Depth Perception from Defocus — A Neural Network Based Approach for Automated Visual Inspection in VLSI Wafer Probing

Neyaz Khan

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

April 1992

ISBN 0-315-73684-4

ABSTRACT

Depth Perception from Defocus — A Neural Network Based Approach for Automated Visual Inspection in VLSI Wafer Probing

Neyaz Khan

This thesis presents a novel approach for the determination of depth from the degree of blur in images for automated visual inspection in VLSI wafer probing. There exists a smooth gradient of focus as a function of depth when a micromanipulator probe is lowered onto a test pad on a VLSI chip. Therefore, by measuring the amount of defocus, the distance from the pad can be estimated. This is very important for automated visual inspection, since contact has to be accurately sensed for non-destructive probing. Because of the very small dimensions involved (a few microns), this is an extremely sensitive operation, and involves visual feedback. Several images of the probe approaching the pad, with different degrees of defocus are taken. Fourier feature extraction, with its inherent property of shift-invariance, is used to extract significant feature vectors. These vectors contain information on the degree of defocus, and hence the distance from the probe. Neural networks are then employed to map these feature-vectors into actual distances. By successfully mapping the feature-vector space into the distance space, the neural network develops the mathematical model for this mapping. The network is then used in the recall mode to linearly interpolate the distance corresponding to the significant Fourier features of a blurred image. Because of the nature of neural networks, this operation is found to be extremely robust and insensitive to noise, vibrations and differences in illumination.

DEDICATION

To my parents, Mrs. Nigar Khan and Mr. Tarique Fareed Khan,
with love and reverence.

ACKNOWLEDGEMENTS

I take this opportunity to thank all my supervisors and friends, Rajnikant Patel, Asim Al-Khalili and Khashayar Khorasani. I thank them all for their financial support, guidance and patience in preparing me for this thesis. My special thanks go to Dr. Baher Haroun for all the help and stimulating discussions which have helped me a lot in my work.

I also wish to thank all the members of the Computer support group. Nicky, Gustavo, Dave and Guy have been extremely helpful and have provided excellent support to the computing facilities. My special thanks to Nicky and Gustavo for their time and patience and for bearing with my ever-increasing demands for computer resources.

I would also like to thank all my friends and fellow graduate students for the mental support and comradierie. I found the atmosphere and working environment very warm and friendly, and conducive to fruitful research. I wish to particularly thank my friends Malek, Rajeev, Howard, Gao, Zheng, Jean-Francois and Hamid for all those fruitful conversations at the coffee breaks. Without their friendly support this thesis would have been a burden.

Contents

List of Figures

Chapter 1

Depth From Defocus

1.1 Introduction

Computer vision provides important sensory information in many robotic tasks, one of the most important stages being the determination of depth, i.e. the distance of an object, from 2-dimensional images. This information is crucial for obstacle avoidance, navigation, poise determination, inspection, manipulation and assembly of objects, etc. There are generally two methods employed; monocular and non-monocular. Non-monocular vision requires two or more images from cameras displaced laterally along a defined baseline. The depth is then calculated by measuring the displacement of image features in one image relative to the other. In monocular vision methods, the depth is estimated as a function of features present in the image. In this thesis, monocular vision is utilized to quantize depth, as a function of blurring of objects viewed under a microscope.

1.2 Measurement of Depth from Defocus

In any photograph taken with a small depth of focus, there is a positive clue to the depth or distance of objects present. Those objects in focus are sharp and clear, while the ones in the background are blurred, so are the ones in the foreground. The further away from the focused plane an object is, the more blurred is its image.

1.2.1 The Point Spread Function

Let Θ be an operation which maps a scene onto images. Given the input scene f, the result of applying Θ to f is denoted by $\Theta(f)$. The operation Θ is considered linear if

$$\Theta(af + bg) = a\Theta(f) + b\Theta(g) \tag{1.1}$$

where a and b are constants. In the analysis of *linear* operations on pictures, the concept of a *point source* is very convenient. Considering the image f to be a sum of point sources, a knowledge of the operation's (Θ's) output for a point source input can be used to determine the output for f. The output of Θ for a point source input is called the **Point Spread Function** (PSF) of Θ.

1.2.2 The Mechanism of Blurring

The mechanism of blurring of the image is displayed in Fig. 1.1 For a thin lens, [1],

$$\frac{1}{u}+\frac{1}{v}=\frac{1}{F} \tag{1.2}$$

where u is the distance between the point in the scene and the lens, v is the distance between the lens and the plane on which the image is in perfect focus, and F the focal length of the lens. Thus

$$u=\frac{Fv}{v-F} \tag{1.3}$$

For a given lens F is constant, and also by fixing the distance between the lens and the image plane $v = v_0 = constant$. The locus of all points in perfect focus at a distance $u = u_0$ is given by:

$$u_0=\frac{Fv_0}{v_0-F} \tag{1.4}$$

Points at a distance $u > u_0$ will be focused at a distance v behind the lens, but in front of the image plane as shown in Fig.1.1 Thus, a blur circle is formed on the image plane. The blurring of the image is better described by the point spread function than by a blur-circle which can be approximated in space by a *2-D Gaussian* [1].

1.2.3 Dependence of Blur on the Spread of the Point Spread Function

The blurred image of a point source is considered as the **Point Spread Function** of the optical system, which corresponds to the degree of defocus. It is formed by the diffraction effect of the lens, and can be approximated in space by a 2-D Gaussian $G(r, \sigma)$ with

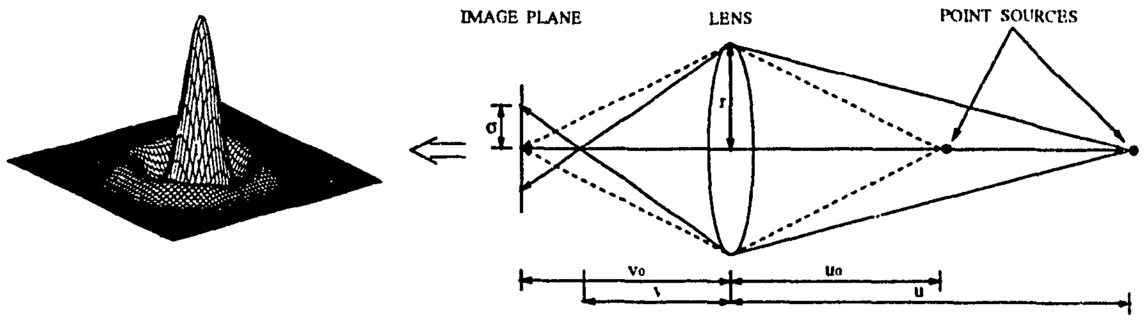


Figure 1.1: *Image formation - v_0 is the distance between the image pulse and the lens, u_0 is the distance between the lens and the locus of perfect focus, and r is the radius of the lens. When a point at $u > u_0$ is projected through the lens, it focuses at a distance $v < v_0$, so that a blur circle is formed, which can be approximated in 3D by a 2D Gaussian.*

spatial constant σ and radial distance r [1]. The value of σ in this model is the radius of the imaged point's "blur circle" or the blur parameter [1]. It can be seen from Fig.1.1 that

$$\tan(\theta) = \frac{r}{v} = \frac{\sigma}{v_0 - v} \tag{1.5}$$

From (1.3) and (1.5), we have

$$D = \frac{Frv_0}{rv_0 - F(r + \rho)} \tag{1.6}$$

or

$$D = \frac{Fv_0}{v_0 - F(+\sigma f)} \tag{1.7}$$

where v_0 is the distance between the lens and the image plane, (the sensor location in the camera), f is the f-number of the lens system, F is the focal length of the lens system, and σ is the spatial constant of the point spread function. Thus, there exists a direct relationship between the distance D and the value of σ i.e., the spread parameter σ of the Gaussian distribution is inversely proportional to the depth of the object in the scene.

1.3 Quantization of Depth as a Function of the Degree of Defocus

Several approaches for quantifying the degree of defocus present in an object in terms of distance have been undertaken by different researchers. A method of obtaining depth profiles from sharpness of the edges was first used by Grossman [2]. In this method, Grossman computes the first derivative of the intensity profile perpendicular to the edges, which is found to be bell shaped with a peak at the location of the edge. The width of this distribution peak is found to be proportional to the amount of defocus of the object. However, this method is very sensitive to noise and also fails when the edges are not well defined, as in the case of an object greatly out of focus.

Pentland [1] utilized the second derivative of the intensity profile for calculating the amount of blur in an image. By using the Laplacian of the intensity profile near the edges, he derived the following expression for calculating the spread parameter σ of the Gaussian distribution as a linear regression of x^2:

$$\ln \frac{\delta}{\sqrt{2\pi\sigma^3}} - \frac{x^2}{2\sigma^2} = \ln \left| \frac{C(x,y)}{x} \right| \tag{1.8}$$

where x is the variable perpendicular to the edge and δ is the step height of the edge. By calculating the Laplacian $C(x,y)$ around each edge, and keeping the value of δ constant, Pentland calculated the value of σ, i.e., the distance of the image point.

Subbarao [3] considered the first derivative of the intensity profile perpendicular to each edge and calculated the distance of the object using:

$$\sigma = kDs \left(\frac{1}{f} - \frac{1}{u} - \frac{1}{s} \right) \tag{1.9}$$

where k is the camera constant, f is the focal length of the lens, D is the diameter of the lens, s is the distance from the lens to the image detector plane, u is the distance of the object from the lens, and σ is the spread parameter of the line spread function. The first derivative g_x along the intensity gradient was calculated by Subbarao by taking the difference of the grey levels of the object perpendicular to the edges. For N pixels the line spread function $\theta(i)$ was then calculated by:

$$\theta(i) = \frac{g_x(i)}{\sum_{i=0}^{N} g_x(i)} \tag{1.10}$$

The spread σ of the line spread function was then calculated using the following expression for the standard deviation of the line spread function:

$$\sigma = \pm \left(\sum_{i=C}^{N} (i - \bar{i})^2 \, \theta(i) \right)^{\frac{1}{2}} \tag{1.11}$$

where $\bar{i}$ is the edge location. This algorithm does not function very well in a noisy environment and also where the edges are not well defined, resulting in erroneous measurements.

1.4 Measurement of Proximity of Probe Tip in VLSI wafer probing

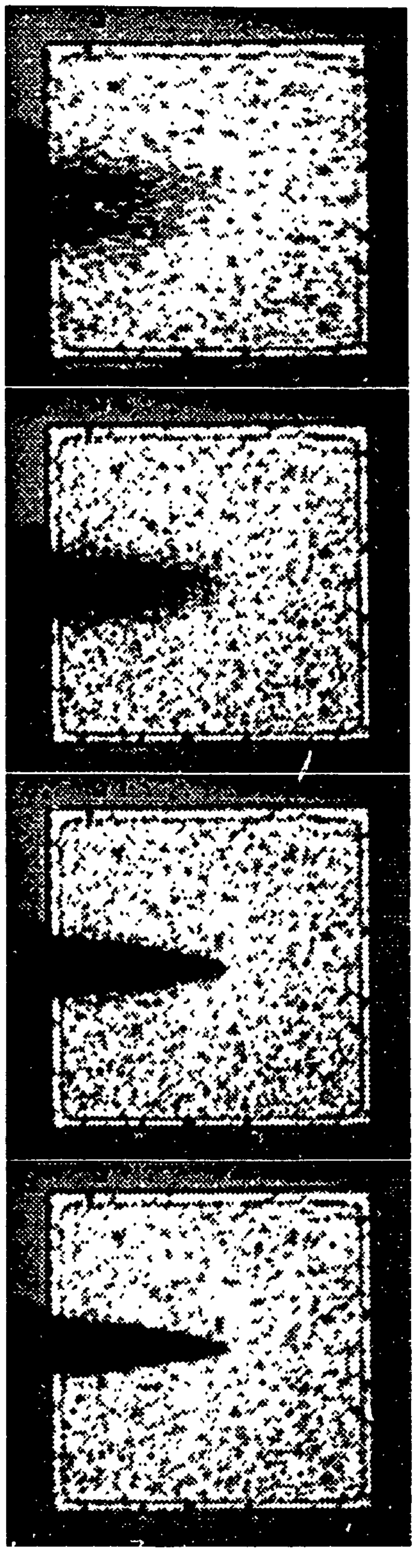
Wafer probing is an important stage in the production cycle of integrated circuits. In this process, test vectors are injected into manufactured chips through the Input/Output or test pads to determine the correct functionality and electrical characteristics of the finished product. Very fine metallic probes are used for this operation, which is an extremely sensitive operation, and is carried out by highly skilled human operators to avoid destructive testing. In order to guide the probe to its target pad in a closed loop control system, the position of the probe relative to its target must be established. Hence, the measurement of the depth of the probe is a vital task in the automation of this task.

1.4.1 Proximity as a Function of Focal Gradient

Our experimental set-up is monocular. There is only a single view of the probe along the optical axis of the light source. Thus, the only available data for the determination of the distance is available from the degree of defocus, or the relative changes in the pixel values of the images.

Various images of the probe are taken as it approaches the surface of the pad. If the microscope is focused onto the surface of the pad, the probe is in full focus only when it touches the pad, and defocused otherwise. It is observed that there exists a focal gradient, Fig.1.2, as a function of the distance of the probe tip from the pad, and the tip gets progressively out of focus as it gets further away from the surface.

All the methods described in the previous section attempt to estimate the spread σ of the Point Spread Function modeled as a Gaussian, and assume that the surfaces separated by the edges are homogeneous and smooth. These methods would be inappropriate in our case, since the boundaries or edges of the probe tip are not well defined, and there is a mix of blurred and focused regions. The task is made more difficult by the presence of non-uniform illumination, background noise and vibrations.



(a) *Distance to touch* = 60 μ

(b) *Distance to touch* = 40 μ

(c) *Distance to touch* = 20 μ

(d) *Distance to touch* = 0 μ

Figure 1.2: *Shows snap-shots of a Probe-tip being lowered onto Pad surface. There is a smooth focal gradient as a function of distance. The tip is in full focus only when it just touches the surface (d).*

1.4.2 Dantu's Algorithm

An approach for finding the distance of a probe tip from the pad was undertaken by Dantu [4] in which he calculated the value of σ or the spread of the PSF contributing to the blurring of the edge. He derived a workable solution by dividing the region around an edge into three sections. The first and the third belong to the background and the blurred object respectively while the second one belongs to the edge itself. Considering a step edge $f(x)$ with magnitude δ i.e.

$$f(x) = \begin{cases} k & \text{if } x < 0 \\ k+\delta & \text{if } x \geq 0 \end{cases} \qquad (1.12)$$

and the line spread function of the camera with the lens system to be modeled as a Gaussian

$$g(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}} \qquad (1.13)$$

The step edge, which is convolved with the line spread function, is shown to be represented by the convolution

$$h(x,\sigma) = k + \frac{\delta}{2} + \delta \int_0^x \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}}\, dx \qquad (1.14)$$

By using piece wise linear segments around the point of zero crossing of the edge, he showed that

$$\left.\frac{\partial h(x,\sigma)}{\partial x}\right|_{x=0} = \frac{\delta}{\sigma\sqrt{2\pi}} \qquad (1.15)$$

Thus, the value of σ can be calculated from the above equation to obtain the distance of the probe tip from the pad by correlating it with the measured distance.

There are three basic tasks for evaluating the value of σ; i.e. (a) Obtaining an edge-map of the probe, (b) using the estimate to calculate the actual zero-crossing along each edge; and (c) correlating the calculated value of σ with the measured distance.

1.4.3 Sources of Error

There are several sources of error in Dantu's [4] algorithm:

- Due to the presence of noise in the images, the computed edge-map is not continuous, but contains gaps within the neighbouring edges. These are interpolated with linear segments in the algorithm, which contributes to error in the system.

- Chaining of edges is a procedure which creates a linked list of the edges starting from the tip of the probe. The end-point of the tip is considered as the first element of the edge-map and the next element located by searching the neighbourhood and then linking to the first one. This operation, which is used in the above algorithm, is very susceptible to background noise in the system and can also contribute to significant error by producing several false chains created by noise in the system.

- Extraction of the three regions in the edge profile is also prone to errors, especially when the edges are not well defined and there is already some error present in the edge map.

- Piecewise plane fitting around the surface of the edge in order to determine the slope of the edge profile is an operation which involves significant approximation, thus contributing to error.

- The step size itself varies for different regions of the edge profile, and are also sensitive to differences in the levels of illumination. Since illumination varies non-linearly in various regions of the image, and also from one image to the other, there is room for error.

The advantages of using Fourier feature signatures were discussed in section 1.5.4 These inherent advantages are utilized for overcoming the shortcomings of Dantu's algorithm [4] for the measurement of distances of the probe tip for automated visual inspection in VLSI wafer probing. As discussed here, a fresh approach is taken in this thesis for quantifying the distance of the probe tip based on the spread parameter σ of the Point Spread

Function. However, this is done in the frequency domain, which is found to be a much more compact and efficient way of solving the problem.

1.5 Fourier Feature Extraction From Images

1.5.1 The Two-Dimensional Fourier Transform

Any one-dimensional signal such as speech, music or radar reflections, no matter how complex, may be described in terms of its simplest wave components. The simplest building-blocks are pure sinusoidal waves, each having unique frequency, amplitude and phase values. The Fourier transform is a powerful tool which maps exactly, the distribution of component frequencies of any signal.

Similarly, two-dimensional signals like images may also be precisely described in terms of their simplest frequency components. The simplest building blocks of images, no matter how complex, are thus pure sinusoidal gratings. Each of these component spatial-frequencies have a *frequency* (spacing of the lines), an *amplitude* (intensity of the grating), a *phase* (relative position of the gratings), and also a fourth degree of freedom not present in one-dimensional sine waves, the *orientation* of the gratings.

1.5.2 Two-Dimensional Fourier transforms for Image analysis

The Fourier Transform is in reality a very useful tool for image analysis. It represents all the information in a complex object on a consistent coordinate system, which shows the relative contribution of the range of low and high spatial frequency components. Low spatial frequencies are those aspects of an object which have gentle contours, broad transitions, wide expanses of a single shape and texture. High frequency components of the object are those with high contrast, sharp edges and frequent transitions.

Every object has a unique Fourier transform which represents its unique combination of low and high frequencies. All objects, of any type, can be represented on the same consistent coordinate system. Because the spatial frequency analysis is in terms of pure sinusoidal frequencies, it provides a universal basis for comparing images on a consistent basis. The global analysis of an object is a complete characterization of the spatial frequency content of the image, mapped onto a universal coordinate system. This universality makes it possible to systematically compare the overall appearance of any object to the appearance of any other object. Thus in the Fourier space one can meaningfully compare apples and oranges, or find the subtle differences in the degree of focus of the object, as has been done in this thesis. Through it, one can exactly analyze and quantify the difference in the universal terms of spatial frequencies. This technique, although more abstract and computationally expensive, is more predictable and universal than systems which compare objects in image or pixel form.

1.5.3 Fourier Coefficient Feature Extraction

The complete Fourier representation of an object is still a full frame of information, and would require a powerful computer to process it. Thus some form of data compression is required, at the same time preserving the features of interest.

Objects in imaged scenes are described by some set of relevant attributes or features. In the case of physical objects, these features can include, for example, size, temperature, colour, shape, texture, chemical structure, spectral response, etc. The images are then subjected to problem-dependent transforms to extract the features of interest, which are encoded as numerical variables.

The Fourier domain is particularly useful in image processing because it lends itself to feature extraction sampling techniques for generating unique feature signatures. In our case, the feature of interest is the degree of defocus, or how well defined are the edges of the probe tip. It is found from experimental observation that there exists a monotonic relationship between the frequency components close to the zero-frequency of the Fourier spectrum. and the degree of defocus of the probe tip. Thus, these frequency coefficients serve as effective *Fourier feature signatures*, and can be used for classification purposes. This results in an effective data compression of approximately 600:1.

1.5.4 Advantages of using Fourier Feature Signatures

There are important properties of the Fourier transform that make it very powerful and practical to use for image analysis and machine vision inspection tasks

Contains all the Image Information

The Fourier transform contains all the information present in the image, but analyzed by spatial frequencies, which are mapped in precise polar coordinates. Thus, any given frequency relationship will be found in the same location in the Fourier transform, whereas on the image itself, any given feature might be found anywhere. The spatial frequency relationships of every point in the image plane are analyzed versus every other point in the image plane, resulting in a complete spatial frequency analysis of the image. However, in the process of creating a global analysis, local detail and relationships are lost. The Fourier transform can only consider the entire field of view as one frame. Everything in the frame is analyzed together and all relationships are mapped, even if it were more desirable to keep some separate. In practical terms, dimensional information is lost, and one is unable to count objects or locate any particular feature on the image plane. Only one object can appear in the field of view, multiple objects being compounded into one transform. This however, does not affect our experiment, since there occurs only a single object in each image, and we are interested in a global feature i.e. the degree of defocus, and not in any local information pertaining to the probe tip. There exists a consistent relationship between the degree of defocus and the resultant Fourier coefficients, which is of prime importance.

Shift and Rotation Invariance

Fourier transforms have the inherent property of being shift invariant. This means that no matter where in the field of view the object may be, the transform will be the same. Whether the object is in the upper left, center, lower right, or anywhere in the image frame, the Fourier amplitude spectrum always appears unchanged in the center of the transform plane, provided, that the frequency content of the object does not change. This property leads to the loss of dimensional information. However, it eliminates the need for edge finding and offset processing, and loosens fixturing constraints in machine vision applications.

Rotation in the Fourier plane is treated differently. Because the orientation of the frequencies is directly represented in the transform, when an object rotates in the field of view, these features rotate as well, but the overall shape of the transform is unchanged. Thus, rotation can be measured with some sensitivity, or completely ignored by simply working with the magnitudes and ignoring the orientation related features.

Uniqueness

Every image has a unique Fourier transform, and any variations will be reflected in changes in the transform coefficients. These changes are in turn reflected in the Fourier feature signatures extracted from the transform coefficients. Thus several images of the same object with even minor changes, like different degrees of focus, have variations in their Fourier feature signatures. Since, these signatures are usually small, it enables effective classification with minimal computing. A comparison of Fourier spectra can even reveal defects in an object without the need for pre-specifying where or what the defect is. This is extensively used in industrial manufacturing and inspection tasks.

Chapter 2

An Overview of Artificial Neural Networks

2.1 The Biological Neuron

The basic anatomical unit responsible for the processing of information in the nervous system is a very specialized cell called the neuron. The classical neuron is equipped with a tree of filamentary dendrites that aggregate synaptic inputs from other neurons. The input currents are integrated by the capacitance of the cell until a critical threshold potential is reached, at which point an output is generated in the form of a nerve pulse – the action potential [5]. This output pulse propagates down the axon, which ends in a tree of synaptic contacts to the dendrites of other neurons.

The resistance of a nerve's cytoplasm is sufficiently high that signals cannot be transmitted more than about 1 millimeter before they are hopelessly spread out in time, and their information lost. For this reason, axons are equipped with an active amplification mechanism that restores the nerve pulse as it propagates. Many axons are wrapped with a special insulating material called myelin, which reduces the capacitance between the cytoplasm and the extracellular fluid, and thereby increases the velocity at which signals propagate. The sheaths of these myelinated axons have gaps called nodes of Ranvier every few millimeters. These nodes act as repeater sites, where the signal is periodically restored. A single myelinated fiber can carry signals over a distance of 1 meter or more. Even the most casual exploration of nervous tissue with an electrode reveals a host of signals encoded as trains of action potentials. For this reason, the mechanism of initiation of the nerve pulse, and of its restoration as it propagates down the axon, became the center of early physiological investigations. the first quantitative work was carried out by Hodgkin, Huxley and Katz [6] on the giant axon of a squid, which revealed the following:

- The cytoplasm in the cell's interior is normally polarized – charged to a potential of approximately – 80 millivolts with respect to the extracellular fluid.
- This potential difference is supported across a cell membrane so thin that it can be resolved only by an electron microscope.

- If sufficient current is injected into the cytoplasm in the direction to depolarize the membrane to a threshold potential of approximately –40 millivolts, a nerve pulse is initiated.
- The pulse travels in both directions from the initiation point, and its shape rapidly becomes independent of the mechanism through which the initiation took place.

2.1.1 Nerve Membrane

All electrical activity in a neuron takes place in the thin membrane that electrically separates the neuron's interior from the extracellular fluid [5]. The nerve membrane is formed from phospholipid molecules arranged in a bilayer about 50 angstroms (5 x 10^{-9} meter) thick. The approximately 100 millivolt potential across the membrane creates a very high electric field, approximately 2 x 10^{7} volts. This extremely thin structure is able to support such a large electrical gradient due to the strong electro-chemical forces within the nerve membrane.

The cross-section of the bilayer structure [5] that forms the nerve membrane consists of individual lipid molecules that have polar head-groups containing positive and negative charges. The hydrocarbon tails of the lipid molecules turn inward to avoid confronting the water. The energy of the electric dipole head-groups is much lower in the water surrounding the entire configuration than in hydrocarbon, which stabilizes the entire structure. The energy of an ion is much higher in the hydrocarbon membrane core, where the polarizability is lower than it is in the water. The membrane thus forms an energy barrier to the passage of ions.

2.1.2 Electrical Operation

The energy barrier formed by the nerve membrane is so high that, at room temperature, vanishingly few ions are able to surmount it. For this reason, it is possible to treat the membrane as a perfect insulator. Any current flow through it will have to be mediated by some agent other than the bare ions in the aqueous solution on either side. It is by manipulation of these agents that living systems achieve the gain in signal energy required for information processing.

Power Supply

Before there is gain, there must be a power supply. The most basic charge transfer agents in all nerve membranes are the metabolically driven pumps that actively expel sodium ions from the cytoplasm and concomitantly import potassium ions from the extracellular fluid. As a result of this pumping process, the cytoplasm is enriched in potassium and depleted of sodium, whereas the converse is true of the solution outside the cell.

The concentration gradient of any charged particles can be used to power electrical activity. If the membrane is permeable to only one type of charge ion, potassium for example, a net negative charge will accumulate inside the cell due to the flow of ions caused by the gradient in charge density. This negative charge will accumulate on the capacitance of the cell membrane, causing a negative potential in the cytoplasm relative to the extracellular fluid. The diffusion of ions outwards will be exactly counter balanced by the drift inward when the voltage across the membrane reaches the threshold value V_r. If the potential inside the cell is raised higher than V_r, a positive current flows outwards, and vice-versa. This is called the reversal potential for the ion.

Equivalent Circuit

The schematic in Fig.2.1 summarizes the contribution of the three ionic gradients to the nerve-membrane current [5]. A neuron at rest is polarized to a negative potential because its membrane is selectively permeable to potassium. A nerve pulse is a transient excursion of the cytoplasmic potential in a positive direction; it is an example of an excitatory signal because it depolarizes the membrane. If the membrane is charged more negatively than its resting voltage, it is said to be hyperpolarized, in which case the signal is inhibitory.

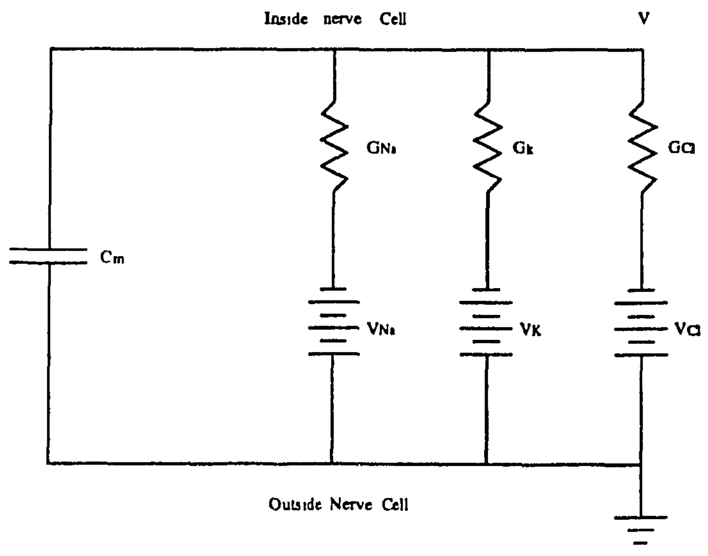


Figure 2.1: *Equivalent circuit of a patch of nerve membrane. The batteries represent the reverse potentials for particular ions; the conductances represent the membrane permeability for the same ion. The membrane capacitance is shown as a lumped capacitor.*

2.1.3 The Action Potential

If the axon is stimulated by 2-millisecond current pulses of increasing magnitude, it is observed that when the pulse drives the potential of the cytoplasm higher than -40 millivolts relative to the extracellular fluid, an action potential is generated [5]. If the current is terminated befor the potential has reached -40 millivolts, the membrane recovers, and no pulse is generated. Once the potential is more positive than -40 millivolts, however, a pulse is generated even if the driving current is terminated. Once this action potential is triggered, it acquires a constant shape, independent of the circumstances under which it originated [7]. That potential is therefore the threshold beyond which the neuron fires a pulse down the axon.

2.1.4 Ionic Channels

Current through the nerve membrane is a function of time. The ion-specific conductance changes in discrete steps; the height of each step is approximately linear in the membrane potential relative to the reversal potential of the ion. At low currents, the number of steps and the width of each step are both exponential functions of the membrane potential. At any given voltage, the steps are all the same height [8]. This suggests that each step is the result of an atomic action on the part of a single molecular entity. The molecular entities responsible for selective permeability of nerve membranes to specific ions are aggregates called channels [5]. The channels responsible for propagating the nerve pulse in an axon are voltage-controlled [8].

2.1.5 Synapses

An action potential generated across the nerve membrane propagates down it as an electrical current, which changes exponentially with the value of the potential. However, the ability to control the current into or out of an electrical node by the potential or another node is the key to all information processing. This capability is provided in neural systems by synapses. A single synapse is the neural counterpart of a transistor [5]. The tip of every neural process ends in a synapse, and there are many synaptic contacts along the branches of the dendritic tree. As an electronic computational machinery, synapses occur not in isolation, but rather in circuit arrangements. The specialization of function of many areas of the nervous system is largely a result of these synaptic circuit arrangements [9].

2.2 The Artificial Neuron

The artificial neuron was designed to mimic the first-order characteristics of the biological neuron [10]. A set of inputs are applied, each representing the output of another neuron. Each input is multiplied by a corresponding weight, analogous to a synaptic strength, and all of the weighted inputs are then summed up to determine the activation level of the neuron. Fig.2.2 shows a model of the artificial neuron that implements this idea and is the basis for all the different network paradigms. An input vector X, consisting of a set of inputs $x_1, x_2, \cdots, x_n$ is applied to the artificial neuron. Each signal is multiplied by an associated weight $w_1, w_2, \cdots, w_n$ in the weight vector W before it is applied to a summation block, labeled $\sum$. Each weight corresponds to the strength of a single biological synaptic connection. The summation block, corresponding roughly to the biological cell body, adds all of the weighted inputs algebraically, producing an output vector $NET = XW$, where X is a row vector and W is a column vector.

2.2.1 Activation Functions

The neuron's output is produced by a nonlinear activation function F acting on the summed output vector NET as shown in Figure.2.2. In vector notation this is given by $OUT = F(NET)$. This function F compresses the range of NET, so that OUT never exceeds some low limits regardless of the value of NET. This is called a *squashing function*, and is often chosen to be a *sigmoid* for which $OUT = 1/(1 + e^{-NET})$. A bipolar function like the *hyperbolic tangent*, $OUT = tanh(NET)$ (Fig.2.3), is also sometimes used to obtain a higher range

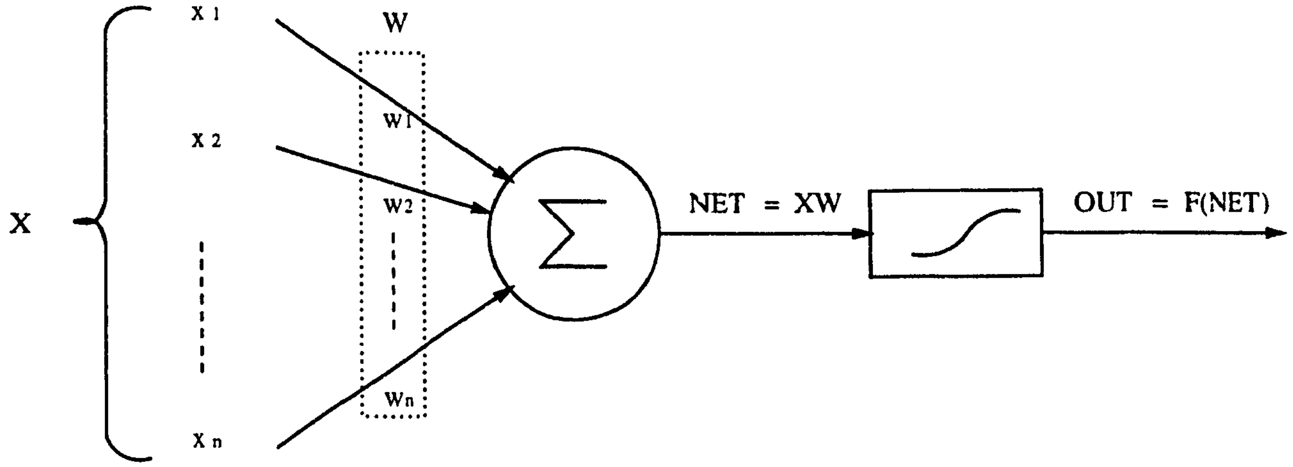


Figure 2.2: *The mathematical model of the artificial Neuron.* $NET = x_1.w_1 + x_2.w_2 + \cdots + x_n.w_n.$; *and* $OUT = F(NET)$.

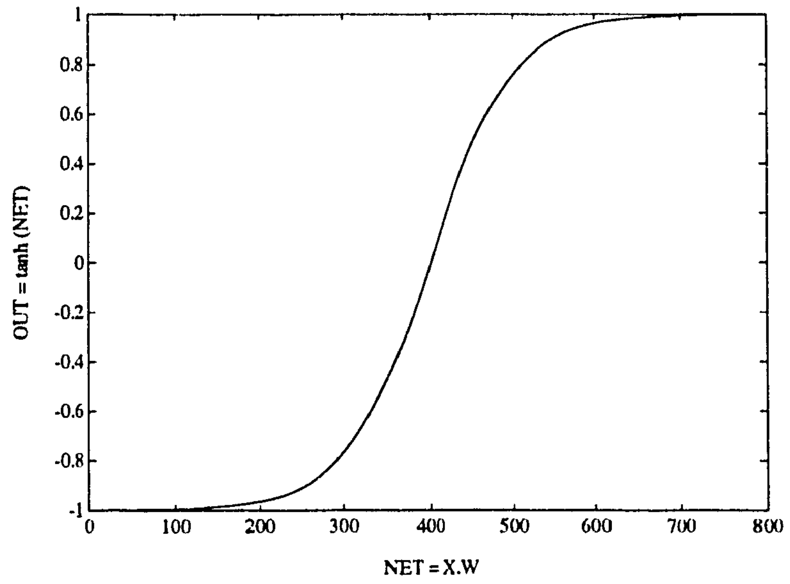


Figure 2.3: *The Hyperbolic Tangent Activation Function of the Neuron.*

By analogy to analog electronic systems, the activation function may be thought of as a nonlinear gain for the artificial neuron. This gain may be calculated by finding the ratio of the change in OUT to a small change in NET. Thus, gain is the slope of the curve at a specific excitation level. It varies from a low value at large negative excitations, where the curve is nearly flat, to a high value at zero excitation, and it drops back as excitation becomes very large and positive [11]. This characteristic has been shown to be very beneficial in some kinds of networks, specially the Backpropagation network.

The simple model of the artificial neuron ignores many of the characteristics of its biological counterpart. For example, it does not take into account time delays that affect the dynamics of the system; inputs produce an immediate output. More important, it does not include the effects of synchronization or the frequency modulation function of the biological neuron. Despite these limitations, networks formed of these neurons exhibit attributes that are strongly reminiscent of the biological system. Many standard paradigms exist as standard network types, based on the applications they are intended for. However, the basic model of the neuron remains the same, the network type being determined by their interconnection and training and recall strategies.

2.2.2 Single-Layer Artificial Neural Networks

Although a single neuron can perform certain simple pattern detection functions, the power of neural computation comes from connecting neurons into networks. The simplest network is a group of neurons arranged in a layer as shown in Fig.2.4. The neurons in the first layer perform no computation and serve only to distribute the input. Hence, they are not considered as forming a layer. The set of inputs of X, i.e., $x_1, x_2, \cdots, x_m$ are each connected to every neuron in the processing layer through separate weights $w_{11}, w_{12}, \cdots, w_{21}, w_{22}, \cdots, w_{mn}$, which is conveniently represented as a matrix W. The dimensions of the matrix are m rows by n columns, where m is the number of inputs and n the number of neurons. Thus, the NET outputs for a layer is a simple matrix multiplication given by $Y = XW$, where Y and X are row vectors.

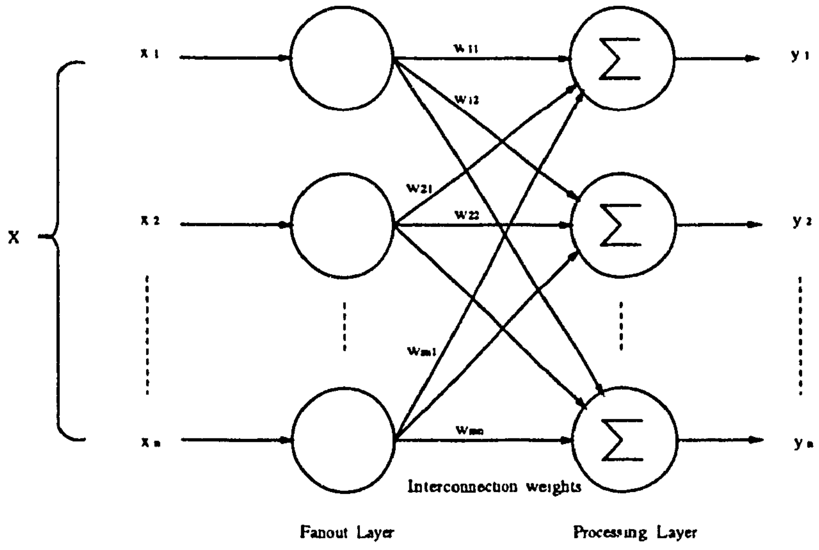


Figure 2.4: *Single Layer Neural Network.*

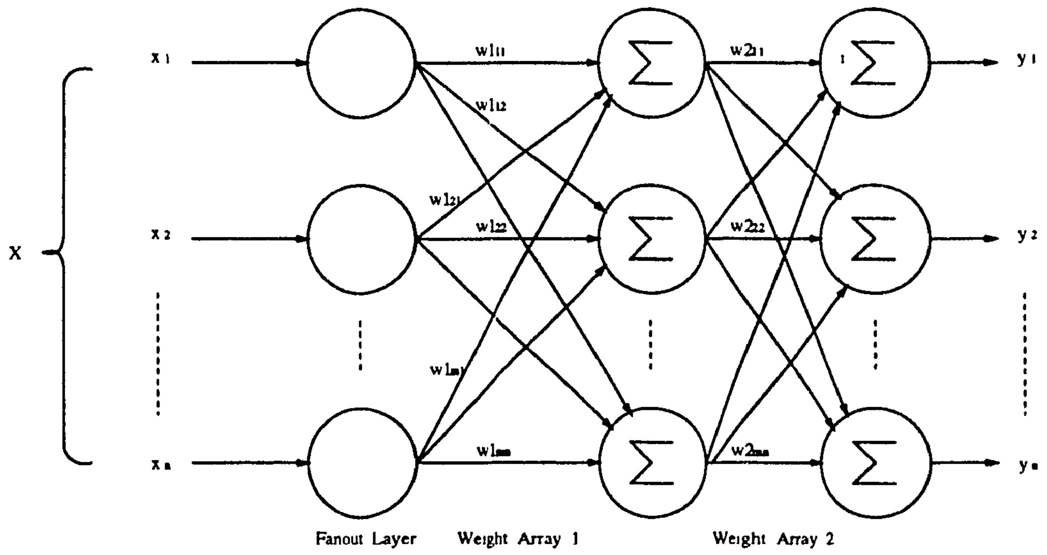


Figure 2.5: *Two Layer Neural Network.*

2.2.3 Multilayer Artificial Neural Networks

More complex networks with specialized functions are constructed by simply cascading groups of single layers (Fig.2.5). Multilayer networks provide no increase in computational power over a single layer network unless there is a nonlinear activation function between the layers [10]. This is because, calculating the output of a layer consists of multiplying the input vector by the first weight matrix and then, if no nonlinear function is present, multiplying the resulting vector by the second weight matrix. This may be expressed as $(XW_1)W_2 = X(W_1W_2)$, which are identical since matrix multiplication is associative. Single layer networks are in general severely limited in their computational capability, and the nonlinear activation functions help in enhancing the network's capability by providing multilayer networks.

2.2.4 Recurrent Networks

Recurrent networks have feedback input connections through weights extending from the output of a layer to the inputs of the same or previous layers. Nonrecurrent networks have no memory, their outputs being solely determined by the current inputs and the values of the weights. In some configurations, recurrent networks recirculate previous outputs back to inputs. Their outputs are thus determined by both their current inputs and their previous outputs. For this reason these networks exhibit properties very similar to short term memory in humans in that the state of the network outputs depends in part upon their previous inputs [10].

2.2.5 Training in Neural Networks

Training involves adjusting the weights of the network so that the application of a set of inputs produces the desired outputs. The weights are sequentially adjusted according to some predetermined procedures, so that the outputs gradually converge to the desired values.

Supervised Training

Supervised training involves the association of a target vector consisting of desired output values with each input vector, together called the training pair. Training involves computing the output values of the network for an applied input vector and comparing them with the target values. The difference is then fed back to the input as an error signal, and the network weights adjusted according to an algorithm that tends to minimize the error. The vectors of the training set are applied sequentially, and the training procedure repeated until the error for the entire training set is at an acceptably low level.

Unsupervised Training

Unsupervised training was developed by Kohonen [12] and others in 1984. It requires no target vectors for the output, and hence no comparison to predetermined output responses. The training set consists solely of input vectors, and the training algorithm modifies network weights to produce consistent outputs. The training process extracts the statistical properties of the training set and group similar vectors into classes. Applying a vector from a given class to the input from a given training set will result in a specific output vector, and there is no way to know *a priori* the outcome of a particular input vector class.

2.2.6 Training Algorithms

The learning rule determines how the weights at each level in a network change during training in response to training data points. Some commonly used learning rules are discussed in this section. In all cases supervised learning is assumed. The following notation is used in defining the effects of learning on the weights of the j^{th} node of the network :

I_j	*the value of the k^{th} input to node j*
O_j	*the output of node j*
A_j	*the activation of node j*
D_j	*the desired value of the output of node j*
w_{jk}	*the current weight of the k^{th} input*
Δw_{jk}	*change in w_{jk} due to current learning iteration*
n_j	*the number of inputs to node j , $1 \leq k \leq n_j$*

Hebbian Learning

Hebb's original learning rule [13] was based on the principle of increasing synaptic strength whenever the corresponding actual input and desired output are simultaneously active.

$$\Delta w_{jk} = \begin{cases} c_1, & \text{if } D_j > c_2 \text{ } and \text{ } I_{jk} > c_2 \\ 0, & \text{otherwise} \end{cases}$$

where

$c_1 > 0,$	is the learning rate
c_2	is the threshold above which a node is considered active

This rule, although of great historical importance, has the major drawback of having no mechanism for reducing weights. Several modifications of Hebb's original rule offer this facility. Hopfield's modification [14] consists of :

$$\Delta w_{jk} = \begin{cases} c_1, & \text{if } D_j > c_2 \text{ } and \text{ } I_{jk} > c_2 \text{ } or \text{ } D_j < c_2 \text{ } and \text{ } I_{jk} < c_2 \\ -c_1, & \text{otherwise} \end{cases}$$

The effect of this is to increase the k^{th} weight whenever the k^{th} input is the same as the desired output, i.e., both are either active or inactive.

Perceptron Learning

Hebbian learning and its variations have the shortcoming of not taking into account the actual output of the node whose input weights are being trained. The weights are thus modified even when the actual output matches the desired inputs. This usually results in degradation of network performance and eventually network paralysis. The Perceptron learning algorithm [15] avoids this problem by measuring the difference between actual and desired outputs and modifying the weights in proportion to this error :

$$\Delta w_{jk} = \begin{cases} c_1 * (D_j - O_j)/n_j, & \text{if } I_{jk} > 0,\ D_j > 0 \text{ and } O_j \leq 0 \\ -c_1 * (D_j - O_j)/n_j, & \text{if } I_{jk} > 0,\ D_j \leq 0 \text{ and } O_j > 0 \\ 0, & \text{otherwise} \end{cases}$$

The weight changes become increasingly smaller as the error becomes smaller and the weights converge to their final values. For linearly separable input sets convergence is also guaranteed [15].

Widrow-Hoff rule

The Widrow-Hoff learning rule was developed for the Adaline network [16]. It combines dependencies on the actual input, the weighted sum of all the inputs, and the desired output :

$$\Delta w_{jk} = c_1 * (D_j - A_j) * I_{jk}/n_j$$

By using the node's activation function A_j instead of the actual output O_j, a variable amount of weight change is allowed, while maintaining a nonlinear output function. The output has two possible values -1 and +1. The term I_j only affects the sign of the weight change. In Anderson's BSB model [17] however, O_j is employed instead of A_j to calculate weight upgrades because unlike the Adaline, the BSB's middle layer uses linear output functions.

Delta Rule and Backpropagation

All the training rules discussed above need *a priori* knowledge of desired outputs for all the nodes being trained. In general, only the external inputs and outputs of the net are known, with no knowledge of the outputs of hidden nodes. As a result, only the output layer whose desired output is known, can be trained by the preceding rules. These learning rules are thus good only for a single layer network since the weights of only the output layer are changed during training, and the weights of intermediate layers remain unaltered.

The generalized delta rule [18, 19] is very similar to the Widrow-Hoff rule except that it deals with continuous signals :

$$\Delta w_{jk} = c_1 * \delta_j * I_{jk}$$

where δ_j is the error signal. The error signal for each node j in the output layer is given by :

$$\delta = (D_j - O_j) * f_j'(A_j)$$

where $f_j'(A_j)$ is the derivative of the j^{th} node's output function $O_j = f_j(A_j)$. For each node j not in the output layer, the error signal is given by :

$$\delta_j = f_j'(A_j) * \sum_{q \epsilon Q} (\delta_q * w_{qj})$$

where the summation is taken over all nodes in the layer directly above node j, and w_{qj} is the weight on the connection from node j to node q.

The backpropagation learning procedure applies the delta rule to multilayer feedforward nets. The following algorithm is executed for each (input – desired output) data pair :

1. The inputs are applied and fed forward through the network, until all node outputs stabilize.

2. Starting with the output layer, the error signals are computed for each node and the input weights to all nodes in that layer updated in accordance with the above equations.

3. Step 2 is repeated layer by layer going downwards until the lowest hidden layer is reached.

The output function must be continuous and differentiable. If the output function is linear, then its derivative is a constant and the delta rule reduces to the Widrow-Hoff rule.

Other Schemes

Many other variations and modifications to the above learning algorithms have been proposed. Kohonen's learning rules [12] involves changing the weights so that they correspond to an average of all the input vectors presented during learning. This scheme is used in Hecht-Nielsen's Counterpropagation network [20].

Some networks describe the state of each node in a network based on the probability density function. Anderson and Abrahams proposed a Bayesian probability network [21] in which all inputs and outputs were given as probabilities, and the state of the overall system was represented as an energy function analogous to Hopfield's energy function [14].

A practical problem encountered during training is the occurrence of spurious states. Several methods have been proposed to reduce this effect, the one used for Hopfield networks for reducing this effect is called "unlearning" [22]. The trained networks are presented with random inputs whose resulting outputs are to modify the weights in a direction opposite to that in learning. This procedure has to be carried out carefully in order not to unlearn previously learned states.

A procedure for escaping local minima is called simulated annealing [23, 24]. The name originated from the annealing process used to cool molten metals to their lowest and most stable energy states. In the training algorithm, a "temperature" coefficient is slowly reduced to simulate this effect, while the units in the network are allowed to change states according to a probability function. At high temperatures the states change dramatically with relatively little dependence upon the system energy, while at low temperatures they change so as to reduce the energy of the system. Reducing the temperatures slowly allows the system to escape local energy minima, and to eventually settle close to the global minima [23].

2.2.7 Simulation Control Strategies

Whereas biological neural networks are massively parallel systems and are analog in nature, most artificial networks are simulated on digital computers. Thus a network designer must take into account effects due to discretization, and those due to the training being a sequential rather than a parallel process.

Synchronous vs. Asynchronous Updates

In a real network, each node updates its output continuously based on its inputs, and hence does not need a global synchronization strategy. However in digitally simulated networks, the output of each node is computed one at a time, starting with the input node. The output of one layer is then used to compute the weights, one at a time in the next layer, and so on. Furthermore, each node might have a different response time. The asynchronous behaviour of a real network is simulated using a pseudo-random control strategy in digitally simulated networks, where training pairs are selected at random from the training set. The analysis of such a strategy is mathematically very cumbersome, but experimentally found to give better results with multilayer networks.

Relaxation Time

In several kinds of networks, considerable time is required for the system to stabilize after an input is applied. The process called relaxation corresponds to the network converging to an energy minimum. An example of such a network is the Bidirectional Associative Memory [25]. Simulation involves repeatedly updating the node outputs until some stability criterion is satisfied, which is usually either a certain number of iterations, or some predefined error margin. The choice of this stability criterion is a major factor in successfully training a network.

Chapter 3

Neural Networks for Pattern Recognition

3.1 Introduction

Pattern recognition can be viewed as the mapping of a pattern correctly from pattern space into class-membership space [26]. In biological systems, it consists of an *opaque mapping*, $R_{op}(\underline{x})$, where the pattern is mapped by an observer into the correct class membership without knowing the details of the mapping process itself. In computer vision, however, the task of pattern-recognition involves replacing the opaque mapping by a *transparent mapping*, $R_{tr}(f(\underline{x})) \rightarrow \underline{c}(\underline{x}) = X$, that can be precisely and algorithmically described to a computer, (Fig.3.1). The entire procedure is carried out in two steps [27]. The object X itself many have several different instantiations ($\underline{x}$). So in the first step, we describe a specific manifestation of the object in terms of appropriate features, i.e., we go from $\underline{x}$ to $f(\underline{x})$. In the second step the machine carries out an unambiguous procedure to achieve the mapping $R_{tr}(f(\underline{x}))$, to go from $f(\underline{x})$ to $\underline{c}(\underline{x}) = X$.

3.1.1 Feature Extraction

Of the two, the first is much more difficult to achieve. It involves obtaining $f(\underline{x})$ based upon significant features of interest of the object in the scene, and is also referred to as feature extraction Fig.3.1(b). It consists of creating a data base of features associated with each pattern, based upon relationships between the significant features of interest. The choice of features is often a difficult task. There are essentially no right or wrong choices, as long as sufficient information has been included in the set of feature vectors. It is usually performed at the preprocessing stage in computer-vision applications, and may be based upon differences in colour, texture, shape, etc.

3.1.2 Classification

The focus of Pattern recognition varies with different applications. In some cases, similarity is of prime interest, in others, the task is one of estimating beliefs. The distinction is important, although in general both similarity and belief need to be estimated [27].

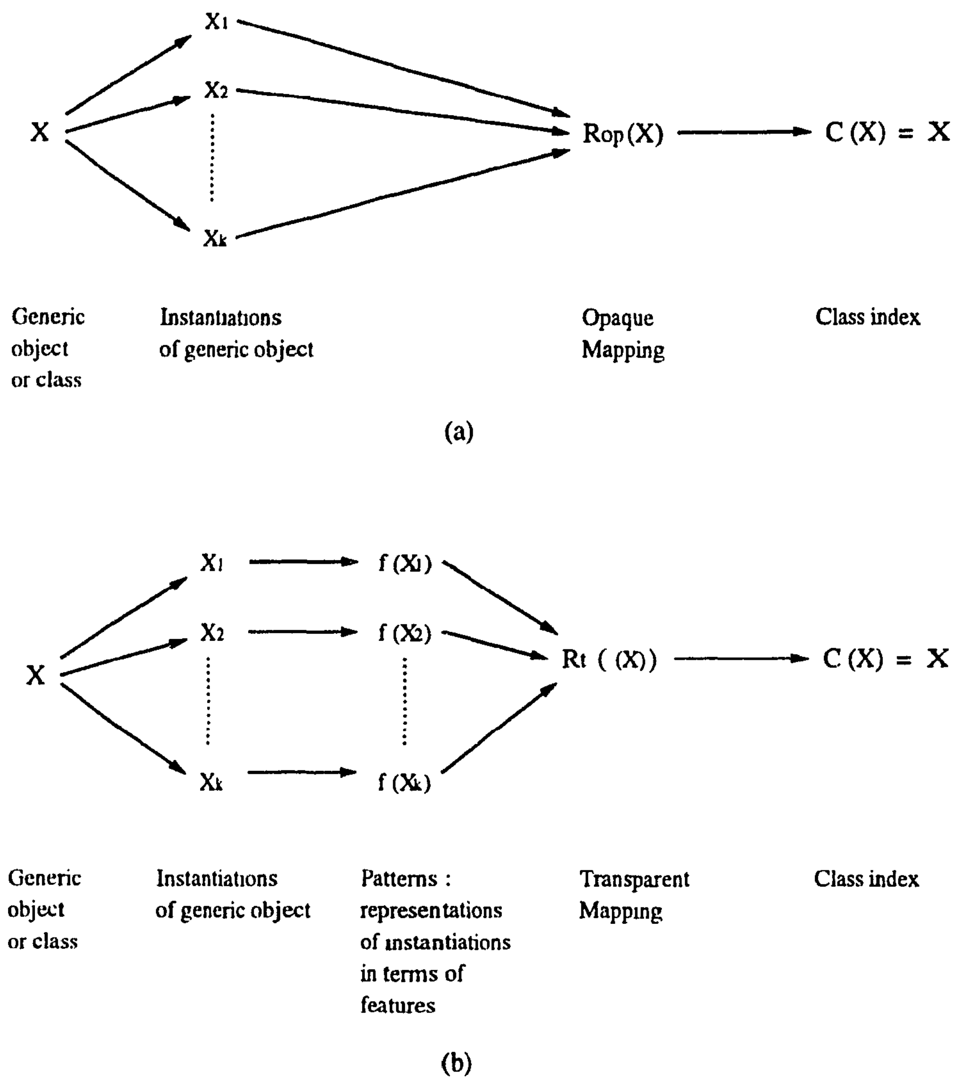


Figure 3.1: *Schematic representation of Pattern Recognition. (a) As carried out in Biological Systems (b) In Computer Vision Applications.*

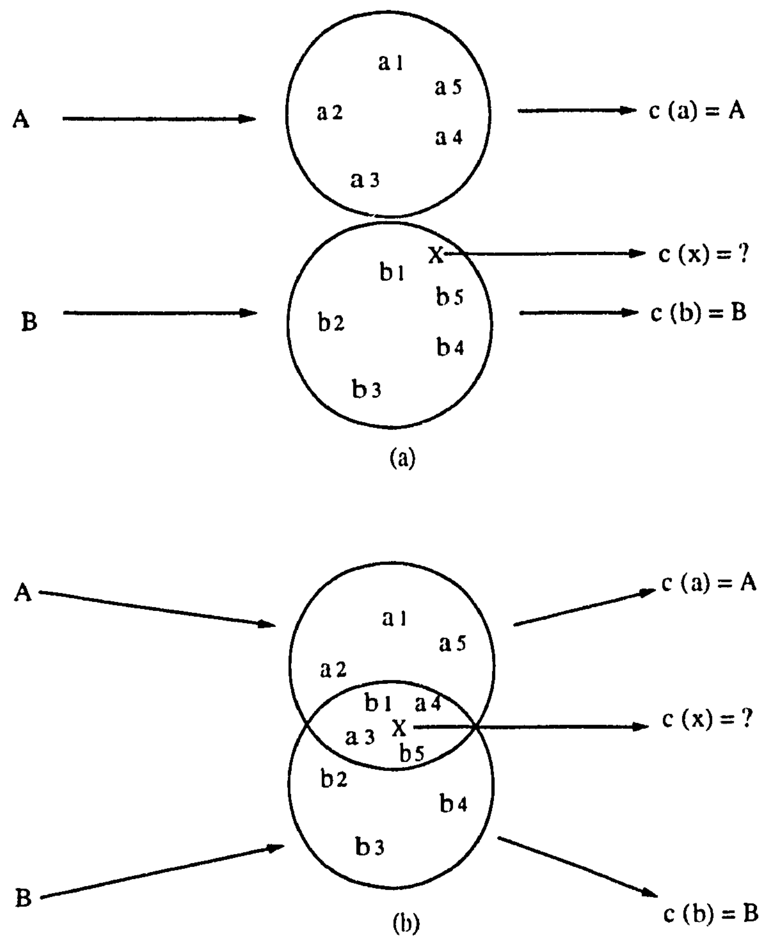


Figure 3.2: *Estimating Class Membership. (a) Generalization based upon estimating similarities. (b) Estimating whether X belongs to class a or b.*

The classification rule is that, if pattern $\underline{x}$ is most similar to pattern of class A, then pattern $\underline{x}$ belongs to class A [27]. Fig.3.2 illustrates the two extreme cases of patterns represented in some feature space. Fig.3.2(a) shows the deterministic situation where any new pattern encountered belongs to any one of several previously encountered classes. Thus, in Fig.3.2(a) the new pattern X belongs to class B. On the other hand, in Fig.3.2(b), X could represent both class A as well as class B. In this case the focus is on estimating beliefs rather than on estimating similarities. Based on some empirical beliefs, the probability of pattern X belonging to either class is estimated. In reality, actual pattern-recognition tasks lie somewhere in between the two extremes.

3.1.3 Classification Model of Pattern Recognition

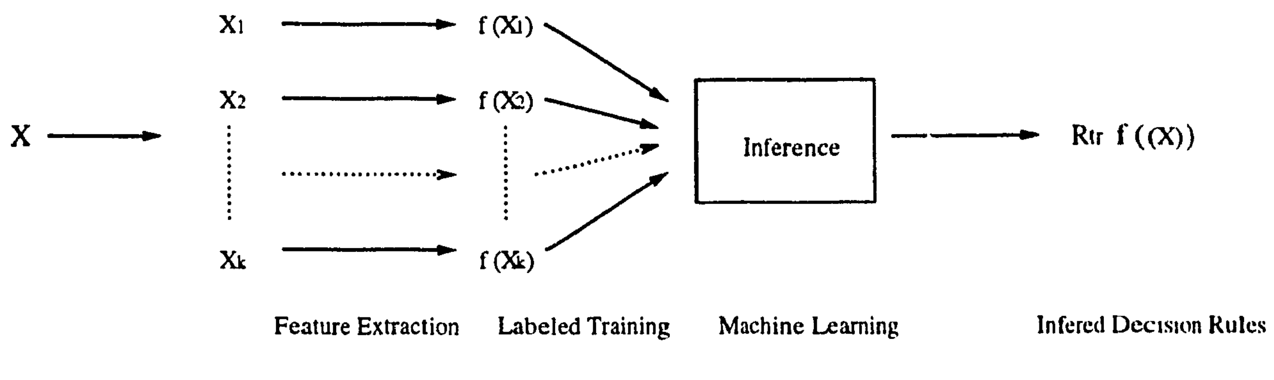
The Classification model of pattern recognition [27] is shown in Fig.3.3. It consists of:

- Performing feature extraction, i.e., deciding how the manifestation $\underline{x}$ of object X should be described symbolically in the form $f(\underline{x})$.
- Learning the transparent mapping $R_{tr}(f(\underline{x}))$, i.e., using a set of labeled training-set patterns to infer decision rules.
- Exercising the mapping $R_{tr}(f(\underline{x}))$ to estimate class membership.

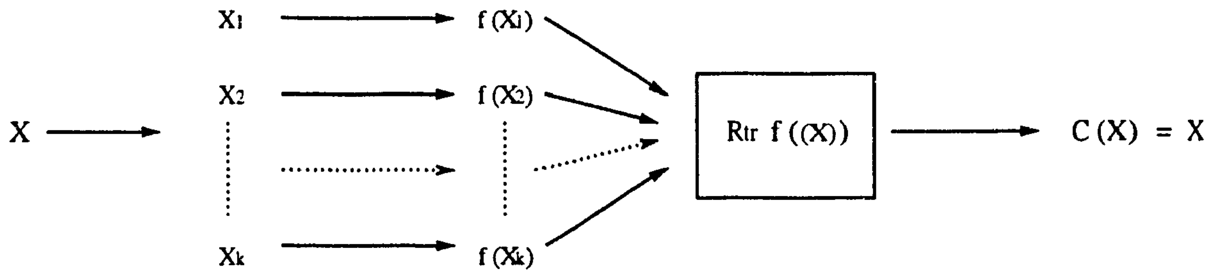
3.2 Estimating Class Membership: The Bayesian Approach

The Bayesian approach is appropriate for statistical pattern recognition when there is no ambiguity about the pattern itself. The Bayes relation can be used for estimating values of the *a priori* probability $P(c_i|\underline{x}_k)$ if those statistics are not known directly and if the class conditional probabilities and a priori probabilities are known. It is given by:

$$P(c_i|\underline{x}_k) = P(\underline{x}_k|c_i)P(c_i)/P(\underline{x}_k) \tag{3.1}$$



(a)



(b)

Figure 3.3: *Classification model of Pattern Recognition. (a) The process of learning the classification model based on labeled training sets. (b) Using the learned model for classification.*

where

$P(c_i)$	$\equiv$	*The a priori probability that a pattern belongs to class c_i, regardless of the identity of the pattern.*
$P(\underline{x}_k)$	$\equiv$	*The probability that a pattern is $\underline{x}_k$, regardless of its class membership.*
$P(\underline{x}_k \vert c_i)$	$\equiv$	*The class conditional probability that the pattern is $\underline{x}_k$, given that it belongs to class c_i.*
$P(c_i \vert \underline{x}_k)$	$\equiv$	*The a posteriori conditional probability that the pattern's class membership is c_i, given that the pattern is x_k.*
$P(c_i, \underline{x}_k)$	$\equiv$	*The joint probability that the pattern is x_k and that its class membership is c_i.*

In the Bayes approach to pattern classification, the *a posteriori* probability is treated in the same manner as are all the other probability measures. The *a posteriori* probability is an objective probability, a statistical quantity which indicates the chance or relative frequency of occurrence in a random experiment.

3.2.1 Bayes Decision Rule

Given x_k, the value of $P(c_i|\underline{x}_k)$ is evaluated for all the classes $i = 1, 2, \cdots, I$, and the class membership decided in favour of the class i for which $P(c|\underline{x}_k)$ is the largest. In general, the approach is to construct decision functions $g_i(\underline{x}_k)$, one for each class. Bayes decision rule is then used for classification as follows:

The given case x_k belongs to class $c = c_j$ if and only if

$$g_j(\underline{x}_k) > g_i(\underline{x}_k) \tag{3.2}$$

for all $i = 1, 2, \cdots, I; i \neq j$, where $g_i(\underline{x}_k)$ is the decision function for class i.

3.2.2 Risk Functions

In practice $g_i(\underline{x}_k)$ could be $P(c_i|\underline{x}_k)$ itself or some function of $P(c_i|\underline{x}_k)$. Risk functions are then synthesized from such conditional-probability functions, and decisions made on the basis of minimum risk or maximum gain [27].

The risk function is defined based upon decision errors for any arbitrary $\underline{x}_k$ when deciding class membership. Thus, the risk undertaken in deciding that $\underline{x}_k$ belongs to class c_i when it actually belongs to c_j is given by:

$$R_i(\underline{x}_k) = l_{ii}P(c_i|\underline{x}_k) + \sum_{j\neq i} l_{ij}P(c_j|\underline{x}_k) \tag{3.3}$$

where l_{ij} is the loss sustained in deciding wrong class membership c_i when it actually belongs to class c_j.

For a two-class problem,

$$R_1(\underline{x}_k) = l_{11}P(c_1|\underline{x}_k) + l_{12}P(c_2|\underline{x}_k) \tag{3.4}$$

$$R_2(\underline{x}_k) = l_{21}P(c_1|\underline{x}_k) + l_{22}P(c_2|\underline{x}_k) \tag{3.5}$$

and the decision rule would be to decide that $\underline{x}_k$ belongs to class c_1 if and only if

$$R_1(\underline{x}_k) < R_2(\underline{x}_k) \tag{3.6}$$

or

$$(l_{12} - l_{22})P(c_2|\underline{x}_k) < (l_{21} - l_{11})P(c_1|\underline{x}_k) \tag{3.7}$$

or

$$\frac{P(c_1|\underline{x}_k)}{P(c_1|\underline{x}_k)} > \frac{l_{12} - l_{22}}{l_{21} - l_{11}} \tag{3.8}$$

In case of correct classification $l_{11} = l_{22}$ and $l_{12} = l_{21}$, and would belong to class c_i if and only if:

$$P(c_1|\underline{x}_k) > P(c_2|\underline{x}_k). \tag{3.9}$$

3.2.3 Effect of System Error on Classification

In Pattern recognition tasks, classification is done on the basis of maximum *a posteriori* probability from pattern information represented as continuous class conditional or joint distributions. This is the cause of system error, which results in considerable classification error [27]. The origin and nature of such errors is illustrated in Fig.3.4. which shows the joint probability distribution $p(c_i, x)$ for two smooth unimodal distributions. For the binary classification task, based on the maximum *a posteriori* probability rule:

$$\underline{x} \in c_1 \; if\, and\, only\, if \;\; P(c_1|\underline{x}) > P(c_2|\underline{x}) \qquad and\, \underline{x} \in c_2\, otherwise \tag{3.10}$$

For any $\underline{x}$, then, the probability density of making an error is given by:

$$P_e(\underline{x}) = min\{P(c_1|\underline{x}),\, P(c_2|\underline{x})\} \tag{3.11}$$

If $x_{threshold}$ is the value of x where the two joint probability distributions intersect, and $p(x)dx$ is the probability of occurrence of the pattern within interval x and dx. Then, according to eqn.3.1, $x \in c_1$ if and only if $x > x_{threshold}$, else, $x \in c_2$. For the case when $x = x_{threshold}$, the probability of error is averaged over all values of x as below:

$$System\, error = \int_{-\infty}^{x_{threshold}} P(c_1|x)p(x)dx + \int_{threshold}^{\infty} P(c_2|x)p(x)dx \tag{3.12}$$

$$= \int_{-\infty}^{x_{threshold}} \frac{P(x|c_1)P(c_1)}{p(x)} p(x)dx$$

$$+ \int_{-\infty}^{x_{threshold}} \frac{P(x|c_2)P(c_2)}{p(x)} p(x)dx \tag{3.13}$$

$$= \int_{-\infty}^{x_{threshold}} p(x,c_1)dx + \int_{threshold}^{\infty} p(x,c_2)dx \tag{3.14}$$

$$\equiv shaded\, areas\, in\; Figure.3.4. \tag{3.15}$$

Thus, the act of classification does not require detailed knowledge of the probability distribution function [27]. The knowledge of $x_{threshold}$ is sufficient for this, and serves as the *discriminant* . For more complex situations the knowledge of several discriminants is required. For higher dimensions, the thresholds are lines, surfaces, or hypersurfaces [27].

3.3 The Nature and Role of Discriminants

3.3.1 Definition of Discriminants

The discriminant is defined by Pao [27] as a function or operator that, when applied to a pattern, yields an output that is either an estimate of the class membership of that pattern or an estimate of the values of one or more of the attributes of the pattern. In other words, the action of a discriminant is to produce a mapping from pattern space to attribute space.

For numeric-valued patterns, the discriminants are based on either of two approaches, as discussed in the following sub-sections. In the first case, the discriminant consists of a measure of distance from the class membership of nearest neighbour. In the second case, the discriminants are hypersurfaces, and patterns are classified according to their relative positions on this hypersurface.

3.3.2 Classification based on Distance to Nearest Neighbour

The distribution of all patterns amongst all classes are described by the joint probability density function $p(\underline{x}, c_i)$. That is, the joint probability that a pattern will be $\underline{x}$ and belong to class c_i is

$$p(\underline{x}, c_i) = p(\underline{x}|c_i)P(c_i) \tag{3.16}$$

and the normalization condition is

$$\sum_i P(c_i) = 1 \tag{3.17}$$

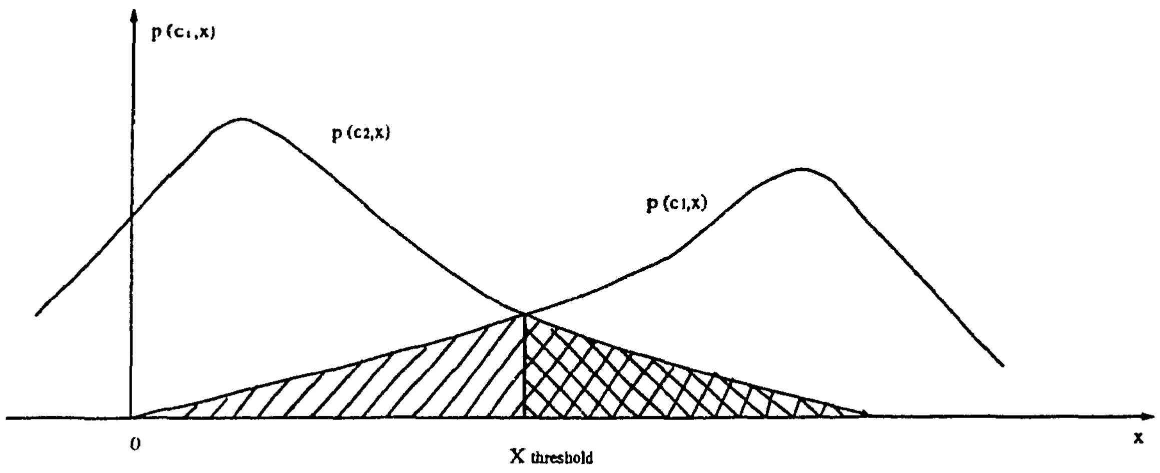


Figure 3.4: *System error in case of continuous distributions*

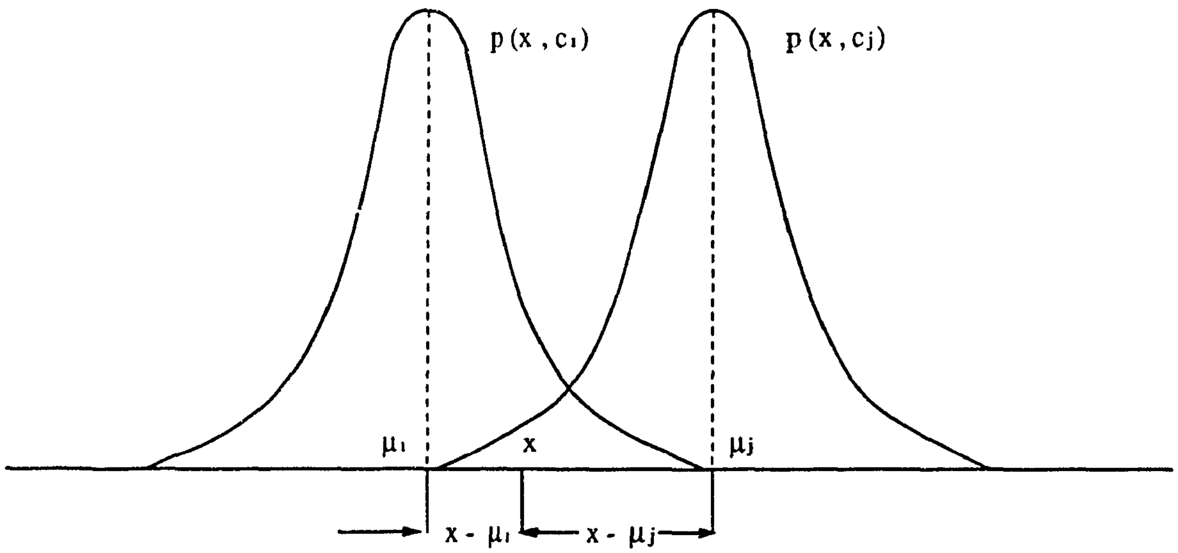


Figure 3.5: *Classification based upon nearest neighbour.*

Considering the distribution to be Gaussian, then for the one-dimensional case:

$$p(x, c_i) = (\sigma_i\sqrt{2\pi})^{-1} exp\left[-\frac{1}{2}(x - \mu_i)^2/\sigma_i^2\right] P(c_i) \tag{3.18}$$

where μ_i is the mean, and σ_i is the standard deviation, of the distribution. It is also assumed that for the two contiguous overlapping distributions, the *a posteriori* probabilities $P(c_i)$ and $P(c_j)$ are equal in magnitude, and so are the widths of the distributions, i.e., $\sigma_i = \sigma_j = \sigma$.

From the Bayes decision rule, $\underline{x}$ belongs to c_i if and only if

$$p(x, c_i) > p(x, c_j) \qquad for\ all \quad j \neq i \tag{3.19}$$

That is

$$\frac{exp\left[-\frac{1}{2}(x - \mu_i)^2/\sigma^2)\right]}{exp\left[-\frac{1}{2}(x - \mu_j)^2/\sigma^2)\right]} > 1 \tag{3.20}$$

or

$$(x - \mu_i)^2 < (x - \mu_j)^2 \tag{3.21}$$

The quantities $(x-\mu_i)^2$ and $(x-\mu_j)^2$ are interpreted as the one-dimensional Euclidean distances of x from the centers of the i and j distributions [27]. This analysis can be further extended to multidimensional distributions for classification based on finding the nearest cluster center.

There is considerable error in using this approach unless, all the $P(c_i)$ are equal to one another, and unless all the σ_i are equal. This approach is near optimal only when appropriate additional weights are introduced to correct this error [27].

For the N-Dimensional case, the general multidimensional Gaussian density can be written as

$$p(\underline{x}|c_i) = \left((2\pi)^{N/2}|\varphi_i|^{1/2}\right)^{-1} exp\left[-\frac{1}{2}(\underline{x} - \underline{\mu}_i)^T\varphi_i^{-1}(\underline{x} - \underline{\mu}_i)\right] \tag{3.22}$$

where $\underline{x}$ is the N-component column vector, $\underline{\mu}_i$ is the N-component mean vector, φ is the N-by-N covariance matrix, $(\underline{x} - \underline{\mu}_i)^T$ is the transpose of $(\underline{x} - \underline{\mu}_i)$, φ^{-1} is the inverse of φ, and $|\varphi|$ is the determinant of φ.

In this case, if all the classes have the same covariance matrix, that is, if $\varphi_i = \varphi$ for all i, and if $P(c_i) = P(c)$ for all i, then [27] $\underline{x}$ belongs to c_i if and only if

$$(\underline{x} - \underline{\mu}_i)^T\varphi^{-1}(\underline{x} - \underline{\mu}_i) < (\underline{x} - \underline{\mu}_j)^T\varphi^{-1}(\underline{x} - \underline{\mu}_j) \tag{3.23}$$

3.3.3 Classification based on Discriminant Vectors or Hyperplanes

In this case, the multidimensional normal density for the joint probabilities are used as discriminants in the form of surfaces that serve to separate patterns of one class from patterns of other classes. In the linear case, these surfaces are hyperplanes, and discriminants can take the form of vectors normal to these planes [27].

From Bayes decision rule, $\underline{x}$ belongs to c_i if and only if

$$p(\underline{x}, c_i) > p(\underline{x}, c_j) \qquad for\ all \quad j \neq i \tag{3.24}$$

Thus the boundary between two classes is described analytically by the condition that, for all patterns $\underline{x}$ on the boundary, we have

$$\frac{P(c_i)\left((2\pi)^{N/2}|\varphi_i|^{1/2}\right)^{-1} exp\left[-\frac{1}{2}(\underline{x} - \underline{\mu}_i)^T\varphi_i^{-1}(\underline{x} - \underline{\mu}_i)\right]}{P(c_j)\left((2\pi)^{N/2}|\varphi_j|^{1/2}\right)^{-1} exp\left[-\frac{1}{2}(\underline{x} - \underline{\mu}_j)^T\varphi_j^{-1}(\underline{x} - \underline{\mu}_j)\right]} = 1 \tag{3.25}$$

The above expression describes a surface in N-dimensional space, separating the c_i class patterns from the c_j class patterns. This hypersurface serves as our discriminant [27].

Assuming that $\varphi_i = \varphi_j$ and taking natural logarithms on both sides, we get

$$\underline{x}^t\varphi^{-1}(\underline{\mu_j} - \underline{\mu_i}) + (\underline{\mu_j} - \underline{\mu_i})^T\varphi^{-1}\underline{x} = 2lnP(c_i)/P(c_j) \tag{3.26}$$

Furthermore, since φ^{-1} is a symmetric matrix, the expression further reduces to

$$2\underline{x}^t\left[\varphi^{-1}(\underline{\mu_j} - \underline{\mu_i})\right] - 2lnP(c_i)/P(c_j) = 0 \tag{3.27}$$

or

$$\underline{x}^t\underline{a} = lnP(c_i)/P(c_j) \tag{3.28}$$

Equation (3.27) describes a Hyperplane rather than a more general hypersurface, and Eqn. (3.28) states that in order to classify such patterns, a discriminant vector $\underline{a}$ has to be found. If the scalar product of the pattern vector $\underline{x}$ and $\underline{a}$ is greater than $lnP(c_i)/P(c_j)$, the pattern belongs to c_i, otherwise it belongs to c_j.

3.4 Learning Discriminants

Based upon the developments in the previous sections, an analytical description for the discriminant surface can be obtained for a two category classification problem. Each pattern is considered as a two dimensional Gaussian distribution, with its principal axes along the x_1 and x_2 coordinate axes, and with diagonal covariance matrices. The mean of the two distributions are $\underline{\mu}_1 = \{\mu_{11}, \mu_{12}\}$ and $\underline{\mu}_2 = \{\mu_{21}, \mu_{22}\}$ and the diagonal components of the covariance matrices are $\{\sigma_{11}, \sigma_{12}\}$ and $\{\sigma_{21}, \sigma_{22}\}$.

The class conditional probabilities are thus

$$P(\underline{x}|c_1) = N_1 exp\left[-\frac{1}{2}(x_1 - \mu_{11})^2/\sigma_{11}^2 - \frac{1}{2}(x_2 - \mu_{12})^2\sigma_{12}^2\right] \tag{3.29}$$

and

$$P(\underline{x}|c_2) = N_2 exp\left[-\frac{1}{2}(x_1 - \mu_{21})^2/\sigma_{21}^2 - \frac{1}{2}(x_2 - \mu_{22})^2\sigma_{22}^2\right] \tag{3.30}$$

where N_1 and N_2 are normalizing factors. Equation (3.25) gives the expression for the line that provides an optimal separation of the two classes. Thus,

$$-(x_1 - \mu_{11})^2/\sigma_{11}^2 + (x_1 - \mu_{21})^2/\sigma_{21}^2 - (x_2 - \mu_{12})^2/\sigma_{12}^2 + (x_2 - \mu_{22})^2/\sigma_{22}^2$$
$$= 2ln[P(c_1)N_1/P(c_2)N_2] \tag{3.31}$$

3.4.1 Linear Discriminants

If $\sigma_{11} = \sigma_{12} = \sigma_{21} = \sigma_{22} = \sigma$, then linear separability is obtained as explained in Eqn. (3.28). The discriminants are thus

$$2(\mu_{11} - \mu_{21})x_1 + 2(\mu_{12} - \mu_{22})x_2 = (\mu_{11}^2 - \mu_{21}^2) + (\mu_{12}^2 - \mu_{22}^2)$$
$$+ 2\sigma ln P(c_1)/P(c_2) \tag{3.32}$$

or

$$x_2 = \left(\frac{\mu_{21} - \mu_{11}}{\mu_{12} - \mu_{22}}\right) x_1 + [(\mu_{11}^2 - \mu_{21}^2) + (\mu_{12}^2 - \mu_{22}^2)$$
$$+ 2ln P(c_1)/P(c_2)]/2(\mu_{12} - \mu_{22}) \tag{3.33}$$

which is the equation of a straight line of the form $y = mx + c$, where m is the slope of the line and c is the intercept on the y axis. Thus (3.32) is an equation of a linear discriminant.

3.4.2 Neural Networks Perspective to Learning of Discriminants

Equation (3.32) can also be viewed in terms of a neural network. Thus, when the inputs are x_1 and x_2, the network learns that the weights need to be $2(\mu_{11} - \mu_{21})$ and $2(\mu_{12} - \mu_{22})$, respectively. The sum of the linearly weighted inputs are then compared to

$$(\mu_{11}^2 - \mu_{21}^2) + (\mu_{12}^2 - \mu_{22}^2) + 2\sigma ln P(c_1)/P(c_2) \qquad (3.34)$$

and the pattern classified into class c_1 or c_2 depending upon whether the sum of the weighted inputs is greater or less than this number. Thus, during the learning phase, the network learns the training pairs $\{x_{i1}, x_{i2}\}$ for each pattern vector $\underline{x}$, and compares it to eqn. (3.34) in the recall mode to classify the patterns into appropriate classes.

3.4.3 Non-linear Discriminants

In the case when the standard deviations are not equal, the discriminant is not a straight line. The general expression for the determinant is then

$$\begin{aligned}
&2(\mu_{11}/\sigma_{11}^2 - \mu_{21}/\sigma_{21}^2)x_1 + (1/\sigma_{11}^2 - 1/\sigma_{21}^2)x_1^2 \\
&+2(\mu_{22}/\sigma_{22}^2 - \mu_{12}/\sigma_{12}^2)x_2 + (1/\sigma_{12}^2 - 1/\sigma_{22}^2)x_2^2 \\
&= (\mu_{11}^2/\sigma_{11}^2 - \mu_{21}^2/\sigma_{21}^2) + (\mu_{22}^2/\sigma_{22}^2 - \mu_{12}^2/\sigma_{12}^2) \\
&\qquad + ln[P(c_1)N_1/P(c_2)N_2] \qquad (3.35)
\end{aligned}$$

The determinant is now a quadratic function of x_1 and x_2. In other words, although the two populations can still be separated by a line, it is not straight anymore. Since the two populations are now not linearly separable, the use of linearly separable discriminant would lead to significant system error.

Linear separability can still be achieved by enhancing the input pattern. In our case by adding the additional components x_1^2 and x_2^2 to the input pattern. Thus if we make the substitution

$$z_1 = 2(\mu_{11}/\sigma_{11}^2 - \mu_{21}/\sigma_{21}^2)x_1 + (1/\sigma_{11}^2 - 1/\sigma_{21}^2)x_1^2$$

and

$$z_2 = 2(\mu_{22}/\sigma_{22}^2 - \mu_{12}/\sigma_{12}^2)x_2 + (1/\sigma_{12}^2 - 1/\sigma_{22}^2)x_2^2$$

which is of the form

$$z_1 + z_2 = C$$

where C is the right hand side of eqn. (3.35), the distributions are now separable in $\{z_1, z_2\}$ space. No new information is added here to achieve linear separability. In neural networks, an additional nonlinearity is introduced due to the presence of a nonlinear activation function at the output of the neurons, (sec. 2.2.1).

3.5 Learning Discriminants – A Neural Networks Approach

3.5.1 The Perceptron as a Two-class Linear Discriminant

A linear discriminant can be represented in the form of an array of multipliers and summing units as shown in Fig. 3.6. The discriminant can also be treated as a black-box, denoted by the dotted-lines, with the pattern-features as inputs. Looking into the box, each input is connected to the output by a link containing a multiplier. The inputs to the summing unit are thus appropriately weighted. In the case of a two-class classification problem, the output takes the value +1 and −1 corresponding to the two classes c_1 and $\sim c_1$ respectively. This black-box can thus be replaced by a linear Perceptron [28]. The weights are then adjusted to obtain this binary classification.

3.5.2 The Neural Network as a Multi-Class Linear Discriminant

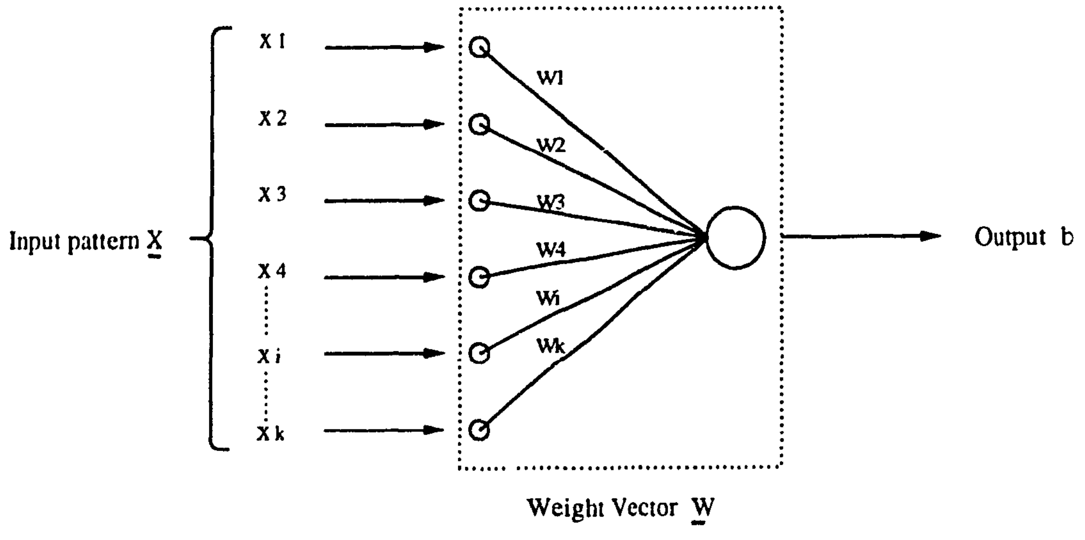
A multi-class classification task would require several outputs as shown in Fig. 3.7. In the network implementation of a linear discriminant, all the nodes and links are linear. This is however not true in the case of learning algorithms used by conventional neural networks, since a nonlinearity is usually introduced at the output of each node. In our case, linear links and nodes are used to illustrate the use of a network as a multi-class linear discriminant. The Linear Perceptron Algorithm [28] is used iteratively to adjust the weights of the network.

The features of a pattern are defined in terms of a column vector $\underline{x}$, so that $\underline{x}^t = \{x_1, x_2, \cdots, x_N\}$. Patterns are then grouped into a matr' X. The linear discriminant is then obtained by solving

$$X\underline{w} = \underline{b} \tag{3.36}$$

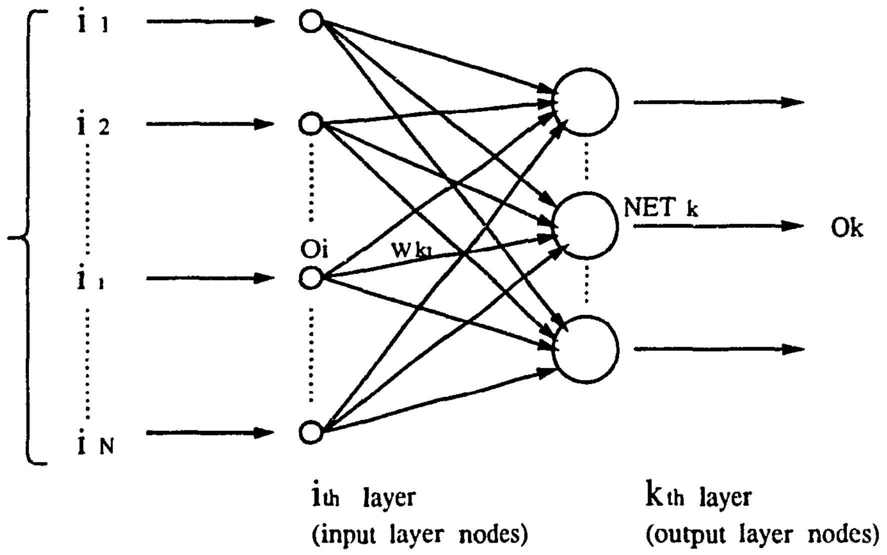
to obtain the column weight vector $\underline{w}$
where the elements of $\underline{b}$ are the desired outputs of the network. In the binary classification problem, the values of the elements in $\underline{b}$ will be $+1$ if $\underline{x}$ belongs to class c_1, and -1 if $\underline{x}$ belongs to class $\sim c_1$. The weights of the network are adjusted according to the Perceptron-Learning Algorithm.

The linear multi-class discriminant is thus determined analytically, but found to fail in the case of a nonlinear distribution of training patterns. A linear Perceptron with no internal layer is incapable of solving the Exclusive-OR, or parity problem and cannot also classify the cluster of c_1 and $\sim c_1$ patterns shown in Fig. 3.8. This is a major drawback in pattern recognition examples since the patterns are usually distributed in a nonlinearly separable pattern space. This is also true in the case of biological-vision. Nonlinear discriminants are provided by introducing a nonlinear activation function at the output of each node. Several learning algorithms have evolved around this concept, and are found to be very useful in case of networks used for pattern recognition.



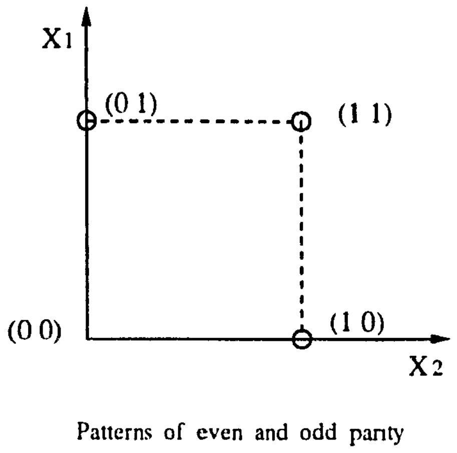
$$\underline{x}^t = b \begin{cases} +1 & if x \in class\, C \\ -1 & if x \notin class\, C \end{cases}$$

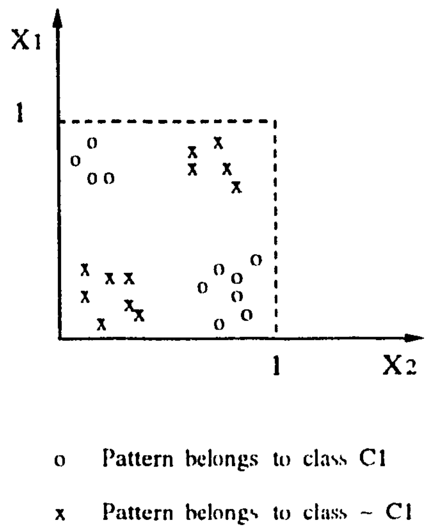
Figure 3.6: *Schematic representation of a linear discriminant - the Perceptron.*



NET_k = input to the k_{th} node in output layer = $\sum W_{ki} O_i$

Figure 3.7: *Schematic representation of a neural network as a multi-output linear discriminant.*




X1
1
1
X2
o Pattern belongs to class C1
x Pattern belongs to class ~ C1

(a) (b)

Figure 3.8: *Patterns that cannot be classified by a linear Perceptron (a) an even and odd parity or Exclusive-OR pattern, and (b) a linearly non-separable cluster.*

Chapter 4

Functional approximation using Backpropagation Neural Networks

4.1 Mapping Neural Networks

The information processing operation that mapping neural networks are intended to carry out [29] is the approximation of a bounded mapping or function $f : A \subset R^n \mapsto R^m$, i.e., from a compact subset A of an n-dimensional Euclidean space to a bounded subset $f[A]$ of an m-dimensional Euclidean space, by means of training using sample vectors $(\underline{x}_1, \underline{y}_1), (\underline{x}_2, \underline{y}_2) \cdots (\underline{x}_k, \underline{y}_k)$, where $\underline{y}_k = \underline{f}(\underline{x}_k)$. The mapping $\underline{f}$ is generated by selecting the vectors $\underline{x}_k$ randomly from A in accordance with a fixed probability density function $\rho(\underline{x})$, where $\rho = 0$ outside A. During the recall phase, input vectors $\underline{x}_j$ are also randomly selected in accordance with $\rho(\underline{x})$. There are generally two categories of mapping networks: *feature-based* networks, and *prototype-based* networks.

4.1.1 Functional Approximation – the Mapping Perspective

The action of a neural net may be viewed principally as a mapping through which points in the input pattern space are transformed into corresponding points in an output space on the basis of designated attribute values, like class membership [30]. This may also be viewed as the functional approximation of the input pattern space into the designated class membership based upon significant features of the input patterns.

The input space is a N-dimensional Euclidean space, and is well mapped. The Euclidean distance is small for patterns close together, and large for others. The neural network transforms this into the output space based upon class membership index values, i.e., in accordance with the values of the designated attributes in the input space. The dimensionality of the output space is also generally different from that of the input space, and the patterns are arranged in terms of their class membership, rather than in terms of their positions in the original input space.

4.1.2 Approximation Accuracy

Since this mapping is a functional approximation, based upon the designated attributes of the input patterns, an appropriate measure of the approximation accuracy has to be determined. This helps avoid error in the output space due to overfitting, or underfitting of the training set data.

To test the approximation accuracy of the network, we compare the actual output vector of the network $\underline{y}(\underline{x}, \underline{w})$ with the desired theoretical value $\underline{f}(\underline{x})$ for an input vector $\underline{x}$, where $\underline{w}$ is the weight vector of the network. The input vector $\underline{x}$ is always chosen with a fixed probability density function $\rho(\underline{x})$. This is done over a large number of testing trials, consisting of randomly selected examples $(\underline{x}_1, \underline{y}_1), (\underline{x}_2, \underline{y}_2), \cdots, (\underline{x}_k, \underline{y}_k), \cdots$, which constitute a test set. Thus, for the k^{th} testing trial, done randomly with $\underline{x}_k$ chosen with some fixed probability density function ρ, the output $\underline{y}_k = \underline{f}(\underline{x}_k)$. The square of the error is given by:

$$F_k(\underline{x}_k, \underline{w}) = |\underline{f}(\underline{x}_k) - \underline{y}(\underline{x}_k, \underline{w})|^2 \tag{4.1}$$

It is assumed that the $\underline{w}$ is fixed during this process, i.e., the network is not being trained and weights do not adapt during the testing phase. Assuming that the limit exists for almost any set of randomly chosen $\underline{x}_k$, the *mean squared error* $F(\underline{w})$ of the network is defined [29] to be:

$$F(\underline{w}) \equiv \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} F_k(\underline{x}_k, \underline{w}) \tag{4.2}$$

4.1.3 The Error Surface

The mean squared error function $F(\underline{w}) \geq 0$, because F is the average of non-negative quantities. It is usually well defined for most neural networks, like the Backpropagation network, and is a function of the weight vector $\underline{w}$ of the network being evaluated. For each

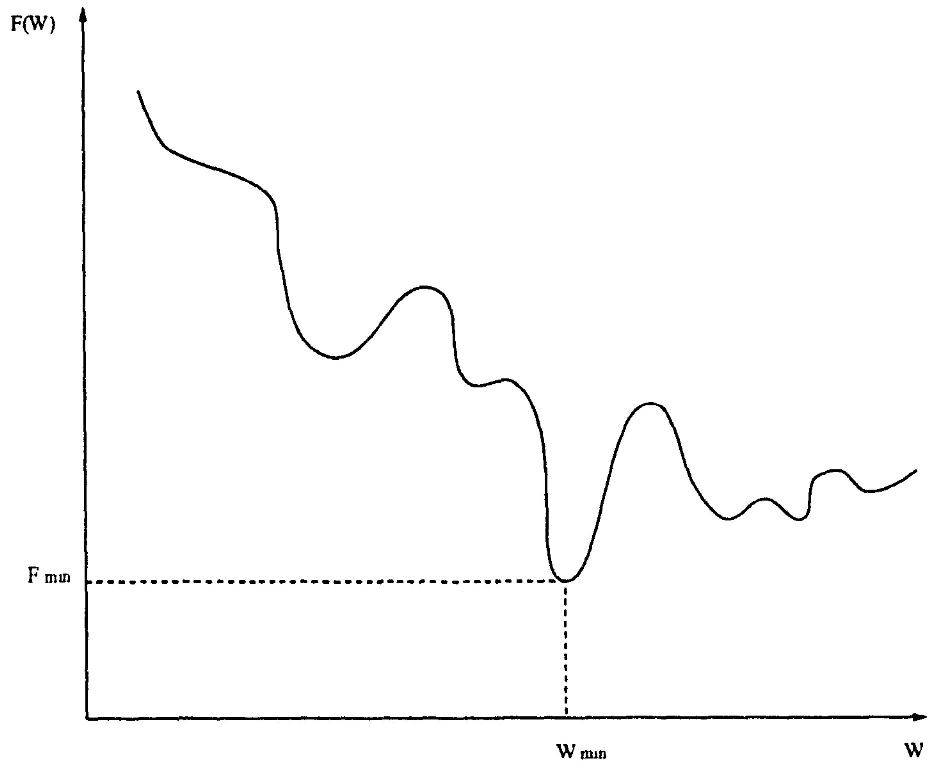


Figure 4.1: *A Typical Neural Network Error Surface. The goal of network training is to find the training vector W_{min} that minimizes the error F of the network, shown by F_{min} above. Note that in this case W consists of a single element.*

selection of weights, a different mean squared error arises. Thus, it can also be regarded as a surface, known as the *error surface* [29], sitting above the weight space of the network with height F above the surface at weight value $\underline{w}$. Since F is a non-negative function, the error surface also lies at a non-negative altitude above the weight space. Figure 4.1 shows a typical error surface.

Training a neural network involves finding the set of weights $\underline{w}_{min}$ that minimizes F. Typically $F_{min} > 0$, since the mapping done by the neural network is an approximation, and not an exact implementation of the desired mapping. The structure of the error surface is also crucial in training the network, specially the Backpropagation neural network. This is because during training the network gets trapped in local minima occurring on the error surface, rather than reaching the global minimum, which is the desired goal of all network training algorithms.

4.2 The Backpropagation Neural Network

A typical Backpropagation network Fig. 4.2, consists of an input layer, an output layer, and at least one hidden layer. Although there is no limit on the number of hidden layers, typically a network has one or two such layers. Each layer is fully connected to the succeeding layer. The arrows indicate flow of information during recall. During learning, information is also propagated back through the network and is used to upgrade the connection weights.

Input vectors are introduced to the network through the input layer. The Processing Elements (PEs) in this layer have no processing power of their own, and serve only as a fan-out to other processing elements in the next layer. The data flows along the connections towards the hidden and the output layers. Each hidden layer PE transforms the incoming data by executing the equations specified in Section 4.2.1. The transformed data is then output to the next layer. Each output layer PE performs a similar transformation on the data received from the last hidden layer, and from the input layer. The final result

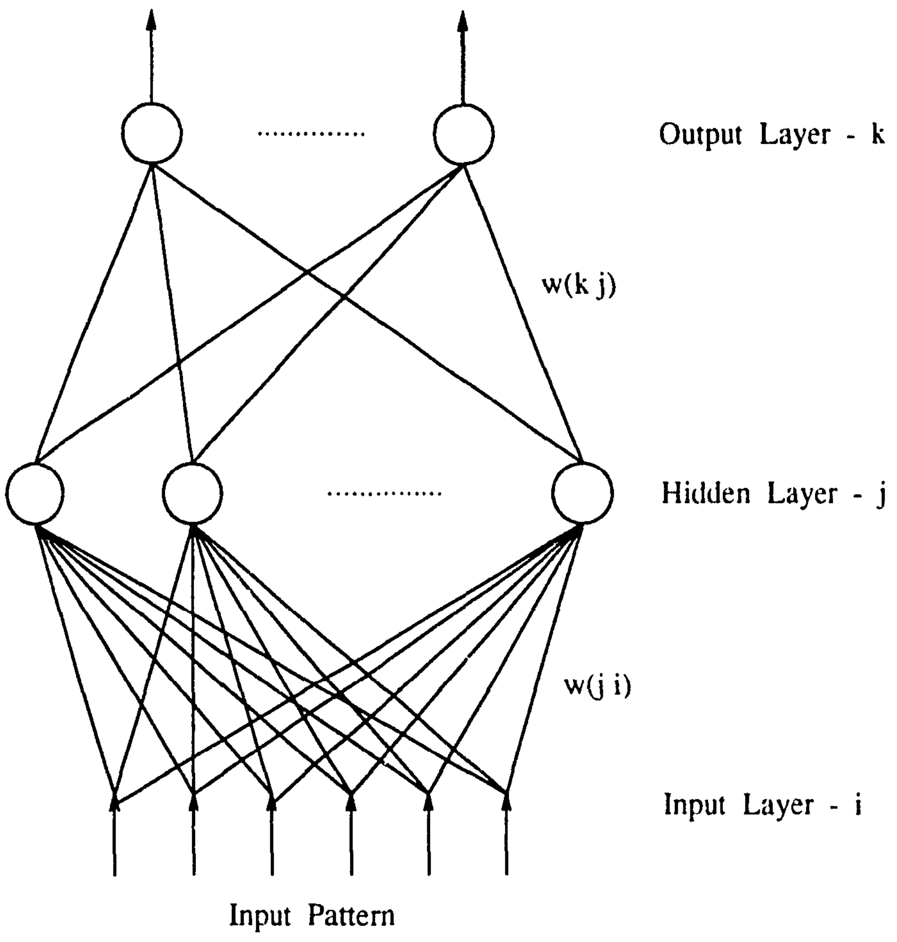


Figure 4.2: *Backpropagation Neural Network with Single Hidden Layer*

is that each input vector is transformed into some corresponding output vector, based upon the mapping function being approximated by the network. During the learning phase, the weights of all the hidden and output layer PEs are systematically upgraded at a rate dependent upon the magnitude of the error signals calculated as the difference between the actual and the desired outputs. This process is repeated until the mapping has been learned by the network to the desired level of accuracy, or until it appears that the network has learned as well as it can.

For a given set of training data, a particular set of weight values will result in some degree of mapping accuracy. The main idea of training the network is to find a set of weight values that result in maximum accuracy and minimum error. This is achieved by using the Mean Squared Error (MSE) criterion for estimating the error signal for training the network. For a given set of input/training pairs, the MSE is the average over all pairs of the squared difference between the desired output and the actual output.

The network weight vector is defined as the vector made up of all the weights of all hidden and output layer PEs concatenated together. If this vector contains N weights, it can be thought of as representing a point in N-dimensional space. Given a set of input/output data vectors, a value of MSE can be calculated for all possible network weight vectors, by successfully training the network. This function maps an N-dimensional network weight vector into an MSE value, defined as a surface in $N+1$-dimensional space [31]. This surface is the *error surface* of the network. The objective is to find the lowest point on the error surface, as explained in Section 4.1.3, known as the *global minimum.* During training, the weights are adjusted so as to move down this error surface in the steepest direction, referred to as *gradient* or *steepest descent.* The position of the weight vector will move downhill on the surface until it reaches the bottom of a valley, called a *local minima.* The weight vector represented by this point is locally optimal, in the sense that the network has a lower mapping error over the training set with these set of weights than with any other adjacent set of weight vectors. Global optimization involves finding the global minimum on the error surface.

A common problem encountered during network training is that the error surface may have multiple local minima with different error values, only one of which may be the global minimum. There are two possible ways of dealing with this problem. The first is to run the network several times from different random starting weight vectors. The second is to jog the weights randomly during training by an appropriate small amount, so as to change the position of the weight vector on the error surface. This causes the network to escape being trapped in local minima on the error surface. A typical network has multiple weights, and therefore, many possible directions to move. The possibility of moving in a down-hill path is thus high, and thus the probability of encountering local minima is small.

The step size which determines the rate and extent of gradient descent is a crucial parameter in adjusting the weight vector of a network during training. It determines how far the weight vector moves down-hill on the error surface, and is determined by two factors:

- The size of the gradient vector. – In addition to a direction, the gradient vector has a magnitude which determines the slope of descent. The larger the magnitude, the steeper the descent.

- The learning rate. – This is usually controlled by the user during training as a software parameter called the *training coefficient*, β in eqn. 4.7. It has a value between 0 and 1, and is multiplied by the gradient vector magnitude to control the step-size.

There is a tradeoff between the learning rate and the training time. With a very low learning rate the path of descent is very smooth, almost continuous [31], but requires many steps or training iterations. Time taken for training is therefore higher. With a higher learning rate, the steps are larger and the time to reach a local minimum smaller, but it is possible to miss the global minimum if the steps are too large. Therefore, the learning rate has to be chosen judiciously by the user to successfully train the network.

4.2.1 The Backpropagation Learning Rules

To avoid confusion from one layer to the next, the following notation is used here for describing the learning rules. A superscript in square brackets is used to indicate which layer is being considered:

${y_j}^{[s]}$	*current output of j^{th} neuron in layer s.*
${w_{ji}}^{[s]}$	*weight on connection joining i^{th} neuron in layer $s-1$ to j^{th} neuron in layer s.*
${Net_j}^{[s]}$	*weighted summation of inputs to j^{th} neuron in layer s.*

The output of each Processing Element in the network is given by:

$$\begin{aligned} {y_j}^{[s]} &= f\left(\sum_i \left({w_{ji}}^{[s]}\, {y_i}^{[s-1]}\right)\right) \\ &= f\left({Net_j}^{[s]}\right) \qquad (4.3) \end{aligned}$$

where f is the activation function of the network, discussed in Section 2.2.1. Traditionally, it is a sigmoid function, but many other variations are used in practical training algorithms.

4.2.2 Backpropagating the Local Error

The network is assumed to have some global error function E associated with it which is a differentiable function of all the connection weights in the network. The error present at the j^{th} neuron in layer s is given by [32] :

$$ {\delta_j}^{[s]} = -\partial E / \partial {Net_j}^{[s]} \qquad (4.4) $$

$$ = f'({Net_j}^{[s]}) \sum_k \left({\delta_k}^{[s+1]}\, {w_{kj}}^{[s+1]}\right) \qquad (4.5) $$

If f in eqn. (4.3) is a sigmoid function, $f'(z) = f(z)\,(1 - f(z))$, and (eqn 4.5) is given by:

$$\delta_j{}^{[s]} = y_j{}^{[s]}\,(1 - y_j{}^{[s]}) \sum_k \left(\delta_k{}^{[s+1]}\, w_{kj}{}^{[s+1]}\right) \tag{4.6}$$

This gives a measure of the error present at the output of each node, which is then propagated back by the network and is used to update weights during the training mode.

4.2.3 Minimizing the Global Error

Given the knowledge of the error at each PE, the aim of the learning process is to minimize the global error E of the system. This is done by using a gradient descent rule:

$$\Delta w_{ji}{}^{[s]} = -\beta\,(\partial E/\partial w_{ji}{}^{[s]}) \tag{4.7}$$

where β is a learning coefficient. The weights are changed according to the size and direction of the negative gradient on the error surface. The partial derivatives are calculated directly from the local error values at each node. Thus:

$$\begin{aligned} \partial E/\partial w_{ji}{}^{[s]} &= \partial E/\partial Net_j{}^{[s]}\;\partial Net_j{}^{[s]}/\partial w_{ji}{}^{[s]} \\ &= -\delta_j{}^{[s]}\, y_i{}^{[s-1]} \end{aligned} \tag{4.8}$$

From eqns. (4.7) and (4.8) we have

$$\Delta w_{ji}{}^{[s]} = \beta\, \delta_j^{[s]}\, y_i^{[s-1]} \tag{4.9}$$

This gives a measure of the slope of the gradient on the error surface and also indicates the amount by which the weights linking nodes i and j should be changed during training.

4.2.4 Network Architecture

The macro-scale detail of the backpropagation architecture as discussed by Hecht-Nielsen [33] is shown in Fig. 4.3. In, general the architecture consists of K rows or layers of processing elements, numbered from the bottom up beginning with 1. The first layer consists of n fanout processing elements that simply accept the individual components $\underline{x}_i$ of the input vector $\underline{x}$ and distribute them, without modification, to all the unit of the second layer. Each unit in each layer receives the output signal of each of the units of the layer below. This continues through all the layers of the network until the final, K^{th} layer, which consists of m units and produces the network's estimate $\underline{y}'$ of the correct output vector $\underline{y}$. Besides the forward connection, each unit of each hidden layer receives an error feedback connection from each of the units above it, which are not merely fanned out copies of a broadcast output (like the forward connections), but are each separate connections, carrying separate error signals.

Details of each individual units are shown in Fig. 4.4. Each unit consists of a single sun processing element and several planet processing elements. Each planet produces an output signal that is distributed to both its sun and the sun of the previous layer that supplied input to it. Each planet receives input from one of the suns of the previous layer as well as its own sun. The output row suns receive the desired value of the output $\underline{y}_i$ for their component of the output vector on each training trial. The network functions in two stages: a forward pass, and a backward pass. A network scheduling processing element (not shown in the figure) controls the operation of each individual PE in the network. At any instant, depending upon the status of the network (i.e., whether the network is in the training or recall state), it controls the operation being performed by each individual PE.

The scheduling of the network's operation consists of two sweeps through the network. The first sweep or forward pass starts by inserting the vector $\underline{x}_k$ into the network's first layer. The processing elements then transmit all the components of $\underline{x}_k$ to all of the units of the second layer of the network, which are transmitted to all the units of the next layer and so on, until finally the m output units emit the components of the $\underline{y}'_k$ (the network's

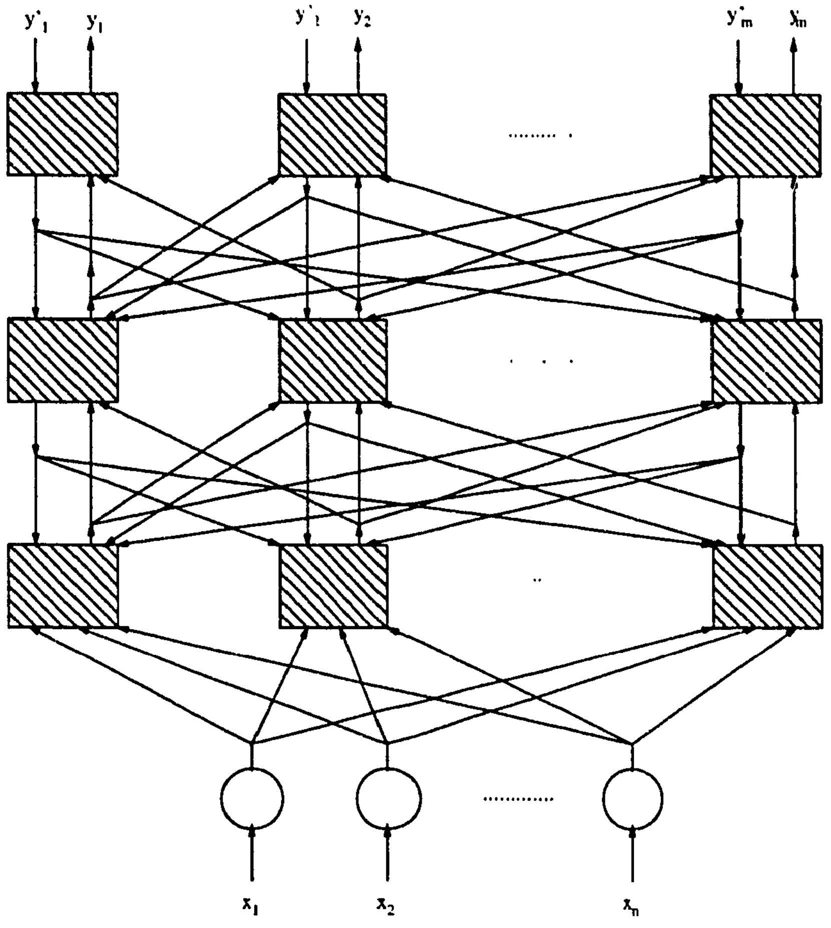


Figure 4.3: *Macroscopic Architecture of the Backpropagation Neural Network. Each shaded box denotes an architectural unit, details of which are shown in Fig.4.4.*

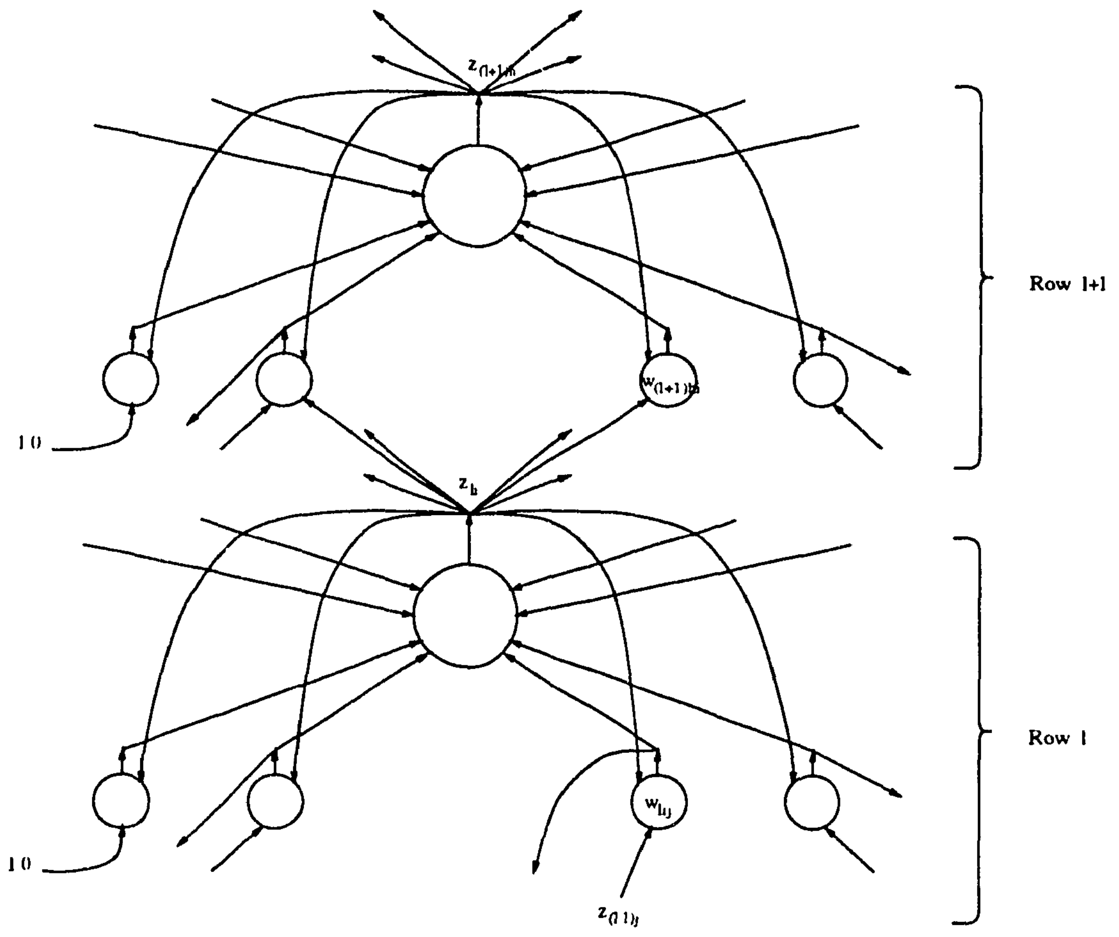


Figure 4.4: *Architecture details of each Processing Unit of the Back-propagation Neural Network. The interaction of the units in adjacent rows (l below, and l+1 above) are shown. Each unit consists of a sun and several planets. Each hidden layer sends its output to one planet of each unit in the next layer. All bias planets receive a constant input* $z_{(l-1)0} \equiv 1.0$ *from a single processing element not shown here. Each planet receives an input from the previous layer and sends back a connection carrying the backpropagated error to the sun that supplied that input.*

estimate of the desired output $\underline{y}_k$). After the estimate $\underline{y}'_k$ is emitted, each of the output units is supplied with it's component of the correct output vector $\underline{y}_k$, starting the second or backward pass through the network. The output suns compute their δK_is and transmit these to their planets. The planets then update their ΔK_{ij} values and then transmit the values $\underline{w}^{old}_{K_{ij}} \delta K_i$ to the suns of the previous row. This process continues until the planets of layer 2, i.e. the first hidden later have been updated, and the cycle is then repeated, each cycle in effect consisting of the inputs to the network "bubbling up" from the bottom to the top and the errors "percolating down" from the top to the bottom [33].

4.2.5 Backpropagation Error Surface

Given a function f, an associated x-section probability density function ρ, and an associated backpropagation architecture intended to approximate f, then a means of measuring the accuracy of this approximation as a function of the network's weights can be defined along with an error surface. Let $\underline{w}$ be the weight vector of the network, consisting of components of the weights of all the planets of the network, starting with the weight of the first planet of the first processing element of hidden layer 1 and ending with the weight of the last planet of the last processing element of the output layer K. For simplification, the components of $\underline{w}$ are referred to as $\underline{w}_1, \underline{w}_2 \cdots$; rather than $\underline{w}_{210}, \underline{w}_{211} \cdots$ and the components of B (the network's estimate $\underline{y}'$ of the correct output $\underline{y}$) as $z_{k1}, z_{k2} \cdots z_{km}$. Thus, one can write $B(\underline{x}, \underline{w})$ and $(\underline{x}_k, \underline{y}_k)$ the example used on the k_{th} testing trial i.e., $\underline{y}_k = f(\underline{x}_k)$, where the $\underline{x}_k$s are drawn from A in accordance with a fixed probability density function ρ.

Let the mean squared error on the k_{th} trial be $F_k = \mid f(\underline{x}_k) - B(\underline{x}_k, \underline{w}) \mid^2$, where Assuming $\underline{w}$ to be fixed, i.e., batch size is set to ∞ to shut off learning, we can define $F(\underline{w})$, the *mean squared error function* of the network to be:

$$F(\underline{w}) \equiv \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} F_k \tag{4.10}$$

The error surface is of a Backpropagation network is defined by the equation $F = F(\underline{w})$ in the $Q+1$-dimensional space of vectors $(\underline{w}, F)$, where Q is the number of dimensions in the vector $\underline{w}$, i.e., the number of planets in the network [33]. The variable $\underline{w}$ ranges over the Q-dimensional space R^Q and for each $\underline{w}$ a non-negative surface with height F is defined by $F(\underline{w})$. In other words, given any selection of weights $\underline{w}$, the network will make an average squared error $F(\underline{w})$ in its approximation of the function f.

The generalized delta rule used for learning in the Backpropagation network, has the property that given any starting point $\underline{w}_0$ on the error surface that is not a minimum, the learning law will modify the weight vector $\underline{w}$ so that $F(\underline{w})$ will decrease. The learning law uses examples provided during training to decide how to modify the weight vector so that the network approximates f with a lesser error. Three basic facts are known about the backpropagation error surface [33]:

- The Backpropagation error surfaces have extensive flat areas and troughs that have very little slope. In these areas it is necessary to move the weight value a considerable distance before a significant drop in error occurs. This usually effects training time, as it is difficult to determine which way to move the weights during training when the slope is very shallow.

- There exist local minimas in which the network gets trapped occasionally. A lot depends on the initial conditions, which governs the shape of the error surface and the energy content of the network, prior to training.

- The Backpropagation error surface has many global minima. This is due to the fact that for each set of weights, there are many weight permutations that yield exactly the same network input/output function.

4.3 Functional Approximation using Multilayer Backpropagation Network

Clear insight into the versatility of the Backpropagation Neural Networks for use in functional approximation came with the discovery that the classic mathematical result of Kolmogrov was actually a statement that for any continuous mapping $f : [0,1]^n \subset R^n \mapsto R^m$, there exists a three-layer neural network that approximates this mapping exactly [33]. Kolmogorov's Mapping Neural Network Existence Theorem [33] states that *given any continuous function $f : [0,1]^n \mapsto R^m$, and a mapping $y = f(\underline{x})$, f can be implemented exactly by a three-layer feed-forward neural network having n fanout processing elements in the first (x-input) layer, (2n+1) processing elements in the middle layer, and m processing elements in the top (y-output) layer.* The function f belongs to L_2 if each of f's coordinate functions is square-integrable on the unit cube. For functions of this class it was shown by Hecht-Nielsen [33] that:

Given any $\epsilon > 0$ and any L_2 function $f : [0,1]^n \subset R^n \mapsto R^m$, there exists a three-layer backpropagation neural network that can approximate f to within ϵ mean squared error accuracy.

Although this theorem proves that three layers are always enough in solving real world problems, it is often essential to have four, five, or even more layers (it is often argued that beyond 5 layers, i.e. three hidden layers the performance does not increase noticeably). For many problems, an approximation with three layers would require an impractically large number of hidden units. Such networks are also not easily trained, whereas an adequate solution can be obtained with a tractable network size by using more than three layers. Furthermore, although the above theorem guarantees the ability of a multilayer network with the correct weights to accurately implement an arbitrary L_2 function, it does not comment on whether or not these weights can be learned using any existing learning law.

Chapter 5

Application of Neural Networks for Depth Perception

5.1 Neural Networks Approach to Depth Perception

In Chapter 1 we examined the problem of determining distance based on the degree of blurring of an object in an image. Blurring was studied with respect to the point spread function, and different approaches for the determination of depth as a function of distance were examined. Fourier analysis is a useful tool for effective data compression in image analysis and feature extraction. Neural network applications for pattern recognition and functional approximation were discussed in Chapters 3 and 4 respectively. In this chapter we present the main contribution of this thesis, which consists of a neural network based approach for the determination of depth as a function of blurring for use in VLSI wafer probing.

It can be seen from Fig. 1.2 and Fig. 5.10 that there exists a smooth relationship between the degree of blur and the distance of a probe from a test pad on a VLSI chip. Therefore, by measuring the amount of blurring, the distance from contact can be estimated. The determination of the amount of blur present in an image was studied with respect to the point spread function in Chapter 1. In this chapter, the effect of blurring on a point-object is studied in the frequency domain, and a monotonic relationship is found between the degree of blurring of an object and the frequency content of the image (Section 5.3.1). Fourier feature extraction, with its inherent property of shift and rotation invariance (Section 1.5.4), is utilized to extract significant feature vectors. These vectors contain information on the degree of blurring, and hence the distance from the probe. Neural networks are then employed to map these feature vectors onto the actual distances. Various different approaches for obtaining the appropriate training set for this mapping are discussed, and the most suitable approach proposed. The experimental methodology is shown in Fig. 5.7, and discussed in detail in Section 5.5.

5.1.1 Why Neural Networks

Significant work was done by Dantu [4] on the problem of the determination of distance-to-touch in VLSI wafer probing. It was shown by Dantu that there exists a continuous mapping between the blur space and the distance space, with respect to the amount of blur present in a probe-tip as a function of distance from the pad surface in the VLSI probing operation [34]. Furthermore, it was seen in section 4.3 that given any continuous function $f : [0,1]^n \mapsto R^m$, and a mapping $y = f(\underline{x})$, f can be implemented exactly by a three-layer feed-forward neural network [33]. Thus, it is assumed that by employing a suitable three-layer neural network, the mapping between the blur space and the distance space can be effectively approximated.

5.1.2 Advantages of using Neural Networks

The main advantages of using neural networks are the following:

- Due to their massively parallel structure, neural networks, are inherently robust and highly fault-tolerant. In particular they are insensitive to slight variations in operating conditions.
- A fully trained Backpropagation neural network provides a continuous functional approximation of a bounded mapping. Thus, during the recall mode distances can be found in a continuous fashion anywhere within the range of the bounded mapping.
- The mapping itself is independent of the operational details of the process involved. Thus, details of the actual mathematical models for the mapping between the blur space and the distance space are not required. All that is required for successfully training the network are the actual and target vectors.

All these advantages make neural networks very suitable for practical applications.

5.1.3 Choice of Network Type

It was shown in Section 4.3 that given any $\epsilon > 0$ and any L_2 function $f : [0,1]^n \subset R^n \mapsto R^m$, there exists a three-layer backpropagation neural network that can approximate f to within ϵ mean squared error accuracy [33]. Backpropagation is thus a powerful mapping tool. The network chosen for our application is a multi-layer backpropagation network, with the number of inputs equal to the size of the Fourier feature vector extracted in Section 5.5.3, and one output element. The network chosen has two hidden layers in order to make the network trainable and to avoid having too many PEs as would be in the case with having only one hidden layer.

5.2 Training the Network to Map Feature Vectors onto Distance-to-Contact

The information present in the Feature vectors is sufficient for training a suitable Backpropagation neural network. In doing so, the network approximates the bounded mapping from the blur space on to the distance-to-contact space, (Fig. 5.2).

In Section 4.1, we saw that the information processing operation that backpropagation networks are intended to carry out, is the approximation of a bounded mapping or function $f : A \subset R^n \mapsto R^m$, from a compact subset A of n-dimensional Euclidean space to a bounded subset $f[A]$ of m-dimensional Euclidean space, by means of training on samples $(\underline{x}_1, \underline{y}_1), (\underline{x}_2, \underline{y}_2) \cdots (\underline{x}_k, \underline{y}_k)$, of the mapping, where $\underline{y}_k = f(\underline{x}_k)$. In our case, the compact subset A consists of the normalized Fourier feature vectors derived from each image, cf. Section 5.5.3, and $f : A \subset R^n \mapsto R^m$ maps these onto the corresponding subset $f[A]$ on the distance-to-contact space. The challenge then is to find and successfully train a suitable Backpropagation neural network to approximate this mapping to within ϵ mean squared error accuracy, where the value of ϵ has to be minimized as much as possible without overtraining, and without getting trapped in local minima on the error-surface of the network, (Section 4.2). Various approaches were tried to implement this mapping by using multi-layer backpropagation networks as discussed in the following subsections.

5.2.1 Training With Raw Images

At first, the network was trained directly with raw images of the probe-tip with various levels of defocus. The size of each image was reduced to 100 x 100 pixels, to include only the tips of the probe. But, due to the large size of input vectors (10,000 data points in this case) the network size was very large. It was also extremely difficult to successfully train such a network. In the case when training was possible for such large networks, it was unable to recall correctly the values of distance-to-contact for unseen images. Thus the network was unable to correctly approximate the desired continuous mapping. This approach was thus discontinued, and some form of data compression required to effectively reduce the size of training vectors. This lead us to investigate the properties of Fourier transforms for feature extraction, as a means of obtaining feature vectors for training the networks.

5.2.2 Training With Averaged Fourier Spectrum of Images

Fast Fourier Transforms (FFT) were performed on each image to obtain the corresponding frequency contents. In order to keep the total number of data points in the Fourier spectrum small, FFTs were performed on small (10 x 10) overlapping windows moved over the whole image. The magnitude of each individual data point in the window was summed up to obtain the averaged spectrum of the whole image. This technique was not very suitable because averaging destroyed the information about the amount of blur present in each image. The network was thus unable to map this information on to distance-to-contact. This approach was thus abandoned.

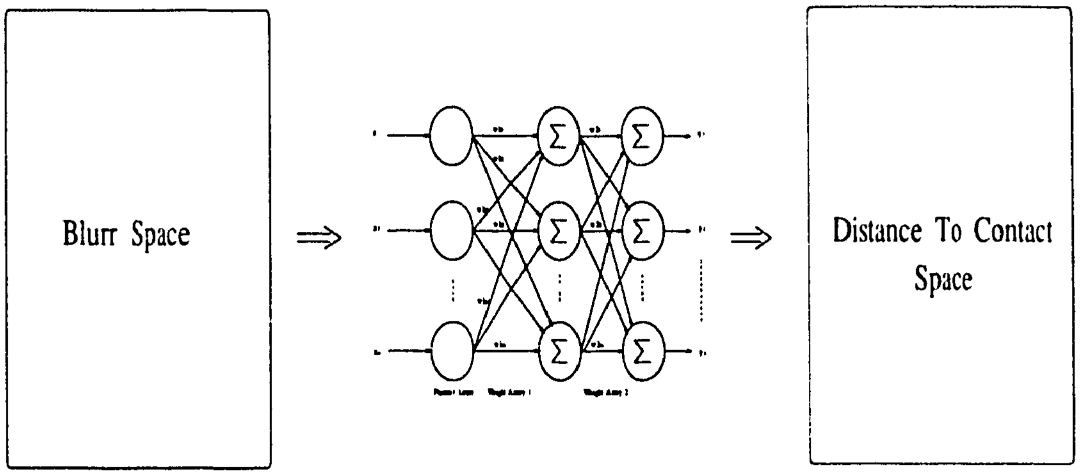


Figure 5.1: *Proposed approach for mapping Blur-space on to Distance-to-contact space by using multi-layer Backpropagation neural networks.*

5.2.3 Training with Blur Extracted by Deconvolution with respect to Most Focused Image

In this approach the amount of blur was extracted by deconvoluting each image with the most focused image in each set. Since convolution is multiplicative in the frequency domain, this was done by subtracting the logarithm of the magnitude spectrum of each image with that of the most focused. This approach was also not suitable because the amount of blur (and hence distance-to-contact) were measured relative to the most focused image in the set, and not on absolute terms. Furthermore, the training set for the network was still large and no monotonic relationship seen in the extracted information with respect to the amount of blur present. This approach was also thus abandoned.

5.2.4 Training With Feature Vectors Extracted from Fourier Spectrum.

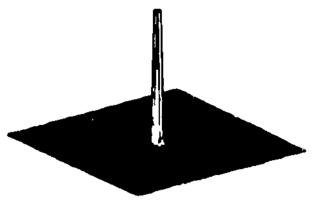
FFTs were then performed on the full image with a view to extract significant feature vectors with adequate information about the amount of blurring present in each image. Details are discussed in Section 5.5.

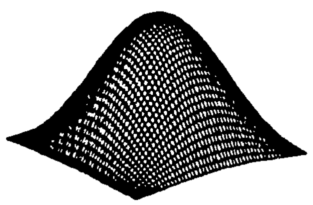
5.3 The Proposed Approach

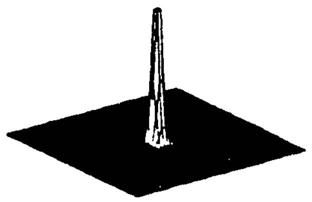
5.3.1 Effect of Variation of the Point Spread Function in Frequency Domain

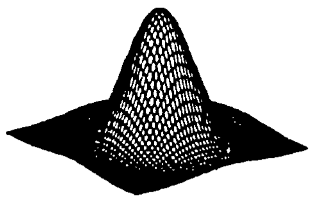
In this section, the effect of variation of the PSF is experimentally studied in the frequency domain. We start by studying the changes in the frequency content of an image with variations in the size of the point object. Extensive simulations were carried out to map these changes in size to changes in the frequency spectrum.

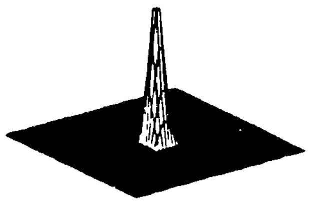
In Fig.5.2, the point object is replaced by a 2-D Gaussian. Based on the discussion in Section 5.3.1, the variations in the spread σ of the Gaussian is interpreted as the variation

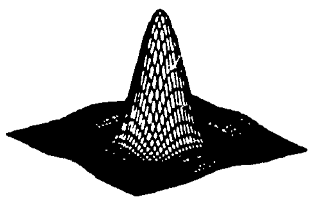

(a) spatial distribution - 3x3

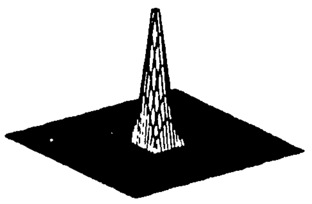

(b) frequency distribution - 3x3


(c) spatial distribution - 5x5


(d) frequency distribution - 5x5


(e) spatial distribution - 7x7


(f) frequency distribution 7x7


(g) spatial distribution - 9x9

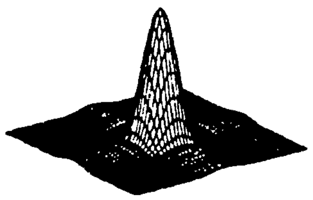

(h) frequency distribution - 9x9

Figure 5.2: *The change in frequency content of an image with changes in the Point Spread Function.* (a), (c), (e) & (g) *represent the PSF in the space domain, modeled as two dimensional Gaussians, with increasing values of σ. As σ of the PSF increases, due to increase in blurring, the spread in the frequency domain decreases, as shown in* (b), (d), (f) & (h)

in the PSF of a point object. The larger the spread of the Gaussian in the space domain, the more defocused is the object. Figure 5.2 (a), (c), (e) & (g) thus represent different levels of defocus of the formed image of a point object, by the defocusing function of the lens and camera system. As the image gets progressively more blurred (in the order Fig.5.2 (a) (c), (e), (g)), the changes are reflected in the corresponding frequency contents (Fig.5.2 (b), (d), (f), (h)). It is observed that:

- the spread of the Gaussian formed by the low frequency elements in the central part of the frequency spectrum is inversely proportional to the amount of spread of the Gaussian in the space domain. Thus, if the object is in focus or very close to it, as in Fig.5.2 (a), the corresponding spread of the Gaussian in the frequency domain is very high. The central lobe in this case covers the entire span of the frequency spectrum. In other words, the slope formed by the lower frequency components in the central part of the spectrum is low, (Fig.5.2 (b)), compared to that of a highly defocused object, as in (Fig.5.2 (h)).

- the side lobes in the frequency spectrum are suppressed, and their magnitudes are very small compared to those of the central frequency elements in the main lobe.

PSF in Frequency – Spread inversely proportional to the spread in Space

The mathematical relationship between the spread of the PSF modeled as a 2-D Gaussian in space, and the corresponding spread of the Gaussian formed in the frequency domain, is derived in this section. For simplicity, the analysis is done initially in one dimension, but will then be extended to two dimensions.

The Gaussian, in 1-D, can be represented by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{5.1}$$

Transforming to the frequency domain by taking Fourier transforms, we get

$$F(\omega) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-j\omega x} e^{\frac{-(x-\mu)^2}{2\sigma^2}} dx \quad (5.2)$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\left(\frac{2j\omega x\sigma^2+(x-\mu)^2}{2\sigma^2}\right)} dx \quad (5.3)$$

$$= \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-\left(\omega^2\sigma^4-2j\omega\mu\sigma^2\right)}{2\sigma^2}} \int_{-\infty}^{\infty} e^{\frac{-\left(x-(\mu+j\omega\sigma^2)\right)^2}{2\sigma^2}} dx \quad (5.4)$$

$$= e^{\frac{-\left(\omega^2\sigma^4-2j\omega\mu\sigma^2\right)}{2\sigma^2}} \quad (5.5)$$

$$= e^{\frac{\mu^2}{2\sigma^2}} \cdot e^{\frac{-\left(\omega-\frac{2\mu}{\sigma^2}\right)^2}{2\left(\frac{1}{\sigma^2}\right)}} \quad (5.6)$$

which implies that

$$\sigma_{freq.} \propto \frac{1}{\sigma_{space}} \quad (5.7)$$

Thus, extending this to 2-D, it can be seen that the spread of the Gaussian formed in the frequency domain is inversely proportional to that in the space domain. This can also be seen from Fig.5.2 The frequency spectrum (magnitudes) of a well focused object, i.e. of an object with a very small value of σ_{space} (Fig.5.2(a)) is seen to have a large value of σ_{freq}, (Fig.5.2(b)), and vice-versa.

This relationship holds for each point object in the image. The image itself is considered as a collection of individual point objects, or pixels. Each point object in turn contributes to the frequency content of the whole image. This relationship between the degree of blur, or the spread of the PSF, and the slope of the central lobe should thus hold for the whole image. This is observed to be true.

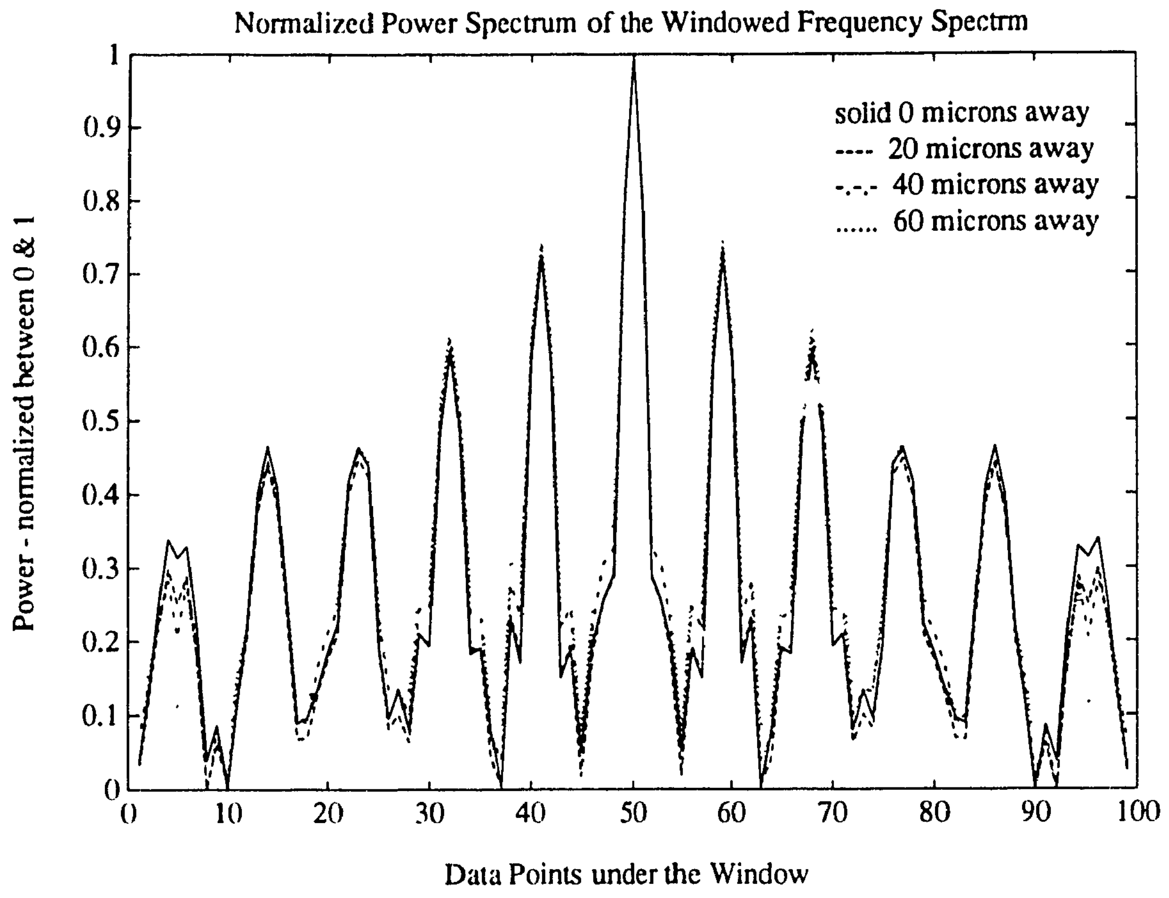


Figure 5.3: *The normalized low frequency components of the Fourier spectrum of each of the four images. Note that there is a monotonic relationship between the degree of focus of the image and the magnitudes of the plots.*

5.4 Experimental Set-up

In this section, we describe the experimental set-up used for determining the distances to contact of the probe tip in VLSI wafer probing. The wafer probing station used in our set-up is shown in Fig.5.4. The various components are described below:

5.4.1 Wafer Chuck and Probe Platform

The wafer chuck is a stable mobile platform with four degrees of freedom: linear motion in the X-Y-Z directions, and rotation about the Z-axis, (Figs.5.5,5.6). It is used for mounting the chip, or wafer, under test. The resolution of movement is 30 threads per cm.

The probe platform is a uniform, smooth surface for holding up to twelve micro-manipulators. Each micro-manipulator has a magnetic base, and three degrees of freedom (X-Y-Z directions). The X-Y control aligns the probe tips with the test pads, while the Z-control gradually lowers the probe onto the pad-surface. The probe tips are magnetically loaded to avoid scratching the pad-surface during the probing operation. There are also leads available for electrical contact.

5.4.2 Optical set-up – Microscope and Camera

A microscope is used to view the probe tip along the Z-direction. The probe tip can be viewed with different magnifications as desired. A camera is mounted onto the microscope to record these images. The sensor array is aligned such that it is possible to obtain a focussed image of the wafer on it using the focusing mechanism of the microscope. The camera used has a 760 x 585 element Charged Coupled Device (CCD) sensor array, with 256 grey levels.

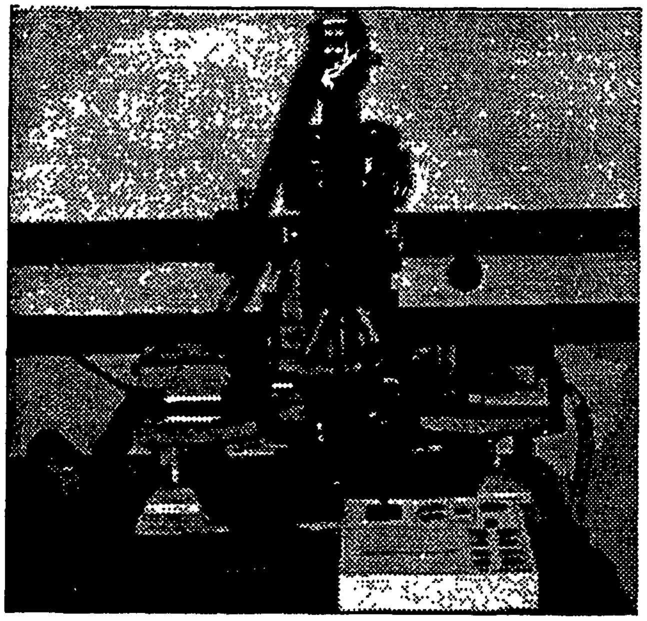


Figure 5.4: *Shows a VLSI Wafer-Probing Station with Camera mounted on the Microscope. This set-up is used for grabbing images of the Probe-tip as it is gradually lowered onto the test-pads of the chip under test.*

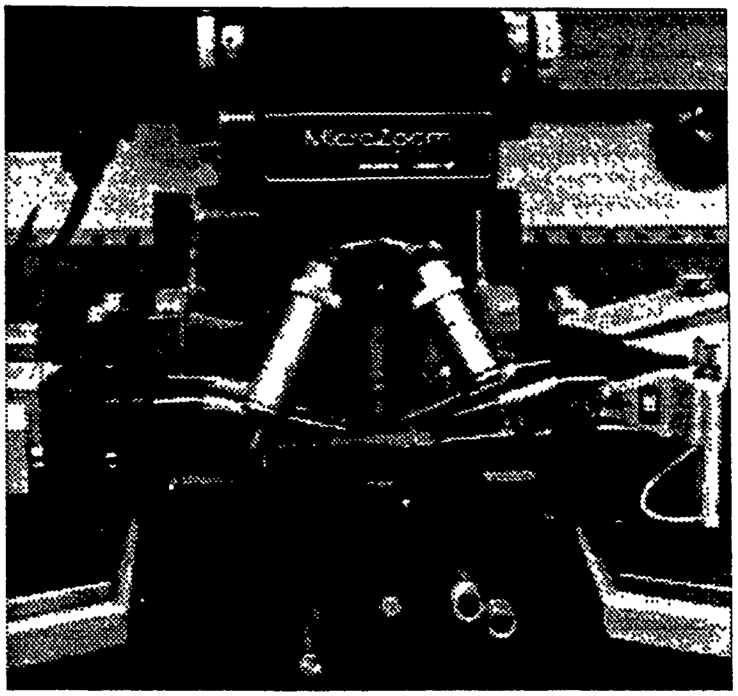


Figure 5.5: *The wafer chuck and probes being lowered onto chip.*

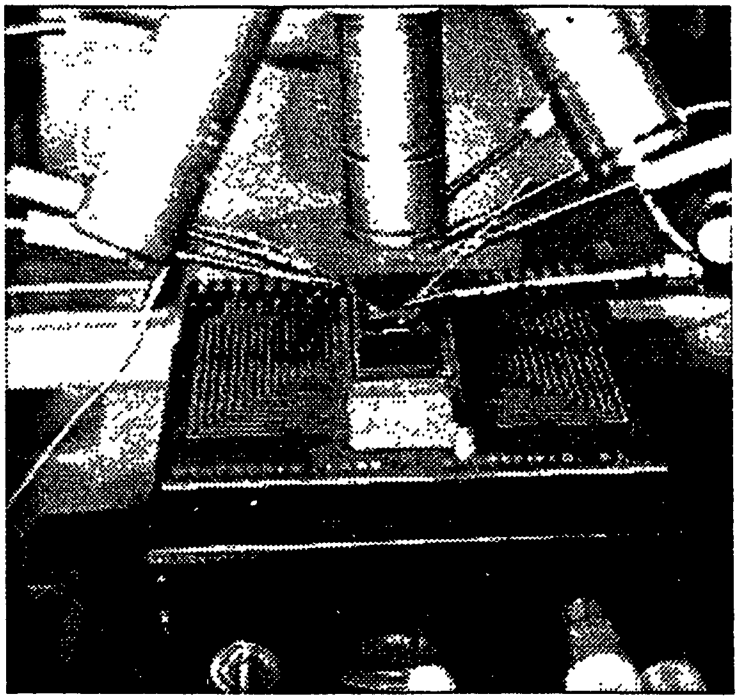


Figure 5.6: *The close-up of the wafer chuck and probes.*

5.5 Experimental Methodology to Determine Distance to Contact

In this section we describe the Experimental methodology of determining the *Distance To Contact* of the Probe tip. The experimental set-up is shown in Fig.5.7, and the various stages in the process are discussed in detail.

5.5.1 Image Grabbing

This stage consists of two steps, scanning and digitization. As shown in Fig.5.8, an optical stage which forms the image $I(x,y)$ usually precedes scanning and digitization. The complete transformation results in a digital image $I(i,j)$ from an analog image $L_1(x,y)$ or $L_2(x,y)$. A digital image $I(i,j)$ is defined as a function of two real discrete variables i and j whose value is referred to as a grey shade, tone, or level. The latter is a nonnegative number or value assigned to an element of the array, which is proportional to either $L_1(x,y)$ or $L_2(x,y)$ in a small area centered around (x,y) [35]. These are the discrete resolution cells of the analog image and are called "pixels". The overall operation thus results in spatial sampling of the image into pixels and quantization of the grey levels into the integer set $I = \{I_1, I_2, I_3, \cdots, I_n\}$ where n is the total number of grey levels.

The three most important parameters that must be considered in designing or evaluating a scanner are the spatial resolution, dynamic range, and digitization plus read-in time. The choice of particular values is usually application specific.

In our set-up we have used the OC-300 Video Digitizer which is connected to a Hitachi camera with a 760 x 485 pixels CCD Array sensor. The camera provides an analog RGB signal which is appropriately sampled and digitized by the digitizer. A block diagram showing the different hardware features of the digitizer is shown in Fig.5.9 and the details are discussed below:

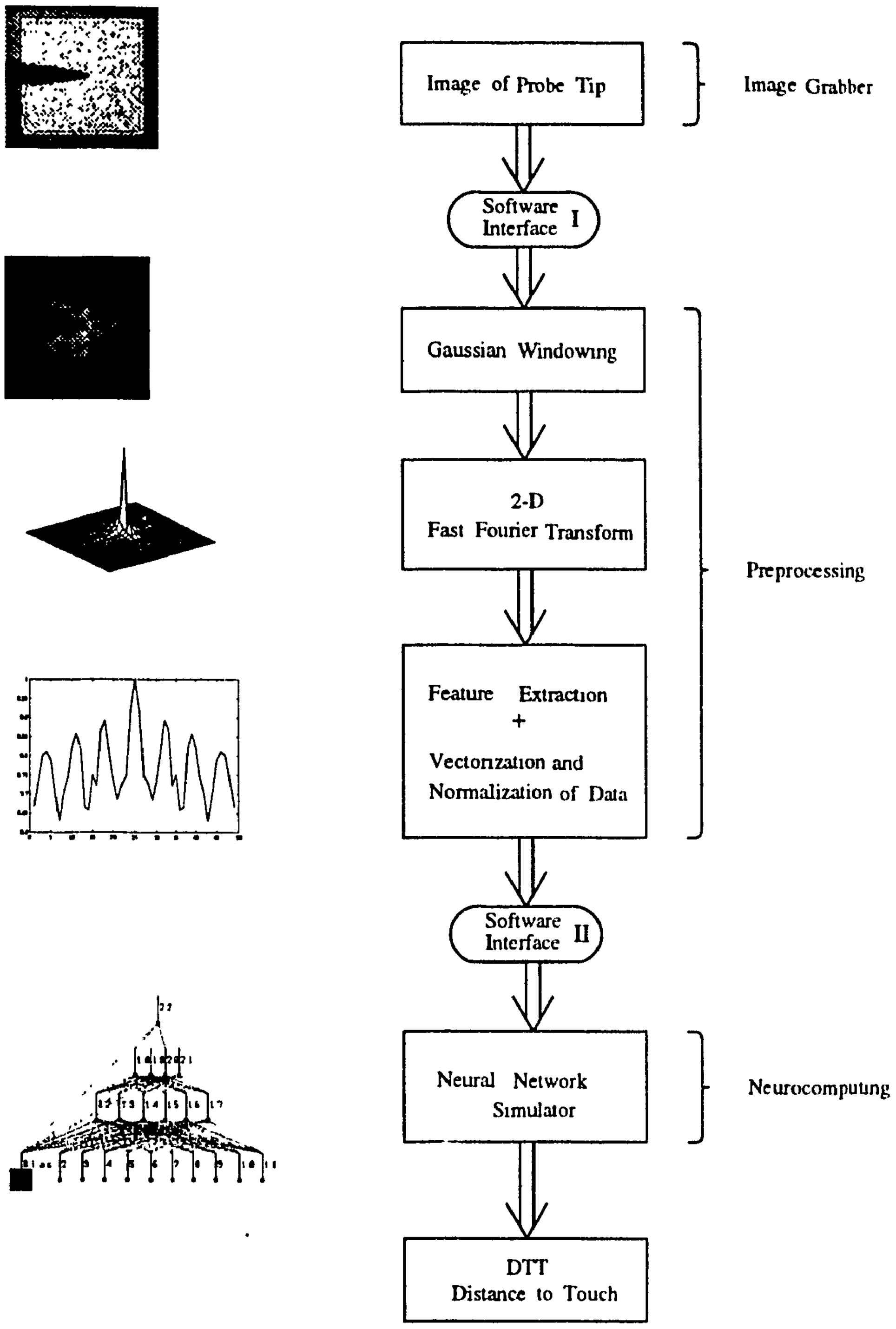


Figure 5.7: *The experimental methodology for determining distance to contact of probe tip in VLSI wafer probing.*

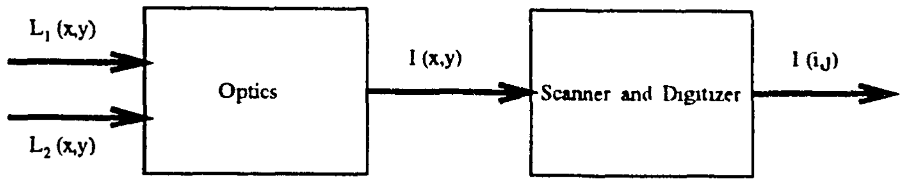


Figure 5.8: *Scanning and digitization of a scene into images.*

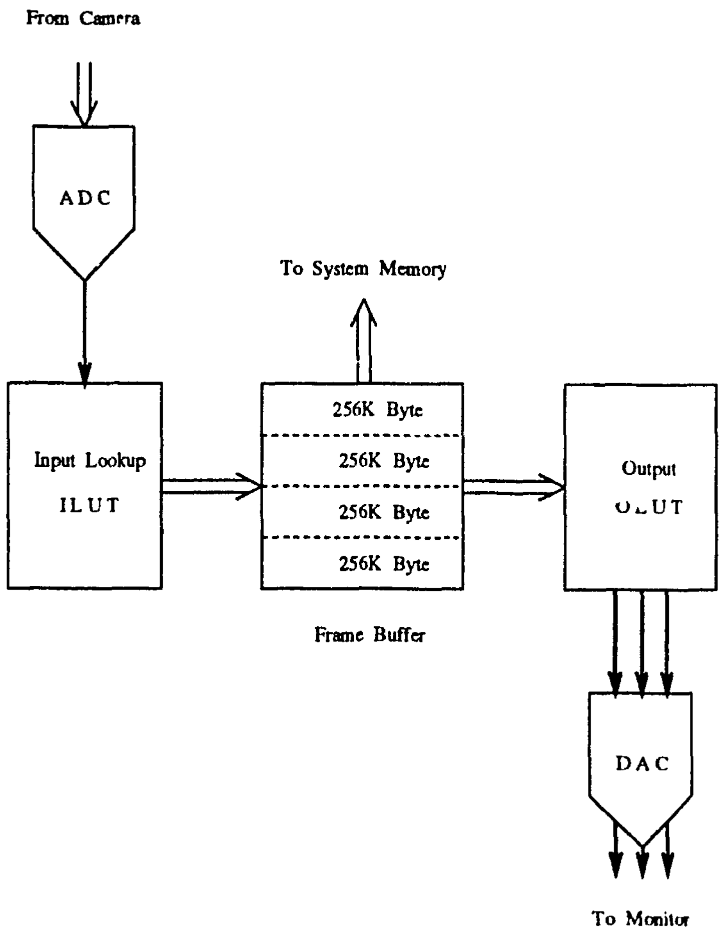


Figure 5.9: *Block diagram of the video digitizer used in our experimental set-up.*

Analog to Digital Converter

The Analog to Digital Converter (ADC) takes the camera signal and converts it into grey levels. It produces an 8-bit output at 10-12 MHz. The ADC allows the analog signal from the camera to be controlled via its range and offset parameters. The range and offset control the amount of information digitized much in the same way as the brightness and contrast affect a monitor. With them, one can limit the amount of information digitized, and process only certain desired grey levels. The range controls and defines the set of values mapped into the 256 grey levels produced by the ADC. It can map just a narrow part of the signal to these grey-levels or the whole signal, depending upon the selection of the range value. The offset helps to manipulate the relevant 256 grey levels of interest from the much larger range available from the video signal. It controls what section of the overall grey spectrum is digitized.

ILUT – The Input Look-Up Table

The Input Look-Up Table or ILUT is a table indexed by pixels. In general, a look-up table takes a pixel as an address, and outputs the grey level stored at that address. It is a one-pixel-in one-pixel-out arrangement which serves as an efficient pre-frame operations processor. Thus, the ILUT takes the 8 bits from the ADC for each pixel and gives the corresponding grey level as an output. It is a very fast operation since Look-Up tables are based on pointer operations and thus operate very fast.

Frame Buffers

Frame Buffers consist of four banks of high speed Video RAMs, of size 256K Bytes each. They are called the *Display Frame Buffer* or the *Access Frame Buffer*, depending on whether the buffer is connected to the monitor, or receives pixels from the ILUT or host CPU. The frame buffer accesses and temporarily stores a linear vector of bytes from an image corresponding to a single line of the image on the monitor. The values are then transferred along with their X-Y coordinate tags to the host processor, and stored on disk as digitized images. These images are accessed from the disk and displayed on the monitor in the reverse operation. The individual pixels are read using the X-Y coordinates from a memory map.

OLUT – The Output Look-Up Table

The OLUT consists of a set of three 256 x 8-bit vectors accessed simultaneously. The purpose is to apply pseudo-colour to the digitized image. The entries are indexed by the grey levels from the current Display Frame Buffer. These three tables are then converted into analog signals to be displayed in black and white or pseudo-color on an RGB monitor.

In our set-up, images of the probe tip are grabbed as it is gradually lowered onto the surface of the probe pad under test. Several images are taken at various distances from touch (see Fig.1.2). There is only a single view along the optical axis of the microscope, and the camera is mounted as shown in Fig. 5.4. The microscope is focused onto the pad surface. Thus the images get progressively more focused as the probe tip approaches the pad surface, or as the distance to contact decreases, and they are perfectly in focus at the point of contact of the probe tip with the pad. The focal gradient is utilized in determining the distance to contact in this thesis.

5.5.2 Software Interface – I

TIFF – Raster Conversion

There is a difference in the organization of the data structures of the images stored under different computing environments. The OC-300 Video Digitizer used in our set-up operates in an IBM-PC environment. Thus, the images grabbed are stored in the TIFF format. The rest of the processing is done on SUN-SPARC workstations which handle images in the SUN-raster format. Thus there is a compatibility problem between the two, and appropriate filters were designed for data conversion from one format to the other. Such filters were written in the C programming language which reads the header of the

TIFF files and then interprets the organization of the data structure of the images. These are then stored on a raster file with appropriate headers, in the SUN-raster format.

Raster – MATLAB Conversion

For the sake of simplicity and to reduce programming complexities, MATLAB programming tools were utilized for most of the computations at the preprocessing stage. This involves further conversion of the SUN-raster files to a format readable by the MATLAB preprocessor. Appropriate filters were designed and written in the C programming language for this purpose.

5.5.3 Preprocessing and Conditioning of Digitized Images

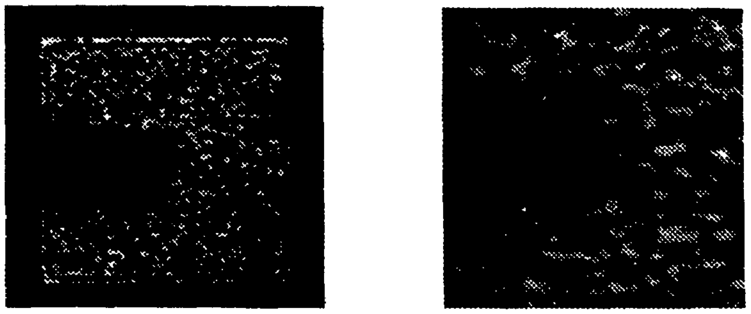
After the images have been grabbed and made compatible to MATLAB-tools with the help of various filters, the necessary preprocessing is done to condition the data before it is input to the Neural Network.

Clipping the Image Size

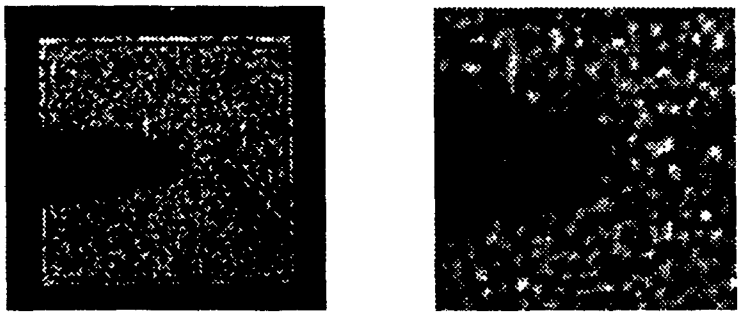
In order to reduce the total amount of information processed, the first step is to reduce the size of each image itself. In our case this is done by including only a small area around the probe tip. As shown in Fig.5.10, this is appropriate because the feature of interest is the amount of blur present in each image, and the rate of change of blurring is maximum at the probe tip. This reduces significantly the total amount of data processed.

Gaussian Windowing

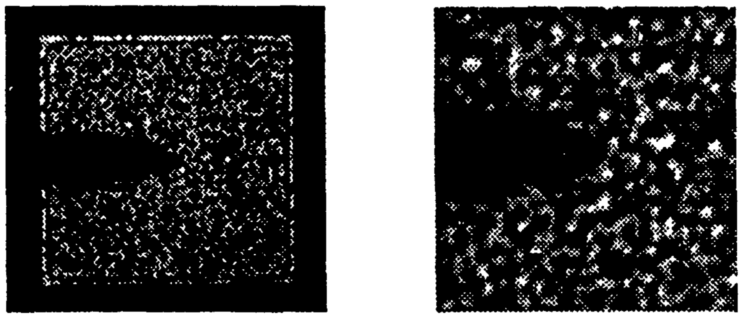
The next step is to window each image by an appropriately. This is done in order to avoid the presence of high frequency components in the resulting Fourier spectrum, contributed by the presence of sharp contrast at the edges or boundaries of each image. The windowing function also contributes to the resultant frequency spectrum, the effect being the convolution of the frequency spectrum of the image by the frequency spectrum of



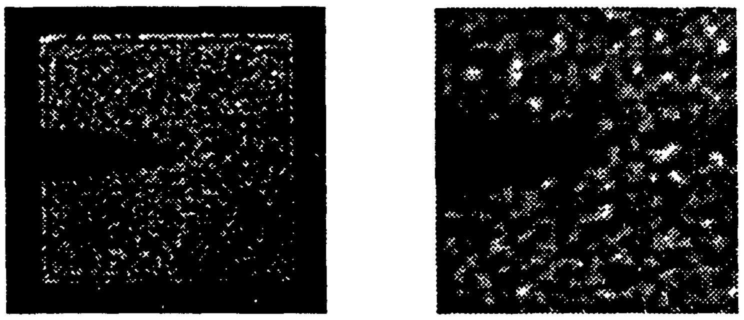
(a) *Distance to touch* = 60 μ



(b) *Distance to touch* = 40 μ



(c) *Distance to touch* = 20 μ



(d) *Distance to touch* = 0 μ

Figure 5.10: *The clipped images of only the probe tips. Note that the rate of change of blurring is maximum around the tip.*

the windowing function [36]. The effect of such a windowing function is negligibly small since the Fourier transform of a Gaussian is itself a Gaussian, and the spread in frequency domain σ_{freq} contributed by a large Gaussian window is small, (section 5.3.1).

A Gaussian window is chosen for this purpose with $\sigma_x = \sigma_y$. The value of σ is chosen so as to include most of the information present in the central portion of the image, gracefully degrading to zero at the edges as shown in Fig.5.11. It is ensured that for each image the probe tip lies close to the center, so that most of the information present in the blurred edge is included in the windowed image.

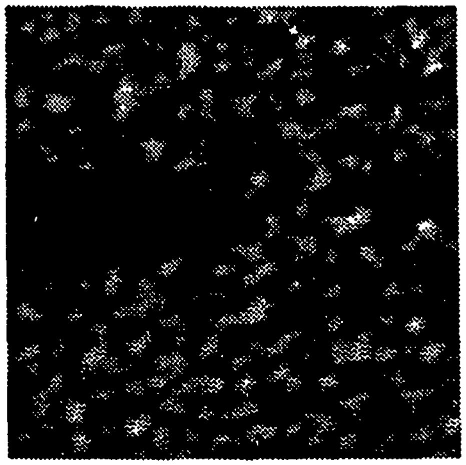
2-D Fast Fourier Transforms

The frequency content of each image is obtained by performing 2-D Fast Fourier transforms of each windowed image. The inherent advantages of working in the frequency domain are enumerated in Section 1.5.4. The Gaussian windowing function effectively suppresses the side-lobes in the Fourier spectrum, with most of the information lying in the central part of the spectrum close to the zero frequency.

Fourier Feature Extraction

There is still a huge amount of redundant data present in the Fourier spectrum of each image. This would "choke" any neural network. Thus some form of data compression is required for effective computation. Fourier feature signatures, are thus extracted from the magnitude spectrum of each image based on features of interest, (Section 1.5.3).

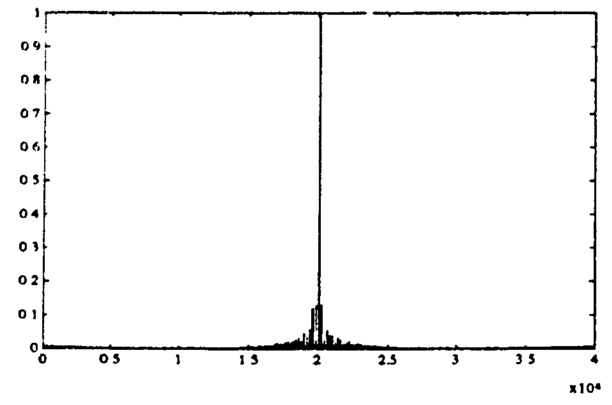
These feature signatures are obtained by further windowing the Fourier magnitude spectrum about the central zero-frequency to obtain the low-frequency components, which are observed to contain information on the amount of blur present. The width of this square window, containing the Fourier feature signature is chosen so as to contain 90% of the power of the resulting spectrum for the most blurred image, that is, the image with the smallest value of σ_{freq}. This n x n window size, corresponding to the minimum value of σ_{freq}, is kept constant and the frequency contents under the window observed for



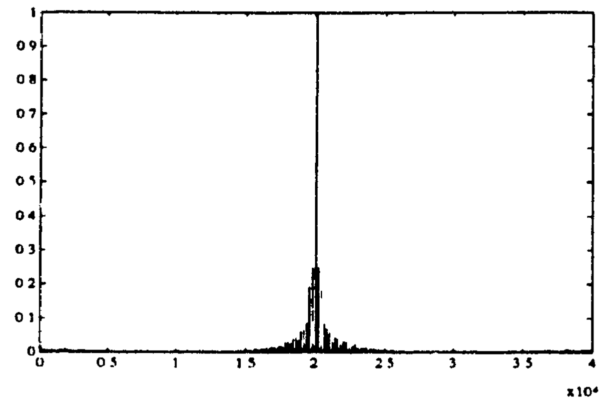
(a)

(b)

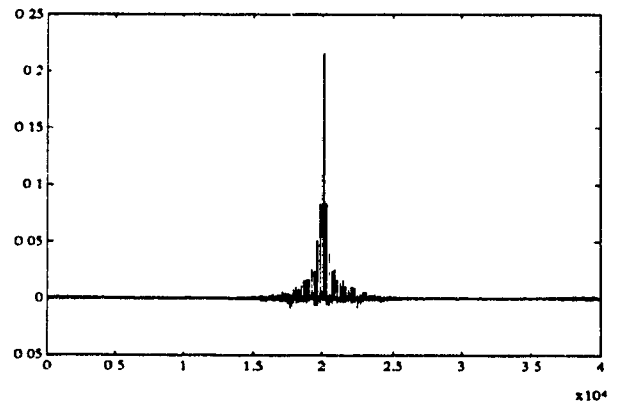
Figure 5.11: *Gaussian windowing of an image to reduce the high-frequency components in the resulting Fourier spectrum, contributed by the presence of sharp contrast at the edges or boundaries of the images. (a) The original image with sharp contrast at the edges (b) The same image windowed by a 2-D Gaussian window.*



(a)



(b)



(c)

Figure 5.12: *Reduction in high-frequency contents of an image by Gaussian windowing. (a) Normalized frequency spectrum (amplitude) of an image. (b) Normalized frequency spectrum (amplitude) of windowed image. (c) Error in spectrum between windowed and un-windowed images.*

each image. Thus, feature signatures for each image with different degrees of defocus are obtained.

Vectorization and Normalization of Data

To train the neural networks, feature vectors containing information on the degree of blur present in each image are required. These are obtained by vectorizing the power spectrum of the windows containing the feature signatures, obtained as explained in the previous section. The feature vectors so obtained, are observed to have a monotonic relationship with respect to the amount of blurring, or distance from contact, as shown in Fig.5.13. In the figure, the difference of the power in the feature vectors is plotted, with respect to the most focused image. It is observed that the curves are monotonically arranged in order of their distance to contact.

5.5.4 Software Interface – II

The feature vectors obtained from the Fourier magnitude spectrum have different peak values for different images. This is not suitable for training the neural network. It was seen in section 4.3 that the function f being approximated by the neural network is a bounded mapping having values between limits 0 and 1. Thus, for effectively training the neural network, the input vectors have to be normalized between these limits. The training and recall vectors have also to be organized in a format suitable for training the network on the NeuralWorks Neurocomputing simulator [32] used in our set-up. In the training set, desired values of the distance corresponding to the input feature vectors are assigned based on the actual measured distances during the probing operation. These values are accurately measured by an automated prober which can be moved in steps of either 20 μ, or 1 μ.

5.6 Neural Networks Simulation

5.6.1 Selection of Network Parameters

All simulations were carried out on the NeuralWorks Professional - II neurocomputing software [32]. Although the Backpropagation approximation theorem [33] states that three-layer networks with $(2n + 1)$ processing elements in the middle hidden layer are always enough in solving real world problems, it is often necessary to have four, five, or even more layers, in order to make the network trainable and of manageable size, (Section 4.3). In our case, the network used has n^2 inputs elements corresponding to n x n feature vectors (Section 5.5.3), two hidden-layer, and one output element. Different sizes of the hidden layers were tried. While there are as yet no clear guidelines for the optimum choice of the network parameters, the appropriate size for our application was found by including enough processing elements in the hidden layers. It was found that by using 40 and 25 elements in the first and second hidden layers respectively, for a 10x1('nput network, we achieve convergence to within 3% error for training set vectors. The training time for such a network is high, since the gain is kept low in order to keep the slope of the transfer function small. Actual simulations took approximately 36 hours for successfully training a backpropagation neural network (with 100 inputs, and 40 and 25 elements in the first and second hidden layers respectively) to within 3% error at the output of the network. This is roughly equivalent to 25000 iterations, or training cycles for simulations carried out on a SUN 3/60 workstation. The gain was kept very low (the actual value being 0.009) for successfully training the network. This is done in order to avoid getting trapped in local minima on the error surface of the network, (Section 4.1.3).

5.6.2 Recalls with Unseen Images

Since f is a continuous function, the fully trained network can be used in the recall mode to interpolate the distance-to-contact for similar, previously unseen feature vectors. This is observed to be true as shown in Fig. 5.15. The first plot shows the recall values of the

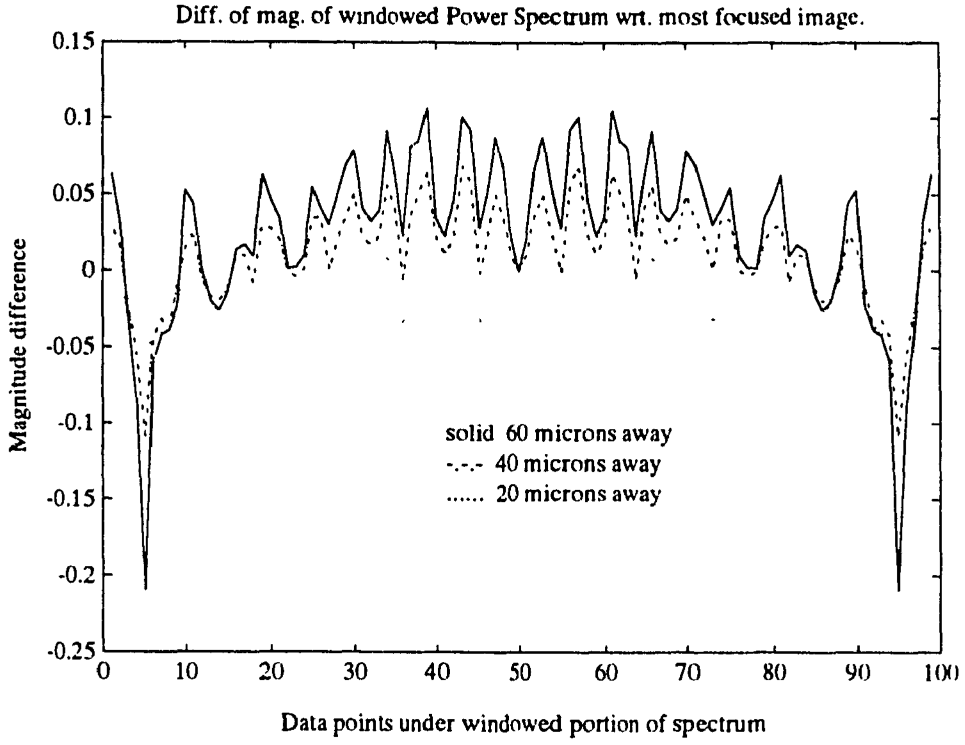


Figure 5.13: *Relationship between distance-to-contact and the relative difference in the power spectrum of the feature vector of each image. The differences are plotted with reference to the most focused image, c the image of the probe at contact. These are used for training the neural network.*

fully trained network for the 4 training sets (Set10 - Set13), each consisting of 9 different images, with different levels of blurring as the probe tip approaches the pad surface (steps of 20 microns), and their normalized values of distance-to-contact (0 - 1). The second plot shows the percentage error for distance-to-contact recalled by the network for 4 different sets (Set14 - Set17) of 9 images each, never before seen by the network, with a maximum error of 8% from measured values of actual distances. These images consist of the same probe tip with different backgrounds. Since the network is able to recall the values of distance-to-contact for images it has never seen before, it is concluded that the network has successfully approximated the mapping from the the Fourier-feature space to the distance-to-contact space.

5.6.3 Backpropagation versus Counterpropagation

Simulations were also carried out with the Counterpropagation neural networks. However, the mapping carried out by these networks is not continuous. The output of the network is quantized to N levels. Since the desired mapping has to be continuous in order to obtain values for distance-to-contact correctly for previously unseen images, this is not acceptable. If interpolation were to be used, it reduces the effect of output quantization. Hence, Counterpropagation networks are not suitable for our application.

5.7 Advantages of using Proposed Approach

5.7.1 Insensitivity to Shift in Position of the Probe

There are important properties of the Fourier transform that make it very useful in image analysis and machine vision inspection tasks. One of the main advantages of using Fourier analysis for feature extraction is that the Fourier spectrum is insensitive to effects of shift and rotation of an object in an image. Thus, any given frequency relationship will be found in the same location in the Fourier spectrum, whereas on the image itself, a given feature might be found anywhere.

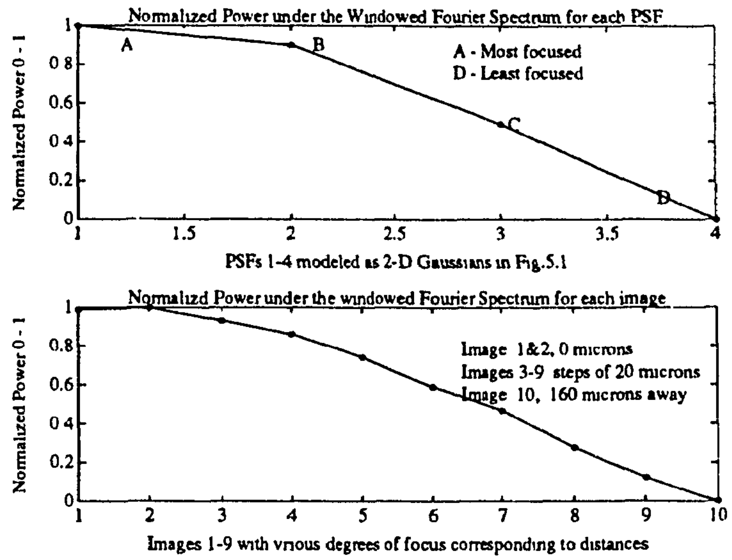


Figure 5.14: *Monotonic relationship between the Normalized Power under Fourier Feature spectrum, with respect to the degree of defocus, or distance-to-touch of Probe-tip.*

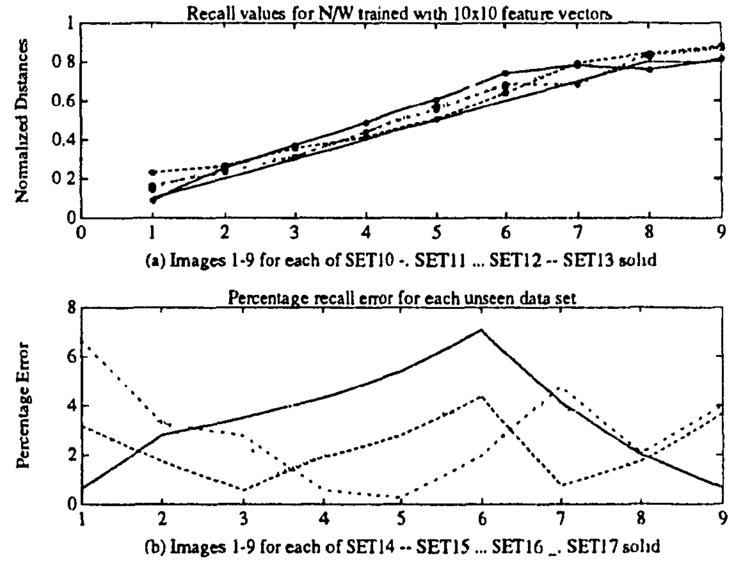


Figure 5.15: *Recalled values of distance-to-touch corresponding to the Fourier feature vectors of each image. Plot (a) for training sets and plot (b) for previously unseen images, cf. section 5.6.2.*

The effect of shifts in the position of the probe tip on the resulting Fourier spectrum was studied. The position of the probe-tip was varied by up to 20% within the image, and the power spectrum plotted after windowing with a Gaussian window of fixed size It is observed from Fig. 5.16 that there is less than 5% error in the power spectrum for up to a 20% variation in the positions of the probe tip, and that the total power of the image (with the same amount of blur) remains fairly constant. Thus, the effect of shifts in the position of the probe tip are negligible. This is very useful in a vibration prone industrial environment, since the exact position of the probe tip can vary frequently because of mechanical vibrations and because of the microscopic dimensions involved.

5.7.2 Insensitive to Rotation of Probe

It is observed that rotation of the probe tip in the image causes very little error in the power spectrum. From Fig. 5.17 we observe that there is less than 3% error for rotations of up to 90 degrees of the probe tip. The total power in the image does not change much with rotation. Rotation causes the frequency elements in the center of the spectrum to be reshuffled. This contributes to the sharp peaks shown in the center of the plots. Simulations were carried out with Gaussian windowed images to suppress the high frequencies contributed by the sharp edges of the image.

5.7.3 Extremely Robust

In the experimental set-up several irregularities are present that can cause significant error in the determination of distance-to-contact. These are: (a) the presence of background noise on the image of the wafer surface, (b) mechanical vibrations that affect the position of the probe, and (c) non-uniform illumination of the wafer surface because of the optical set-up of the microscope system. These irregularities contribute to error in the measurement of distance of probe tip from the wafer surface using by Dantu's algorithm [4]. Such sources of error are unavoidable in an industrial environment; hence, a robust algorithm is necessary for such environments.

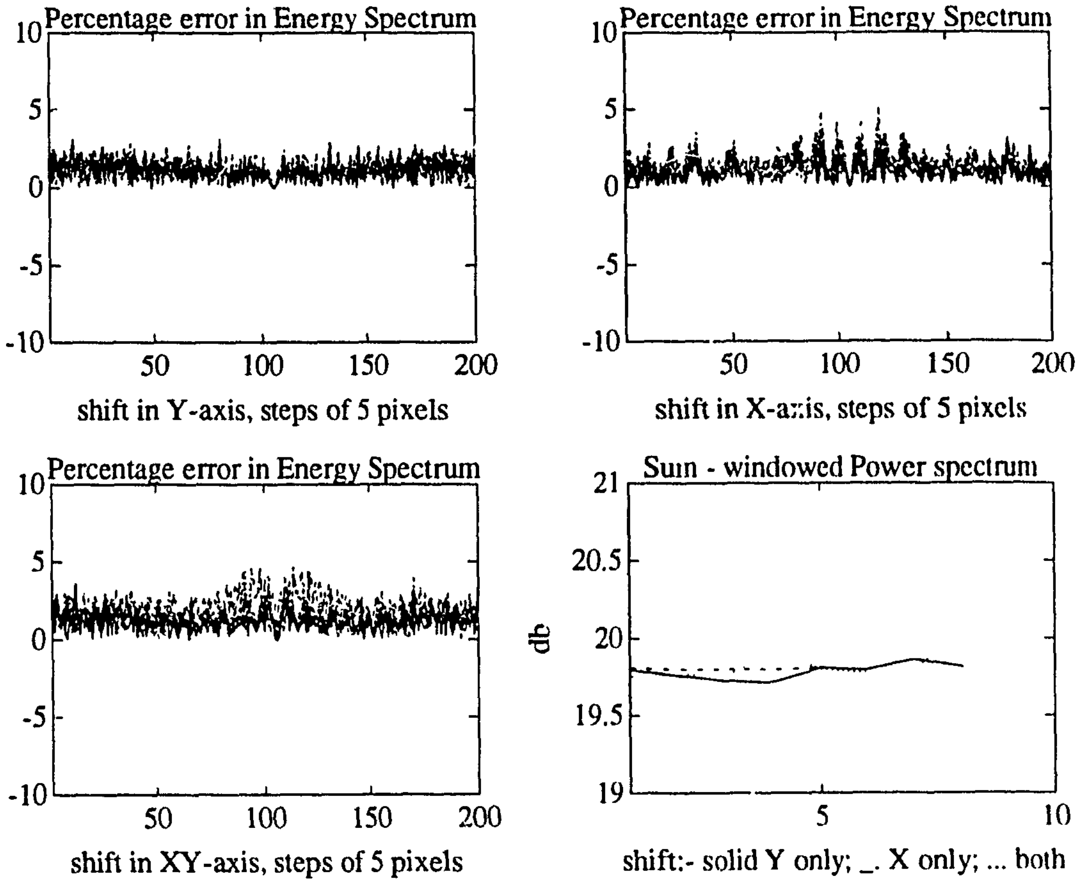


Figure 5.16: *Percentage error in the power spectrum due to shift in the position of the probe-tip in the image. There is less than 5% error in the power spectrum for up to a 20% variation in the positions of the probe-tip, and that the total power of the image remains fairly constant.*

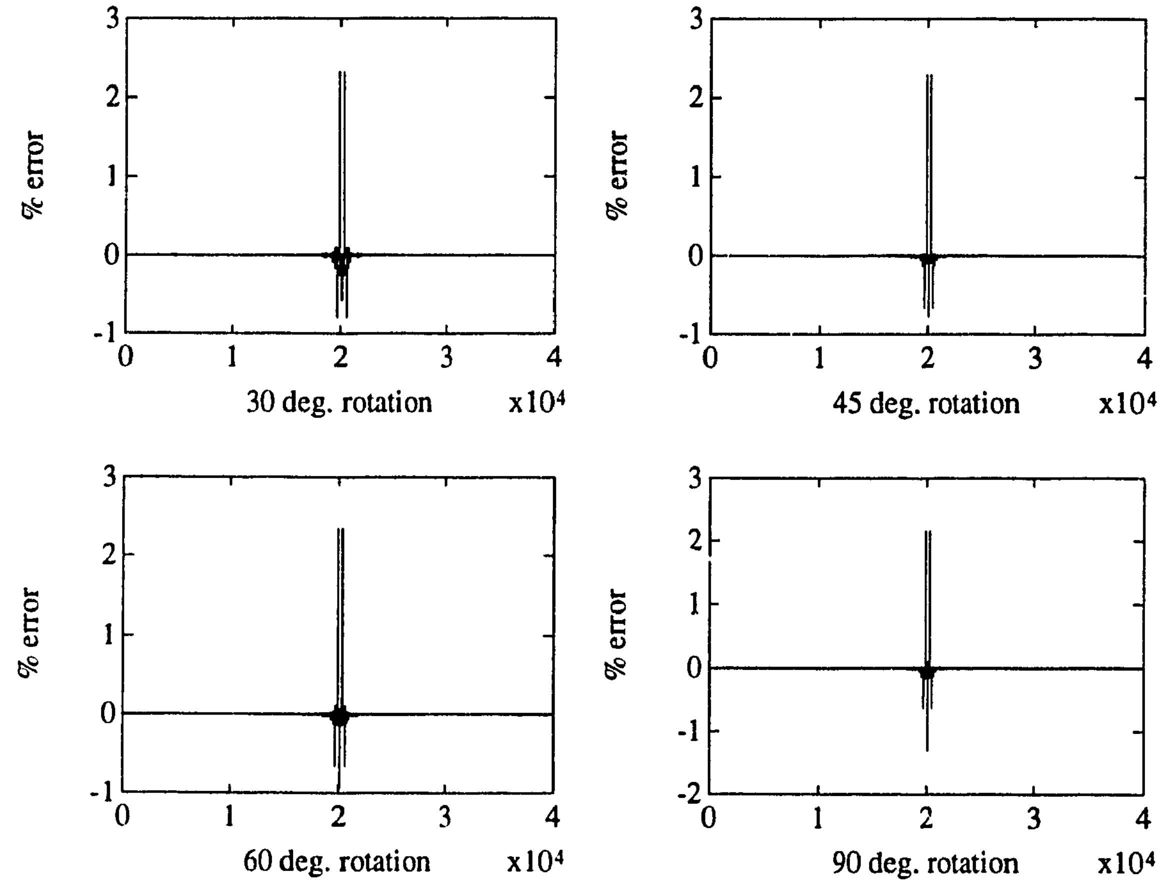


Figure 5.17: *Plot of percentage error in Power Spectrum due to rotation of the probe-tip.*

It is observed from our results that the algorithm proposed in this thesis is robust and insensitive to the effects of noise, vibrations, and non-uniform illumination of the wafer and the probe. It was shown in Sections 5.7.1 and 5.7.2 that there is very little effect on the frequency contents of the blurred image due to the effects of shift and rotation of the probe tip. Thus, the overall effect of mechanical vibrations on the system is negligible.

Background noise contributes to changes in the frequency content of the image. The effect is insignificant in the extracted feature vectors, as the corresponding frequency elements lie outside the window used for sampling the feature vectors.

Non-uniform illumination is caused by the lens system of the microscope, with bright illumination at the center of the image which gets progressively less towards the periphery. The locus of constant illumination forms concentric circles, with the brightest circles lying in the center of the image. This does not affect our system since Gaussian windows are used in our algorithm with the probe-tip in the center of this window before extracting the frequency content of the image. This nullifies the effect of non-uniform illumination.

Neural networks are inherently fault-tolerant systems because of the massive parallelism in their structure. Thus slight variations of specific patterns in the pattern space are tolerated by the network in a classification system. Consequently, slight variations in the feature vectors due to the effect of noise do not affect the correct classification by the network into appropriate distances. This makes our algorithm robust and fault-tolerant, and very suitable for use in an industrial environment.

Chapter 6

Conclusions and Further Work

6.1 Contributions of this Thesis

The major contribution of this thesis is in the use of neural networks for depth perception and a its application to automated visual inspection of VLSI wafers, with significant improvement to Dantu's algorithm [4] for depth perception in VLSI wafer probing. There are three major shortcomings in Dantu's algorithm: (a) It is very sensitive to background noise and vibrations, elements that are invariably present in an industrial environment. (b) It is computationally very expensive. (c) It is very sensitive to differences in the level of illumination of the probe surface, and intensity gradient. All of these make it difficult to use in a practical environment. In this thesis an extremely robust algorithm has been proposed which is insensitive to shift and rotation of the probe, and differences in the level of illumination. It was observed that, there is less than 5% error in case of shifts in position of the probe tip by up to 20%, and less than 3% error for rotations of up to 90 degrees of the probe tip. The effects of non-uniform illumination because of the optical set-up of the microscope, and through-the-lens illumination were effectively eliminated in this thesis by Gaussian windowing of the image of the probe tip. The effects of background noise was eliminated by removing the frequency components contributed by the noise. Backpropagation neural networks were successfully used for mapping the Fourier feature vectors to the appropriate distance-to-contact. The trained network was used in the recall mode to obtain the distance-to-contact with a maximum error of 8% (Fig. 5.15) for previously unseen images.

6.2 Further Work

There are a number of aspects of the topic considered in this thesis which can benefit from further investigation. Some of the most important are the following:

6.2.1 Application of Neural Networks for Fourier Feature Extraction

The backpropagation network has proven to be very successful in approximating the mapping from the Fourier Feature Vector space to the distance-to-contact space. It is sen from Fig. 5.15 that the network recalled the values of distance-to-contact for previously unseen images with less than 8% error. The network was trained based on Fourier feature vectors extracted at the preprocessing stage. fig. 5.7. This operation involves significant computation. The use of neural networks for this stage may help to reduce the computation time involved considerably, since the networks operating in the recall mode will require very little computation. However the training time would be high for such operations.

6.2.2 Implementation of the Algorithm in VLSI

It would be highly desirable to implement the entire algorithm on a dedicated VLSI chip, or a set of chips for the individual stages of the process. This would speed up the computation tremendously by utilizing hardware implementations for the stages that are computationally expensive, such as the 2-D FFTs.

6.2.3 Multiple Neural Network Approach

Another interesting approach would be to utilize several different types of neural networks. Each individual stage of the process would be performed by a different network and they would be finally coupled together to implement the algorithm. For example, feature extraction could be performed by a separate network, which would then be connected to another network for implementing the mapping from feature space to distance space etc.

6.2.4 Optimization of Network Parameters

Optimization of the network parameters has not been investigated in this thesis. There exists no mathematical basis for the optimum selection of these parameters in general.

However, this aspect cannot be totally ignored. Some means of optimizing the size, number of layers, number of elements in each layer, training and recall strategies etc have to be developed, and should form the basis of further work.

6.2.5 Use of Other Network Paradigms

The use of multi-layer Backpropagation neural networks has been justified in this thesis. However, the use Functional Link neural network is also very promising in terms of training time, and its use in functional approximation. Due to limitations of time, simulations with other network types were not carried out. One reason, why Functional Link networks were not used in our application is because of the large number of inputs required for such networks. The use of these networks versus the Backpropagation neural network for functional approximation remains to be investigated, and should form the basis of further work.

Bibliography

[1] A. Pentland, "A new sense for depth of field," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, 1987.

[2] P. Grossmann, "Depth from focus," *Pattern Recognition Letters*, vol. 5, pp. 63–69, January 1987.

[3] M. Subbarao and G. Natrajan, "Depth recovery from blurred edges," *IEEE Computer Society International Conference on Computer Vision and Pattern Recognition (Ann Arbor, Michigan)*, pp. 498–503, June 1988.

[4] R. Dantu, *A Computer Vision System for VLSI Wafer Probing.* PhD thesis, Concordia University, Montreal, 1990.

[5] C. Mead, *Analog VLSI and Neural Systems.* Reading: Addison Wesley, 1989.

[6] A. Hodgkin, A. Huxley, and B. Katz, "Measurement of current – voltage relations in the membrane of the giant axon of loligo," *Journal of Physiology (London)*, pp. 116–449, 1952.

[7] B. Katz, *Nerve, Muscle, and Synapse.* New York: McGraw-Hill, 1966.

[8] B. Keller, R. Talvenheimo, and J. Catterall, "Sodium channels in planar lipid bilayers," *Journal of General Physiology*, 1986.

[9] G. Shepherd, *The Synaptic Organization of the Brain.* New York: Oxford University Press, 1979.

[10] P. Wasserman, *Neural Computing Theory and Practice.* New York: Van Nostrand Reinhold, 1989.

[11] S. Grossberg, "Contour enhancement, short-term memory, and consistencies in reverberating neural networks," *Studies in Applied Mathematics*, vol. 52, 1973.

[12] T. Kohonen, *Self-Organization and Associative Memory.* Berlin: Springer-Verlag, 8 ed., 1984.

[13] D. Hebb, *The Organization of Behavior.* New York: Wiley, 1949. Partially reprinted in [37].

[14] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences, USA*, vol. 79, 1982. Reprinted in [37].

[15] M. Minsky and S. Papert, *Perceptrons.* Cambridge: MIT Press, 1969. Partially reprinted in [37].

[16] B. Widrow and M. Hoff, "Adaptive switching circuits," in *1960 IRE WESCON Convention Record*, vol. 4, pp. 96–104, New York: IRE, 1960.

[17] J. Anderson, "Cognitive and psychological computation with neural models," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, pp. 799–815, 1983.

[18] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing* (D. Rumelhart and J. McClelland, eds.), vol. 1, ch. 8, pp. 318-362, Cambridge: MIT Press, 1986. Reprinted in [37].

[19] D. Rumelhart, J. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge: MIT Press, 1986.

[20] R. Hecht-Nielsen, "Counterpropagation networks," *IEEE International Conference on Neural Networks*, vol. 2, pp. 19–32, 1987.

[21] C. Anderson, "A bayesian probability network," in *Neural Networks for Computing* (J. Denker, ed.), pp. 7–11, American Institute of Physics, New York, 1986.

[22] R. Sasiela, "Forgetting as a way to improve neural-net behaviour," in *Neural Networks for Computing* (J. Denker, ed.), pp. 386–391, American Institute of Physics, New York, 1986.

[23] D. Ackley, G. Hinton, and T. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive Science*, vol. 9, 1985. Reprinted in [37].

[24] J. McClelland, D. Rumelhart, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 2. Cambridge: MIT Press, 1986.

[25] B. Kosko, "Competetive adaptive bi-directional associative memories," in *IEEE International Conference on Neural Networks*, vol. 2, (San Diego 1987), pp. 759–766, IEEE, New York, 1987.

[26] L. Zadeh, "Fuzzy sets and their application to classification and clustering," in *Classification and Clustering* (J. V. Ryzin, ed.), pp. 251–299, New York: Academic Press, 1977.

[27] Y. Pao, *Adaptive Pattern Recognition and Neural Networks.* Reading, Massachusetts: Addison-Wesley Publishing Company Inc., 1989.

[28] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.

[29] R. Hecht-Nielsen, *Mapping Neural Network - The Mapping Implementation Problem, pp.110 111.* Reading, Massachusettes: Addision-Wesley Publishing Company, 1990. Neurocomputing.

[30] Y. Pao and D. Sobajic, "Metric synthesis and concept discovery with connectionist networks," in *Proceedings of the IEEE Systems, Man and Cybernetics Conference*, (Alexandria, VA), 1987.

[31] *Multilayer Backpropogation - Description of MBPN Operation and Function.* San Diego CA: HNC, Inc., 1991. Neurocomputing Software Tools - HNC Neurosoftware.

[32] *Backpropogation - Minimizing the Global Error.* Pittsburg PA: NeuralWare, Inc., 1989. Neural Computing Applications Notes - NeuralWorks Professional II.

[33] R. Hecht-Nielsen, *The Backpropogation Neural network.* Reading, Massachusettes: Addision-Wesley Publishing Company, 1990. Neurocomputing.

[34] R. Dantu, N. Dimopoulus, R. Patel, and A. Al-Khalili, "Depth perception using blurring and its application in vlsi wafer probing," *International Journal of Machine Vision and Applications*, Accepted for publication.

[35] M. Levine, *Vision in Man and Machine.* New York: McGraw-Hill, 1985.

[36] R. Ramirez, *Rectangularly Windowed Waveforms.* Englewood Cliffs, N.J. 0763: Prentice-Hall, Inc., 1985. The FFT Fundamentals and Concepts.

[37] J. Anderson and E. Rosenfeld, eds., *Neurocomputing: Foundations of Research.* Cambridge: MIT Press, 1988.