is not an unrealistic assumption, but it does not need to be exactly true in practice. ICA can be used to estimate the contribution coefficients from the two signals, which allows us to separate the two original signals from each other. Hyvärinen and Oja have written a good tutorial about ICA which contains more details about the algorithms involved [28]. In a task such as face recognition, much of the important information may be contained in the high-order relationships among the image pixels. Some success has been attained using data-driven face representations based on PCA, such as eigenfaces. PCA is based on the second-order statistics of the image set, and does not address high-order statistical dependencies such as the relationships among three or more pixels. Independent component analysis (ICA) however separates the high-order moments of the input in addition to the second-order moments. ICA thus in some ways provide a more powerful data representation than PCA, as its goal is to provide an independent rather than an uncorrelated image decomposition and representation.

EVALUATION: In 1999 Liu and Wechsler also claimed that ICA produced better results or matched the results that were obtained purely by PCA [29]. This was later in 2001 contradicted by Baek, Draper, Beveridge and She [30]. They showed that PCA outperformed ICA when the distance method is selected to maximize performance. Both experiments were conducted using the FERET database. The most recent contradicting results from 2001 however showed that the differences in recognition rates between PCA and ICA are only minor, and very much depend on how the algorithms in detail are implemented. Global properties like coloring, width and length are more easily captured by PCA than ICA, since ICA basis vectors are more spatially localized than their PCA counterparts. Recognizing more localized features, like face expressions, may produce significantly different results.

V) HIDDEN MARKOV MODELS (HMM):

The use of hidden Markov models is a powerful statistical technique that has been applied to many subject areas, from predicting political crises to the reconstruction of DNA and the recognition of speech. In order to use HMM for recognition, an observation sequence is obtained from the test signal and then the likelihood of each HMM generating this signal is computed. The HMM which has the highest likelihood then identifies the test signal. Finding the state sequence which maximizes the probability of an observation is done using the Viterbi algorithm, which is a simple dynamic programming optimization procedure. More details describing all the technical details concerning the algorithms used can be found in a great tutorial written by Rabiner [31]. One pioneer of using hidden Markov models for face recognition was Samaria at Trinity College [32]. Nefian and Hayes [33] at Georgia Institute of Technology have written several papers on pseudo 2D HMM and some on embedded HMM. Eickeler, Müller and Rigoll [34] have written about how to get high performance using pseudo 2D HMM. Some attempts have also been made by Othman and Aboulnasr [35] on 2D HMM

HMM has been extensively used for speech recognition, where data is naturally one dimensional along the time axis. The equivalent fully-connected two-dimensional HMM would lead to a very high computational cost problem. Samaria has proposed using the 1D continuous HMM for face recognition. For a frontal face the states of the Markov model include hair, forehead, eyes, nose and mouth, each representing a state. These states always occur in the same order, from top to bottom, even if faces undergo small rotations in the image plane. Each facial region will be assigned to a state, in a left-to-right one dimensional hidden Markov model. Only transitions between adjacent states in a top to bottom manner are allowed. An observation sequence is generated from a face image using a sampling window with overlap. The observation sequence is composed of vectors that represent the consecutive horizontal strips, where each vector contains the pixel values from the associated strips. The goal of the training stage is to optimize the hidden Markov model parameters to best describe the observations. This is done by maximizing the probability of the observed sequence given a set of variable parameters. Recognition is done by matching the test image against each of the trained models. To do this the image is converted to an observation sequence and then model likelihoods for all database images are computed. The model with the highest likelihood reveals the identity of the unknown face.

EVALUATION: HMM-based methods have shown better performances compared to the traditional eigenfaces method. Error rates of about 5% were reported when pseudo 2D HMM was used compared to about 10% with eigenfaces on the same dataset. The 1D HMM had an error rate of 13% in the same experiment. The Baum-Welch algorithm, which is used for the training of the HMM for each person, provides the HMM parameters corresponding to a local maximum of the likelihood function depending on the initial model parameters. It is therefore very important to use a good initial model for the training. Also as much training data as possible is needed in the estimation of hidden Markov model parameters, to estimate good models for recognition. Block overlap helps in providing higher statistical resolution. However large overlap results in increasing the computational load and memory requirements for all parts of the system. The time required by the recognition system is critical. It is a function of the size of the database. Recognition time must be less than the time between two consecutive occurrences of people in a scene. Depending on the parameterization used the Viterbi algorithm can require a large number of calculations. This implies that sometimes the algorithm runs slowly.

The related work in the field of Face Recognition has been dealt with in some detail in the above section. The various advantages and disadvantages have been portrayed.

With knowledge of the various face recognition models let us now proceed to understand Face Recognition using Gabor Wavelets. Using wavelets for face recognition overcomes many of the disadvantages that are prevalent in the above mentioned methods. Before going into the implementation details using Gabor wavelets let us understand the concept of wavelets.

2.6. WAVELETS AND WAVELET TRANSFORMS:

2.6.1. WAVELET OVERVIEW:

The fundamental idea behind wavelets is to analyze according to scale. Indeed, some researchers in the wavelet field feel that, by using wavelets, one is adopting a whole new mindset or perspective in processing data.

Wavelets are functions that satisfy certain mathematical requirements and are used in representing data or other functions. This idea is not new. Approximation using superposition of functions has existed since the early 1800's, when Joseph Fourier discovered that he could superpose sines and cosines to represent other functions. However, in wavelet analysis, the scale that we use to look at data plays a special role. Wavelet algorithms process data at different scales or resolutions. If we look at a signal with a large "window", we would notice gross features. Similarly, if we look at a signal with a small "window", we would notice small features. The result in wavelet analysis is to see both the forest and the trees, so to speak.

This makes wavelets interesting and useful. For many decades, scientists have wanted more appropriate functions than the sines and cosines which comprise the bases of Fourier analysis, to approximate choppy signals. By their definition, these functions are non-local (and stretch out to infinity). They therefore do a very poor job in approximating sharp spikes. But with wavelet analysis, we can use approximating functions that are contained neatly in finite domains. Wavelets are well-suited for approximating data with sharp discontinuities.

The wavelet analysis procedure is to adopt a wavelet prototype function, called an analyzing wavelet or mother wavelet. Temporal analysis is performed with a contracted, high-frequency version of the prototype wavelet, while frequency analysis is performed with a dilated, low-frequency version of the same wavelet. Because the original signal or function can be represented in terms of a wavelet expansion (using coefficients in a linear combination of the wavelet functions), data operations can be performed using just the corresponding wavelet coefficients. And if you further choose the best wavelets adapted to your data, or truncate the coefficients below a threshold, your data is sparsely represented. This sparse coding makes wavelets an excellent tool in the field of data compression.

Other applied fields that are making use of wavelets include astronomy, acoustics, nuclear engineering, sub-band coding, signal and image processing, neurophysiology, music, magnetic resonance imaging, speech discrimination, optics, fractals, turbulence, earthquake-prediction, radar, human vision, and pure mathematics applications such as solving partial differential equations.

2.6.2. HISTORICAL PERSPECTIVE:

In the history of mathematics, wavelet analysis shows many different origins. Much of the work was performed in the 1930s, and, at the time, the separate efforts did not appear to be parts of a coherent theory.

PRE-1930:

Before 1930, the main branch of mathematics leading to wavelets began with Joseph Fourier (1807) with his theories of frequency analysis, now often referred to as Fourier synthesis. He asserted that any 2π periodic function f(x) is the sum

$$a_0 + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx)$$

of its Fourier series. The coefficients a_0, a_k and b_k are calculated by

$$a_0 = \frac{1}{2\pi}\int_0^{2\pi} f(x)dx, \quad a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos(kx)dx, \quad b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin(kx)dx$$

Fourier’s assertion played an essential role in the evolution of the ideas mathematicians had about the functions. He opened up the door to a new functional universe.

After 1807, by exploring the meaning of functions, Fourier series convergence, and orthogonal systems, mathematicians gradually were led from their previous notion of frequency analysis to the notion of scale analysis. That is, analyzing f(x) by creating mathematical structures that vary in scale. How? Construct a function, shift it by some amount, and change its scale. Apply that structure in approximating a signal. Now repeat the procedure. Take that basic structure, shift it, and scale it again. Apply it to the same signal to get a new approximation. And so on. It turns out that this sort of scale analysis is less sensitive to noise because it measures the average fluctuations of the signal at different scales.

The first mention of wavelets appeared in an appendix to the thesis of A. Haar (1909). One property of the Haar wavelet is that it has compact support, which means that it vanishes outside of a finite interval. Unfortunately, Haar wavelets are not continuously differentiable which somewhat limits their applications.

THE 1930S:

In the 1930s, several groups working independently researched the representation of functions using scale-varying basis functions. Understanding the concepts of basis functions and scale-varying basis functions is key to understanding wavelets.

What are Basis Functions?

It is simpler to explain a basis function if we move out of the realm of analog (functions) and into the realm of digital (vectors).

Every two-dimensional vector (x, y) is a combination of the vector (1, 0) and (0, 1). These two vectors are the basis vectors for (x, y). Why? Notice that x multiplied by (1, 0) is the vector (x, 0), and y multiplied by (0, 1) is the vector (0, y). The sum is (x, y).

The best basis vectors have the valuable extra property that the vectors are perpendicular, or orthogonal to each other. For the basis (1, 0) and (0, 1), this criteria is satisfied.

Now let’s go back to the analog world, and see how to relate these concepts to basis functions. Instead of the vector (x, y), we have a function f(x). Imagine that f(x) is a musical tone; say the note A in a particular octave. We can construct A by adding sines and cosines using combinations of amplitudes and frequencies. The sines and cosines are the basis functions in this example, and the elements of Fourier synthesis. For the sines and cosines chosen, we can set the additional requirement that they be orthogonal. How? By choosing the appropriate combination of sine and cosine function terms whose inner product add up to zero. The particular set of functions that are orthogonal and that construct f(x) are our orthogonal basis functions for this problem.

What are Scale-varying Basis Functions?

A basis function varies in scale by chopping up the same function or data space using different scale sizes. For example, imagine we have a signal over the domain from 0 to 1. We can divide the signal with two step functions that range from 0 to 1/2 and 1/2 to 1. Then we can divide the original signal again using four step functions from 0 to 1/4, 1/4 to 1/2, 1/2 to 3/4, and 3/4 to 1. And so on. Each set of representations code the original signal with a particular resolution or scale.

By using a scale-varying basis function called the Haar basis function (more on this later) Paul Levy, a 1930s physicist, investigated Brownian motion, a type of random. He found the Haar basis function superior to the Fourier basis functions for studying small complicated details in the Brownian motion.

Another 1930s research effort by Littlewood, Paley, and Stein involved computing the energy of a function f(x):

$$energy = \frac{1}{2}\int_{0}^{2\pi} |f(x)|^2 \, dx$$

The computation produced different results if the energy was concentrated around a few points or distributed over a larger interval. This result disturbed the scientists because it indicated that energy might not be conserved. The researchers discovered a function that can vary in scale and can conserve energy when computing the functional energy. Their work provided David Marr with an effective algorithm for numerical image processing using wavelets in the early 1980s [36].

1960-1980:

Between 1960 and 1980, the mathematicians Guido Weiss and Ronald R. Coifman [37] studied the simplest elements of a function space, called atoms, with the goal of finding the atoms for a common function and finding the “assembly rules” that allow the reconstruction of all the elements of the function space using these atoms. Grossman and Morlet, a physicist and an engineer, broadly defined wavelets in the context of quantum physics [38]. These two researchers provided a way of thinking for wavelets based on physical intuition.

POST-1980:

Stephane Mallat gave wavelets an additional jump-start through his work in digital signal processing [39]. He discovered some relationships between quadrature mirror filters, pyramid algorithms, and orthonormal wavelet bases (more on these later). Inspired in part by these results, Y. Meyer constructed the first non-trivial wavelets. Unlike the Haar wavelets, the Meyer wavelets are continuously differentiable; however they do not have compact support. A couple of years later, Ingrid Daubechies used Mallat’s work to construct a set of wavelet orthonormal basis functions [40] that are perhaps the most elegant, and have become the cornerstone of wavelet applications today.

2.6.3. FOURIER ANALYSIS:

Fourier’s representation of functions as a superposition of sines and cosines has become ubiquitous for both the analytic and numerical solution of differential equations and for the analysis and treatment of communication signals. Fourier and wavelet analysis have some very strong links.

FOURIER TRANSFORMS:

The Fourier transform’s utility lies in its ability to analyze a signal in the time domain for its frequency content. The transform works by first translating a function in the time domain into a function in the frequency domain. The signal can then be analyzed for its frequency content because the Fourier coefficients of the transformed function represent the contribution of each sine and cosine function at each frequency. An inverse Fourier transform does just what you’d expect; transform data from the frequency domain into the time domain.

DISCRETE FOURIER TRANSFORMS:

The discrete Fourier transform (DFT) estimates the Fourier transform of a function from a finite number of its sampled points. The sampled points are supposed to be typical of what the signal looks like at all other times. The DFT has symmetry properties almost exactly the same as the continuous Fourier transform. In addition, the formula for the inverse discrete Fourier transform is easily calculated using the one for the discrete Fourier transform because the two formulas are almost identical.

WINDOWED FOURIER TRANSFORMS:

If f(t) is a non periodic signal, the summation of the periodic functions, sine and cosine, does not accurately represent the signal. You could artificially extend the signal to make it periodic but it would require additional continuity at the endpoints. The windowed Fourier transform (WFT) is one solution to the problem of better representing the non periodic signal. The WFT can be used to give information about signals simultaneously in the time domain and in the frequency domain.

With the WFT, the input signal f(t) is chopped up into sections, and each section is analyzed for its frequency content separately. If the signal has sharp transitions, we window the input data so that the sections converge to zero at the endpoints. This windowing is accomplished via a weight function that places less emphasis near the interval’s endpoints than in the middle. The effect of the window is to localize the signal in time.

FAST FOURIER TRANSFORMS:

To approximate a function by samples, and to approximate the Fourier integral by the discrete Fourier transform, requires applying a matrix whose order is the number sample points n. Since multiplying an n x n matrix by a vector costs on the order of n^2 arithmetic operations, the problem gets quickly worse as the number of sample points increases. However, if the samples are uniformly spaced, then the Fourier matrix can be factored into a product of just a few sparse matrices, and the resulting factors can be applied to a vector in a total of order n log n arithmetic operations. This is the so-called fast Fourier transform or FFT

2.6.4. WAVELET TRANSFORM V/S FOURIER TRANSFORM:

SIMILARITIES BETWEEN FOURIER AND WAVELET TRANSFORM:

The fast Fourier transform (FFT) and the discrete wavelet transform (DWT) are both linear operations that generate a data structure that contains $\log_2 n$ segments of various lengths, usually filling and transforming it into a different data vector of length 2^n.

The mathematical properties of the matrices involved in the transforms are similar as well. The inverse transform matrix for both the FFT and the DWT is the transpose of the original. As a result, both transforms can be viewed as a rotation in function space to a different domain. For the FFT, this new domain contains basis functions that are sines and cosines. For the wavelet transform, this new domain contains more complicated basis functions called wavelets, mother wavelets, or analyzing wavelets.

Both transforms have another similarity. The basis functions are localized in frequency, making mathematical tools such as power spectra (how much power is contained in a frequency interval) and scalegrams useful at picking out frequencies and calculating power distributions.

DISSIMILARITIES BETWEEN FOURIER AND WAVELET TRANSFORM:

The most interesting dissimilarity between these two kinds of transforms is that individual wavelet functions are localized in space. Fourier sine and cosine functions are not. This localization feature, along with wavelets' localization of frequency, makes many functions and operators using wavelets "sparse" when transformed into the wavelet domain. This sparseness, in turn, results in a number of useful applications such as data compression, detecting features in images, and removing noise from time series.

One way to see the time-frequency resolution differences between the Fourier transform and the wavelet transform is to look at the basis function coverage of the time-frequency plane. Figure 2.7 shows a windowed Fourier transform, where the window is simply a square wave. The square wave window truncates the sine or cosine function to fit a window of a particular width. Because a single window is used for all frequencies in the WFT, the resolution of the analysis is the same at all locations in the time-frequency plane.

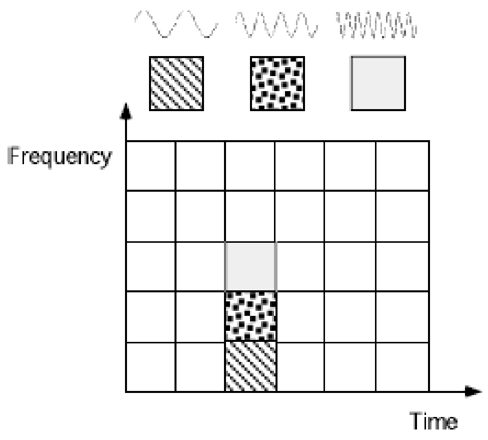


Figure 2.7: Fourier basis functions, time-frequency tiles, and coverage of the time-frequency plane.

An advantage of wavelet transforms is that the windows vary. In order to isolate signal discontinuities, one would like to have some very short basis functions. At the same time, in order to obtain detailed frequency analysis, one would like to have some very long basis functions. A way to achieve this is to have short high-frequency basis functions and long low-frequency ones. This happy medium is exactly what you get with wavelet transforms. Figure 2.8 shows the coverage in the time-frequency plane with one wavelet function, the Daubechies wavelet.

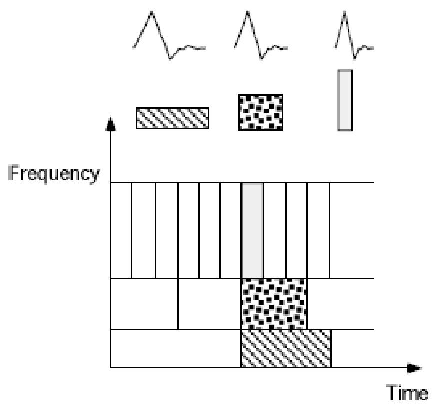


Figure 2.8: Daubechies wavelet basis functions, time-frequency tiles, and coverage of the time-frequency plane.

One thing to remember is that wavelet transforms do not have a single set of basis functions like the Fourier transform, which utilizes just the sine and cosine functions. Instead, wavelet transforms have an infinite set of possible basis functions. Thus wavelet analysis provides immediate access to information that can be obscured by other time-frequency methods such as Fourier analysis.

2.6.5. WHAT DO SOME WAVELETS LOOK LIKE?

Wavelet transforms comprise an infinite set. The different wavelet families make different trade-offs between how compactly the basis functions are localized in space and how smooth they are. Some of the wavelet bases have fractal structure. The Daubechies wavelet family is one example (see Figure 2.9).

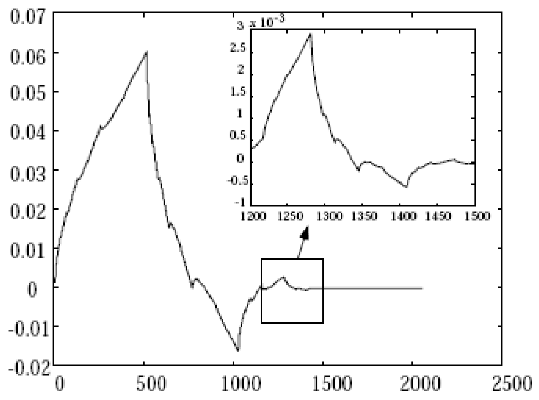


Figure 2.9: The fractal self-similarity of the Daubechies mother wavelet.

Within each family of wavelets (such as the Daubechies family) are wavelet subclasses distinguished by the number of coefficients and by the level of iteration. Wavelets are classified within a family most often by the number of vanishing moments. This is an extra set of mathematical relationships for the coefficients that must be satisfied, and is directly related to the number of coefficients. For example, within the Coiflet wavelet family are Coiflets with two vanishing moments, and Coiflets with three vanishing moments. In Figure 2.10, we illustrate several different wavelet families.

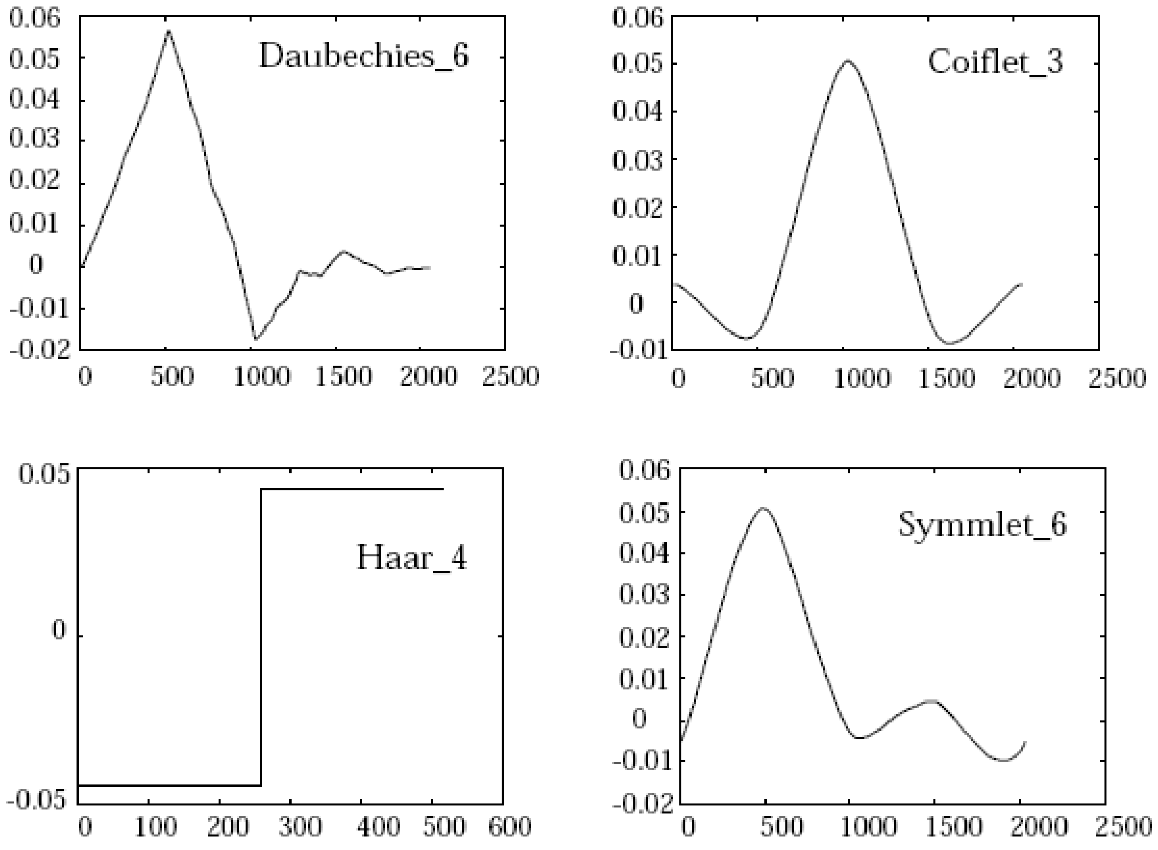


Figure 2.10: Several different families of wavelets. The number next to the wavelet name represents the number of vanishing moments (A stringent mathematical definition related to the number of wavelet coefficients) for the subclass of wavelet. These figures were generated using WaveLab.

2.6.6. WAVELET ANALYSIS:

Now we begin our tour of wavelet theory, where we analyze our signal in time for its frequency content. Unlike Fourier analysis, in which we analyze signals using sines and cosines, now we use wavelet functions.

THE DISCRETE WAVELET TRANSFORM:

Dilations and translations of the “Mother function”, or “analyzing wavelet” ϕ(x); define an orthogonal basis, our wavelet basis:

$$\phi_{(s,l)}(x) = 2^{-\frac{s}{2}}\phi(2^{-s}x - l)$$

The variables *s* and *l* are integers that scale and dilate the mother function ϕ to generate wavelets, such as a Daubechies wavelet family. The scale index s indicates the wavelet’s width, and the location index *l* gives its position. Notice that the mother functions are rescaled, or “dilated” by powers of two, and translated by integers. What makes wavelet bases especially interesting is the self-similarity caused by the scales and dilations. Once we know about the mother functions, we know everything about the basis.

To span our data domain at different resolutions, the analyzing wavelet is used in a scaling equation:

$$W(x) = \sum_{k=-1}^{N-2} (-1)^k c_{k+1}\phi(2x+k)$$

where W(x) is the scaling function for the mother function ϕ and c_k are the wavelet coefficients. The wavelet coefficients must satisfy linear and quadratic constraints of the form

$$\sum_{k=0}^{N-1} c_k = 2, \quad \sum_{k=0}^{N-1} c_k c_{k+2l} = 2\delta_{t,0}$$

where δ is the delta function and *l* is the location index.

One of the most useful features of wavelets is the ease with which a scientist can choose the defining coefficients for a given wavelet system to be adapted for a given problem. In Daubechies' original paper, she developed special families of wavelet systems that were very good for representing polynomial behavior. The Haar wavelet is even simpler, and it is often used for educational purposes.

It is helpful to think of the coefficients $\{c_0 \ldots c_n\}$ as a filter. The filter or coefficients are placed in a transformation matrix, which is applied to a raw data vector. The coefficients are ordered using two dominant patterns, one that works as a smoothing filter (like a moving average), and one pattern that works to bring out the data's "detail" information. These two orderings of the coefficients are called a quadrature mirror filter pair in signal processing parlance. A more detailed description of the transformation matrix can be found elsewhere.

To complete our discussion of the DWT, let's look at how the wavelet coefficient matrix is applied to the data vector. The matrix is applied in a hierarchical algorithm, sometimes called a pyramidal algorithm. The wavelet coefficients are arranged so that odd rows contain an ordering of wavelet coefficients that act as the smoothing filter, and the even rows contain an ordering of wavelet coefficient with different signs that act to bring out the data's detail. The matrix is first applied to the original, full-length vector. Then the vector is smoothed and decimated by half and the matrix is applied again. Then the smoothed, halved vector is smoothed, and halved again, and the matrix applied once more. This process continues until a trivial number of "smooth-smooth- smooth..." data remain. That is, each matrix application brings out a higher resolution of the data while at the same time smoothing the remaining data. The output of the DWT consists of the remaining "smooth (etc.)" components, and all of the accumulated "detail" components.

THE FAST WAVELET TRANSFORM:

The DWT matrix is not sparse in general, so we face the same complexity issues that we had previously faced for the discrete Fourier transform. We solve it as we did for the FFT, by factoring the DWT into a product of a few sparse matrices using self-similarity properties. The result is an algorithm that requires only order n operations to transform an n-sample vector. This is the "fast" DWT of Mallat and Daubechies.

WAVELET PACKETS:

The wavelet transform is actually a subset of a far more versatile transform, the wavelet packet transform. Wavelet packets are particular linear combinations of wavelets. They form bases which retain many of the orthogonality, smoothness, and localization properties of their parent wavelets. The coefficients in the linear combinations are computed by a recursive algorithm making each newly computed wavelet packet coefficient sequence the root of its own analysis tree.

ADAPTED WAVEFORMS:

Because we have a choice among an infinite set of basis functions, we may wish to find the best basis function for a given representation of a signal. A basis of adapted waveform is the best basis function for a given signal representation. The chosen basis carries substantial information about the signal, and if the basis description is efficient (that is, very few terms in the expansion are needed to represent the signal), then that signal information has been compressed.

According to Wickerhauser [41], some desirable properties for adapted wavelet bases are:

1. Speedy computation of inner products with the other basis functions.
2. Speedy superposition of the basis functions.
3. Good spatial localization, so researchers can identify the position of a signal that is contributing a large component.
4. Good frequency localization, so researchers can identify signal oscillations and
5. Independence, so that not too many basis elements match the same portion of the signal.

For adapted waveform analysis, researchers seek a basis in which the coefficients, when rearranged in decreasing order, decrease as rapidly as possible. To measure rates of decrease, they use tools from classical harmonic analysis including calculation of information cost functions. This is defined as the expense of storing the chosen representation. Examples of such functions include the number above a threshold, concentration, entropy, logarithm of energy, Gauss-Markov calculations, and the theoretical dimension of a sequence.

WAVELETS ENDNOTE:

Most of basic wavelet theory has been done. The mathematics has been worked out in excruciating detail and wavelet theory is now in the refinement stage. The refinement stage involves generalizations and extensions of wavelets, such as extending wavelet packet techniques. The future of wavelets lies in the as-yet uncharted territory of applications. Wavelet techniques have not been thoroughly worked out in applications such as practical data analysis, where for example discretely sampled time-series data might need to be analyzed. Such applications offer exciting avenues for exploration.

2.7. GABOR WAVELETS:

Dennis Gabor ***(Gábor Dénes)*** (5th June, 1900, Budapest - 9th February, 1979, London) was a Hungarian physicist who is most notable for inventing holography, an achievement for which he later received the Nobel Prize in Physics in 1971. It was in 1946 that the first time-frequency wavelets (Gabor wavelets) were introduced by Dennis Gabor [42], who at that time was researching into communication theory.

The representation of images by Gabor wavelets is chosen for its biological relevance and technical properties. The Gabor wavelets are of similar shape as the receptive fields of simple cells in the primary visual cortex (V1). They are localized in both space and frequency domains and have the shape of plane waves restricted by a Gaussian envelope function. Simple cells in the primary visual cortex have receptive fields (RFs) which are restricted to small regions of space and highly structured [43]. Earlier examinations by Hubel and Wiesel leaded to a description of these cells as *edge detectors*. More recent examinations [44] showed that the response behavior of simple cells of cats corresponds to local measurements of frequencies.

Jones & Palmer 1987 believe, among others, such as Marcelja 1980 and de Valois & de Valois 1988 that the same type of cells have been observed in both examinations and that the interpretation as edge detectors was a first approach of the real properties.

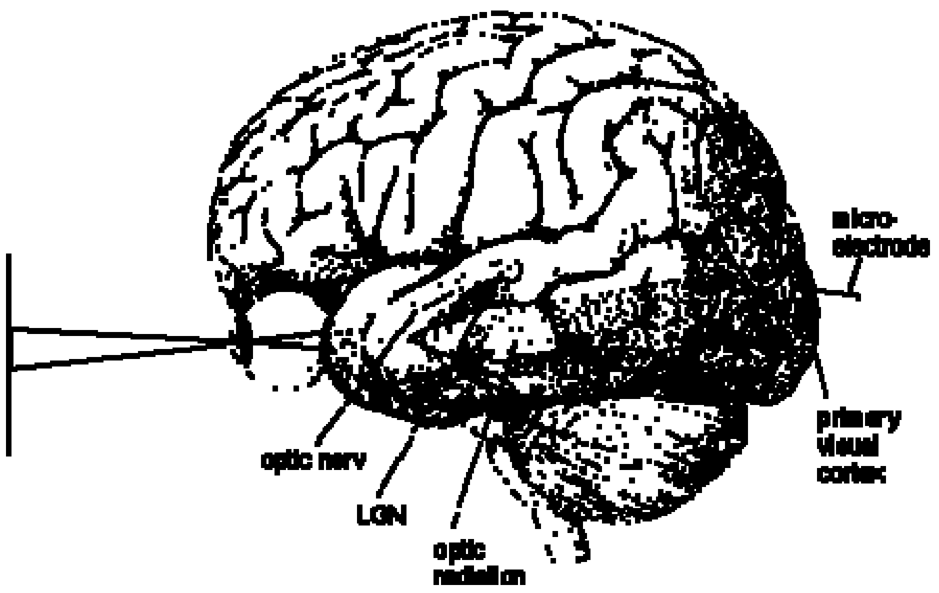


Figure 2.11: The visual pathway in the human brain (Taken from Hubel & Wiesel 1987, p. 19)

In Jones´ & Palmer's experiment, simple cells were measured with a micro electrode. The RF of a certain cell was measured location for location by projecting a dot-like stimulus on a homogeneous screen the corresponding eye looks to. An example of the resulting responses is given in figure 2.12.

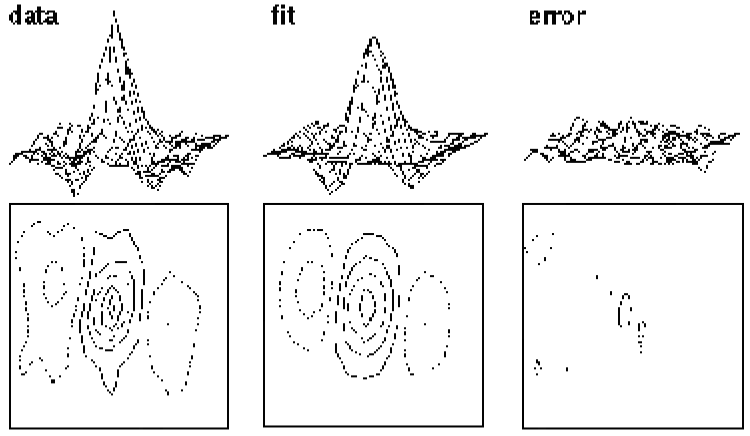


Figure 2.12: Adaptation of a Gabor filter to the data of the measured response behavior of a certain simple cell: *left:* experimental data in the space domain *middle:* adapted Gabor filter *right:* difference of the adapted filter values to the experimental data (Taken from Jones & Palmer 1987)

A simple model for the responses of simple cells in the primary visual cortex (V1) is the linear receptive field (RF)

GABOR FILTERS:

Spatial domain **Frequency domain**

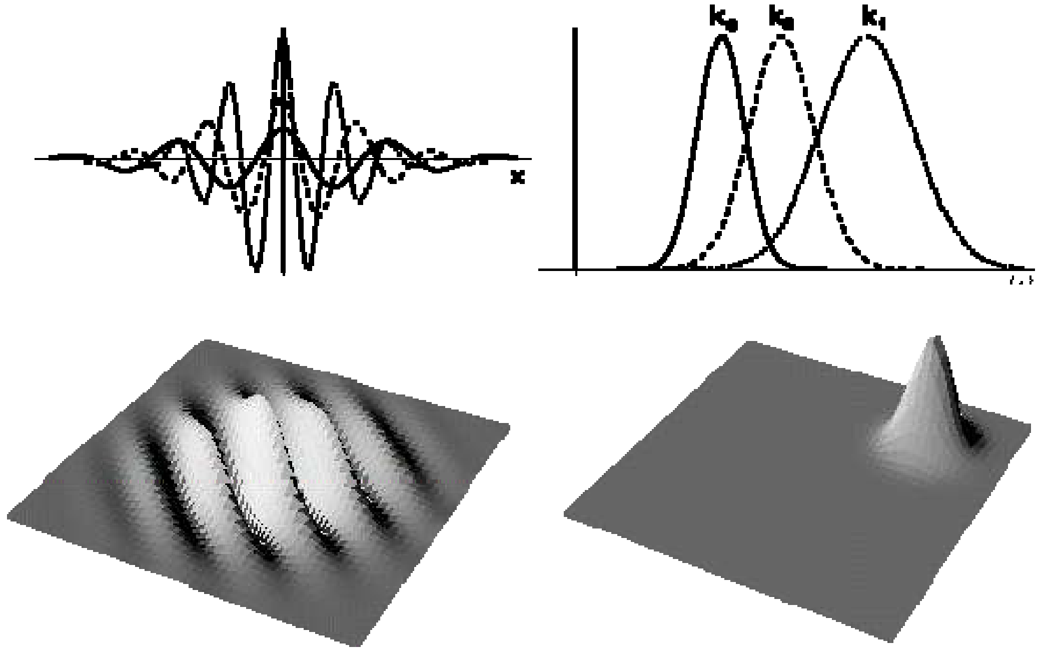


Figure 2.13: A simple model for the responses of simple cells in the primary visual cortex (V1)

Thus, the response *a* of such a cell can be written as a *correlation* of the input data, i.e. an image *I*(**x**), with the modeled RF *p*(**x**):

$$a_k(x_0) = \int I(x) p_k(x - x_0) dx$$

Thus as demonstrated above, we use Gabor wavelets (and not any other wavelets) for image representation because it represents the image based on the way the human mind does. This makes modeling computer vision based on human vision a more efficient and effective process.

2.8. NEURAL NETWORKS:

An artificial neural network (ANN), also called a simulated neural network (SNN) or commonly just neural network (NN) is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network.

In more practical terms neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data.

There are various types of neural networks. Explanation of all these types will not be possible due to constraint in space. But a specific type of neural network "The Multi-layer Perceptron (MLP)" will be dealt with in some detail since we are using a MLP in our face recognition system

MULTI-LAYER PERCEPTRON:

This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications the units of these networks apply a sigmoid function as an activation function.

The *universal approximation theorem* for neural networks states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer. This result holds only for restricted classes of activation functions, e.g. for the sigmoidal functions.

Multi-layer networks use a variety of learning techniques, the most popular being *back-propagation*. Here the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles the network will usually converge to some state where the error of the calculations is small. In this case one says that the network has *learned* a certain target function. To adjust weights properly one applies a general method for non-linear optimization task that is called gradient descent. For this, the derivative of the error function with respect to the network weights is calculated and the weights are then changed such that the error decreases (thus going downhill on the surface of the error function). For this reason back-propagation can only be applied on networks with differentiable activation functions.

In general the problem of teaching a network that performs well, even on samples that were not used as training samples, is a quite subtle issue that requires additional techniques. This is especially important for cases where only very limited numbers of training samples are available. The danger is that the network overfits the training data and fails to capture the true statistical process generating the data. Computational learning theory is concerned with training classifiers on a limited amount of data. In the context of neural networks a simple heuristic, called early stopping, often ensures that the network will generalize well to examples not in the training set.

Other typical problems of the back-propagation algorithm are the speed of convergence and the possibility to end up in a local minimum of the error function. Today there are practical solutions that make back-propagation in multi-layer perceptrons the solution of choice for many machine learning tasks.

A very basic example for implementing the XOR function is as shown in figure 2.14.

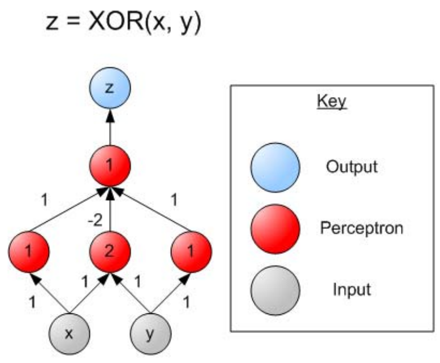


Figure 2.14: An example implementation of XOR function using MLP

3. DESIGN & IMPLEMENTATION

3.1. PROGRAMMING SPECIFICS:

Before we go into the specifics of our project, we would like to mention the implementation tools that we have used.

The entire Face recognition system has been implemented using the C programming language. The programming has been carried out using the VIM (VI Improved) editor. The programs run in Cygwin. Cygwin provides a complete UNIX environment within Windows.

The display programs that we have written will be required to be rewritten using Microsoft Visual C++ for them to run within the Windows environment. But for this we will have to purchase MS Visual C++ and obtain the valid licensing key. Since Cygwin is open source software, it is available for free download and use.

Cygwin is a Linux-like environment for Windows. It consists of two parts

- A DLL (Dynamic Link Library) (cygwin1.dll) which acts as a Linux API (Application Program Interface) emulation layer providing substantial Linux API functionality.
- A collection of tools, which provide Linux look and feel. The Cygwin DLL works with all non-beta, non “release candidate”, ix86 32 bit versions of Windows since Windows 95, with the exception of Windows CE.

3.2. BLOCK DIAGRAM AND ITS EXPLANATION:

The entire system has been modeled as three interconnected sub models. These can be represented by three blocks which as a whole forms our face recognition program. The entire system is as shown in figure 3.1.

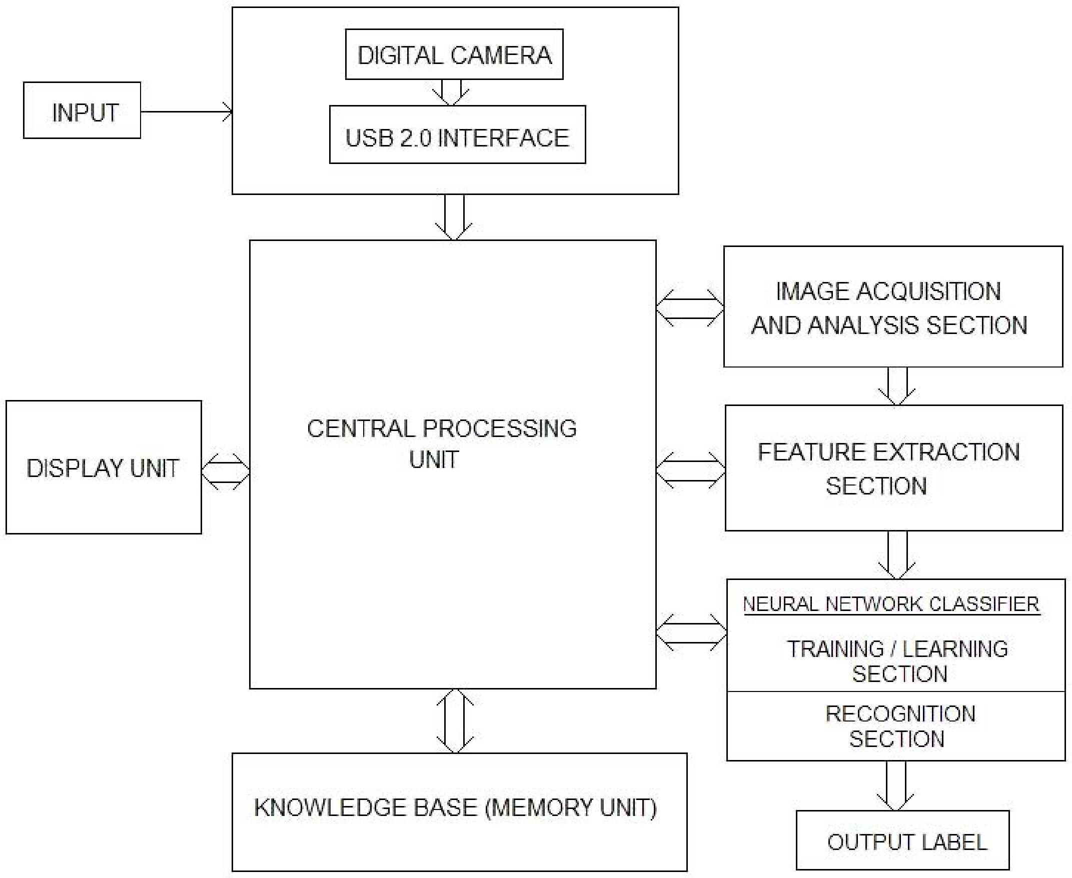

INPUT
DIGITAL CAMERA
USB 2.0 INTERFACE
DISPLAY UNIT
CENTRAL PROCESSING UNIT
IMAGE ACQUISITION AND ANALYSIS SECTION
FEATURE EXTRACTION SECTION
NEURAL NETWORK CLASSIFIER
TRAINING / LEARNING SECTION
RECOGNITION SECTION
KNOWLEDGE BASE (MEMORY UNIT)
OUTPUT LABEL

Figure 3.1: Block diagram depicting the various modules in the face recognition system.

The input consists of the face of the person seeking authentication. The facial image is captured by a digital camera and fed via USB to the central processing unit. The major components attached to the Central Processing Unit are the Display unit and the Memory unit. The entire program has been structured into three modules

- Image acquisition and analysis section
- Feature extraction section
- Neural network classifier

  a) Training/Learning section

  b) Recognition section

Let us consider each of the above sections and understand the design details.

3.2.1. IMAGE ACQUISITION AND ANALYSIS SECTION:

The input to this section is the facial image of the person seeking recognition. The image is preprocessed if required. This preprocessing involves the use of digital image enhancement techniques. The details of digital image enhancement techniques can be found in Appendix A. Here we are assuming only variations with respect to the quality of the image and not with respect to the head poise or expressions.

After the preprocessing step has been completed we need to convert the image to the portable gray map format (PGM format). This makes the image a gray scale image. Details regarding the PGM specifications can be found in Appendix B.

Once the image has been fed to the program, we have to place the wavelets in the "Region of Importance". As our images are faces the region of importance is "The inner face region". Four points are located on the inner face region as shown in figure 3.2. The red dots indicate the corners of the eyes (represented by points 1 and 2) and the corner of the lips (represented by points 3 and 4).

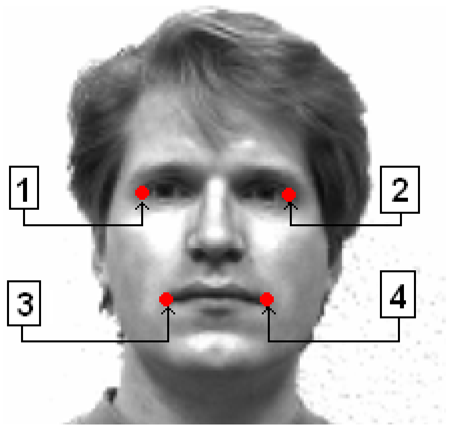


Figure 3.2: Indicates the initial points that are to be placed in order to create the wavenet

The distance between points 1 and 3 and the distance between points 2 and 4 are calculated. Now the wavelet net (wavenet) is created and placed in the inner region of the face. The system is more efficient if these feature points are made to be located automatically rather than manually. More wavelet points gives better representation but at the cost of computational speed. Finding an appropriate trade off between these two factors is of prime importance. The Gabor wavelet expression and feature extraction process is explained in detail in the next section.

3.2.2. FEATURE EXTRACTION SECTION:

This section is the core of our face recognition program. Here the face image is processed with Gabor wavelets and in the process features are extracted.

A set of frequencies and angles are chosen to represent the image at each of the above mentioned points of the wavenet.

Now we define "Lower bound" and "Upper bound" frequencies given by

$$f_{LB} = \frac{1}{x_1\sqrt{2}} \quad \text{and} \quad f_{UB} = \frac{1}{x_2\sqrt{2}}$$

The values of x_1 and x_2 are chosen such that $x_1 > x_2$. A set of frequencies to be used at each wavelet point is obtained by starting at f_{LB} and multiplying by 2 until f_{UB} is reached. The number of frequencies is given by *P*.

For each frequency, a set of orientations is chosen ranging from to $-\pi$ to π. The step size between any two θ is $\frac{2\pi}{n}$, where n is chosen appropriately. The number of orientations is given by *Q*.

The points on the wavenet are denoted by (c_x,c_y) as per the Cartesian co-ordinates. The number of wavelet points is given by *R*.

Now a set of frequencies "f" and orientations "θ" are obtained at each wavelet point (c_x,c_y).

Let $N = P*Q*R$

$I(x, y)$ (or just *I*) represents the input image. 'x' and 'y' represent the coordinates of each pixel in the Cartesian system. Hence 'x' and 'y' ranges from 0 to 'h' and 'w' respectively where 'h' is the height of the image and 'w' is the width of the image.

Now we define a family of *N* Gabor wavelet functions $\Psi = \{\psi_{1,1,1}, \ldots, \psi_{P,Q,R}\}$ of the form

$$\psi_{i,j,k}(x,y) = \frac{f_i^2}{2\pi} \exp\left\{-0.5 f_i^2 \left[\left(x - c_{x_k}\right)^2 + \left(y - c_{y_k}\right)^2\right]\right\} *$$
$$\sin\left\{2\pi f_i \left[\left(x - c_{x_k}\right)\cos\theta_j + \left(y - c_{y_k}\right)\sin\theta_j\right]\right\} \quad (3.1)$$

where f_i denotes the frequency, θ_j denotes orientation and c_{x_k} , c_{y_k} denotes wavelet position.

Now the wavelet function $\psi_{i,j,k}$(x, y) is normalized by scaling it by the reciprocal of its length $\|\psi_{i,j,k}$(x, y) $\|$. Thus we obtain the unit vector $\hat{\psi}_{i,j,k}$(x, y) as shown below

$$\hat{\psi}_{i,j,k} = \frac{\psi_{i,j,k}}{\left\|\psi_{i,j,k}\right\|} \quad (3.2)$$

The first weight w_1 associated with wavelet ψ_1 is given by the dot product of *I* and $(\hat{\psi}_{i,j,k})_1$ as shown below

$$w_1 = I \cdot (\hat{\psi}_{i,j,k})_1 \quad (3.3a)$$

The subsequent weights w_u $(2 \le u \le N)$ associated with wavelets ψ_u are given by the dot products of I_{diff_u} and $(\hat{\psi}_{i,j,k})_u$ as shown below

$$w_u = I_{diff_u} \cdot (\hat{\psi}_{i,j,k})_u \quad (3.3b)$$

where I_{diff_u} represents the intermediate difference image given by

$$I_{diff_u} = I_{diff_{(u-1)}} - \hat{I}_{u-1} \text{ with } I_{diff_1} = I(x,y) \text{, for } 2 \le u \le N \tag{3.4}$$

where $\hat{I}_u$ is the intermediate reconstructed image for each wavelet given by

$$\hat{I}_u = w_u \cdot \left(\hat{\psi}_{i,j,k}\right)_u \text{, for } 1 \le u \le N \tag{3.5}$$

The final reconstructed image is given by

$$\hat{I} = \sum_{u=0}^{N-1} \hat{I}_u \tag{3.6}$$

In the above reconstruction process, the set of weights are obtained. From this set of weights the ‘n’ highest weights are selected, where ‘n’ is chosen suitably. These ‘n’ weights along with their corresponding set of ‘n’ frequencies and orientations are the features. Thus we now obtain a “n*3” feature vector. (Other methods can also be used to choose the feature vectors such as, considering only the maximum projections, considering the frequency and orientation at each point that gives maximum projection, etc). This Feature vector is fed to the neural network for training or recognition as the case maybe.

3.2.3. NEURAL NETWORK CLASSIFIER:

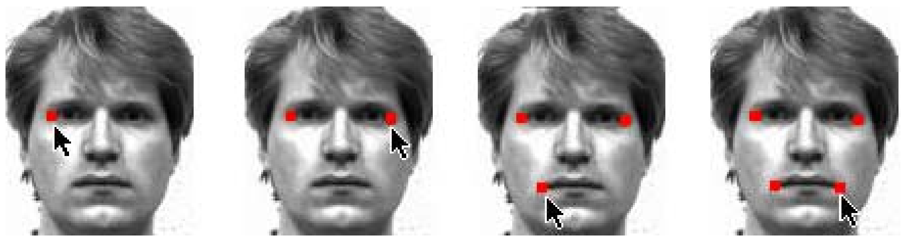
The Neural Network can either be configured in the training mode or recognition mode. Initially the Neural Network is set to training/recognition mode. As the placement of wavelets is of importance, we train the Neural Network with respect to different wavelet placements for each individual. An optimal encoding is used to differentiate between individuals. The Neural Network is trained with respect to all the individuals. Factors such as “number of hidden neurons”, “momentum”, “learning rate” and “number of iterations” are varied for optimal training. While in the recognition mode, the Neural Network outputs a “recognition label” depending upon the training it has undergone.

3.3. IMPLEMENTATION DETAILS:

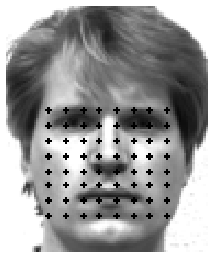
3.3.1. IMAGE ACQUISITION, ANALYSIS AND FEATURE EXTRACTION:

1) The facial photograph of the subject is captured.
2) The image is converted into the PGM format.
3) The image is enhanced if required. This constitutes the preprocessing step. The image enhancement is carried out using Digital Image Enhancement techniques.
4) The image is fed to the program by specifying the image filename at the command line.
5) The display routine displays the facial image of the subject on the screen.

6) In order to create the wavenet, we identify feature points on the face. We choose the corners of the eyes and corner of the lips as shown below. These points are selected by mouse clicks.



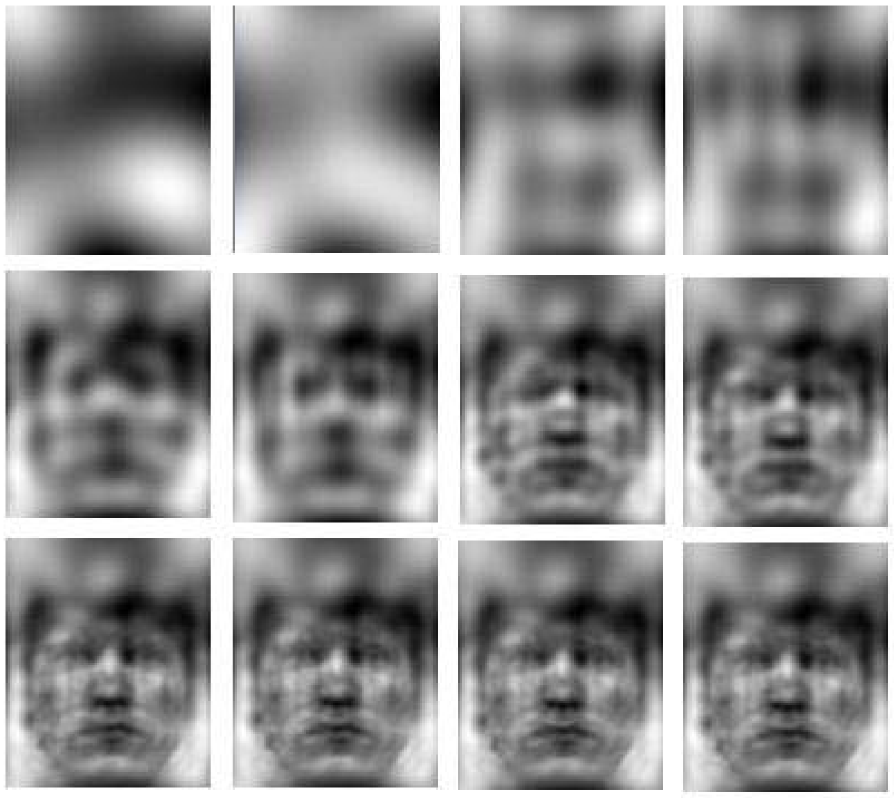
7) Once these four points have been identified the program execution is resumed by right-clicking on the image.
8) The program calculates the dimensions of the inner region of the face depending upon the ratios of the distances between the above placed points. The wavenet is then placed on the inner region of the face. A selection of 8 X 8 points gives the ideal trade off between image representation and computational speed. The wavenet points are as shown below.



9) The set of frequencies are chosen from $\frac{1}{64\sqrt{2}}$ to $\frac{1}{2\sqrt{2}}$ in multiples of 2. Thus we have a set of 6 frequencies.

10) The number of orientations is chosen to be 10. The orientations are chosen from $-\pi$ to π in steps of $\frac{2\pi}{10}$.

11) Total number of iterations is 64 * 6 * 10 (*P**Q*R*). The reconstruction of the image takes place as explained in 4.2.2. Successive images of the reconstruction process are shown in the figure below:



12) The final reconstructed image is obtained as shown below

13) This image is written and named appropriately by the program; specifying the number of wavelet points, number of frequencies used and number of different orientations chosen. For the above mentioned parameters that we have chosen, the image is named as “rec wa=64 f=6 a=10.pgm”.

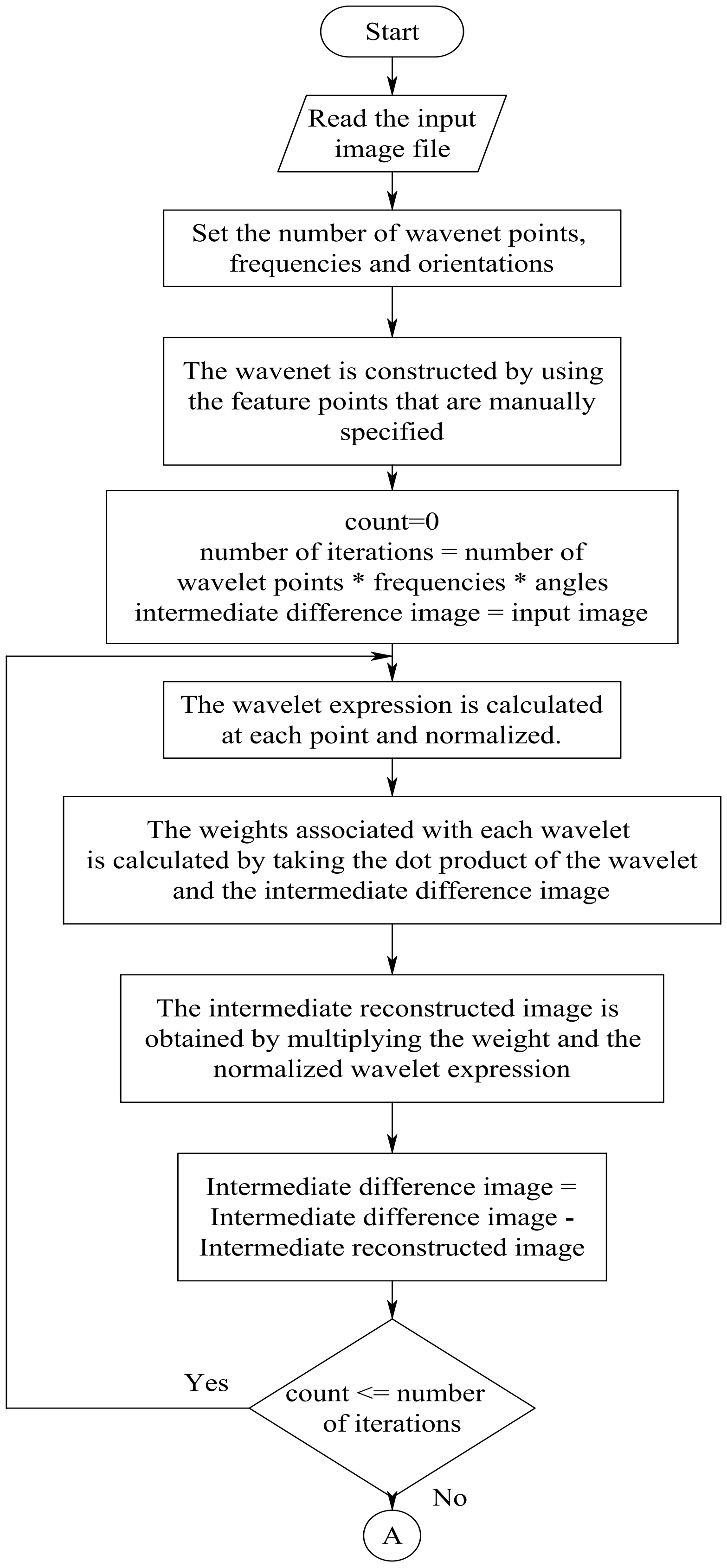
14) As the reconstruction process takes place, the projections (weights) are calculated as given in equations 3.3a and 3.3b. These weights are stored in an array. This array is bubble sorted and the highest 50 projections (in terms of magnitude) and corresponding frequencies and orientations are chosen. These 150 parameters form the feature vector which is fed to the neural network.

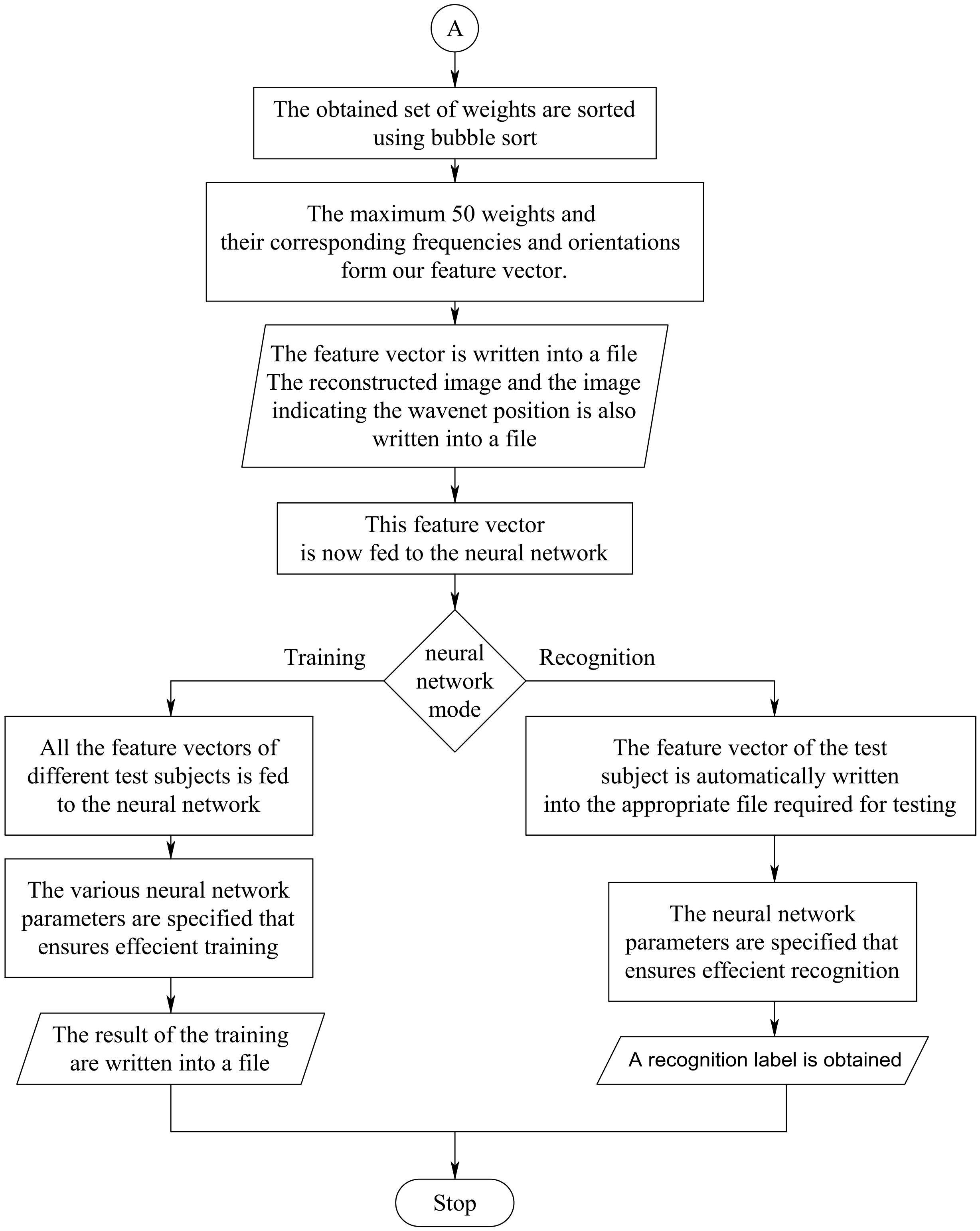
3.3.2. NEURAL NETWORK TRAINING AND RECOGNITION:

We use a MLP (multi layer perceptron). The neural network consists of a set of 6 files that are essential for the functionality of the MLP program. The details of these files can be found in Appendix C.

1) During the training stage, the obtained feature vectors are manually written into the mlp.PATTERNS file.
2) Once all the feature vectors have been written into the mlp.PATTERNS file the neural network is trained.
3) In order to test a subject, the subject’s feature vector is written into test.PATTERN file; which is done by the program automatically.
4) The neural network is now put into testing mode and we get an output recognition label.

3.3.3. FLOWCHART:


Start
Read the input image file
Set the number of wavenet points, frequencies and orientations
The wavenet is constructed by using the feature points that are manually specified
count=0
number of iterations = number of wavelet points * frequencies * angles
intermediate difference image = input image
The wavelet expression is calculated at each point and normalized.
The weights associated with each wavelet is calculated by taking the dot product of the wavelet and the intermediate difference image
The intermediate reconstructed image is obtained by multiplying the weight and the normalized wavelet expression
Intermediate difference image = Intermediate difference image - Intermediate reconstructed image
count <= number of iterations
Yes
No
A


A
The obtained set of weights are sorted using bubble sort
The maximum 50 weights and their corresponding frequencies and orientations form our feature vector.
The feature vector is written into a file The reconstructed image and the image indicating the wavenet position is also written into a file
This feature vector is now fed to the neural network
neural network mode
Training
Recognition
All the feature vectors of different test subjects is fed to the neural network
The various neural network parameters are specified that ensures effecient training
The result of the training are written into a file
The feature vector of the test subject is automatically written into the appropriate file required for testing
The neural network parameters are specified that ensures effecient recognition
A recognition label is obtained
Stop

4. TEST RESULTS

- Wavelet net needs to be placed in the region of importance.
- Placement of wavelets should be consistent with an admissible variation of just few pixels for efficient training and recognition.
- 8 X 8 wavelet matrix gives optimal tradeoff between good representation and computational time required.
- Top 50 projections and corresponding angles and frequencies which form the feature vector is found to be optimal.
- Number of hidden neurons should be about 15% of the number of input neurons.
- System is insensitive to homogenous illumination changes to a certain extent.
- System is extremely robust against facial hair, glasses and small changes in head poise.

  if the number of subjects is less than 5.
- System is not robust against extreme changes in facial expression.
- With 5 subjects overall efficiency of close to 100% was achieved.
- System is not very robust against extreme head pose variations and expression if the number of subjects is more than 5.
- With 10 subjects overall efficiency of close to 90% was achieved.
- Overall efficiency of 90% was achieved with tests on the selected images of Yale and Olivetti face databases.
- Efficiency of the neural network depends on the output encoding. One-Hot encoding was found suitable for less than 5 subjects and binary encoding for 5 or more subjects.

5. CONCLUSION

In this project we developed Face recognition system using Gabor wavelets. As said earlier, the use of Gabor wavelets is biologically motivated. This approach appears to be quite perspective, insensitive to small changes in head poise and homogenous illumination changes, robust against facial hair, glasses and also generally very robust compared to other methods. However it was found to be sensitive to large facial expression variations. Also, it was found that placement of wavelets should be consistent for efficient recognition. It is worth mentioning that Gabor wavelets technique has recently been used not only for face recognition, but also for face tracking and face position estimation.

Face recognition systems are no longer limited to classification, 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.

APPENDIX – A
DIGITAL IMAGE ENHANCEMENT TECHNIQUES

The aim of image enhancement is to improve the interpretability or perception of information in images for human viewers, or to provide 'better' input for other automated image processing techniques. Image enhancement techniques can be divided into two broad categories:

1. Spatial domain methods, which operate directly on pixels, and
2. Frequency domain methods, which operate on the Fourier transform of an image.

SPATIAL DOMAIN METHODS:

The value of a pixel with coordinates (x, y) in the enhanced image $\hat{F}$ is the result of performing some operation on the pixels in the neighbourhood of (x, y) in the input image, F. Neighbourhoods can be any shape, but usually they are rectangular. Some of the spatial domain methods are:

1) GREY SCALE MANIPULATION:

The simplest form of operation is when the operator T only acts on a 1 X 1 pixel neighbourhood in the input image. The simplest case is thresholding where the intensity profile is replaced by a step function, active at a chosen threshold value. In this case any pixel with a grey level below the threshold in the input image gets mapped to 0 in the output image. Other pixels are mapped to 255.

2) HISTOGRAM EQUALIZATION:

Histogram equalization is a common technique for enhancing the appearance of images. Suppose we have an image which is predominantly dark. Then its histogram would be skewed towards the lower end of the grey scale and all the image detail is compressed into the dark end of the histogram. If we could `stretch out' the grey levels at the dark end to produce a more uniformly distributed histogram then the image would become much clearer.

3) IMAGE SMOOTHING:

The aim of image smoothing is to diminish the effects of camera noise, spurious pixel values, missing pixel values etc. There are many different techniques for image smoothing; we will consider neighbourhood averaging and edge-preserving smoothing.

Neighbourhood Averaging:

Each point in the smoothed image $\hat{F}(x, y)$ is obtained from the average pixel value in a neighbourhood of (*x*, *y*) in the input image. For example, if we use a 3 *X* 3 neighbourhood around each pixel we would use the mask

$$\begin{matrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{matrix}$$

Each pixel value is multiplied by 1/9, summed, and then the result placed in the output image. This mask is successively moved across the image until every pixel has been covered. That is, the image is *convolved* with this smoothing mask (also known as a spatial filter or kernel).

Edge preserving smoothing:

Neighbourhood averaging or Gaussian smoothing will tend to blur edges because the high frequencies in the image are attenuated. An alternative approach is to use *median filtering*. Here we set the grey level to be the median of the pixel values in the neighbourhood of that pixel.

The outcome of median filtering is that pixels with outlying values are forced to become more like their neighbours, but at the same time edges are preserved. Of course, median filters are non-linear.

4) IMAGE SHARPENING:

The main aim in image sharpening is to highlight fine detail in the image, or to enhance detail that has been blurred (perhaps due to noise or other effects, such as motion). With image sharpening, we want to enhance the high-frequency components; this implies a spatial filter shape that has a high positive component at the centre.

A simple spatial filter that achieves image sharpening is given by

$$\begin{matrix} -1/9 & -1/9 & -1/9 \\ -1/9 & 8/9 & -1/9 \\ -1/9 & -1/9 & -1/9 \end{matrix}$$

Since the sum of all the weights is zero, the resulting signal will have a zero DC value (that is, the average signal value, or the coefficient of the zero frequency term in the Fourier expansion). For display purposes, we might want to add an offset to keep the result in the 0-255 range.

FREQUENCY DOMAIN METHODS:

Image enhancement in the frequency domain is straightforward. We simply compute the Fourier transform of the image to be enhanced, multiply the result by a filter (rather than convolve in the spatial domain), and take the inverse transform to produce the enhanced image.

The idea of blurring an image by reducing its high frequency components or sharpening an image by increasing the magnitude of its high frequency components is intuitively easy to understand. However, computationally, it is often more efficient to implement these operations as convolutions by small spatial filters in the spatial domain.

FILTERING:

Low pass filtering involves the elimination of the high frequency components in the image. It results in blurring of the image (and thus a reduction in sharp transitions associated with noise). An ideal low pass filter would retain all the low frequency components, and eliminate all the high frequency components. However, ideal filters suffer from two problems: *blurring* and *ringing*. These problems are caused by the shape of the associated spatial domain filter, which has a large number of undulations. Smoother transitions in the frequency domain filter, such as the Butterworth filter, achieve much better results.

GEOMETRIC TRANSFORMATIONS:

Geometric transformations consists of operations such as rotation, scaling and distortion (or undistortion!) of images. There are two basic steps in geometric transformations:

1. A spatial transformation of the physical rearrangement of pixels in the image and
2. A grey level interpolation, which assigns grey levels to the transformed image

Spatial transformation:

Pixel coordinates (x, y) undergo geometric distortion to produce an image with coordinates (x', y') given by $x' = r(x, y)$, $y' = s(x, y)$ where r and s are functions depending on x and y. For example Suppose $r(x, y) = \frac{x}{2}$, $s(x, y) = \frac{y}{2}$. This halves the size of the image. This transformation can be represented using a matrix equation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Grey Level Interpolation:

The problem we have to consider here is that, in general, the distortion correction equations will produce values x' and y' that are not integers. We end up with a set of grey levels for non integer positions in the image. We want to determine what grey levels should be assigned to the integer pixel locations in the output image. The simplest approach is to assign the grey value for $F(x, y)$ to the pixel having closest integer coordinates to $\hat{F}(x', y')$.

APPENDIX – B
PGM FORMAT SPECIFICATIONS

The PGM format is a lowest common denominator grayscale file format. The name “GM” is an acronym derived from “Portable Gray Map”.

Each PGM image consists of the following:

1. A “magic number” for identifying the file type. A pgm image’s magic number is the two characters “P5”.
2. Whitespace (blanks, TABs, CRs, LFs).
3. A width, formatted as ASCII characters in decimal.
4. Whitespace.
5. A height, again in ASCII decimal.
6. Whitespace.
7. The maximum gray value (Maxval), again in ASCII decimal. Must be less than 65536, and more than zero.
8. Newline or other single whitespace character.
9. A raster of Height rows, in order from top to bottom. Each row consists of Width gray values, in order from left to right. Each gray value is a number from 0 through Maxval, with 0 being black and Maxval being white. Each gray value is represented in pure binary by either 1 or 2 bytes. If the Maxval is less than 256, it is 1 byte. Otherwise, it is 2 bytes. The most significant byte is first. A row of an image is horizontal. A column is vertical. The pixels in the image are square and contiguous.
10. Each gray value is a number proportional to the intensity of the pixel, adjusted by the ITU-R Recommendation BT.709 gamma transfer function. (That transfer function specifies a gamma number of 2.2 and has a linear section for small intensities). A value of zero is therefore black. A value of Maxval represents CIE D65 white and the most intense value in the image and any other image to which the image might be compared.
11. Note that a common variation on the PGM format is to have the gray value be “linear”, i.e. as specified above except without the gamma adjustment.

12. In the transparency mask variation on PGM, the value represents opaqueness. It is proportional to the fraction of intensity of a pixel that would show in place of an underlying pixel. So what normally means white represents total opaqueness and what normally means black represents total transparency. In between, you would compute the intensity of a composite pixel of an "under" and "over" pixel as under * (1-(alpha/alpha_maxval)) + over * (alpha/alpha_maxval). Note that there is no gamma transfer function in the transparency mask.
13. Characters from "#" to the next end-of-line, before the maxval line, are comments and are ignored.

Here is an example of a small image in the plain PGM format.

```
P2
#used for comments
24 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

There is a newline character at the end of each of these lines.

APPENDIX – C

NEURAL NETWORK FILES

The Multi Layer Perceptron (MLP) program “**mlp.c”** that we are using requires the following files for execution.

a) **config_mlp_train**: Parameters of the MLP such as “Number of Iterations”, “Number of Patterns”, “Number of Input Neurons”, “Number of Hidden Neurons”, “Number of Output Neurons”, “Learning Rate (eta)”, “Momentum (alpha)”, “Error Reporting Interval” and “Error Threshold Level” that are required for the training of the neural network are specified.

b) **config_mlp_test**: The following parameters - “Number of Input Neurons”, “Number of Hidden Neurons” and “Number of Output Neurons” that are required for testing are specified in this file.

c) **mlp.PATTERNS**: The entire set of feature vectors that are required for training are given in this file along with an appropriate output encoding for the various subjects. We have used binary encoding.

d) **test.PATTERN**: This file contains the feature vector of the test subject. Once the reconstruction program executes, the obtained feature vectors are automatically written into this file.

e) **train.RESULTS**: The results of the training are written into this file.

f) **wts.DAT**: The set of weights generated corresponding to the training that the neural network undergoes are written into this file.

BIBLIOGRAPHY

[1] Gilbert Strang, MIT. Linear algebra and its applications. *Academic Press* (1976)*, Second edition: Harcourt Brace Jovanovich* (1980)*, Third edition: Brooks/Cole* (1988)*, Fourth edition: Brooks/Cole* (2005).

[2] I. Biederman. Recognition by components: A theory of human image understanding. *Psych Rev*, vol. 94, pp.115-147, 1987.

[3] I. Rock. The logic of perception. *Cambridge, MA: MIT Press*, 1983.

[4] V. Brian, Funt, Kobus Barnard, Lindsay Martin. Is machine colour constancy good enough? *Proceedings of the 5th European Conference on Computer Vision*, vol. 1, pp.445-459, June 02-06, 1998.

[5] S. Zeki. A vision of the brain. *Blackwell Scientific Publications Oxford*, 1993.

[6] T. Vetter, V. Blanz. Estimated coloured 3D face models from single images: An example based approach. *In Computer Vision – ECCV '98, Freiburg, Germany*, vol. 11, 1998.

[7] R. Brunelli, T. Poggio, Face recognition: Features versus templates. *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15(10), pp.1042-1052, 1998.

[8] J. Ingemar, Cox, Joumana Ghosn, N. Peter, Yianilos. Feature-based face recognition using mixture-distance. *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, pp.209, June 18-20, 1996.

[9] V. Govindaraju, R. K. Krishnamurthy. Holistic handwritten word recognition using temporal features derived from off-line images, *Pattern Recognition Letters*, *Elsevier Science*, vol. 17(5), pp.537-540(4), May 1, 1996.

[10] R. Chellappa, C. L. Wilson, S. Sirohey. Human and machine recognition of faces: A survey. *Proc. IEEE*, vol. 83, pp.705-740, 1995.

[11] Osamu Nakamura, Shailendra Mathur, Toshi Minami. Identification of human faces based on isodensity maps. *Pattern Recognition Archive*, vol. 24(3), pp.263-272, 1991.

[12] Z. Zhang, M. Lyons, M. Schuster, S. Akamatsu. Comparison between geometry-based and Gabor-wavelets-based facial expression recognition using multi-layer perceptron. *3rd. International Conference on Face & Gesture Recognition*, pp.454.

[13] D. Meyer, J. Denzler, H. Niemann. Model based extraction of articulated objects in image sequences for gait analysis. *Proceedings of 1997 International Conference on Image Processing*, vol. 3, pp.78-81, Oct 26-29, 1997.

[14] Michael Isard and Andrew Blake. CONDENSATION - Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, vol. 29(1), pp. 5-28, August, 1998.

[15] N. P. Costen, D. M. Parker, I. Craw. Effects of high-pass and low-pass spatial filtering on face identification. *Perception and Psychophysics*, vol. 38, pp.602–612, 1996.

[16] M. Kirby, L. Sirovich. Application of the Karhunen-Loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12(1), pp.103-108, January, 1990.

[17] I. Craw, N. Costen, T. Kato, S. Akamatsu. How should we represent faces for automatic recognition? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21(8), pp.725-736, August, 1999.

[18] B. Moghaddam, A. Pentland. Media Lab, MIT, Cambridge, MA. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19(7), pp.696-710, July, 1997.

[19] M. A. Turk, A. P. Pentland, Media Lab, MIT, Cambridge, MA. Face recognition using eigenfaces. *Proceedings of 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp.586-591, June 3-6, 1991.

[20] H. A. Rowley, S. Baluja, T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20(1), pp.23-28, January, 1998.

[21] G. Z Yang, T. S. Huang. Human face detection in a complex background. Pattern Recognition, Pergamon Press, vol. 27(1), pp.53, 1994.

[22] I. T. Jolliffe. Principle component analysis. *Spring-Verlag, New York*, 1986.

[23] M. Loeve. Theory of probability. *Wiley, New York*, 1955.

[24] T. Kanade. Picture processing system by computer complex and recognition of human faces. *Doctoral dissertation, Kyoto University*, November, 1973.

[25] Alan L. Yuille, Peter W. Hallinan, David S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision, Springer Netherlands*, vol. 8(2), pp.99-111, August, 1992.

[26] Daniel L. Swets, John (Juyang) Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18(8), pp.831-836, August, 1996.

[27] P. N. Belhumeur, J. P. Hespanha, D. J. Kriegman. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19(7), pp.711-720, July, 1997.

[28] A. Hyvaerinen, E. Oja. A fast fixed-point algorithm for independent component analysis. *MIT Press, Neural Computation*, vol. 9, pp.1483-1492, 1997.

[29] C. Liu, H. Wechsler. Comparative assessment of independent component analysis (ICA) for face recognition. *Proceedings of the Second International Conference on Audio- and Videobased Biometric Person Authentication, Washington, DC*, March, 1999.

[30] J. R. Beveridge, K. Baek, B. A. Draper, K. She. PCA vs. ICA: a comparison on the FERET data set. *Proceedings of the International Conference on Computer Vision, Pattern Recognition and Image Processing, Durham, NC*, March, 2002.

[31] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, vol. 77(2), pp.257-286, February, 1989.

[32] A. C. Harter, F. S. Samaria. Parameterisation of a stochastic model for human face identification. *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, pp.138-142, December 5-7, 1994.

[33] A. V. Nefian, M. H. Hayes. Hidden Markov models for face recognition. *Proceedings of 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '98*, vol. 5, pp.2721-2724, May 12-15, 1998.

[34] S. Eickeler, S. Müller, G. Rigoll. High performance face recognition using pseudo 2-D hidden Markov models. *European Control Conference (ECC)*, August, 1999.

[35] H. Othman, T. Aboulnasr. A Separable Low Complexity 2D HMM with Application to Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25(10), pp.1229-1238, October, 2003.

[36] H. C. Francis Crick, C. David Marr, Tomaso Poggio. An Information Processing Approach to Understanding the Visual Cortex. *AIM-557*, April 1, 1980

[37] R. Coifman, G. Weiss. Extensions of Hardy spaces and their use in analysis. *Bulletin of AMS*, vol. 83, pp.569–645, 1977.

[38] A. Grossmann, J. Morlet. Decomposition of functions into wavelets of constant shape, and related transforms. *Mathematics* 1 *Physic,* edited by Streit L. Singapore: World Scientific, vol. 1, pp.135–165, 1985.

[39] S. G. Mallat. Multiresolution approximations and wavelet orthonormal bases of $L^2(R)$. *Trans. Amer. Math.* Soc. vol. 315(1), pp.69-87, 1989.

[40] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math*, vol. 41, pp.909-996, 1988.

[41] R. R. Coifman and M. V. Wickerhauser. Wavelets and adapted waveform analysis: a toolkit for signal processing and numerical analysis. *Different Perspectives on Wavelets, American Mathematical Society*, pp. 119-153, 1993.

[42] D. Gabor. Theory of communication. *Journal of Institute of Electrical Engineers*, vol. 93, pp.429-457, 1946.

[43] S. Marcelja. Mathematical description of the responses of simple cortical cells. *J. Opt. Soc. Amer,* vol. 70(11), pp.1297-1300, 1980.

[44] J. P. Jones, L. A. Palmer. An evaluation of the two-dimensional Gabor filter model of simple receptive fields in the cat striate cortex. *J. Neurophysiology*, vol. 58, pp.1,233-1,258, 1987.