Development of an image analysis system to produce a standardised assessment of print quality

Thesis

UNRESTRICTED

Development of an Image Analysis System to Produce a Standardised Assessment of Print Quality

Jack Soning Tchan

A thesis submitted in partial fulfilment of the requirements
of the Open University for the degree of Doctor of Philosophy

August 1998

Sponsoring establishment:
The London Institute: London College of Printing

Acknowledgements

I wish to express my gratitude to Dr Robert Thompson, Dr Andrew Manning and Dr Colin Parsons for their supervision and guidance during this study. I would like to thank the support technicians for their help, in particular Mr Arun Datta and Mr Mike Barber. I am indebted to The London Institute for their financial support for the project. Lastly, I would like to thank Wong Mui, Jenny, Stephen, Mimi and Jane for their support during the study.

Associated studies

Papers delivered	Place	Date
Print Quality Assessment by Human Perception	Institute of Physics London	January 1995
Print Quality Assessment by Human and Machine Perception.	The Europto International Electronic Image Capture and Publishing Symposium Zurich	May 1998

Conferences attended

Name	Place	Date
Neural Computing Applications Forum	University of York	July 1995
Neural Computing Applications Forum	University of Portsmouth	September 1995
Neural Computing Applications Forum	University of Oxford	June 1996
Neural Computing Applications Forum	The Department of Trade and Industry, London	January 1997
Neural Computing Applications Forum	The University of Cambridge	July 1997

Abstract

A method has been developed using an image analysis system that simulates human print quality perception. Previous work in the area of print quality assessment has only produced methods that measure individual print quality variables, or assess small parts of an image. The image analysis system developed in this investigation is different from the previous work because it analyses the combined effects of different variables using neural network technology. In addition, measurements from an entire image can be obtained and the system can assess images irrespective of their shape.

The image analysis system hardware consists of a monochrome CCD camera, a Matrox image acquisition board and a 200 MHz Pentium computer. A data pre-processing program was developed using Visual Basic version 5 to process the image data from the camera. The processed data was fed into a neural network so that empirical models of print quality could be formulated. The neural network code originated from the Matlab neural network toolbox. Backpropagation and radial basis neural network functions were used in the investigation. The hardware and software of the image analysis system were tested for non-impact printing techniques. Images of a square, a circle and text characters with dimensions of 1 cm or less were used as test images for the image analysis system. It was established that it was possible to identify the different printing processes that produced the simple shapes and text characters using the image analysis system. This was achieved by training the neural network using pre-processed image data. This produced multi-dimensional mathematical models that were used to classify the different printing processes.

The classification of the different printing processes involved the objective measurement of print quality variables. Different printing processes can produce print that differs in print quality when assessed by observers. Therefore the successful classification of the printing processes demonstrated that the image analysis system could, in some cases, simulate human print quality perception. To consolidate on the preceding printing process identification result, a simulation of print quality perception was made. A neural network was trained using observer assessments of a simple pictorial image of a face. These face images were produced using a variety of different non-impact printing techniques. The neural network model was used to predict the outcomes of a further set of assessments of face images by the same observer. The accuracy of the predictions was 23 out of 24 for both the backpropagation and radial basis function neural network functions used in the test.

The investigation also produced two possible practical applications for the system. Firstly, it was shown that the system has the potential to be used as a machine that can objectively assess the print quality from photocopiers. Secondly, it was demonstrated that the system might be used for forensic work, since it can identify different printing processes.

Objectives

A system will be developed that can make print quality assessments of images made by non-impact printers. The images used will be monochrome images of simple shapes and text characters. These results will be compared with those made by observers. The purpose of the project is twofold. It will help resolve complex issues in print quality assessment procedures while contributing new facts and information to the field of machine intelligence.

Contents

List of figures

List of Tables

1.0 Introduction

This thesis describes the development of a machine that is capable of simulating human print quality perception. The machine consists of an electronic camera and a computer that uses artificial neural networks to process the data from the camera. This combination of the camera and computer is known as an image analysis system. It will be shown that there are differences between the human visual system and the image analysis system. Aspects of the human cognitive processes involved in visual perception and artificial neural networks employed to model human visual perception will be discussed. It will be demonstrated that even without a full comprehension of the cognitive processes and artificial neural networks, it is still possible to produce an image analysis system that can simulate print quality perception.

This thesis can be separated into four stages. The first stage discusses the cognitive processes involved in visual perception, printing, previous attempts at solving print quality assessment problems, image analysis, artificial intelligence and artificial neural networks. The second stage uses the information from the initial stage to describe the development of an image analysis system that can simulate print quality perception. The third stage states a series of results obtained from the image analysis system that demonstrates it can simulate print quality. The final stage discusses the results from the preceding stage and further work that can be carried out. It also discusses potential practical applications for the image analysis system that have emerged from this investigation.

2.0 Cognitive processes and artificial intelligence

Assessing the quality of any printed image is an easy task for observers. The only action that needs to be taken is to ask an observer to comment on the quality of the print. Anyone who is capable of verbal responses or gestures can give an opinion. A closer examination of this process, however, poses many questions. For example, are the answers absolute or will they vary from observer to observer and, if so, how? Can print quality be standardised? Can a machine measure it?

The first question is a statistical one. It would simply be necessary to ask a large number of observers an identical print quality question and to record the results. This process, as discussed later, is unsatisfactory for standardisation purposes and moreover little will be discovered about subjective print quality assessment. It is therefore proposed to develop a

machine that can make standard print quality measurements. This may also broaden the understanding of print quality assessments by observers since human physiological and psychological behaviour must be considered. Since only opinions are sought after, there will be no correct answers. This lack of exact target responses to questions will make the task of modelling print quality perception more difficult and make it harder to show that the model is correct. This is not due to a lack of knowledge about the neurophysiology of the visual system which is objective in nature. It is because there is a requirement for subjective intelligent behaviour from observers to be understood when the assessments are made. It can be argued that print quality assessments by observers are only subjective because it is unknown how these opinions are formed. If it were known, then it would be objective. The aim of this project is not to make an exact theoretical model of print quality perception. This would, as shown later, be beyond the scope of this project. The approach taken has been to produce a model that can be implemented by computer using existing theories from cognitive psychology and methods from artificial intelligence. Since these theories and methods originate from attempts to understand human intelligence, the approach can also be considered semi-empirical. This can only increase the probability that a discovery will be made about the mechanisms involved in the human perception of print quality during the course of the project.

From the discussion above, a good starting point for the project would be a general overview of visual perception from the eyes to the lower and higher cognitive processes in the brain for static monochrome images. This is followed by a discussion of the difficulties encountered in attempting to explain the mechanisms of human intelligence.

2.1 The neurophysiology of the human visual system [1,2,3]

When a printed image is perceived, an observer can vary the distance of the image to the eyes to receive the optimal information content from it. Focusing is achieved by automatically changing the thickness of the lens in the eye. The eyes can adapt to varying light levels by changing the aperture of the pupil although this may alter the perceived image. The two processes above are control mechanisms that are currently available to automatic cameras, albeit slower. However, other properties of the human visual system such as the characteristics of human eye movement and object recognition are unavailable to machines. In the eye, there are photoreceptors that convert incident light into electrical signals. They consist of 10^8 light-sensitive rods and $6x10^6$ cones. The rods are for monochrome perception and they function at low levels of illumination. The cones are for

colour perception and for vision at high illumination levels. The rods and cones are distributed non-uniformly over the retina. The retina has a fovea and a Macula Lutea region responsible for high visual acuity. The periphery of the retina contains mainly rods and a few scattered cones. There is also an optical disc where the optical nerves leave the eye; it contains no photoreceptors and forms the blind spot. The responses of groups of these receptor cells are pooled at the ganglion cells. Some of these ganglion cells pool their responses from large regions of the retina, usually on its periphery. These cells are described as possessing large receptive fields. Ganglion cells nearer to the fovea have smaller fields. Larger receptive fields are responsible for a coarse quantisation of the image, while the smaller ones are responsible for higher acuity vision. The role of the ganglion cells in feature detection will now be explained.

2.2 Feature detection by ganglion cells [2,3]

The ganglion cells can be described as feature detectors. Groups of these ganglion cells can be used to detect features such as black bars on a white background. Therefore different sub-groups of these cells can be used to detect this feature at different orientations to form, for example, the enclosed outline or edge of an object. In order to find the edges in an image, which correspond to intensity changes with spatial position as shown in **figure 2.1 a**, the intensity can be differentiated against spatial position. The peak of this function will determine the position of the edge. Differentiating for a second time will place the edge on a zero crossing. Both of these operations are illustrated in **figure 2.1 b and c** below.

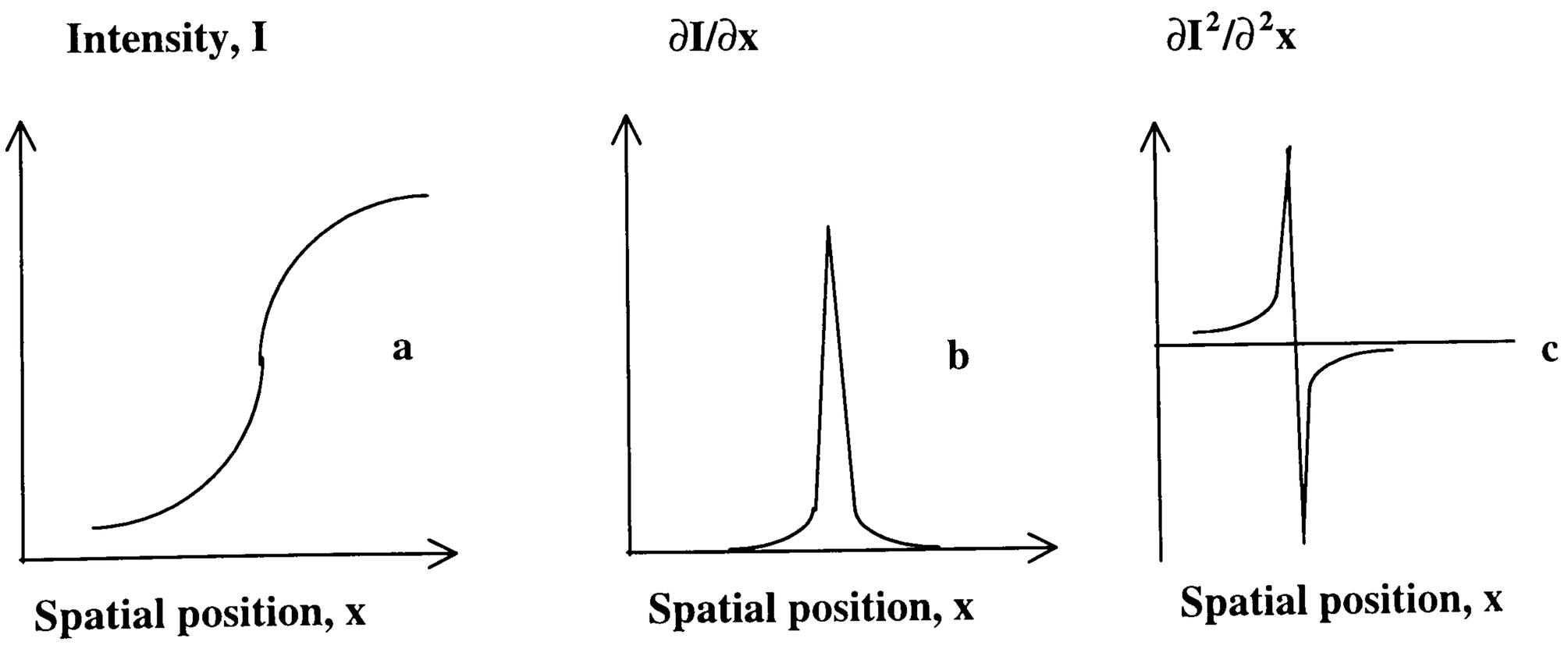


Figure 2.1. Two models of edge detection derived from the intensity profile in a, are shown in b and c.

A simple way to determine the position of an edge is to use a template as in **figure 2.2** below. The only other operation that is required to determine the position of the edge is to differentiate the intensity with respect to spatial position of the detected edge.

1	**0**	**-1**
1	**0**	**-1**
1	**0**	**-1**

Figure 2.2. A simple template that can be used to detect a vertical

The problem with using a template is that it is sensitive to edge sharpness. A sharp edge would be difficult to locate accurately using a template with large cell areas and a blurred edge would also be difficult to quantify using a template with a small cell area. In order to overcome this problem, templates with different cell dimensions can be employed.

Gaussian functions can also be used to detect edges that differ in sharpness **equation 2.1**. σ is the standard deviation which determines the width of the Gaussian. By changing the width, edges with different degrees of sharpness can be detected. It has been suggested that visual cognition uses 2-dimensional Gaussian functions in the templates to filter out unwanted frequencies.

$$\text{Gaussian}(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2} \right) \qquad \textbf{Equation 2.1}$$

The principles of image processing described above are similar to those of its electronic equivalent, the charged couple device or CCD camera. The differences are that firstly, CCD data processing is serial and the human visual system can be considered both serial and parallel. Secondly, current CCD cameras have a lower resolution, if the image is not magnified, and a simpler structure that consists of a uniform array of pixels. The principles of the CCD camera will be given in more detail at a later stage, where it will be seen that there are even more differences between the two. Even at this early stage of imaging processing, it can be seen that there are fundamental differences between human

and machine vision systems. This is only one of many problems that will be encountered if an exact model of subjective print quality perception is to be found. The next section highlights one of these problems. It is a problem that is not exclusive to visual processing, but concerns the entire cognitive process that must be studied here in order to understand the possible mechanisms behind print quality perception.

2.3 The information processing and connectionist paradigms of cognitive psychology [4,5,6]

Two main schools of thought in cognitive psychology are the information processing and connectionist paradigms. The information processing model has its roots based in digital computing. It assumes that the theoretical principles of cognition and digital computers are the same; they are serial processes, which manipulate symbols, and they store information in specific places. This means that all cognitive processes can be attributed to a few mathematical operations involving the manipulation and storage of symbols. **Figure 2.3** illustrates a possible model for the serial approach.

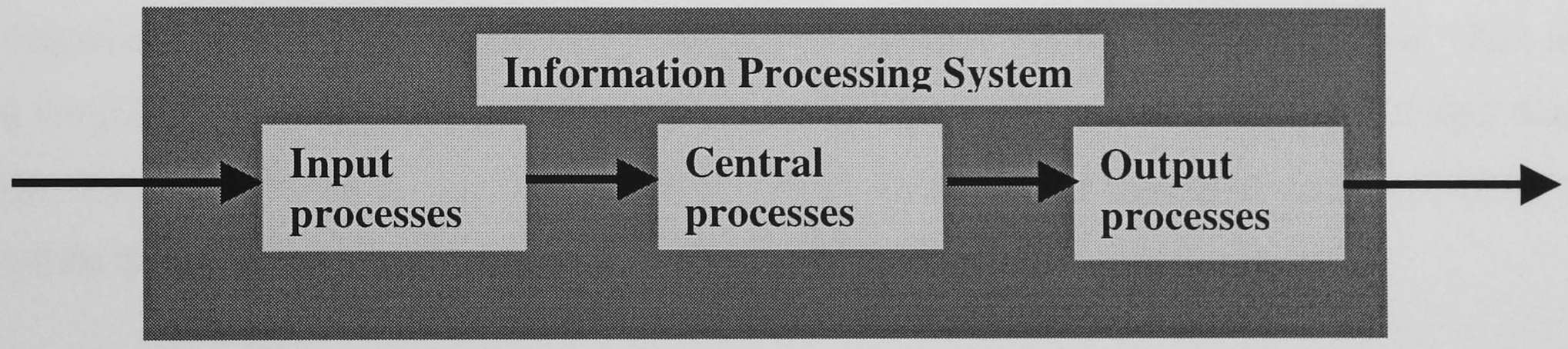


Figure 2.3. The information processing model of cognition.

In the case of cognition, the input stage initially analyses the data, for example the image processed by the retina, in order to recognise objects. The central processes deal with all the functions that occur in the brain. This includes the coding, manipulation, and storage and retrieval of information. A typical operation here might be to decide whether to store an image in long or short-term memory. The output process prepares responses for the external environment. These responses could be either verbal replies or physical gestures. In the case of a computerised imaging system, the CCD camera that processes image data can represent the input stage. The second stage would be responsible for image data coding, manipulation, and storage and retrieval, while the output stage would prepare data for an external printer or visual display unit.

A drawback with the serial approach is that it does not refer to the structure of the brain, which operates differently to the digital computer. A digital computer is a Von Neumann machine; it has a central processing unit that can only execute one instruction at a time using electronic gates that operate digitally or a few of these central processing units connected in parallel. The brain, as will be shown later is still poorly understood. What is known however, is that it consists of approximately 10^{10} neurons that are each connected to 10^4 others via structures called dendrites. **Figure 2.4** shows one of these neurons with its connections in simplified form. A neuron can be considered as a stand-alone analogue logical processing unit. The neuron inputs and outputs are known as the synapses and axons respectively. Each neuron has one output and many inputs. The inputs of a neuron are connected to the outputs of other neurons. The voltages of these inputs are stored and controlled chemically in the synapses of the neuron. The proximity of the synapse to the dendrite in **figure 2.4** represents the degree of electrical connectivity between the two. Above a threshold voltage, a chemical change at the synapse causes conduction, increasing the voltage across the soma. The axon will remain quiet until the voltage across the soma, reaches a critical value known as the activation level. The activation level will cause the output to fire, triggering a response changing the synapse voltage of other neurons. This is a simplified picture of a highly complex process. However, it can still be concluded that the brain can perform parallel operations since each neuron can operate independently whilst being highly interconnected.

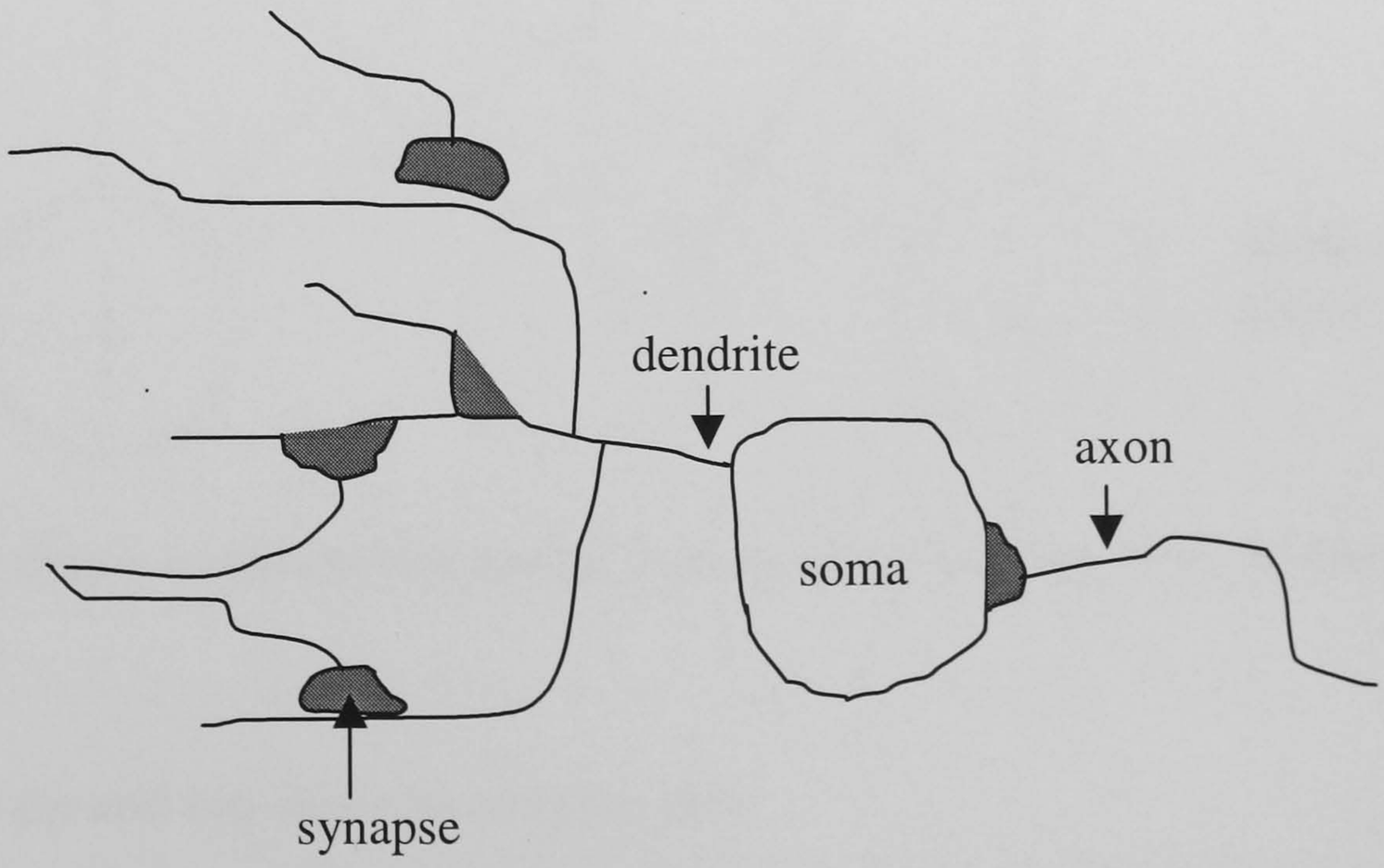


Figure 2.4. A simple representation of a biological neuron.

2.4 The connectionist model of cognition [5,6]

The connectionist model is based on the structure of the brain. It appeared in the early 1980's with the advent of artificial neural networks. This subject is also referred to as connectionism or parallel distributed processing (PDP). The principle of PDP is based on interconnecting units to form a large network that is capable of performing more than one task simultaneously, which is not possible with serial processing. Each unit has a level of activation which depends on the input by other units and on environmental conditions. If the input to a unit is positive, it may raise the activation past a threshold, triggering a response. This is best illustrated with the pictorial example shown in **figure 2.5.** In this example the letter R in Red is to be recognised. The arrows show activated responses and the dots inhibited responses.

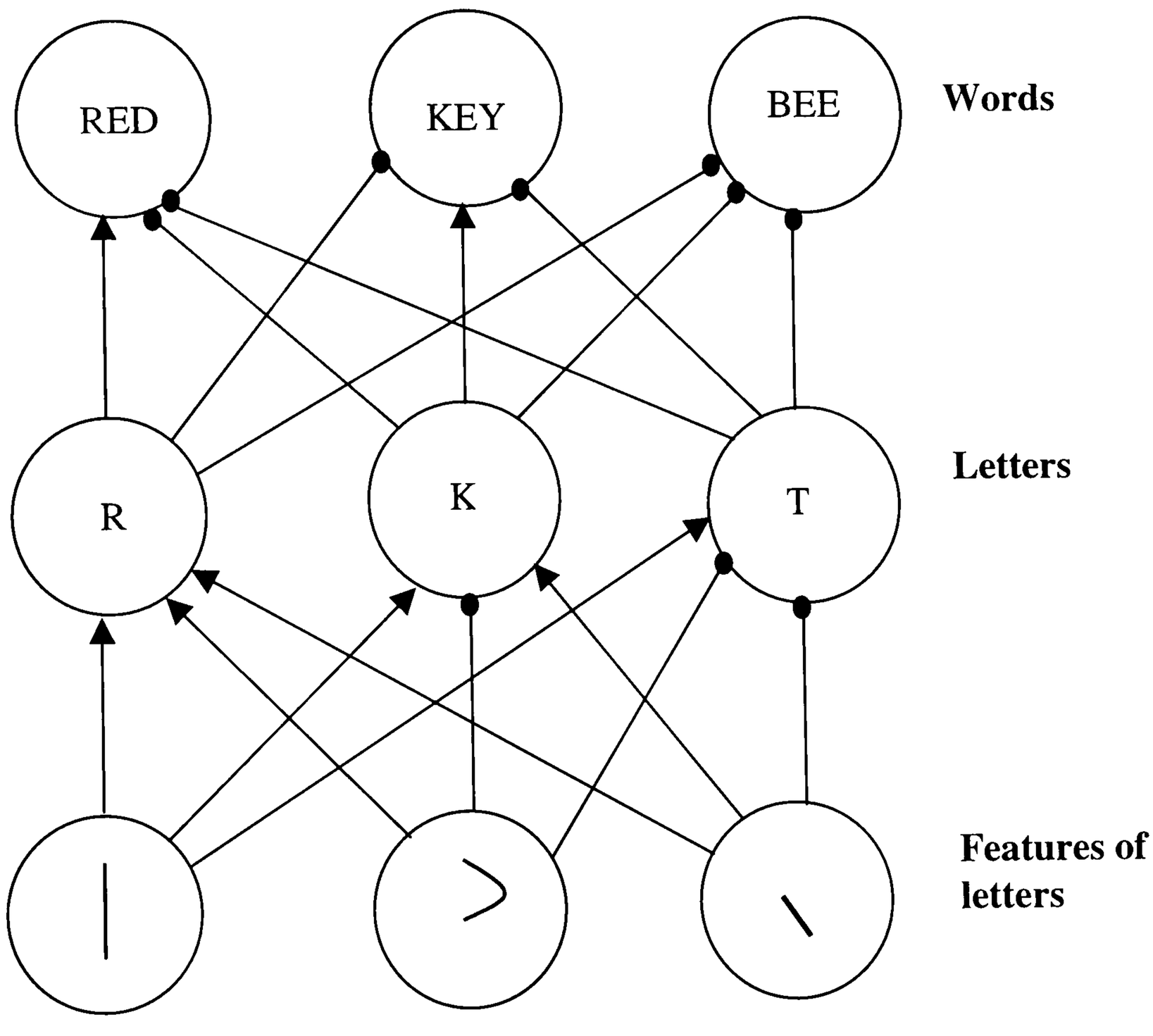


Figure 2.5. A simple connectionist model that enables the letter R in the word Red to be recognised.

2.5 Bottom-up and top-down processing [4,5]

In cognitive psychology, bottom-up processing means taking information from the bottom (defined here as the light input from the image to the retina) and working up to higher levels of perceptual analysis. This is also known as data-driven processing because it is the data from the retina which determines visual perception.

Top-down processing means that concepts in the brain's memory about the environment and the image content influence how it is perceived. If data-driven processing alone can explain visual perception then the development of the print quality assessment system would be easier than if top-down processes are required but this is unlikely as the following character recognition example will show.

Consider the diagram shown below in **figure 2.6**. The two words **THEY** and **BAKE** can be recognised from it despite the fact that the **H** and the **A** are represented by an identical character that does not possess all of the topological features of **A** and is a distortion from **H**s that are normally encountered.

Figure 2.6. An example which demonstrates the existence of top-down processing.

Since the **A** and the **H** are represented by the same character it means that additional information not present in the printed image itself is used to recognise these words. This information must originate from previous knowledge of language stored in the brain.

Cognition can be separated into two categories. Firstly, both basic cognitive and cognitive processes refer to the mechanisms behind perceptual tasks such as object and sound recognition, counting and memorizing and learning simple responses. Secondly, cognitive processes cover higher level processing such as conscious thought and feelings, constructing arguments and deciding what to say. Cognitive processing must therefore include an element of top down processing. Cognitive variables must also be considered in the print quality assessment problems since decision making processes and natural language responses are involved.

So far mental processes have only been discussed up to the point of cognition. It can be concluded that cognitive theories can objectify apparently subjective behaviour only up to a point. It fails however, to explain completely the mechanics of conscious experience and

intelligent behaviour. This problem can be formalised using Searle's Chinese room thought experiment. Prior to the examination of this, a brief overview of artificial intelligence from which Searle's experiment originates will be given.

2.6 The current approach to artificial intelligence [7,8]

Artificial intelligence (AI) concerns the development of computational systems that perform tasks that require human intelligence. These tasks generally exclude ones for which a decision procedure is known, but include those that require an understanding of perception. For example a computer program that solves algebraic equations is not artificially intelligent but a chess program is.

The main problem of AI is the arguments concerning causality of intelligence and whether a machine can be constructed that possesses human mental states. The second problem is well known to AI researchers and is usually expressed in Searle's Chinese room problem.

The Chinese room argument is a thought experiment to demonstrate that AI is not true human intelligence but a simulation of it. It uses an analogy, in which a man is placed in an enclosed room that isolates him from the outside world. He does not understand Chinese but does possess a book that contains a set of instructions about Chinese characters. These instructions give correct answers to questions in Chinese. Outside this room is a man who understands Chinese. His function is to push a written question in Chinese through a letterbox for the man inside the room to answer. The respondent looks up the characters in his book of instructions that tells him the correct characters to write in reply. He is therefore able to give the right answer without understanding Chinese. This, according to Searle is how a computer operates. They can only simulate human intelligence. Computers therefore, are incapable of 'understanding' a problem in the same way humans and can only solve them by looking up answers from a set of rules.

The validity of the Chinese room argument is debatable since a book of rules could be interpreted as a convenient representation of human intelligence. The book could be a machine of which the mechanisms are unobservable. This would take the argument full circle.

Searle also believes that in order for a machine to achieve human intelligence, it must have the causal capability of the human brain. For this to happen the machine must be made a

biological system containing neuroprotein. The validity of Searle's arguments is a major focus of discussion among psychologists and philosophers that is yet to be resolved. However, what definitely is known is that measurements can be made on systems that are regarded as intelligent. Therefore models can be constructed that can simulate human intelligence and can subsequently be tested for accuracy.

It has been demonstrated that it is the current lack of understanding of the causal mechanisms of human intelligence that makes the development of full AI so difficult. The approach usually adopted by AI researchers is to break up the problem into small areas of study called domains and to concentrate on simulations of human intelligence. For example, a chess program or a text character recognition system can be regarded as domains that are simulations. The chess program may be so strong at the game, beating any human opponent, that it cannot be regarded as possessing human intelligence. The same would apply to the text recognition system if it could not recognise handwriting. The last two points illustrate that by breaking down the general AI problem into smaller components, a greater insight into AI can be achieved.

In this project the domain of print quality assessment is dealt with. These domains are not entirely based on human intelligence, as the full nature of human intelligence is not understood, but on other semi-empirical models. This is deemed unimportant, if the research has some practical use as in the two examples above. Even so, there is the possibility of acquiring knowledge about human intelligence.

Evidence has now been produced that shows that the knowledge required to develop a subjective print quality assessment machine based on the precise mechanisms of human intelligence is currently unavailable. Therefore an approach used frequently by artificial intelligence researchers of building empirical models for small AI domains will be adopted.

The next chapter will focus on previous attempts to measure subjective print quality. To understand this a discussion of print quality variables and printing processes must be undertaken beforehand.

3.0 Print quality investigations by other researchers

This section initially concentrates on the nature of print quality variables and the main printing processes. This is necessary so that past print quality researches described in subsequent sections can be understood. The historical survey of print quality research in this section will not only concentrate on paper and ink processes, but will discuss photographic print quality. This related area has yielded important results relevant to this project.

3.1 Print quality variables [9,10]

Print quality variables will be classified into two classes, resolution and noise. A high resolution means there is a large information content in the image. This will produce good image quality. Resolution, as will be seen later, can be split further into spatial and tonal parameters. Noise is independent of the resolution of a printing process. It can degrade the quality of a print by masking it with unwanted effects. Noise appears in different forms, of which edge raggedness and mottle are examples. The next section explains the importance of resolution in the context of print quality. This is followed by a description of the main printing processes and the types of noise that they can produce.

Spatial resolution specifies the capability of a printing system to resolve fine detail. In all binary printing processes (these are all printing processes with the exclusion of intaglio) the reproduction of tones between black and white are produced by a method called halftoning. In halftoning, an image is formed by a regular dot pattern. These dots can vary in size, shape and frequency. These variables can affect the perceived quality of an image. The number of lines formed by the dots per inch (dpi) is known as the screen ruling and specifies the spatial resolution. A high spatial resolution is needed to render type accurately and produce fine artwork. For example, a high resolution is required to print security documents such as cheques and passports. Lithography, which is capable of producing halftone at 300 dpi can do this. At this resolution, the dots cannot be resolved using the eye alone. They can only be seen under magnification. This means that it is possible to manufacture printing systems that are capable of producing images with spatial resolutions beyond the threshold that human perception can assess. Tonal resolution or greyscale rendering capability is the ability of a printing process to produce a wide range of tones. Both spatial and tonal resolution can determine the print quality of an image since they can both change its information content.

3.2 The printing processes and their print characteristics [8,9,10,11]

Printing processes can be divided into analogue and non-impact or digital processes. An important difference between these two is that in analogue processes, the original is produced only once, regardless of the size of the print run. In the case of digital printing, the original has to be rewritten each time a print is required. Analogue processes include flexography, lithography gravure and screen printing, while non-impact printing consists of thermal, inkjet and laser processes. These techniques will now be described.

3.2.1 Flexography

Flexography is a relief process. Using this method, the printing area containing the ink is pressed against the paper. The printing block is made of rubber or plastic. Since the printing surface is raised, the background does not come into contact with the ink. Flexography has the advantages of low cost and high versatility. It can be used to print on most media and it can use both quick-drying solvent and water based inks. Its main applications have traditionally been in low quality printing such as packaging, cheap magazines and newspapers. The quality is low firstly due to the amount of pressure needed to transfer the very fluid ink from the plate to the substrate causing ink to squeeze out around the edges of the image areas. This causes halftone dots to enlarge around the edges. Secondly, the printing plate is very soft and distorts easily under pressure.

3.2.2 Lithography

Lithography is a planographic process. This means that the image producing area of the printing plate lies in the plane of the plate. The lithographic process differentiates between image and non-image areas through relative surface energies. The image area is made hydrophobic and ink attracting, while the non-image area is made hydrophilic and ink repellent. Both ink and water are applied to the entire plate surface. The ink layer is transferred to an intermediate roller called a blanket cylinder. From the rubberised blanket, the ink is transferred to the substrate as the paper is pressed against the blanket to receive the image. Offset lithography is a versatile process because the blanket cylinder has a soft rubberised coating; it can print on many different kinds of substrate including those with quite rough surfaces. This means that it is difficult to recognise lithographic printing by assessing the application alone.

The print quality advantages of lithography over flexography are that it can faithfully reproduce fine photographic detail and fine halftones better. This is due to the "stiffer"

inks and softer rubberised blanket used in lithography which means that less pressure on the paper is required than in flexography.

The use of water in lithography can cause the paper to stretch resulting in distortion. Lithographic inks are tacky and can lift paper fibres creating a rough uneven texture in the print. This effect is often referred to as fluffing or picking. A way in which offset printing can be recognised is when there is too little water present, causing the non-image area to be contaminated with small droplets of ink. This creates a tinting or 'toning' effect on the non-image areas of the print.

3.2.3 Gravure

Gravure is an intaglio process. This means that the imaged areas are recessed into the plate, instead of being flat as in lithography or raised as in flexography. In this process, recessed cells that accept ink are engraved into a cylinder. This can be achieved by using a diamond stylus cutter which oscillates in and out as the cylinder rotates. The cylinder comprises a steel core that is coated with a thin layer of copper. It is into this layer of copper that the cells are engraved. After it has been engraved, the cylinder is coated with chromium for durability. This process which is required to manufacture a gravure cylinder is expensive and is reserved for printing which runs into millions of copies. In conventional gravure, the cell depths are varied to alter the image tones while in halftone gravure the cell area is also changed. Gravure is excellent for reproducing photographs because it is capable of producing fine screens. To remove surplus ink, a doctor blade is pulled across the plate. The paper is fed over the plate by a cylinder covered by rubber which also presses the paper against the recesses to absorb the ink.

In conventional gravure, varying the ink cell depth produces the tonal range. As the cell depth increases the ink delivered from it increases until a solid print is produced. Due to the uniform square cell structure of the screened cylinder, the edges of text can give a fuzzy stairstepping effect that is visible even when inspected using the eye alone.

In halftone gravure the tones are produced from small round cells. Light and dark tones are produced from small and large cells respectively. This produces a halftone dot pattern as in flexography and lithography. A characteristic of this type of printing is the doughnut effect it produces for light tones. The ink transfers only from the edges and not from the centre, leaving a halo effect.

Problems can occur with both types of gravure, firstly, solids can have white spots. This is due to viscous ink that does not join up cell walls completely. However, if the ink is too fluid, the cell walls will join up for solids, but a mottling effect will appear. Secondly, thin vertical streaks of ink can run through a print caused by nicked doctor blades that do not wipe off excess ink perfectly, or by score marks on the cylinder surface. Thirdly, a white speckled appearance with light tones is possible when the print is on rough paper. This is due to the pits on the paper being larger than the cell therefore preventing contact between the paper and the cell.

3.2.4 Screen Printing

Using this method, a photographic stencil is placed on a flexible screen made of synthetic fibre. High viscosity ink is squeezed through the image areas of the stencil onto the screen. A screen can be wrapped around three-dimensional objects. This means that the process can be used to print directly onto three-dimensional objects. Like flexography, screen printing can print on most surfaces. It produces prints with thick films of ink which produce high contrast, bold images making it very suitable for large poster work. However, the thick ink films means that the photographic stencil is unable to produce small characters or fine photographic details.

3.2.5 Non-impact printing (NIP)

NIP can be placed in a sub-class of its own discrete from the processes described above. The reason for this is that the images are stored electronically in a computer memory, instead of physically on a plate. The electronically stored image acts as an instruction set for the print engine. The advantages of NIP over the other processes described above are that it is easier and less time consuming to produce very small print runs. The main NIP technologies are inkjet, electrostatic and thermal transfer printing.

3.2.6 Continuous inkjet

Using this technique, a continuous flow of ink is forced through a nozzle using a pump, and is broken into droplets by a vibrating transducer. These droplets are then charged by an electric field. The droplets are then either deflected onto paper via electrostatic plates or into an ink recirculation system. This is a primitive form of inkjet technology which is unable to produce high resolution images. Its main application is in high speed industrial marking and coding.

3.2.7 Continuous-array inkjet

The principle of this technique is the same as that of the continuous inkjet, except that the single nozzle is replaced by a row of fine nozzles. The nozzles operate independently of each other. This technique produces higher quality printing compared with using a single nozzle since the mechanical motion of the print head is reduced from two to one dimension. Continuous-array inkjet printing is capable of printing at 300dpi. The resolution is limited by the difficulties involved in manufacturing the ink return path plumbing and ink pumps with the required pressure. Another problem with this type of printing is that ink tends to splatter against the paper.

3.2.8 Impulse jet

Impulse jet is a drop-on-demand system. It ejects ink only when it is required, otherwise the ink remains in the nozzle held by surface tension. The requirement for an ink recirculation system is also unnecessary. Two different techniques in this process can be used to deliver the ink onto the paper. These are referred to as phase change and bubble jet and are described below

3.2.9 Phase change inkjet

This process uses wax-based pigments as inks. The inks are kept in a reservoir in the liquid phase. The liquid ink is drawn into a chamber which is connected to the print head. The ink is ejected from the print head onto the paper by squeezing the chamber using a piezoelectric crystal. The wax-based ink does not penetrate the paper as well as a solvent ink since it solidifies on contact with the substrate. This makes the images sharp and of high colour saturation. However since the ink does not penetrate the substrate, the surface is easily scuffed on physical contact.

3.2.10 Bubble jet

The bubble jet process, which is also referred to as thermal inkjet, uses water-based inks with both low viscosity and surface tension. These properties enable the ink to be rapidly ejected from the nozzle. Ink is ejected when it is heated in a small chamber connected to the nozzle. The heated ink evaporates to form a bubble within the chamber. The increase in volume forces an ink droplet out of the nozzle. After this, the chamber cools and is refilled from the reservoir by capillary action. This method of inkjet printing is very common since the print head array mechanism is very easy to manufacture using conventional semiconductor technology.

3.2.11 Thermal transfer printing

In thermal transfer printing, an array of computer controlled heating elements is placed in contact with a heat-sensitive substrate. An image is formed when the substrate is moved past the heating elements which fire to create the image. The spatial resolution across the image is fixed at 300 dpi but can be varied perpendicular to the array by changing its speed. There are several different thermal transfer processes which are discussed below.

3.2.12 Direct thermal printing

This technique uses paper that is specially coated with a dye based material which darkens when heated. Only binary printing of 300dpi resolution is currently possible with this process. This means that the images are usually of poor quality. This fact is reflected in its uses, which include inexpensive fax machines, barcoding and labelling applications. Also, the thermally induced images are restricted to monochrome since the chemical processes required for full colour printing have not been developed.

3.2.13 The thermal wax transfer process

In this process, a heat-sensitive ribbon supplies the colourant. As in direct thermal printing, the heat is supplied by a computer-controlled array of heating elements. In addition to heat, pressure is used to transfer a waxed-based colourant from the ribbon onto the substrate. Due to the low melting temperature of the colourant, there is very little residue left on the ribbon after the transfer stage. The requirement of pressure in transferring the wax onto the substrate means that the substrate must have a smooth surface. This excludes the use of plain paper for this process.

3.2.14 The thermal dye diffusion process.

This technique, which is also known as dye sublimation printing, works on the same principles as the thermal wax transfer process, but with one major difference; dye sublimation produces continuous tone images while the thermal wax transfer process produces binary images. This means that the intensity of colourant delivered by the dye sublimation printhead can vary from point to point, giving it a greater tonal resolution than for the binary thermal wax transfer method. For a monochrome image, the dye sublimation process is capable of 256 grey levels per dot and at 300dpi can produce a print quality approaching that achieved in photography.

3.2.15 Electrophotography

Electrophotography also known as laser printing, works in the following way. A photoconductor drum is used to store the image. This is achieved by electrostatically charging the drum surface and then exposing it to a laser beam which contains the modulated image signal supplied by a computer, or alternatively, by switching a matrix of light emitting diodes. The light has the action of discharging the areas with which it interacts. Two possible processes can now occur. Firstly, once the selected non-image areas have been discharged, toner, a powdered polymer colourant, is applied to the drum and is attracted to the charged imaged areas. Secondly, the toner is charged with the same polarity as the charged drum. Therefore the toner will be repelled to those areas that are discharged by the light, which are now the image areas. The drum is then placed in contact with the paper and the toner is transferred to its surface. The toner is subsequently fused onto the paper surface by the application of heat and pressure. The principle of laser printing is the same as xerography except that a printed image is illuminated by light and its image is focused directly onto the charged drum eliminating the need for a computer to supply the modulated image signal.

3.2.16 Print quality factors in inkjet, thermal and laser printing

Factors that can affect the print quality in digital printing apart from spatial and tonal resolution include: jagged or ragged edges, dashed lines, uneven print across the image, and a rough grainy appearance. These defects can be attributed to dot misplacement and reproduction problems. Small satellites of ink and toner splashes can also have a detrimental effect, along with the problem of mottle which also occurs in analogue printing processes.

3.2.17 The print quality measurement methodology

It can be seen from the discussion in this section that there are many different printing processes. Each has its own set of characteristics that can affect the subjective appearance of the print it produces. This can be a complex problem to solve if a global print quality model is sought. This is not assisted by the other variables involved such as ink and toner formulation, the variable substrates used for printing, and the manufacturer, age and condition of the printing machinery. To make meaningful measurements, a simplification is required to reduce the number of variables involved.

When observers assess print quality they need to know consciously nothing about printing processes and variables. They only decide what is a good or bad print without the necessity of stating a reason for their decision. If it is possible for an untrained observer to make print quality assessments without prior knowledge of printing processes then it may be possible to develop a system that also produces the same results for the assessments without the requirement for prior knowledge of the printing processes. Even if the observer does have prior knowledge, it might be possible to express this knowledge as a mathematical model. This is an important concept that will be adopted in this project to simulate subjective print quality. Print quality measurements containing no known prior knowledge of the printing processes will be objectively measured and processed by a computational engine to produce a subjective assessment model. This will be achieved by using image analysis for the measurements and neural networks and fuzzy logic for the processing of the data. To support this method and to highlight its originality, a brief history of print and image quality research will now be covered.

3.3 Print quality measurements in paper-ink systems

The advent of modern print quality analysis was made possible by the invention of the computer-based image analyser. This made high-speed image analysis possible. The first image analyser of this kind appeared in 1963 **[13]**. These were primitive devices that could only compute binary black and white areas in an image, or the frequency of black to white areas; they did not have any memory to store image data. They consisted of a cathode ray tube camera connected to a monochrome TV monitor and analogue-measuring device. These systems were mainly used to examine the microstructure of metallurgical samples in order to classify their grain sizes. As the technology of the image analyser improved, it was realised that these devices could be used to measure print quality variables. Many researchers, mainly in the area of electrographic printing, have attempted print quality measurements. A summary of print quality and related work using image analysers carried out to date will now be given. This will be used to support the methodology employed in this thesis.

In 1979 **[14]** image analysis was suggested as a tool that could measure contamination in paper and the evaluation of solid and halftone print quality. In this paper, detailed proposals were given on how to achieve this, although no data was published. In 1982 **[15]** the use of image analysis was suggested for the inspection of residue ink in recycled pulp in an industrial plant. By 1993 **[15]** a fully automated system for this task had been

commissioned. The inspection area was 20 cm^2 and particles of 50 microns could be detected under a magnification of x2. These measurements can be classified as print defects since the particles can contaminate a printed surface. The fact that fast automatic measurements can be made means that there should be no difficulty in making these measurements under laboratory conditions.

Research into print quality parameters **[17– 24]** has been carried out quite recently with the advent of powerful low-cost, modern personal computers. This type of research differs from that of recycled pulp because more precise measurements of printed images have been made and these results have been compared with subjective assessments of quality. An account of this work is given below.

In the area of gravure printing, image analysis has been recognised as a useful tool for the measurement of size and distribution of unprinted spots **[18]**. It was reported that the sensitivity of the instrument in the assessment of print quality was high enough to agree with the subjective results from observers. An important point mentioned in this paper is that the limitations of the computer technology at that time restricted the amount of data and hence the size of image that could be processed in a given time.

Properties of halftone dots have been assessed **[18,19]**. These include dot area, circularity, profile and perimeter measurements and greylevel histogramming. Although edge raggedness measurements were mentioned here, there is no evidence in the quoted reference of them actually being made.

Edge raggedness **[20,21,22,23,24]**, voids **[20,21]**, background satellite **[22]** line-broadening **[22]** and template matching measurements **[23]** have been found to be prevalent in digital printing. Measurements in the variation of the diameter at different angles in the character 'O' to evaluate stairstepping has been carried out **[24].** The letter O was chosen in order to eliminate the need for character alignment. Another suggested method was to measure the perimeter of a print character, smooth the edge and then to compare the two **[19]**. The effects of edge raggedness and void rate have been measured **[20]**. These results were compared with subjective assessments and were reported to agree with each other.

Mottle is a major problem in printing and has been experimentally studied **[19,25]**. It results from uneven absorption of ink and affects the perceived texture of the printed surface. The

mottle data was processed using power spectrum analysis and the co-occurrence matrix methods respectively in the references cited above. Power spectrum analysis can assess the noise caused by mottle by plotting the tonal distribution against the angular frequency. The co-occurrence matrix method analyses an image by splitting it into an array of small regular sized sections known as pixels. An entropy measurement, defined by **equation 3.1** below, of the pixel count for tonal values differing by an amount, i-j, for neighbouring pixels distance d apart in the array made. $M_d(i,j)$ is a matrix containing the array of elements i and j stores the counts i-j. Good correlation between the image analysis results and observer assessments were reported in both cases. Since mottle will not be a main part of the initial investigation it will not be discussed in detail until the concluding part of the thesis.

$$\text{Entropy} = \sum_i \sum_j M_d(i,j) \log_e M_d(i,j) \qquad \textbf{Equation 3.1}$$

The work described in **[26]** repeats the work of **[14-24]** exclusively for digital printing. It serves as a useful summary of the work using image analysis and highlights the fragmentation discussed in previous print quality investigations.

Contrast and contrast uniformity of prints has been a neglected area of research by the researchers mentioned above. This could be due to the fact that this type of measurement is difficult to achieve using CCD cameras. The discussion above has only concentrated on the printed region. Another variable is the printed medium itself. The next two paragraphs cover these areas.

Texture can be defined as areas on surfaces that have a visible patterned or random effect. This can be used by visual perception or machine vision to identify adjacent edges when two differently textured areas meet. On paper surfaces, a rough texture can be created by protruding paper fibres and a smooth surface by coating the paper. This effect which governs whether the surface is perceived as rough or smooth can change the print quality. No direct measurements using image analysis for the effect of the 3-dimensional texture property on print quality has been found in the literature. However, the method used to measure it will be the same as that for ink mottle since this is also caused by a 3-dimensional scattering effect by the ink surface.

Whiteness and gloss measurements have been carried out without using image analysis. Paper whiteness assessments have been carried out using specialist colour measuring equipment, the Zeiss Elrepho and the Technidyne Technibrite Micro-TBIC [**27**]. Although the author here describes whiteness as a quality criterion he does not relate it to print quality perception. For the measurement of gloss, a simple reflectometer was used to measure reflected light at different angles **[28]**. This research produced a correlation between observer assessment and measurement. If this variable is important in print quality perception then it should be measured by image analysis with the other measurements so that an integrated model of print quality can be made.

An important observation that can be drawn from the survey is that image analysis measurements have been reported to correlate well with subjective assessments **[17,19,20,22,25]**. However, the drawback of the cited research is that they are specific to single or small groups of printing processes and/or concentrated on a single variable. This makes the entire subject fragmented. The work cited immediately below does consider more than one variable simultaneously. It does, however, originate from photographic research and was carried out prior to the development of modern image analysers. This is unimportant here. What is important is the principle that two variables have been considered simultaneously to produce an image quality model.

3.4 Print quality models

So far all the work carried out deals with print quality defects as separate variables. This section deals with image quality models that analyse more than one variable simultaneously.

The combined influence on the sharpness and graininess of colour prints has been studied **[29]**. In these experiments the graininess was varied by using grain masks and the sharpness changed by varying the degree of enlarger focus. A total of 210 experimental prints of 5 different scenes were studied by two groups of observers. One of these groups comprised experienced technical observers, the other of non-technical observers. It was concluded that the perceived subjective print quality assessments from the experienced observers for the five images were given by the empirical equation:

$$P_Q = [0.413(S)^{-3.4} + 0.422(10 - G)]^{-1/3.4} - 0.532 \qquad \textbf{Equation 3.1}$$

where P_Q is the perceived print quality, S is the sharpness, G is the graininess

The significance of this work is that it has been recognised that print quality variables can be combined with each other to influence perceived print quality. **Equation 3.1** however, only applies to the five images used in the test and is therefore not a general formula. Also it only considers graininess and sharpness for photographic images.

Engledrum and McNeill **[30,31]** have recognised the work described above and have applied the theory to printing processes. The printing processes were an inkjet and a wax thermal transfer printer. They used five different colour originals for these assessments. These originals are described as continuous tone pie charts, a solid colour graphic and a photograph consisting of a graphic and a map. The equation which was developed for these originals and printing processes is:

$$P_Q = (0.187\text{defects}^{-1.73} + 0.475\text{sharpness}^{-1.73} + 0.22\text{colour accuracy}^{-1.73})^{-\frac{1}{1.73}} - 3.53$$

Equation 3.2

If correct, the **equations 3.1** and **3.2** show that it is possible to mathematically model perceptual print quality assessments, albeit for a limited number of images and printers. The question which has not been addressed is whether a general model can be developed that takes into account all processes, substrates and images. If a general model can be developed then it would be possible via an image analyser to produce an artifically intelligent system. This project will attempt to move as far as possible down this route.

To highlight the originality of the work in this thesis further, current commercial products that measure print quality and ongoing research will now be discussed. Since they are commercial they will lack detail on methodology, however it is the results only that are of concern here.

3.5 Commercial systems and software designed to measure print quality

By 1996, two commercial software applications specifically for measuring print quality and a proposal for a third had been produced. The importance of this is that they employ or propose to employ a CCD camera interfaced to a personal computer. A description of these systems is given below, with their limitations, from the available information given in the references.

Optimas **[32]** released Sentinel, a software application that checks that a print has been accurately reproduced. This is achieved by training the software, probably using neural networks, to recognise differences between a template image and an image under test. A subtraction between the template and the test image has to be performed which requires exact pixel to pixel alignment. This software has been designed for high volume industrial inspection for measuring flaws in identical prints from a standard process. However, this software has only been designed for faults such as print dropout and splatter in small text areas and logos.

Another software application that has been developed is ImageXpert by Acuity **[33,34]**. This software package is more versatile than Sentinel in the sense that it claims to measure the following: resolution/MTF, signal to noise ratio, uniformity, dimensional accuracy, dot quality and colour consistency, not only for printed images, but also for LCD displays and CCD sensors. No mention has been made in the literature about combining the effects of different variables and correlating them to subjective perception.

Finally, Xerox **[35]** has proposed a fully automated system for copy quality analysis. They suggest scanning an image with a CCD array after which the software in the system would produce data on image resolution, density, colour rendition and registration.

From the surveys of **sections 3.3** and **3.4**, a summary of the research can be given as follows. Image analysis has been extensively used for the measurement of print quality variables. **Figure 3.1** shows the schematic layout of the equipment used for this. Subjective comparisons have also been made using groups of observers to rank prints that have been image analysed. These results show good correlation between the two but only consider the effects of separate print variables and/or processes and/or for an entire image. The research cited for the work that combines the effects of variables has not been carried out using an image analyser.

The research in this project will take the form of **figure 3.2**. The differences in design of the image analysis system to that of **figure 3.1** are as follows: it will automatically analyse the data from an entire image producing multi-parameter data that will be processed by a neural network or fuzzy logic model. In addition, the subjective analysis of the images will also be different. Instead of concentrating on groups of observers for the subjective

assessments, it will emphasise the use of only a single observer in a Turing type test. This will be discussed in the next section in addition to the image analyser and the fundamental principles of neural networks.

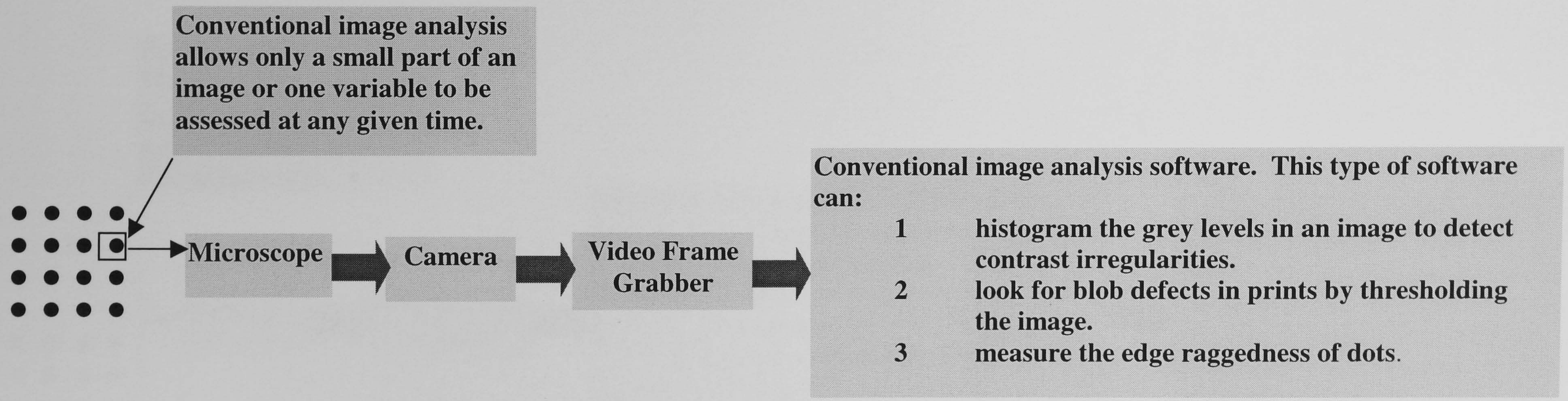


Figure 3.1. A conventional image analysis system used to measure print quality.

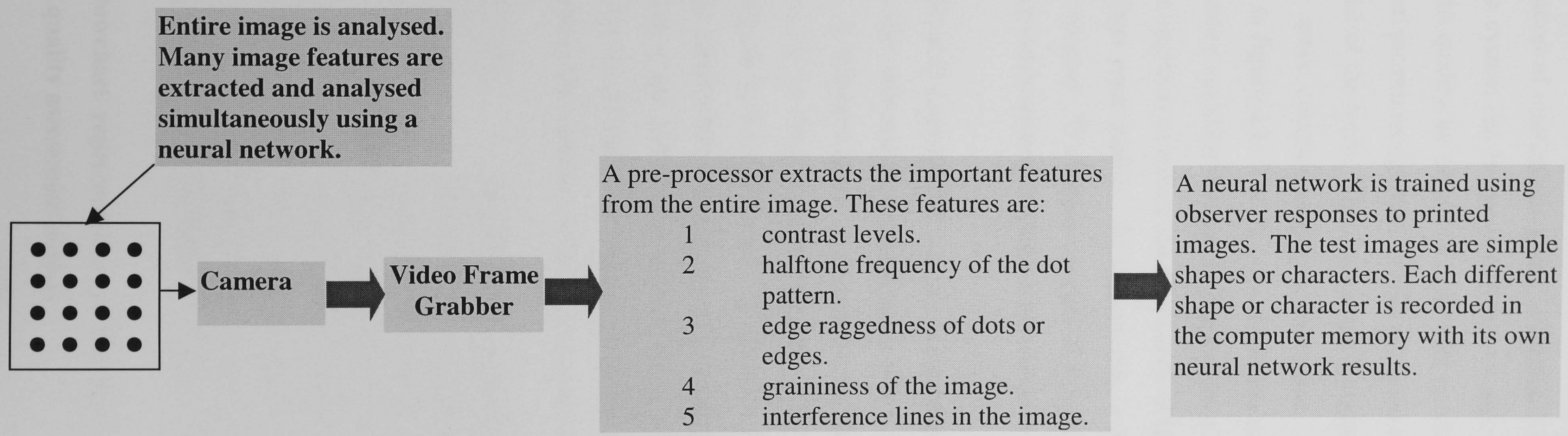


Figure 3.2. The system under development for measuring print quality perception. This system will analyse an entire image and many variables simultaneously.

4.0 Theoretical outline of the experimental work

It is now clear that the domain under investigation is intelligent print quality assessment using the interpretation of intelligence given in **section 2.6**. More precisely, this project concerns whether a system can be developed that is able to predict or imitate subjective assessments of print quality by observers. As will be shown later, the predictions will come from statistical pattern recognition techniques in the form of neural networks and not from an exact model of the brain. This means that a statistical method will be adopted to model empirically more complex cognitive processes. A formal description of this approach is shown in **figure 4.1**. The diagram shows the mapping of an input space which consists of print quality variables that can be mapped into an output space via the engine. The function of the engine is to reduce the dimensions of the data to a single dimension or a variable that defines print quality. To achieve this, a neural network and fuzzy logic approach has been adopted. The engine consists of a digital computer that can implement a neural network algorithm written in a high-level computer language.

If it is possible to develop a system that can pass a test based on human intelligence for the domain of print quality assessment, it would mean that it would be easier to define print quality perception. A standard based on perception will have been produced which does not involve humans in the final stage of analysis. The parameters of a machine that measure print quality can be fixed and used as a reference or duplicated. This is untrue for observers who have irreversible time-dependent parameters that govern their behaviour; for example, observers age and cannot unlearn things like a computer can. The next section explains a test for AI that is well known to workers in this field and shows how it can be utilised in testing the reliability of the print quality assessment system.

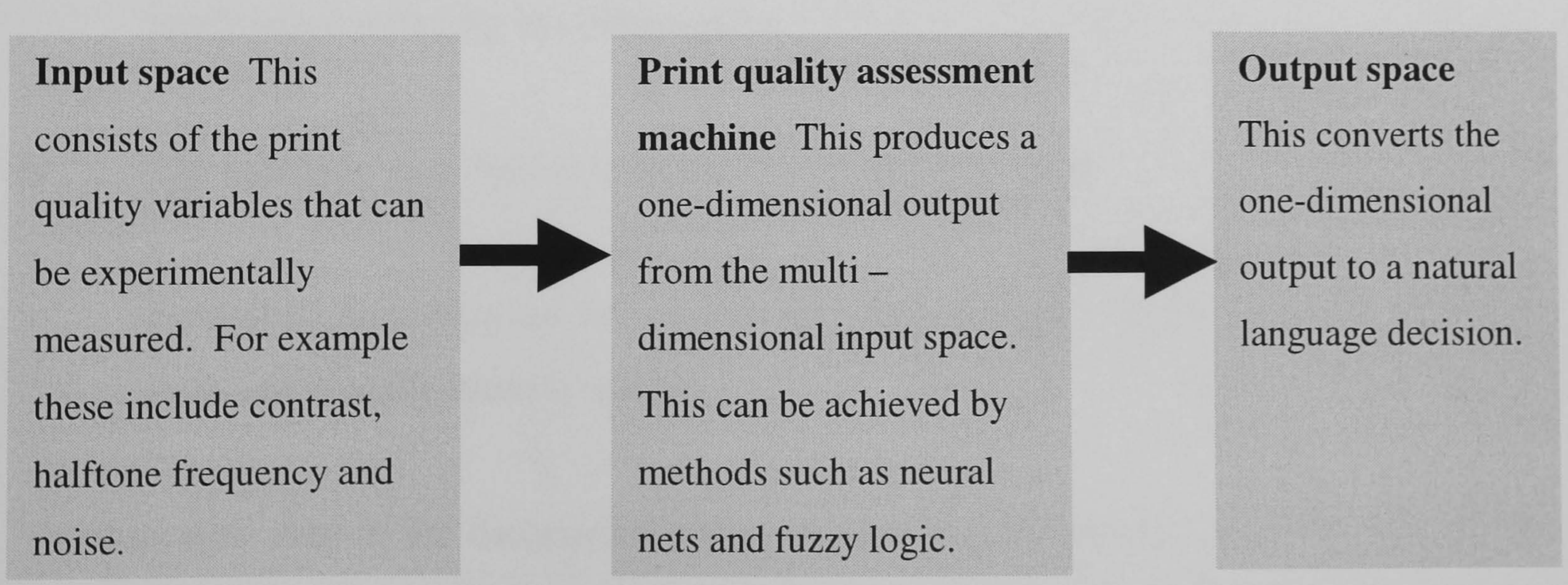


Figure 4.1. A flowchart representation of the tasks that must be performed to produce the print quality assessment model.

4.1 Turing's Test [8,36,37]

Turing's test was devised to see whether machines could possess human intelligence. Turing's test is only a partial test of human intelligence. Although it is an incomplete test, it does expand our understanding of both AI and human intelligence. In this project it helps to formulate a print quality definition.

In the original test proposed by Turing, an interrogator is asked to differentiate between the response of two computer terminals one of which is operated by a deceitful man, and the other by a helpful woman. The interrogator does not know who operates which terminal. The interrogator types out questions to the two terminals and from the responses given by the two respondents has to decide which terminal is operated by the man and which by the woman. The man is now replaced by a computer that is programmed to behave like a human and the process repeated. After both sets of trials have been carried out, an analysis of the interrogator's success rate would determine whether the computer is intelligent. Turing's test is an incomplete test of intelligence for the following reasons:

i It is only a test for artificial intelligence and does not encompass human consciousness. It is only concerned with output responses.

ii In the original test, the interrogator is only allowed five minutes to establish whether a machine or a human operates a terminal.

iii Any object (a hat, chair or a carrot for example) can replace the computer and be tested for intelligence. The objects, although giving no response, may still receive intelligence rating by the interrogator.

iv The computer may be consistently chosen as the woman.

v Perceptual responses are not considered. For example, the computer is not told to look and visually identify objects.

The concepts used in the description above show problems that are not dealt with in the Turing test. They also illustrate the complex nature of human intelligence. It is for these reasons that the AI problem is tackled by splitting it into many smaller and simpler

components called domains. For example, a face-recognition system may represent a domain and a system that recognizes smells is another domain.

The literature survey of **section 3.3** shows that application of AI has been neglected in print quality research. The work mainly has concentrated on physical measurements of print quality using image analysers. Comparisons of these physical measurements with observer assessments have also been made. This project combines knowledge from cognitive psychology and AI with practical techniques using image analysers to advance print quality research. **Figure 3.1** illustrates the image analysis methodology used for print quality measurements from the literature survey in **section 3.3**. The principle is to capture image data using a CCD camera or scanner and then process this data using a personal computer. The approach taken here is the same as that of previous researchers cited in this thesis together with:

i A specially written pre-processing computer program for an image analysis system that analyses print quality variables. It will have features that are currently unavailable in commercial image analysis or print quality software.

ii An artificial intelligence element represented by the neural network and fuzzy logic stage in **figure 3.1**. The function of the neural network and fuzzy logic stage is to take processed image data from the pre-processing and to create a model of subjective print quality assessment by observers.

It has been shown in **sections 2.0-2.5** that the neurophysiological and cognitive processes that are required by an observer to make print quality assessments are complex and not fully understood. This means that to simulate human print quality perception without constraints would be, with current technology, an impossible task. The following part of this section deals with imposing constraints on the variables to simplify the problem so that meaningful measurements can be made using currently available technology. As a starting point, an observer-based definition of print quality has been formulated and will serve as a useful foundation for subsequent research.

4.2 A Simple Observer Based Definition of Print Quality

Print quality assessment by observers can be defined using the following procedure:

Step 1

Two images A and B of equal dimensions reproduced from the same original using the same or different processes are shown to an observer as in **figure 4.2**. Any printing process can be considered. The inkjet and photocopier depicted in the diagram are only examples.

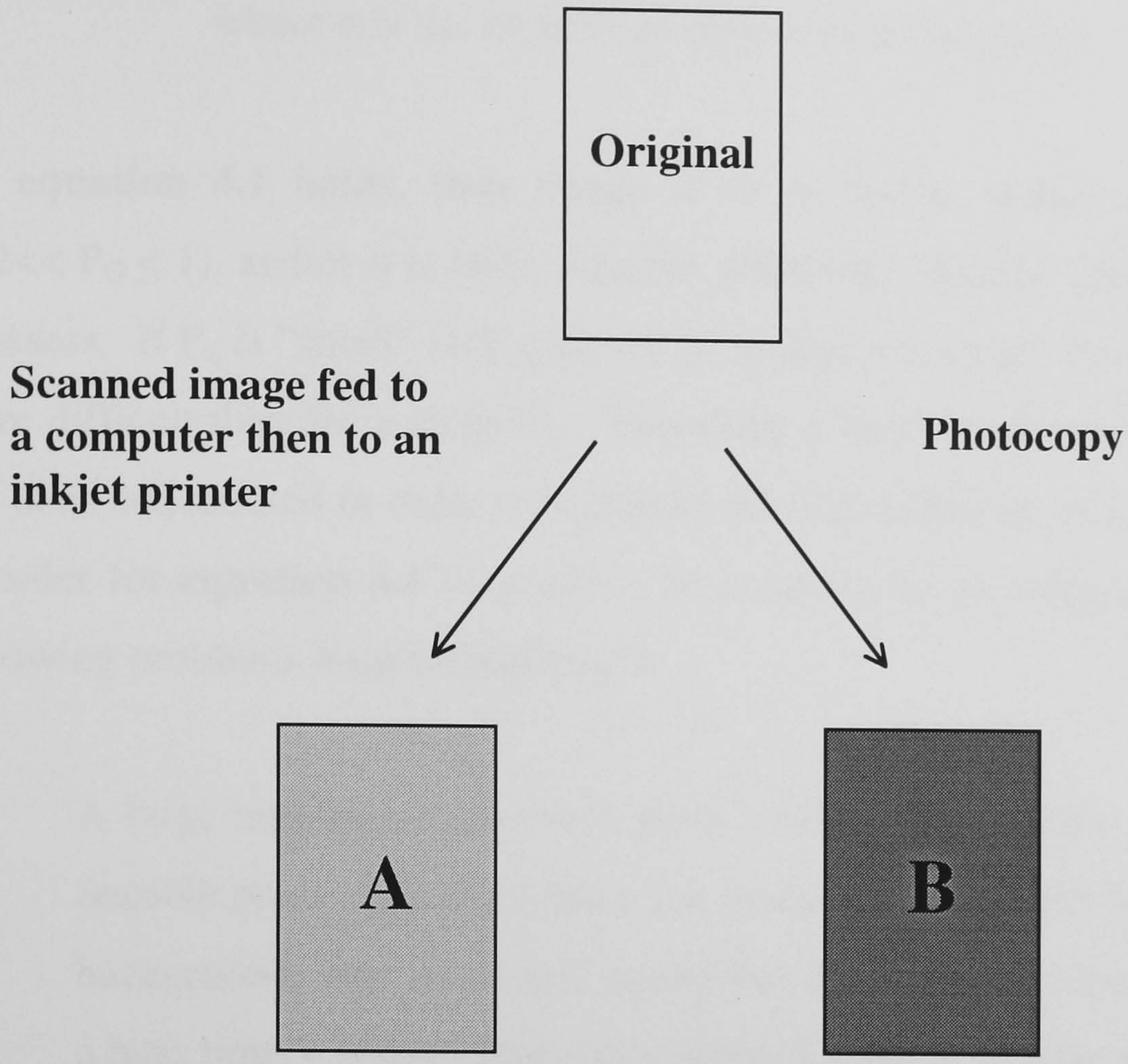


Figure 4.2. The starting point for a simple definition of print quality that will be used in this project.

This definition of print quality removes the option of comparing images of differing sizes (this refers to intentional alterations and not to differences that may appear due to errors in the printing process) and comparing differing image objects. Introducing these two constraints will simplify the experimental procedure, both in psychological analyses and the algorithm development stages later in the project.

Step 2

An observer O compares images A and B under controlled conditions of, time of observation, T; illumination, I; and distance of observer to images, D. If an observer O states that image A is of higher print quality than B, under specified conditions of T,I and D, then the result for that particular observer will be accepted as true. This result will be denoted as O_A. Under these testing conditions and for a group of observers, the following definition of print quality P_Q may be applied:

$$P_Q = \lim_{n \to \infty} \frac{1}{n} \sum_{n=1} O_A > \frac{1}{2} \qquad \textbf{Equation 4.1}$$

where n is the number of observers in the group.

If **equation 4.1** holds, then image A is of higher quality than B. If P_Q is "large" ($1/2 << P_Q \leq 1$), and/or n is large then the difference in print quality will be large and easy to assess. If P_Q is "small" ($1/2 \leq P_Q << 1$), and/or n is small, the assessment results will be more difficult than for a large P_Q. Therefore n must be increased. This simple definition has been constructed in order to highlight the difficulties in measuring the P_Q of an image. In order for **equation 4.1** to produce an accurate or meaningful measurement of P_Q the following problems must be addressed.

i A large number of observers from a wide cross-section is used. This may not be feasible practically since there are many variables such as age, gender, educational background, race and visual acuity that must be considered. To reproduce results at a later time, a similar group of observers must be reassembled.

ii A major constraint in **equation 4.1** is that the distance of the observer to the image and lighting conditions must be specified. This puts an extra burden on problem 1.

iii The decision of the assessment by the observers is absolute and not a function of intrinsic psychological and physical variables that can also be different for different observers. A non-exhaustive list is given below:

a Observers randomly focus attention on different parts of images for differing periods of time at random.

b Observers may ask questions and start "learning" about print quality and become "experts" in this field.

c Observers are prone to tiredness and can become less motivated. Therefore even using a single observer can produce random or systematic variations in the results.

Equation 4.1 does not yield information about print quality perception since no physical measurements of the printed images or observers have been made. Also **equation 4.1** and the associated problems of using observers to measure print quality show that it is difficult for different researchers to agree upon and reach a universal definition of P_Q using only observers. In order to overcome these difficulties it is proposed to develop a framework based on a practical, computational engine that can resolve these problems.

To fulfil the objective an image analysis system that exhibits the external behaviour of human visual perception will be developed. To test the system, Turing's test for artificial intelligence will be applied and neural network and fuzzy logic technology will be used to implement it.

This project takes the approach of developing a system to assess samples of images from a range of non-impact printers. The concept here is to see whether the system can generalise at a later stage. This means whether it can assess other print samples successfully that it has not previously encountered. If it cannot achieve this for a particular image then an attempt will be made to see whether it can learn the correct response in the same way that a human learns the correct answer to a problem. A discussion of the tools used for the development stage will now be given starting with the hardware.

4.3 Image Analysis [13,38]

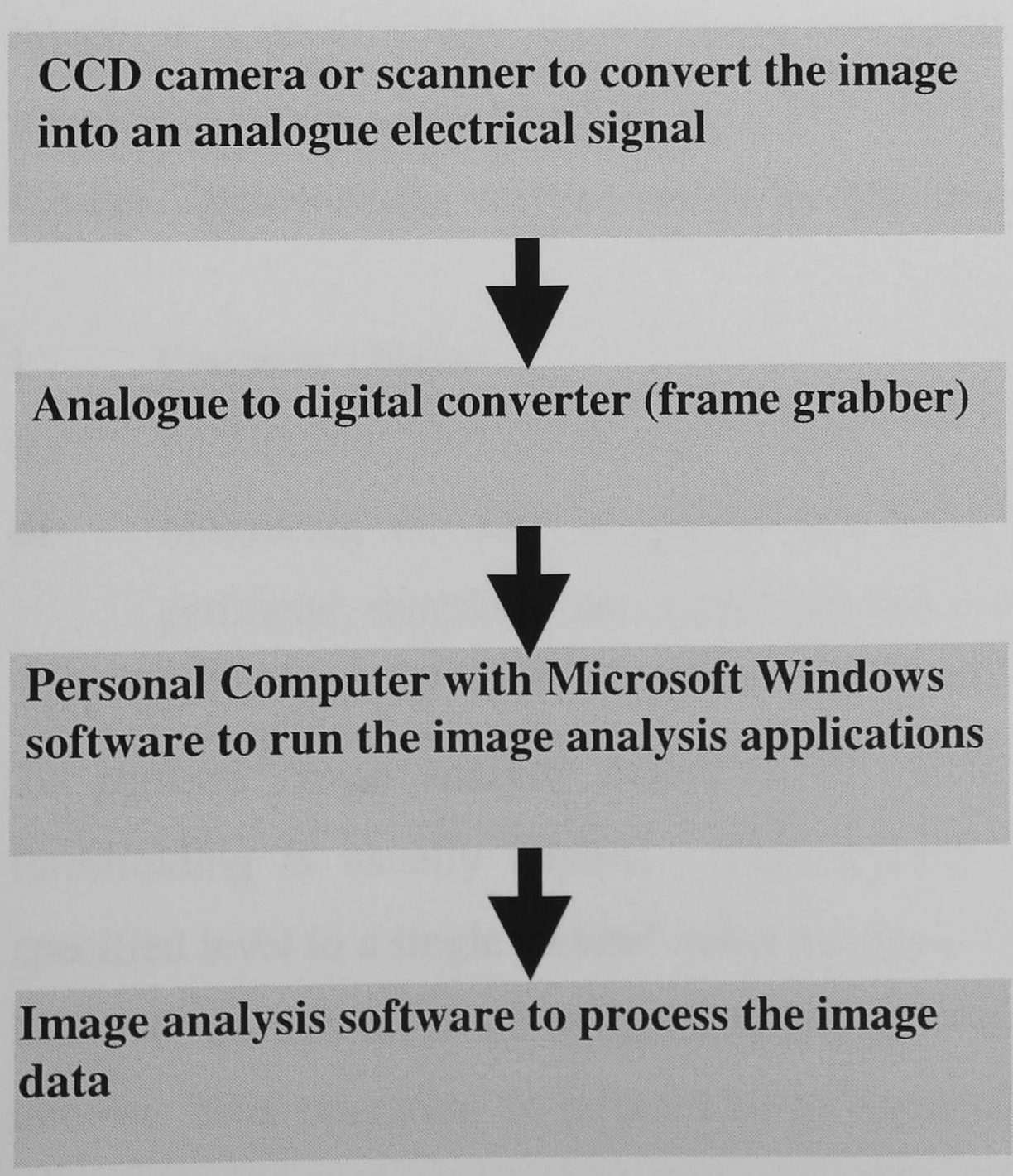


Figure 4.3. A block diagram of the stages of an image analysis system.

The arrangement shown above in **figure 4.3** is a block diagram of the basic components of an image analyser system. They have been extensively used as a tool for print quality research, as has been described in **section 3.1**. There is also a wide range of image analysis software for them An evaluation of commercial software has been carried out in this project. Image analysers are also highly versatile instruments that are used in a variety of applications. The following list indicates some areas where image analysis is employed:

Industrial inspection	Component recognition
	Component fault detection
Medical Science	Dermatology
	Neurology
	Muscle cell analysis
Paper Science	Texture analysis
	Print quality measurements

The image analysis software currently available is capable of many different measurements of the features of images. In order to do this, it can also perform a wide range of image processing operations. It is important at this point to recognise the difference between image analysis and image processing. The former deals with the collection of data from an image; it is about image quantification and yields a result in non-image terms. Image processing is concerned with how to improve an image to facilitate easier interpretation by the eye. Typical image analysis operations that are possible include:

i Counting objects

ii Performing greyscale histogramming

iii Measuring the size of objects and holes within objects. These include area, perimeter, roundness and aspect ratio measurements.

To perform image analysis measurements, an image processing operation known as thresholding is usually applied. Thresholding assigns all greyscale values above a specified level to a single "white" value and these regions are treated as background. Those below or on this level are assigned to a single black value and are treated as regions of interest. This operation is referred to as binarisation. It is possible, by changing the

threshold value, to remove unwanted features of an image so that image analysis operations can be performed only on the objects of interest.

There are other image processing transformations that are available with image analysis applications. Background correction of uneven lighting can be achieved by using image subtraction of a control image. There are also filters that can be applied to images to remove high or low spatial frequencies. These use operations such as Fourier and Laplace transforms**[38]** which convert spatial information into a sinusoidal frequency spectrum. Therefore, an image can be transformed by converting it into frequency information, selectively filtering out frequencies and converting it back to its original spatial form. An example of a practical application of filtering is the removal of scratches on an image. These scratches produce high frequency components when the spatial image is converted into frequency information. These high frequency components can be removed before reconverting the image back to the spatial form, which will now be absent of the scratches.

Commercial software applications are usually designed to be fast and efficient at analysing images. All the applications that have been encountered in this project operate under Microsoft Windows which is important for software integration, and accept a wide variety of image formats such as TIFF, GIF and BMP. They can all be operated using macros. A macro enables a series of operations that require many manual keystrokes on a computer to be recorded and played back using a single keystroke. This feature is important if hundreds of images are to be analysed automatically.

The use of this type of equipment to make print quality measurements has already been referred to in **section 3.3**. Part of the investigation is to determine whether commercial analysers and software which are currently available can make accurate, fast and reliable measurements that can be correlated to subjective print quality when analysing images of entire objects. An entire object is defined here as a simple shape or text character. The efficiency in analysing the data is important, since a large amount of data will be needed to produce meaningful results. The usefulness and limitations of the available hardware and software in the context of print quality measurements will be discussed later.

4.4 The CCD Camera [38]

In order to capture an image for analysis, a CCD camera is frequently used. A CCD (charged coupled device) is a semiconductor shift register. In the case of a CCD camera, it

consists of a two dimensional array of cells. Each cell holds a discrete signal value that is dependent on the incident light intensity. The frame grabber in the computer acquires the camera signal by shifting the data serially into its memory.

A major problem with CCD cameras, which can introduce errors into the acquired data, is noise. Quantum, electrical and thermal fluctuations, and impurities within the CCD crystal cause this noise. Other possible problems include smear; this is caused by the continued integration of light after the charges have been shifted. This elongates a spot of light into a longer stripe. Blooming can also occur when only part of the image is illuminated above the saturation level; the area of saturation increases due to diffusion of the excess charge. These possible defects to CCD cameras indicated here show that the image data must be carefully analysed so that camera noise is not mistaken for print defects. A large part of the data processing will involve analysing and removing or minimising the errors caused by the camera defects.

The image analysis system described above can only process data to give values for individual print quality variables. It cannot combine these effects to produce multi-variable models as has been cited in **section 3.1**. To achieve this combination, neural and fuzzy logic technology will be used. This technology is described in the following section.

4.5 Neural Networks [39,40,41,42,43,44,45]

An artificial neural network (ANN) can be considered as a self-adaptive artificially intelligent tool that is an aid to solving problems that are deemed too difficult to overcome by other means. Neural networks can also be considered as a mathematical tool used to model complex problems in many different subject areas, such as Engineering, Economics and Medicine. A main feature of these networks is that they can be implemented on a computer. ANNs consist of interconnected units called neurons or perceptrons. The function of a perceptron can be considered analogous to that of a biological neuron in the brain described in **section 2.3.** They both work using the same connectionist principles to solve problems in the areas of pattern recognition, classification, prediction and novelty detection. A major difference between the two is that the artificial network architectures are extremely simple in comparison to that of the brain. The brain has over 10^{10} neurons each with over 10^4 connections while current ANNs have a few hundred neurons with a few thousand connections at the most.

As a scientific discipline, neural networks originated in 1943, with the development of a mathematical tool known as a perceptron or artificial neuron by McCulloch and Pitts **[39]**. However single perceptrons can only solve linearly separable problems, which posed severe limitations on their practical uses. It was the discovery of the multilayered perceptron (MLP) which utilised the Backpropagation Rule by Rummelhart, McCelland and Williams **[39]** that enabled non-linearly separable problems to be solved. This greatly enhanced its versatility as a statistical problem-solving tool. Also it was the discovery of the MLP, that has made neural networks a large area of research. There is continuous development in this field by mathematicians and their work is applied by experimental scientists and engineers to real, practical problems. Neural networks have been successful in solving complex engineering problems. In the field of pattern classification an industrial meat- grading system has been successfully implemented in Denmark **[44]**. It has been claimed that this system can grade meat more finely than by expert visual inspection. Some of the diverse research areas of neural technology are: face recognition, classification of cells in cervical smears, robot control, multiphase flow monitoring in oil pipelines, electrical load forecasting and weather forecasting **[45]**.

It must be emphasised that neural networks are only statistical modelling tools and have their limitations as well as advantages. The characteristics of neural networks are briefly summarised in this paragraph. Neural networks can help solve problems in which the rules are unknown. This usually occurs when the number of dimensions in a problem is large and hence difficult to visualise by humans. They can still work when data is missing or incomplete, albeit with lower efficiency. When the network does make a mistake, it can use this knowledge to create a better model. In other words, it learns from its mistakes. The neural network's ability to work with incomplete data means that it can generalise. In other words it can give the correct response to data values that it has not previously encountered.

However, their outputs can be difficult to explain for the reasons that high-dimensional problems are being dealt with and the network weights only describe empirical relationships. Neural networks frequently work badly when irrelevant data as well as relevant data needed for the model is inputted. The irrelevant data increases the number of dimensions in the network solution unnecessarily. A greater number of dimensions requires a larger dataset to fill the higher dimensional space to produce an accurate mapping. It is necessary therefore to discard irrelevant data before implementing a neural

network. This is an important concept and is frequently referred to as pre-processing and will be extensively referred to later.

An initial discussion on the basic principles of neural networks will now follow. In view of the diversity of neural technology, only the areas of this topic that is important for this project will be covered here. The relevant areas come under the subject heading of pattern recognition.

4.5.1 The Single Neuron [39,40,41,42,43,44,45]

A very simple pattern recognition problem is the 2-dimensional linearly separable problem in **figure 4.4**. Two variables that are relevant to this thesis, halftone frequency and contrast have been chosen for illustration purposes. The vectors **1** and **0** represent the results of these measurements made on a sample of images. The **1**s and the **0**s denote better and worse image quality than the images which have vectors that lie on the line.

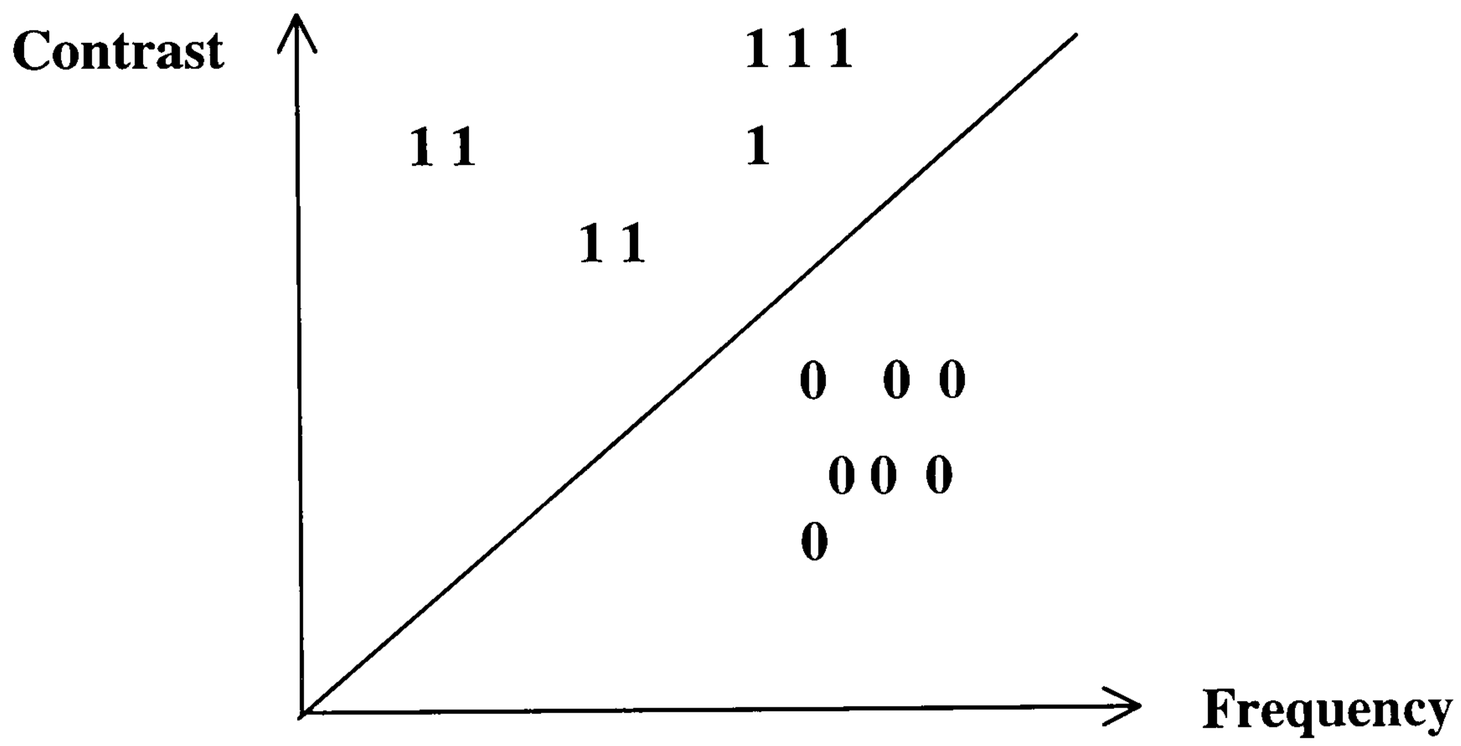


Figure 4.4. A linearly separable classification that can be solved using a single neuron.

The problem is how to find the correct solution that separates the two classes. A single neuron, known also as a perceptron, can solve this problem and also other linearly separable problems that are in higher dimensional space. **Figure 4.4** is a diagram of a perceptron. The inputs P_1 to P_n are parameters that might, in this project, represent contrast, halftone frequency, noise, and the length to width ratio of a print character. For the problem in **figure 4.5** only two inputs P_1 and P_2 would be necessary. These inputs are multiplied by their corresponding weights W_1 to W_n to give an output response that can be compared and subsequently corrected using the true target response provided by the observer assessment.

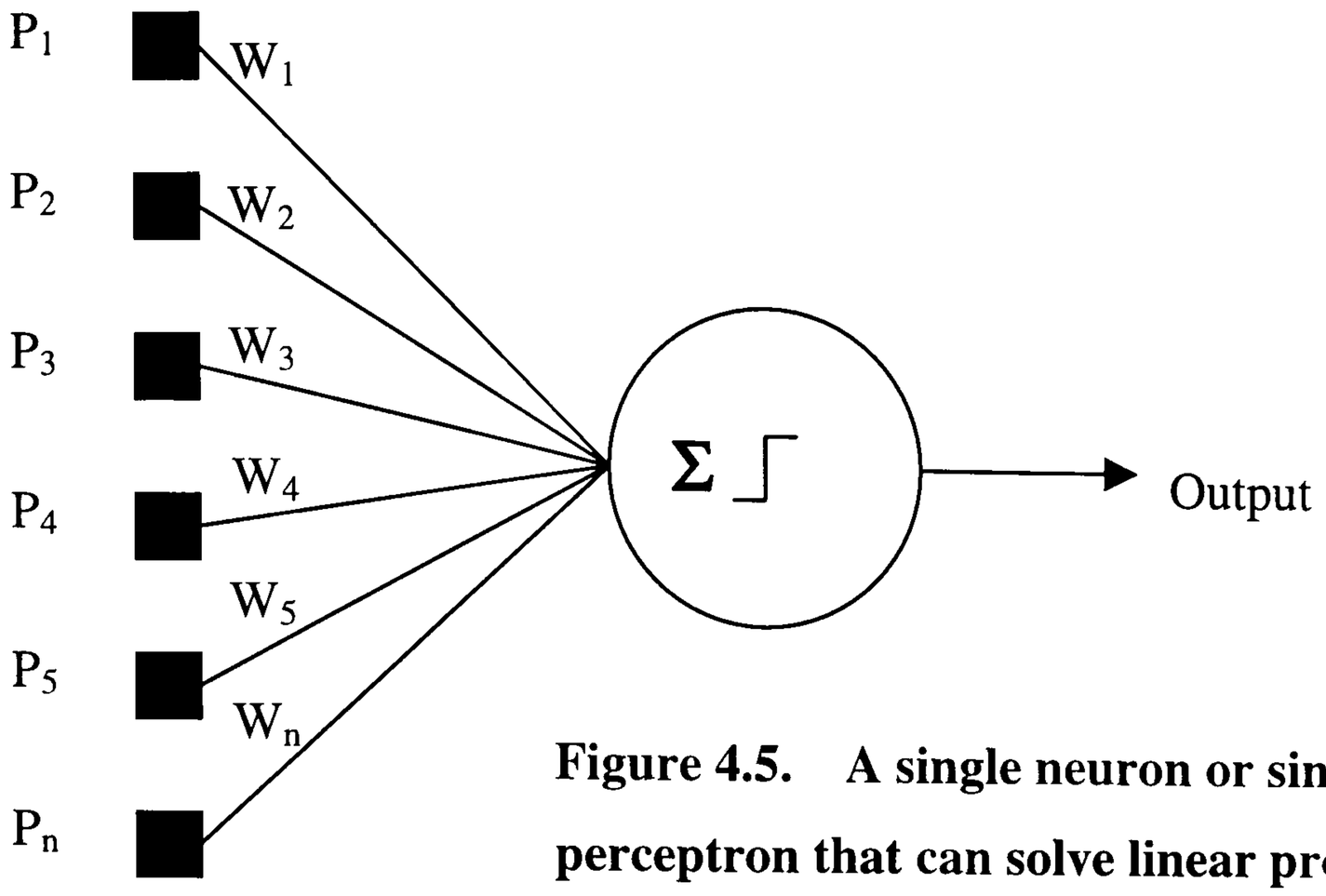


Figure 4.5. A single neuron or single-layered perceptron that can solve linear problems.

A single neuron enables a linearly separable problem to be classified in the following way. A function y sums each input P, multiplied by it's weight, W

$$y = \sum_{i=1}^{n} W_i P_i \qquad \textbf{Equation 4.2}$$

This sum is then compared against a threshold value θ, assigned to the neuron. If $y > \theta$ then the output is 1 and if $y \leq \theta$ then the output is 0. This is known as hardlimit thresholding and is illustrated in **figure 4.6**. It can give a value of only 0 or 1 when it passes the θ threshold. If the resultant output does not correspond with the target response, then the weights of **equation 4.2** are adjusted to give the correct output value. This form of error correction is known as supervised learning.

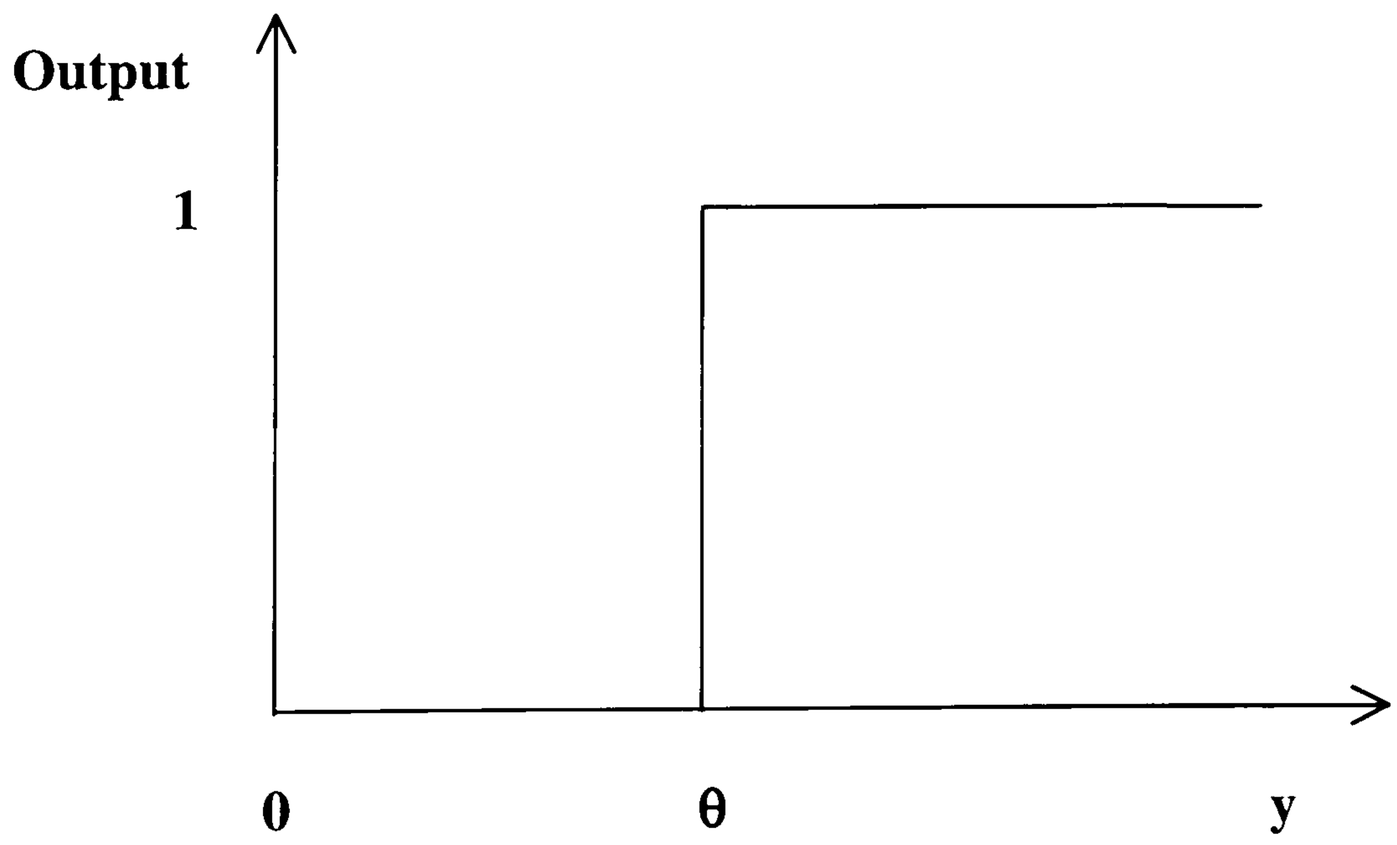


Figure 4.6. The hardlimit threshold for the single layer perceptron.

In order for the single perceptron shown in **figure 4.5** to solve problems, a learning algorithm that can be solved on a computer must be formulated. The learning algorithm for the single perceptron is shown below. The η term is known as the learning rate and has a value $0 \leq \eta \leq 1$. The learning rate value is a critical parameter. If η is small then the solution will be found slowly. If η is large then oscillations that slow down the learning process can also occur.

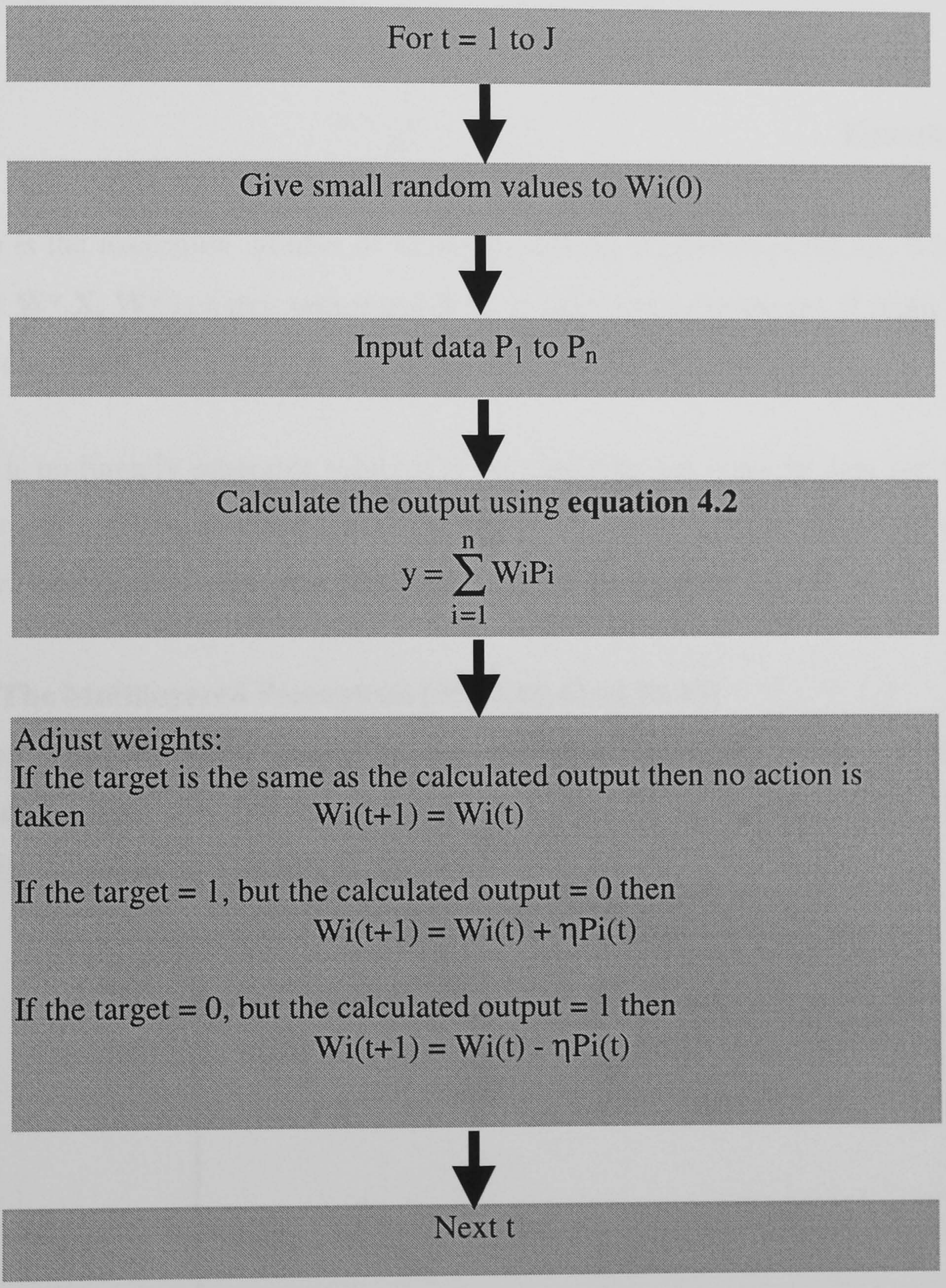


Figure 4.7. A learning algorithm for a single perceptron that can be implemented on a computer.

The learning algorithm above can be implemented in any computer language since it is a set of algebraic calculations that are iterated a finite number of times. The weights W, are

initially set to random values and each example is presented and iterated J times through the loop. As more passes are made through the perceptron the weights converge to a solution that separates the classes, but only if there is a linearly separable solution. There is a proof that shows that if a linearly separable solution exists then the perceptron learning algorithm will always find a fixed solution. It has been demonstrated that if there is a linearly separable solution in a classification problem, the number of iterations for all cases will be **[39]**:

$$n = \frac{1}{\delta^2} \qquad \textbf{Equation 4.3}$$

where n is the maximum number of iterations that the algorithm performs, δ is a constant that is < **W*.X, W*** is a unit vector and **X** is an input vector in the set of input vectors that is to be classified .

If there is no linearly separable solution to the classification problem then the weights will continuously oscillate showing that the perception will not find a solution. The next section discusses non-linearly separable problems and how they can be solved.

4.5.2 The Multilayered Perceptron [39,40,41,42,43,44,45]

Many problems in pattern recognition are non-linearly separable. An example of this is illustrated by the 2- dimensional exclusive OR (XOR) problem in **figure 4.8**. This problem cannot be solved using a single perceptron.

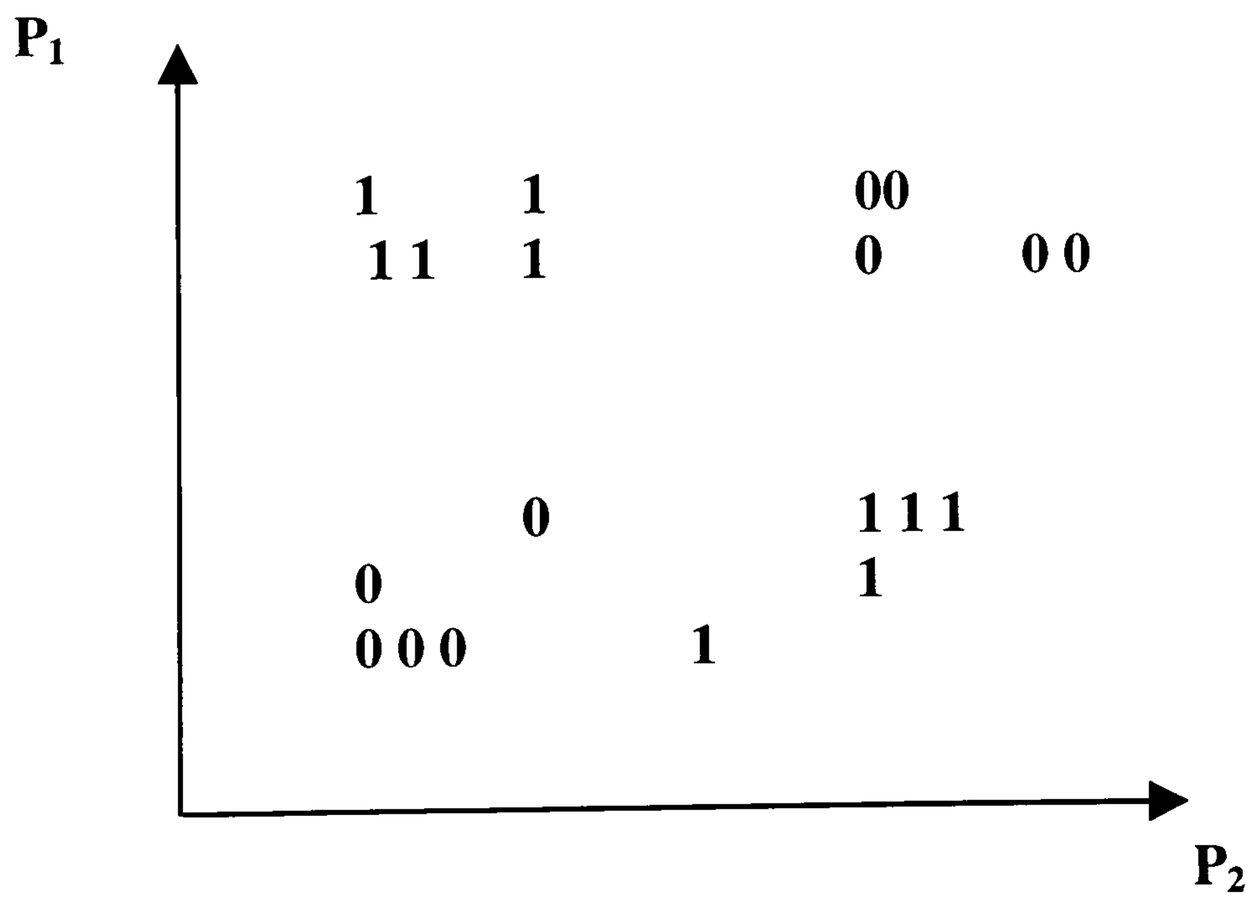


Figure 4.8. The exclusive XOR problem which the single neuron cannot solve.

A neural network with more than one layer of preceptrons is frequently referred to as a multi-layered perceptron or MLP. To solve the non-linear XOR problem of the previous section the following network structure with more than one perceptron layer illustrated in **figure 4.9** can be employed. The principle is to use two perceptrons to model separate linearly separable regions and to use their combined outputs as inputs to a third perceptron that gives the final classification. This type of network is known as feed-forward and although it can solve non-linearly separable problems it cannot find the solution itself for the following reasons.

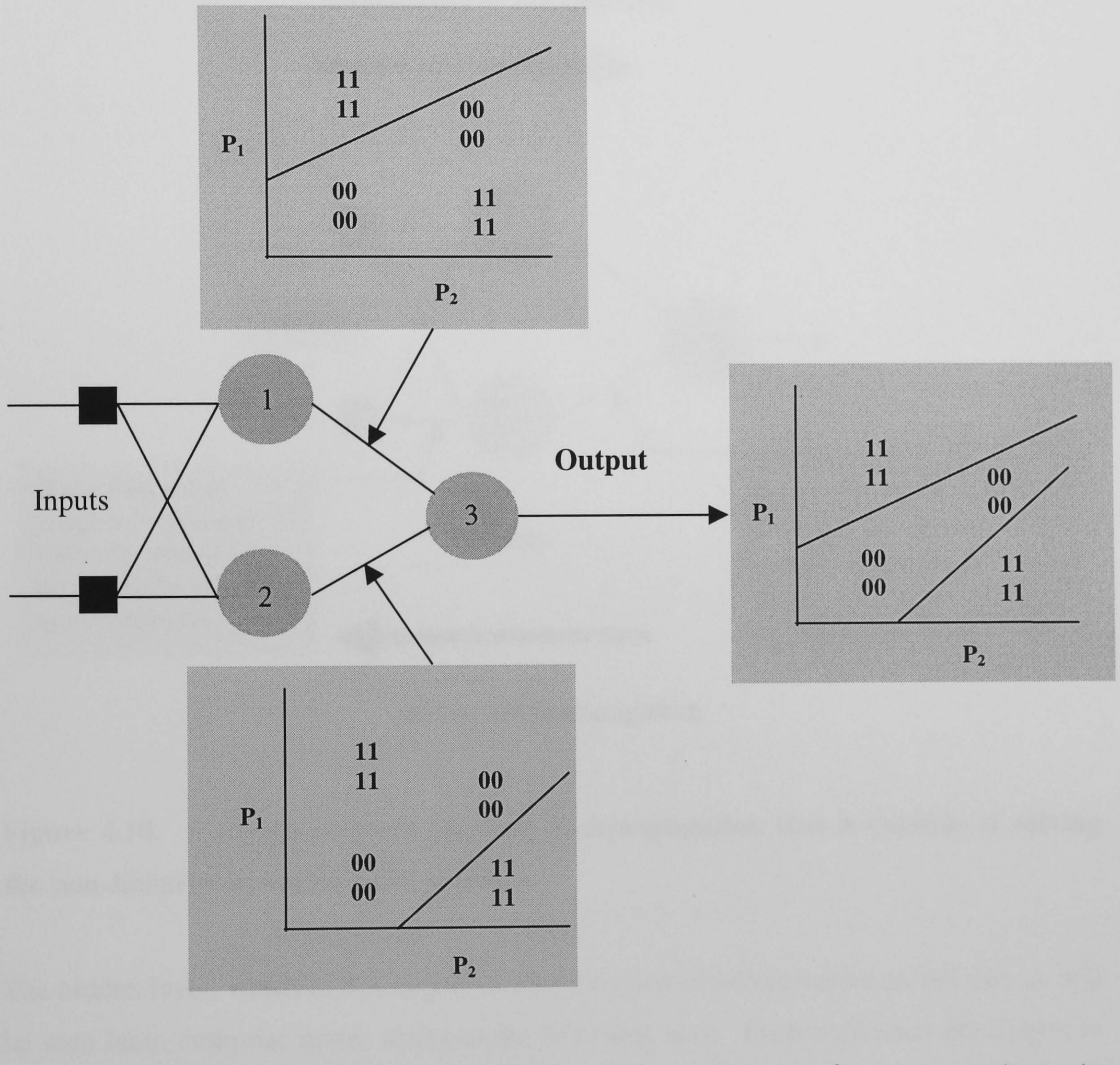


Figure 4.9. An attempt to solve the XOR problem using two input perceptrons to classify the two separate linearly separable regions for the third perceptron to produce the final classification.

The problems with the feed-forward network in **figure 4.9** are that the linearly separable solutions from perceptrons 1 and 2 must be known before perceptron 3 can make the final

classification. There is no way in which the output perceptron will know which weight to adjust since the actual inputs are isolated from it. The hardlimit thresholding function in **figure 4.6** for the single perceptron is also inappropriate for networks with more than one layer since the output neuron of the multi-layered feed-forward network will only have limited weight values (0 and 1) for classification purposes. These problems can be overcome with the same feedforward network architecture shown in **figure 4.9** but using a different algorithm known as backpropagation or the generalised delta rule to adjust the weights. The general principle of backpropagation is shown in **figure 4.10** below.

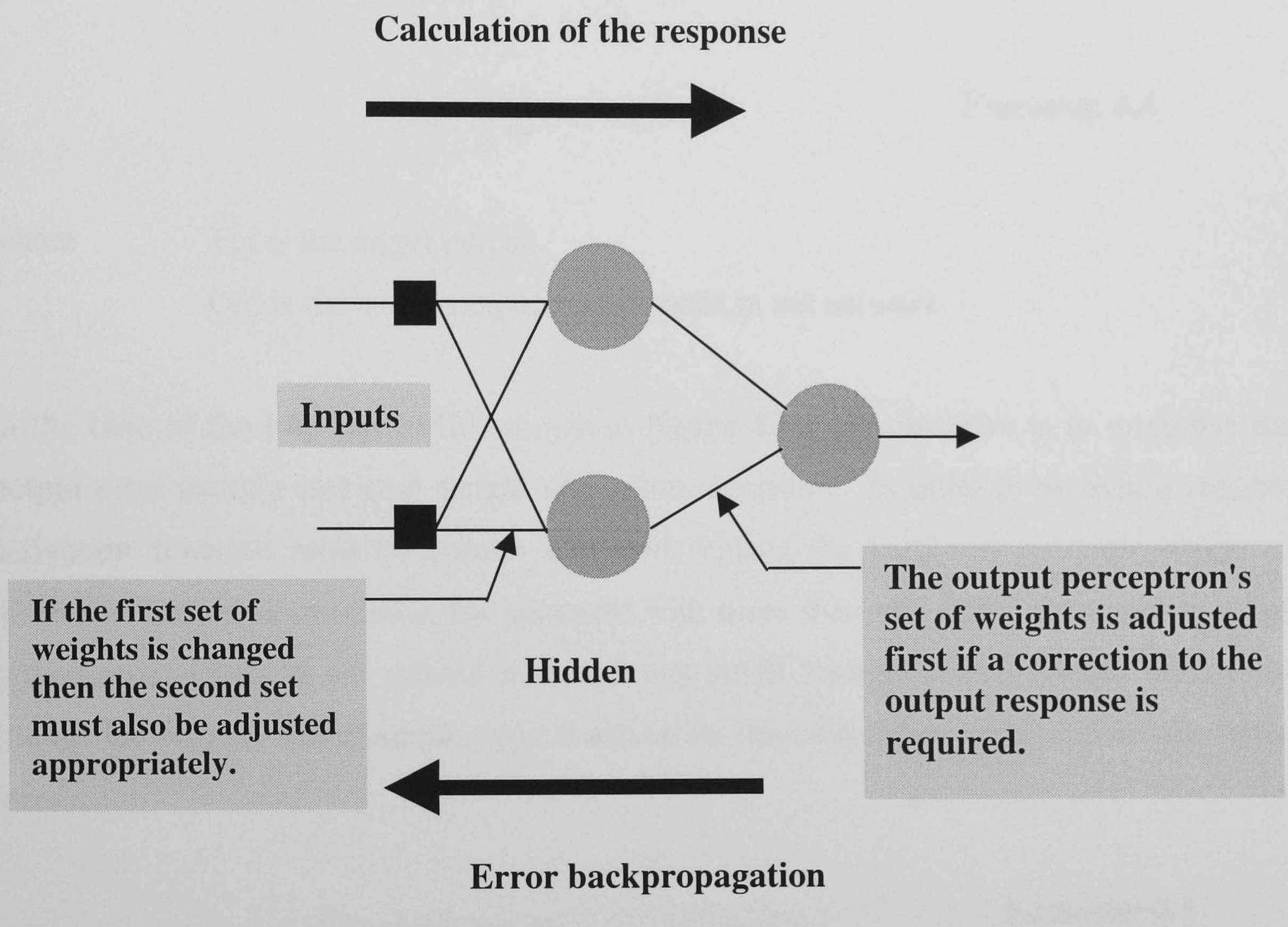


Figure 4.10. A neural network showing backpropagation that is capable of solving the non-linearly separable XOR problem.

The hidden layer, which in this simple example consist of two perceptrons, but can, as will be seen later, comprise more, works in the following way. Each individual perceptron in this layer solves a small part of the complete problem. It is the function of the output layer to assemble the smaller partial solutions into the final complete one. It is for this reason that the hidden layer is sometimes referred to as a feature detector and is analogous to the feature detectors in human visual perception which have been described in **section 2.2**.

The basic principle which backpropagation uses in solving classification problems is similar to that for the single perceptron and feed-forward networks in that a target response is compared with the actual response. If there is a difference between the output and target responses, the weights at the nodes of the network are adjusted to reduce this difference. In feed-forward networks the weights are corrected from the input towards the output. However, the multilayer perceptron corrects the weights from the output towards the input, hence the name backpropagation. The output error E_{pj} of a backpropagation network is defined as:

$$E_{pj} = \frac{1}{2} \sum (T_{pj} - O_{pj})^2 \qquad \textbf{Equation 4.4}$$

where Tpj is the target output

Opj is the actual output at each node in the network.

In the case of the two layer MLP shown in **figure 4.10**, the objective is to minimise the output error using a two step weight correction algorithm. In order to do this, a suitable activation function must be chosen that will replace the hardlimit function which, as discussed earlier, is unsuitable for networks with more than one layer. The choice of the activation function is not critical provided that small changes in the weight values can change the output. For example, typical activation functions often employed are the linear threshold:

$$f(y_{out}) = my_{out} + c \qquad 0 \leq f(y_{out}) \leq 1 \qquad \textbf{Equation 4.5}$$

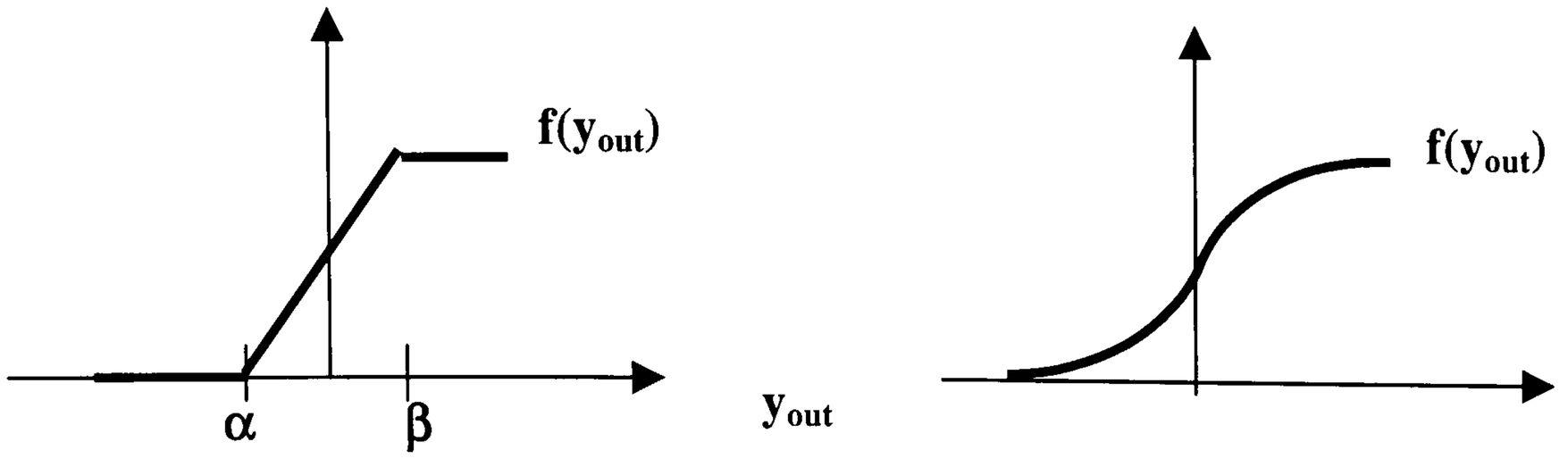
when $\alpha \leq y_{out}$ then $f(y_{out}) = 0$ and when $\beta \geq y_{out}$ then $f(y_{out}) = 1$

where α and β are constants and $f(y_{out})$ is the sigmodial threshold:

$$f(y_{out}) = \frac{1}{1 + \exp(-\lambda y_{out})} \qquad \textbf{Equation 4.6}$$

where λ is an arbitrary constant

These thresholding functions of **equations 4.5 and 4.6** are illustrated below in **fig 4.10 a and b** respectively.



(a). The linear function **(b). The sigmoidal function**

Figure 4.11. Graphs showing the shapes of (a), the linear and (b), the sigmodial functions that are often used to activate perceptrons in multilayered networks.

The method that solves the linearly separable problem for the single perceptron described in previous section is known as gradient descent. The weight adjustment equations are derived from the gradient descent equation which takes the form:

$$W_i = -\eta \frac{\partial E}{\partial W_i} \qquad \textbf{Equation 4.7}$$

This equation is also used to derive the equations used for the backpropagation algorithm. There are many text which give the derivation using the chain rule for these equations. The results of which are given in **equations 4.9** and **4.11** below and in **figure 4.12**.

$$\Delta W_{jk} = -\eta \frac{\partial E}{\partial W_{jk}} = -\eta \delta_k y_j \qquad \textbf{Equation 4.8}$$

$$\delta_k = (y_k - t_k) y_k (1 - y_k) \qquad \textbf{Equation 4.9}$$

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}} = -\eta \delta_j y_i \qquad \textbf{Equation 4.10}$$

$$\delta_j = \sum_k \delta_k W_{jk} y_j (1 - y_i) \qquad \textbf{Equation 4.11}$$

where W_{jk} and W_{ij} are the weights from nodes j to k and i to j respectively, y_i and y_k are the outputs of the hidden layer and output layer respectively. t_k is the target value and η is a gain term. δ_k is the error for pattern p on node k of the output layer. δ_j is the error for pattern p on node j of the hidden layer.

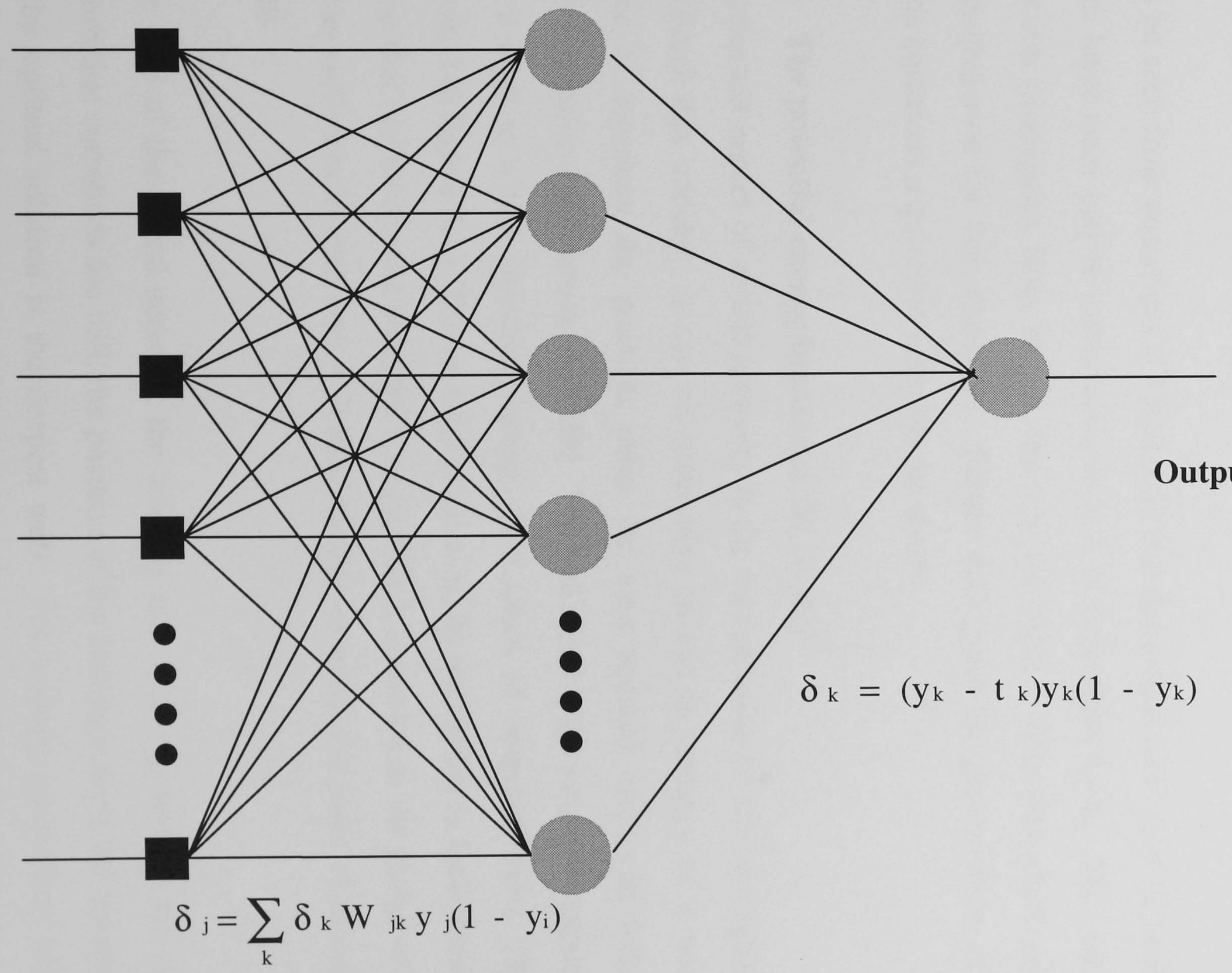


Figure 4.12. Architecture of the backpropagation neural network. The diagram also shows the equations required to adjust the weights so that the errors are reduced.

If the error for the output is required and a sigmoidal threshold is used then it can be calculated using the following summations. These have the same form as in the case of the single perceptron. Each layer calculates their weight summations and compares it with their respective weight functions F_k and F_j respectively as in **equations 4.12 and 4.13** below:

$$y_k = F_k\left[\sum_{i=0}^{n=1} W_i X_i\right] \qquad \textbf{Equation 4.12}$$

for the output layer

$$y_j = F_j\left[\sum_{j=0}^{n=1} W_j X_j\right] \qquad \textbf{Equation 4.13}$$

for the hidden layer.

It can be seen from **equations 4.9 and 4.11** that the output error must be known before the hidden layer error can be calculated, since δ_j is dependent on δ_k. This means that error correction propagates from output towards the input. This procedure gives the name backpropagation for this algorithm. **Figure 4.12** gives the general form of this network and the equations required to compute the errors.

4.5.3 The potential energy landscape [39, 41,42]

An important aspect of neural networks is the minimisation of the sum squared error. To understand this concept, it can be instructive to use an analogy of a potential energy surface to represent the possible paths the sum squared error can follow when the backpropagation algorithm is used **[39]**. The ball which represents the output value, moves from a high to a low potential energy via a series of potential wells. This analogy is illustrated in **figure 4.13**. The minima of these wells are known as local minima except for the one that has the lowest potential energy, which is known as the global minimum. If the ball has sufficient kinetic energy it may be able to reach the point of minimum potential energy.

In the case of the neural network, the learning rate η corresponds to the viscosity of the medium that surrounds the ball, the position of the ball represents the position of the error and the optimal solution is the deepest well. For a single perceptron this energy map always consist of a parabola with only the global minimum. For backpropagation however

the error surface will typically take the form of **Figure 4.13**. However for backpropagation, the existence of minima and therefore the possibility of convergence is, in general, not guaranteed. There is no proof in existence that can determine whether a particular backpropagation network will converge or not. This can only be determined by practical experimentation. This point emphasises the heuristical nature of neural networks.

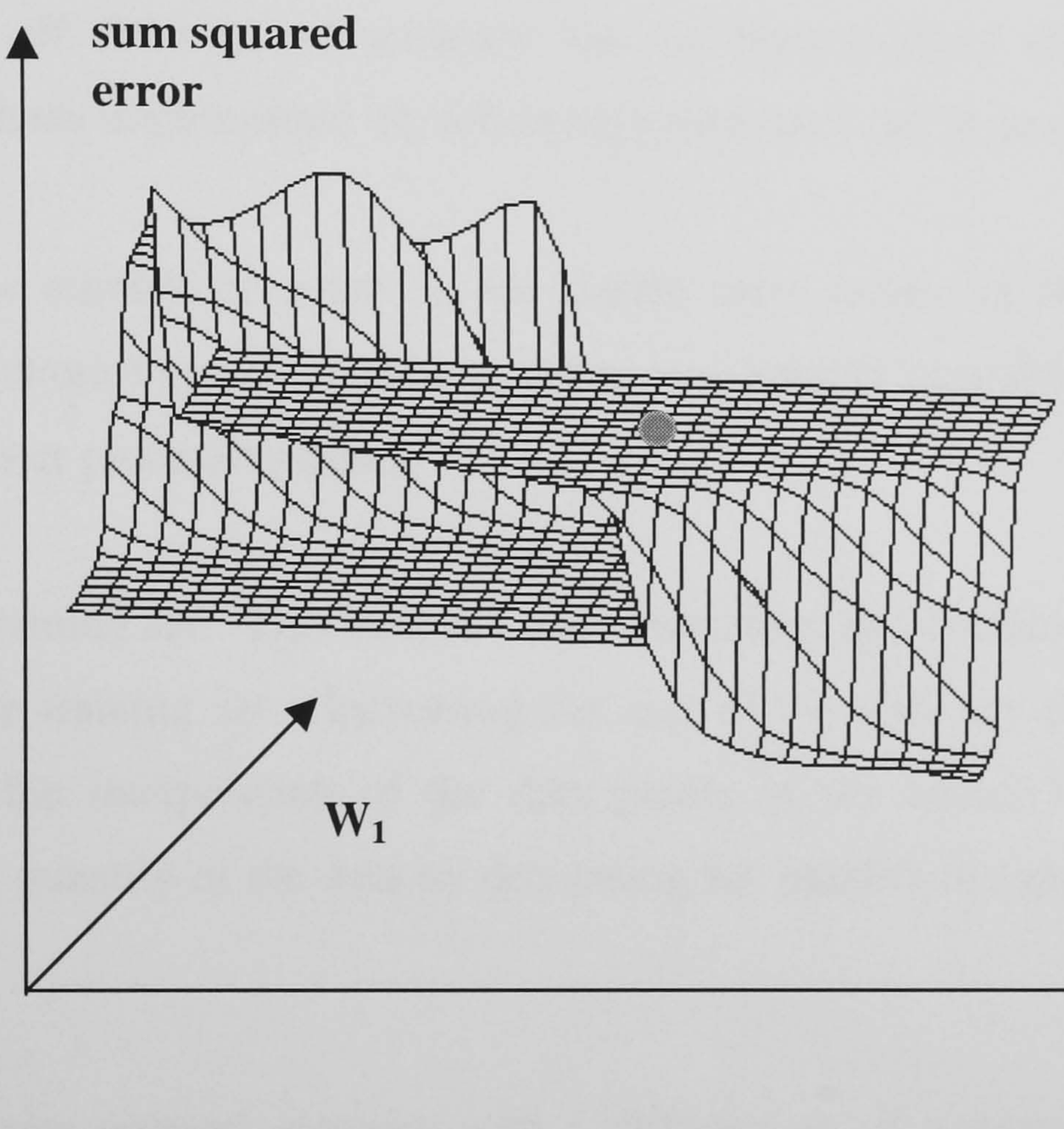


Figure 4.13. A 2-dimensional landscape of the sum squared error function for the adjustment of two weights for an MLP.

The learning rate parameter also highlights the heuristical nature of neural networks. If the learning rate is too small or large the optimal solution may not be found and the network will settle into a local minimum, from which it cannot escape. The concepts of learning rate and the gradient descent method for minimising network errors are very important in MLPs. These concepts must be understood, if backpropagation is to be applied successfully. It will also be shown later that there are error minimisation procedures such as the application of momentum and the Levenberg - Marquart algorithm that can help find solutions more efficiently. The following two sections will discuss the performance characteristics of backpropagation.

4.5.4 The backpropagation network training procedure. [43]

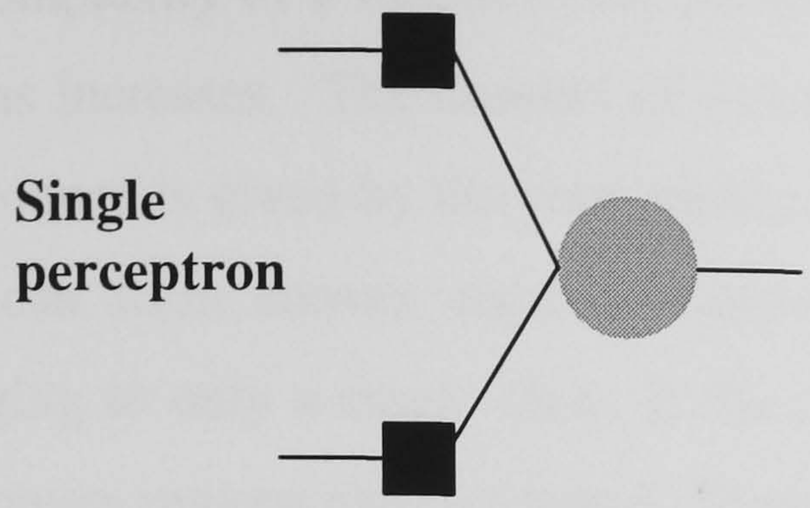
A procedure that can be used in practice to implement a backpropagation neural network solution is to present it with three sets of data. The first set is the training set. The neural network is trained on this set of data to produce an input to output mapping. This mapping is applied to another set of data that has not been seen by the network. This second set of data is known as the validation set and is used to assess the accuracy of the model produced by the training set. This is achieved by monitoring the error in the validation set when the network is in training. When the error in the validation set is at a minimum training is stopped. If the required accuracy has not been reached after the process described above has been implemented the following procedures can be used to obtain it.

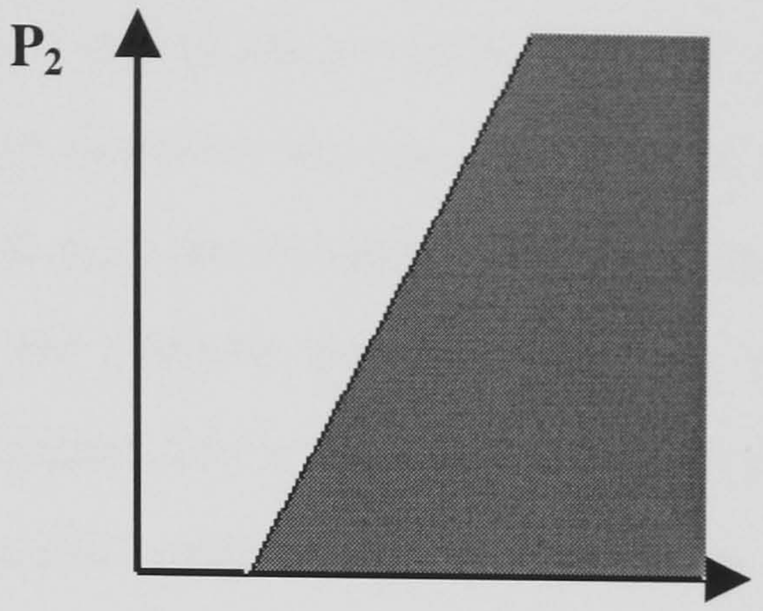
i Increasing the number of nodes in the hidden layer shown in **figure 4.12** will enable the network to produce more complex non-linearly separable models at the expense of faster processing times.

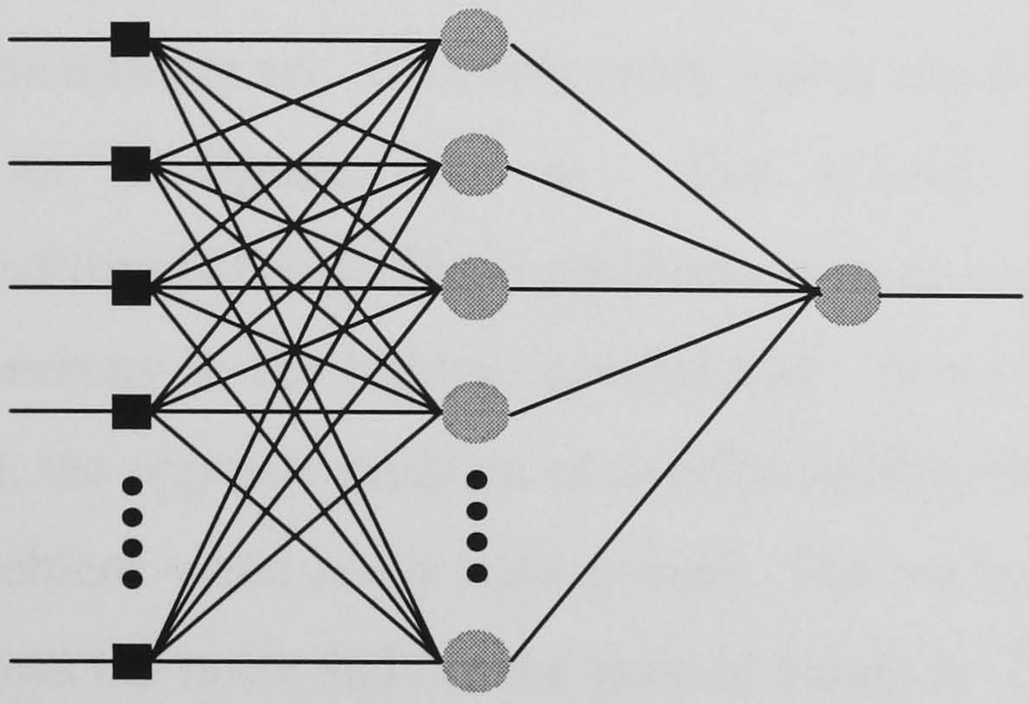
ii Modify the training set. This includes both increasing and decreasing the amount of data in the training set. Increasing the size of the data set can improve the accuracy of the interpolation of the data points in the neural network model. Reducing the quantity of the data by decreasing the number of input variables can also help.

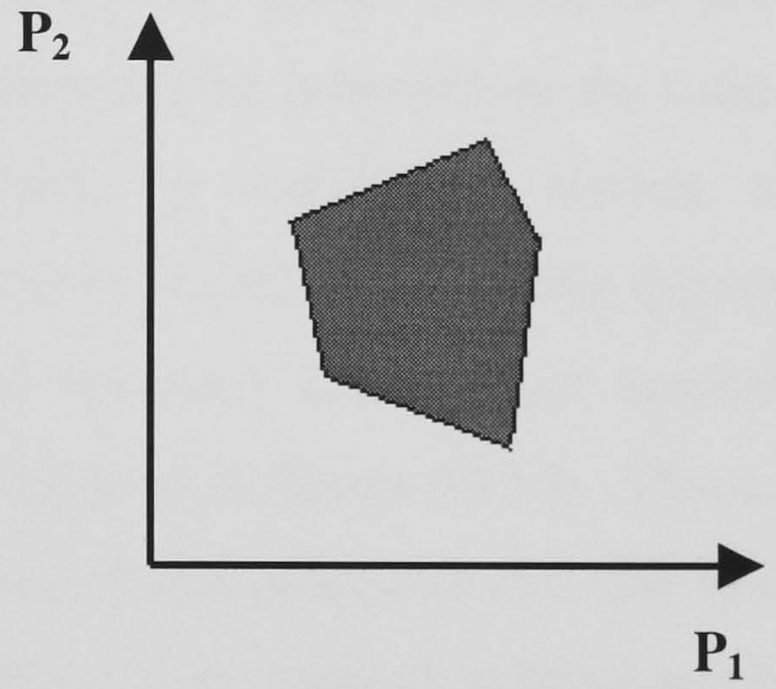
iii Restart the same network program with a different set of weights or change the learning rate of the neural network. The multilayered perceptron uses the gradient descent method for finding the correct mapping. A large learning rate can mean the network misses the correct solution. A small learning rate can mean that the program takes a long time to converge to the correct solution. Finding satisfactory minima can be dependent on the starting point of the network training process. This starting point is determined by the initial weights in the network. Changing the initial weights can change the final solution reached by the network.

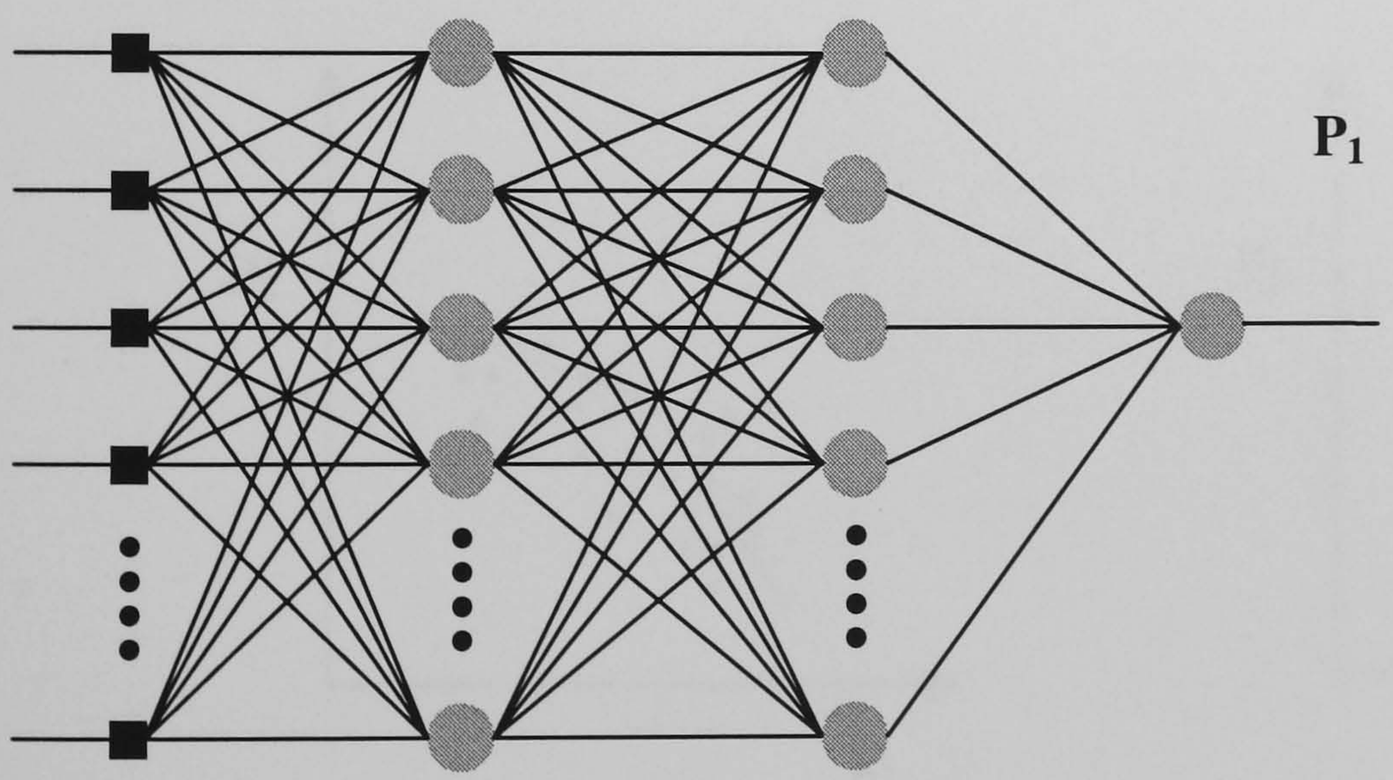
Once the sum squared error in the validation set is at a level that is acceptable for the application that the network is to be applied to, the weights in the network are frozen. Then the network performance is evaluated on another set of data that is new to the network. This data set is known as the test set and tests the generalisation capabilities of the network.











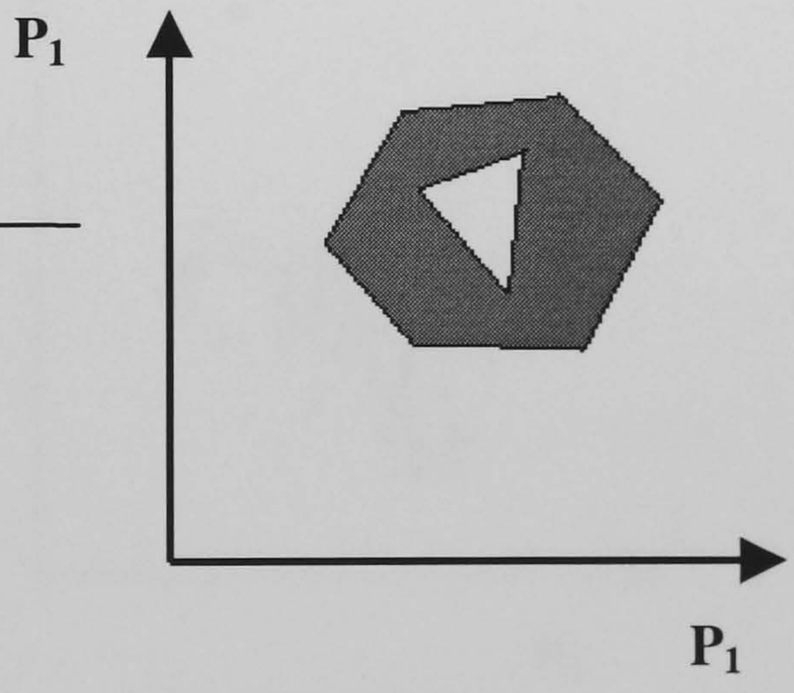
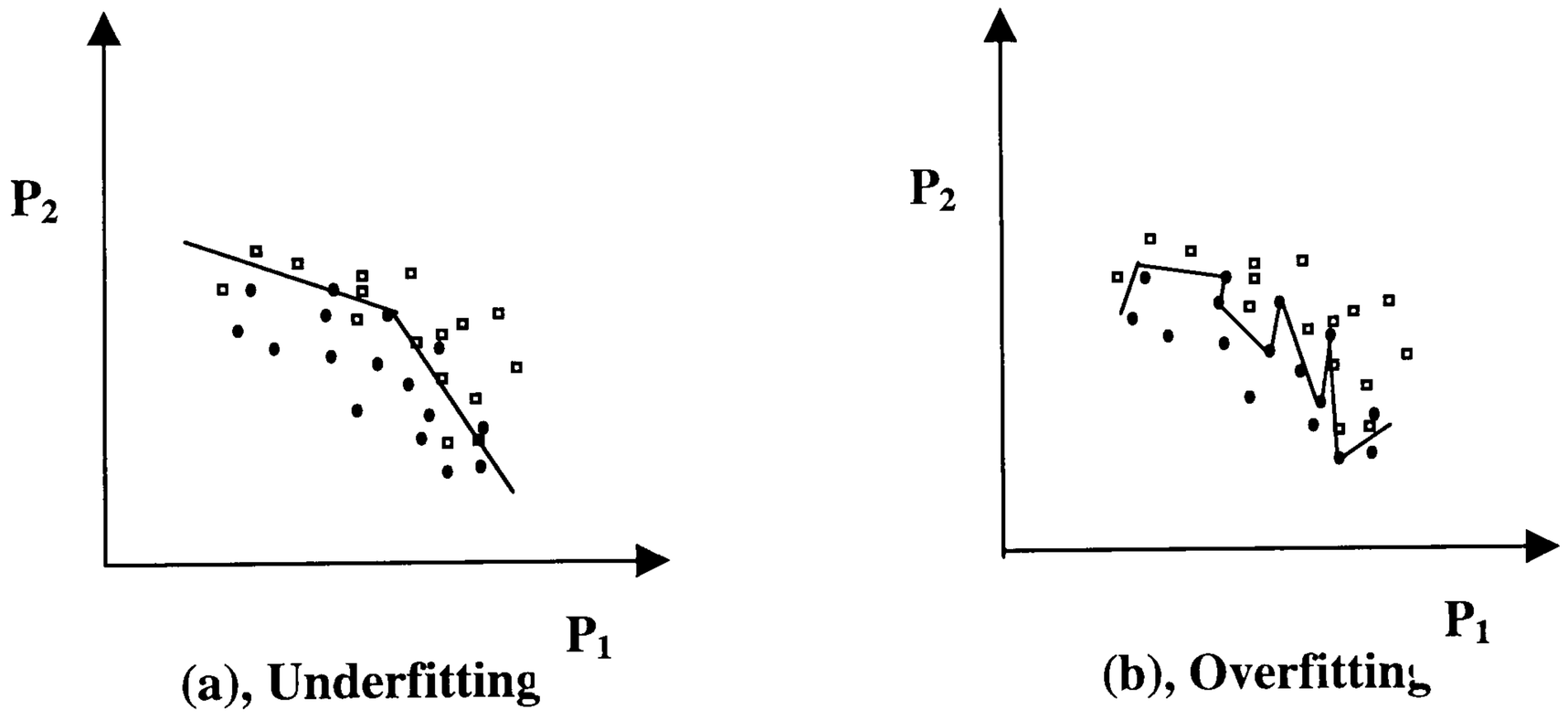


Figure 4.14. An illustration that shows the kinds of mappings that networks with 0,1and 2 hidden layers can produce. The middle example for the network with one hidden layer illustrates a convex region.

4.5.5 Decision boundaries and interpolation using MLP backpropagation [39,41,43]

The complexity of a function that an MLP can model increases as the number of hidden neurons increases. The number of linearly separable decision boundaries that a network can produce is given by the relationship n_h - 1 where n_h is the number of hidden neurons. They can form convex regions. Convex regions are regions that enclose data points belonging to only a single class. If the number of hidden layers is increased to two then non-convex regions can be formed. These regions are illustrated in **figure 4.14** above.

If too few hidden neurons are used to model non-linearly separable problems, two difficulties can arise. Firstly, there may be no solution to the problem using the network on the training set. Secondly, only a poor classification may be achieved on the validation set as in **figure 4.15 a**. The second problem is commonly referred to as underfitting**[41,43]**. These problems in some cases can be solved by increasing the number of neurons in the hidden layer**[41,43]**. However if too many hidden layer neurons are used, the opposite problem of overfitting**[41,43]**can occur as in **figure 4.15 b**. This can be a problem when noisy data is used. The interpolation is too precise in that the function follows the noise and not the general solution. Therefore, there exist an optimal number of hidden neurons between these two extremes that can only be discovered using trial and error. This problem also serves to highlight the heuristical nature of neural networks.



(a), Underfitting **(b), Overfitting**

Figure 4.15. (a), shows a function that underfits two classes of data and (b), shows a function that overfits two classes of data.

4.6 Radial basis function networks (RBF networks) [39,40,41,42,43]

In the preceding discussion, it has been explained that a single perceptron can only model linearly separable behaviour and that an MLP can model non-linearly separable behaviour using a hidden layer of neurons, which learns the appropriate functions through backpropagation. Another method that is frequently used in pattern recognition is the radial basis function or RBF. This technique can also model non-linearly separable problems. It does so by mapping data onto a higher-dimensional space so that a linear classification can be made **[40]**.

In the case of radial basis function networks, the hidden layer of neurons has fixed functions and the output layer performs the learning. The principle behind this network is to use the fixed functions to transform the data into a higher dimensional space, where the data is linearly separable. This is achieved by measuring the distance of the input vector to a fixed centre and applying a fixed function to this distance. This enables the output layer which is a single neuron to perform a linear separation. These fixed functions are called radial basis functions and have the general form:

$$Z_j(\mathbf{x}) = F_j(\| \mathbf{x} - \mathbf{a}_j \|) \qquad \textbf{Equation 4.14}$$

where ($\|\mathbf{x} - \mathbf{a}_j\|$) is a distance measure, $\mathbf{x}$ is the input vector and $\mathbf{a_j}$ is a centre from where $\mathbf{x}$ is measured from. F_j is a strictly positive radially symmetric function with a unique maximum at its centre $\mathbf{a_j}$ and is usually Gaussian of the form given in **Equation 4.15** below. σ_j is the standard deviation or width of the Gaussian.

$$F_j(\| \mathbf{x} - \mathbf{a}_j \|) = \exp\left(- \frac{\| \mathbf{x} - \mathbf{a}_j \|}{2\sigma_j^2} \right) \qquad \textbf{Equation 4.15}$$

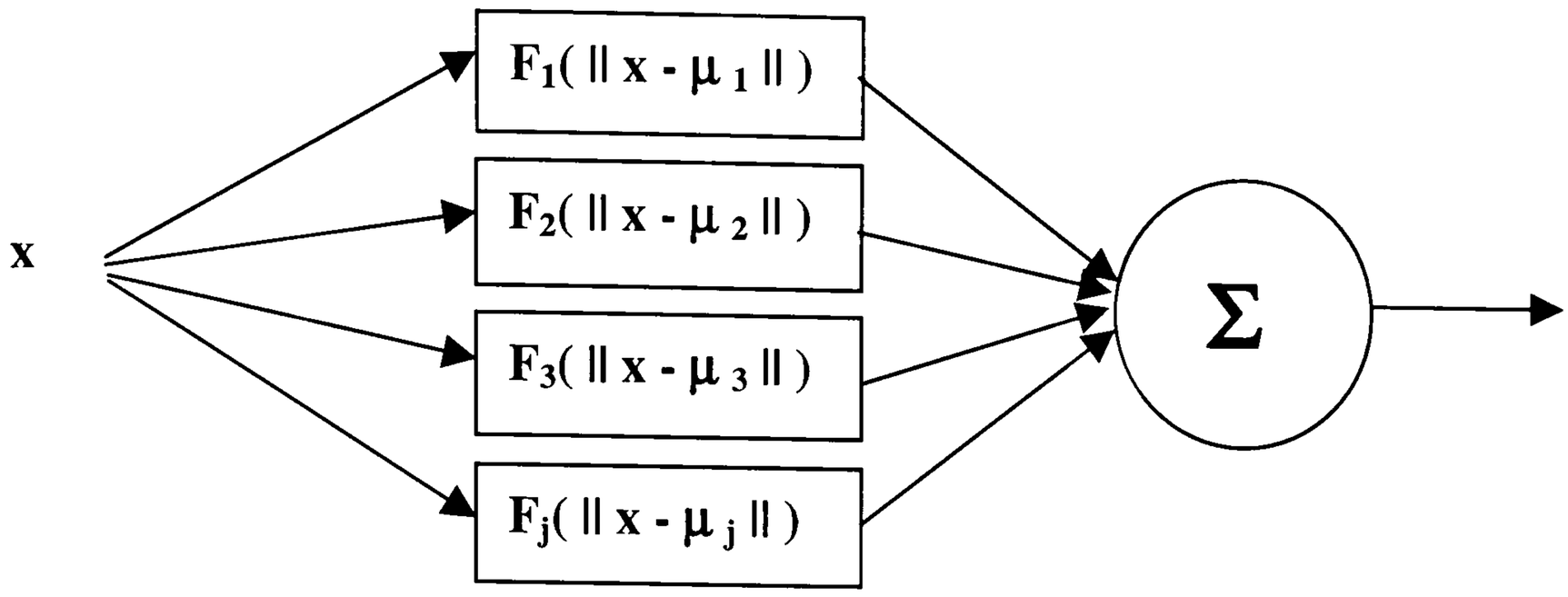


Figure 4.16. The radial basis function architecture.

The layout of the RBF network is illustrated in **figure 4.16** above. An advantage of radial basis functions over the MLP is that they do not have local minima since non-linearly separable problems are transformed into linearly separable ones. If a solution exists it will be unique. Therefore a validation dataset is not required as for the backpropagation network. A full non-linear optimisation on the weights of the network is unnecessary This will help the RBF network to converge much faster than a multilayered perceptron.

A problem with using an RBF is that finding the correct set of basis functions to obtain classes that are linearly separable can be difficult. The radial basis function (RBF) works using a method known as exact interpolation. It fits a curve exactly between datapoints. This is illustrated using **figure 4.17**, which compares a backpropagation network with a radial basis network when they perform the same pattern recognition task. A backpropagation network separates classes using hyperplanes while the RBF fits curves in localised regions of space between points. This means that RBF networks are in general worse at generalising and handling noisy data than are MLPs. unless the number of basis functions is high.

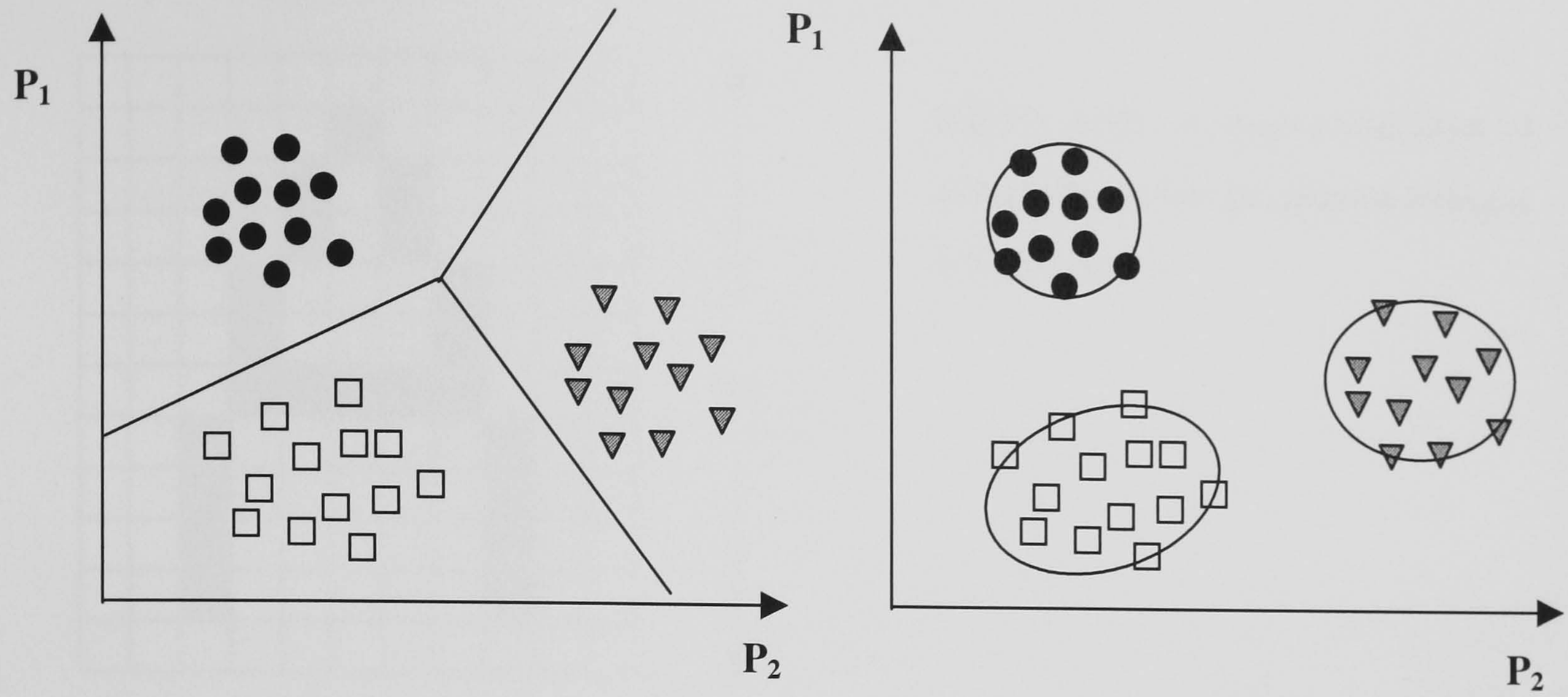


Figure 4.17. **(a), shows the method of hyperplanes used by MLP to partition the different classes, while (b), shows the method of exact interpolation adopted by the RBF.**

4.7 Data pre-processing [41]

Care and thought must be taken when preparing data for a neural network. This is important because irrelevant data that is inputted into a neural network can reduce its efficiency and accuracy since it creates extra computation and can cause the number of dimensions to be increased. If the data occupies a higher dimension space, then a larger amount of data is necessary to produce accurate classification, since a higher dimensional space needs more data to fill it. The process of preparing data for neural networks is known as pre-processing. An example of this very important stage of neural computation is given below.

The aim of pre-processing is to reduce the number of dimensions in the network making it more efficient and accurate. This can be explained using a character recognition problem. **Figure 4.18** represents a digital image of a character A that can be generated using a CCD camera. It requires 132 pixels for its representation. In practice a monochrome CCD camera can also generate 255 grey values. In order to distinguish this character from a B, C or D it is possible to use a network with 132x255 inputs. This is a very high

dimensional space that will produce poor classifications unless a large number of examples are used in the training data.

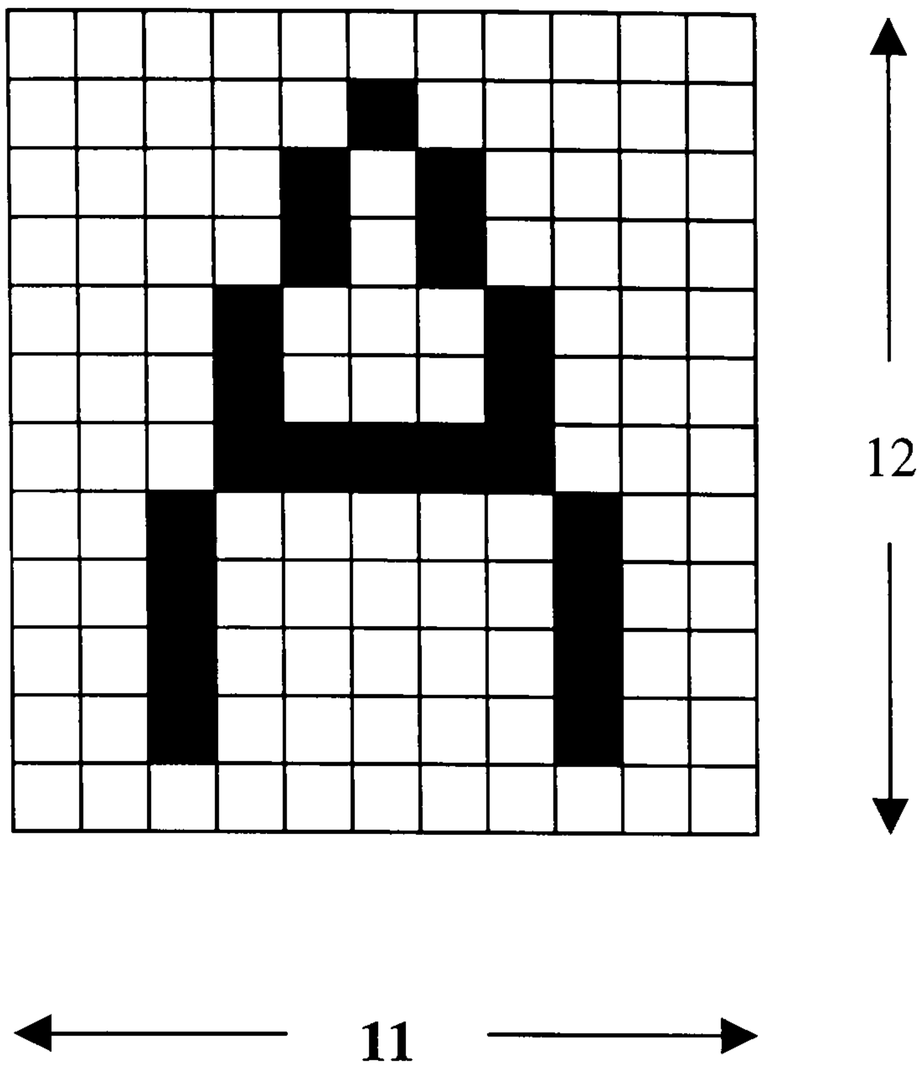


Figure 4.18. A representation of a text character produced using a CCD camera.

The number of input parameters can be greatly reduced by the application of prior knowledge. Prior knowledge is defined as additional information about the problem that does not exist in the training data itself. For example, prior knowledge such as using the height and width of the A,B,C and D to replace the 132 x 255 dimensional problem might produce a good classification in only a 2-input network. If this produces a poor result, the height and width measurements can be coupled with information about the number of enclosed regions the characters possess (C = 0, A =1, D =1, B=2), in order to obtain a better result. This process of reducing the number of dimensions to eliminate irrelevant data is known as feature extraction.

Therefore much thought must go into the preparation of the data before it is applied to the network. The process of preparing data for neural networks is known as pre-processing. In practical neural network projects, this area can be far more important than the implementation of the neural network itself. This is because in some cases the neural networks employed are well developed and tested research tools, whilst the input data is a freshly obtained experimental quantity.

Neural networks cannot be regarded as universal problem solvers. If they are not suitable for a particular problem other techniques such as Bayesian or fuzzy logic methods can be used. Neural network techniques have been chosen for this project because they can be implemented on a computer and can be computationally very efficient. The number of

dimensions that will be initially presented to the network will be relatively small and has been estimated at approximately 10. The use of only a low number of variables will increase the probability of producing good working models. Neural networks also have the advantage, if the mathematics is disregarded, of being conceptually easy to understand. All that needs to be understood is that an input produces a correct or incorrect output or answer. If the answer is correct, nothing further is done. If the answer is incorrect, then the network is trained or corrected to improve its future performance. This process of correction can be viewed as the same as the process that humans and animals use when they learn how to perform tasks.

MLP and RBF technologies are well researched and developed subjects and it is possible to obtain commercial software that applies these techniques to solve classification problems. This means that if a neural approach is adopted, it will be unnecessary to develop the software that performs the backpropagation and RBF algorithms. However it is still important to understand the concepts of neural technology if they are to be applied successfully.

4.8 Bayesian techniques in pattern recognition [41,43,47]

Pattern recognition problems can be solved statistically and have been used prior to the discovery of neural networks. A common technique is Bayesian classification which can be expressed as follows:

The objective is to classify a feature vector X to a class Gi where i=1,2,3,4,5....n. If $P(Gi|X)>P(Gj|X)$, where $i \neq j$ then Bayes' rule assigns X to the class Gi, the class with the highest conditional probability. P(Gi|X) is calculated using the equation:

$$P(Gi \mid X) = \frac{P(Gi)P(X \mid Gi)}{\sum_{n} P(X \mid Gn)P(Gn)} \qquad \textbf{Equation 4.16}$$

The difficulty in using Bayesian classifiers is that the quantity P(X|Gi) is difficult to calculate unless the data set is very large. The approach that is taken is to estimate either the probability distribution or to assume it to be Gaussian. However, for infinitely sized datasets; the error that a Bayesian classifier makes is the theoretical minimum any classifier can make. It is for this reason that this statistical technique has been used as a benchmark for neural networks.

A neural network therefore cannot perform better than a Bayesian classifier in the theoretical limit. Since a neural network system performs the same task as a Bayesian classifier, it can also be viewed as a statistical classifier. In fact neural networks and Bayesian classifiers usually perform with the same level of accuracy. The reason why a statistical technique has been mentioned is to emphasise the point that the problem here is a statistical one. Understanding this point will be an aid that will be used later in the experimental work.

However there are two practical reasons for using neural networks instead of statistical methods. Firstly, they do not suffer so greatly from the curse of dimensionality. This is a difficulty when to find a solution, problems are moved to higher dimensional spaces and the volume of data has to be increased accordingly to produce meaningful models. Secondly, neural network algorithms are computationally faster. For this project, the exact reason why a neural network can achieve a particular accuracy is of less importance than the accuracy itself. The main purpose of neural networks here is as a tool that can be used to recognise patterns from data produced by the print quality measurement system.

4.9 Fuzzy logic [42,48,49,50]

The method which fuzzy logic uses to map an input space onto an output space is conceptually easy to understand since it is mathematically straightforward and is also based on natural language. Although the primary use of fuzzy logic is in industrial control, the technique can be used for pattern recognition problems with which this project is concerned.

Fuzzy logic was primarily developed for industrial control problems. Zadeh, an industrial control specialist who wanted to model mathematically large control systems so that they could be physically built, first suggested it. He discovered that the larger a system became, the more difficult it became to model, using conventional mathematical techniques, and eventually reached a point where it became impossible. In order to overcome this, he used the concepts of fuzzy logic in his designs. Zadeh claimed that despite sacrificing precision by using the natural language procedures of fuzzy logic, qualitative models with good predictability could still be built. It was also claimed that this lack of precision was an advantage. This is justified by the example of learning to park a car. He states the obvious fact that it is more difficult to solve this problem using an equation than by human

qualitative methods. In other words the precise procedures of robots are less efficient than the more qualitative methods of humans in performing this task.

Fuzzy logic combines (1) quantitative knowledge (mathematical models and data) with (2) linguistic information that cannot be quantified mathematically. A classical set C can be defined as:

$$C = (x|x>k) \qquad \textbf{Equation 4.17}$$

where k is a constant that determines whether x belongs to C.

A fuzzy set F is an extension of the classical set **C**:

$$F= (x, M_F(x)|x\in X) \qquad \textbf{Equation 4.18}$$

$M_F(x)$ is known as the membership function that can be any normalised function. The shape of this function is determined by the attributes of the concept. The membership function maps each element of X to a membership value in F in the interval [0,1] which is referred to as the degree of membership in F. This process is known as fuzzification of the crisply defined input data.

After fuzzyfying the data, logical rules are then applied to it. These consist of logical operators (AND, OR and NOT) combined within IF THEN statements. This takes the form IF x = A AND y = B THEN z = C. The IF component is often referred to as the antecedent and the THEN component is called the consequent. In fuzzy logic, these rules are expanded to give more than a 1 or 0 output normally associated with logic. The result is a number in the interval [0,1]. This is known as the degree of support for the rule.

Real applications usually require more than one logical rule. The usual requirement is to use two or more rules each with their own fuzzy output sets. These output sets can be combined with each other to give a single integrated fuzzy output set. These rules are formulated and combined by human expert interaction or by neural networks in the part of the fuzzy logic system known as the inference engine.

A set of aggregated areas calculated from the previous output sets represent the single fuzzy output set. To obtain a single value for the system, a method known as centroid

calculation can be employed. The process of extracting a single value from the fuzzy output set is known as defuzzification. **Figure 4.19** below illustrates the fuzzy logic process.

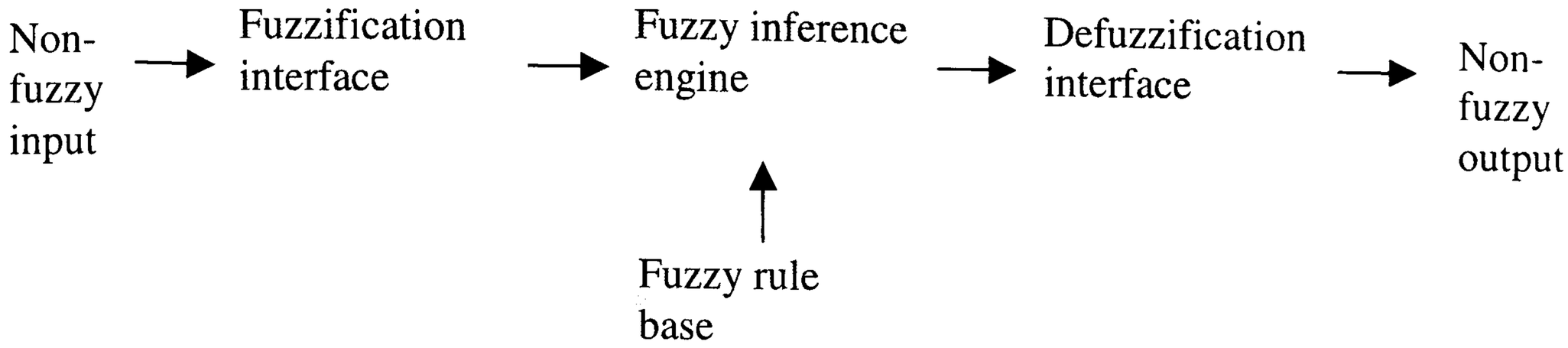


Figure 4.19. Diagram showing the components of a fuzzy system.

The fuzzy inference system described above is known as Mamdani's method. Another fuzzy logic process that is used is the Sugeno fuzzy inference method. The difference between the Sugeno and Mamdani systems is in the form of the fuzzy rules. In the Mamdani method a fuzzy rule takes the form IF x = A AND y = B THEN z = C where A and B are the antecedent fuzzy sets and C is the consequent fuzzy set. In fuzzy logic systems there can be many consequent fuzzy sets from different rules. After the aggregation of the consequent fuzzy sets the centroid has to be computed. The computation of the centroid can be simplified if the Sugeno system is adopted, since its rules take the form IF x = A AND y = B THEN z = k where k is a crisply defined constant. Instead of integrating a continuously varying function to determine the centroid, defuzzification can be achieved by calculating the average of a few datapoints

The disadvantages of fuzzy logic when compared with neural networks are as follows. They do not learn patterns in the training data. They rely entirely on expert human knowledge to detect patterns in data. However, backpropagation is a computationally slow algorithm with no guarantee of convergence even when a solution exists. Methods have been devised that use fuzzy logic to help neural networks converge more efficiently. These methods are known collectively as neurofuzzy logic. One such method known as ANFIS (Adaptive Neural Fuzzy Inference System) uses backpropagation neural networks to adjust membership functions. The description of neurofuzzy logic given here serves as an outline to an extensively researched area. It is its wide range of successful industrial applications that makes it worth pursuing in this project.

Another reason for applying fuzzy logic is that it approaches data analysis from a different paradigm. This paradigm combines mathematical precision with the elastic nature of human decision making that was stated at the beginning of this section. Since human subjective analysis can be elastic the imprecise nature of fuzzy logic may produce better models than precise mathematical techniques alone.

4.10 Experimental structure and approach

So far all the theoretical aspects and information needed to start the experimental work have been explained. The most important aspect of this experimental section is the collection of data for subsequent analysis. This is because the equipment used and the neural network software had already been developed. In order to collect the data, image analysis hardware needs to be interfaced with commercial image analysis software to see what can be achieved. The next stage is to develop software to analyse halftone and solid print from non-impact printers. Therefore an algorithm is required and a computer language needed to implement it. The final stage of the systems development will be to interface a neural network to the pre-processing software and to train it with assessments from an observer.

5.0 Preview of the experimental section.

So far the cognitive processes that are necessary for the perception of print quality have been analysed. This has been followed by a discussion of the printing processes and the variables that affect print quality. It was deduced from these discussions that firstly, a model of print quality which is derived from an exact theory of cognitive processes is outside the scope of this project, since cognition is not fully understood. Secondly, it is impracticable to investigate the print quality of all the printing processes simultaneously due to the large number of variables that must be considered. The enormous variety of images that can be produced should also be considered in relation to these points.

To overcome these problems in order that a coherent study of print quality can proceed, an image analysis system has been produced based on **figure 3.2**. This system can implement a procedure based on the Turing test discussed in **section 4.1**. The image analysis system achieves this in the following way.

1 The development of the system adopts an expandable modular approach.

2 A limited number of printing processes and a few monochrome images of simple shapes and text characters were used in the investigation.

3 The system was tested under these limitations, but can be expanded at a later stage.

The following system shown in **figure 5.1** has been developed for the measurement of print quality. This diagram emphasises the components of the system represented in **figure 3.1** where the software development was undertaken. The software development was the main component of the experimental work. The system can be fully automated and has the possibility of expansion beyond the current project.

The image analysis approach to the measurement of print quality can be used to predict measurements made by observers. In fact, this was the nature of previous research carried out in this field (see **section 3.3**). Print quality assessments using an image analysis system involved taking a series of measurements from images and these would be compared with decisions made by observers. These studies did not consider the possibility of extracting features from entire images, investigating a wide range of printing methods and the automation of machine assessments for print quality, using an integrated method that takes into account all three problems simultaneously (see **section 3.3**).

JVC CDD monochrome camera attached to a zoom lens that produces an image with dimensions of 768(horizontal) x 576(vertical) pixels and 256 grey levels. This camera has an automatic lighting adjustment circuit. The camera is connected to a Pentium personal computer.

↓

Inside the computer there is a Matrox Meteor frame capture board. This converts the analogue signal from the camera to a digital signal. The digital image data is then stored in specially designated areas of the computer memory called buffers.

↓

The image data is processed by a high-level computer language. This is achieved using the Matrox imaging library(MIL) application development software, to convert the data in the buffers to a form that can be read in Microsoft Windows 95. The code to do this was written in Microsoft Visual Basic version 5.

↓

A pre-processing program was coded in Visual Basic V5. The algorithm for this was developed specifically for this project. It was designed to process both solid and halftone information from the printed images under investigation. It extracts information about tone intensities, halftone frequency patterns, edge defects and properties of the paper from the raw image data of the previous stage. This information is presented in the form of a 52 dimensional model that is processed by neural networks from the MATLAB neural network and fuzzy logic toolboxes.

↓

Selected variables from the pre-processing stage are inputted in neural network solutions. These variables are reduced to a single value that is a measure of print quality. These are provided by the MATLAB neural network and fuzzy logic toolboxes and comprise of the backprogation, radial basis and ANFIS neurofuzzy logic algorithms.

Figure 5.1. A flow chart showing the components of the image analysis system used in this investigation

The project described in this thesis is different from previous research into print quality for the following reasons. Firstly, it takes an artificial intelligence approach to print quality. It investigates the possibility of developing an image analysis system that behaves like an observer in the assessment of print quality. The system can be viewed as a machine that simulates this aspect of human behaviour, instead of a machine that makes precise measurements from printed images.

Secondly, it also assesses a wide range of printing processes and many variables simultaneously. It uses a CCD camera to capture data from an entire image of a simple shape or text character and uses a method that gives an output response for the whole image.

A major part of this thesis is concerned with the development of a computer program for the image analysis of image data from a CCD camera. The main reason for this is that commercial software could not make the measurements required for this project. The algorithm that has been specifically developed for this project has many features which are not to be found in commercially available systems. A description and analysis of the algorithm will be given at a later stage.

So far all the theoretical aspects and the original features of the work have been explained. Referring to **figure 5.1**, the most important aspects of the experimental work are the collection of data for subsequent analysis and the development of the pre-processing algorithm. This is because the hardware used and the neural network software have already been commercially built. The next sections describe how the final system illustrated in the flow chart of **figure 5.1** was developed, followed by the type of results that can be obtained from it.

5.1 The CCD cameras used in the project

A JVC TK - 1070E CCD colour camera with a zoom lens was initially used for the early stages of the experimental work when assessments were made of the suitability of commercial software for the project. This colour camera was subsequently replaced by a JVC TK - S350 black and white model since only monochrome images were to be investigated. The monochrome camera has the following advantages in this project. Firstly, it employs a single CCD chip instead of the three which are needed for the colour

camera. Thus less complex software can be used and the risk of camera malfunction is minimised.

Secondly, the monochrome replacement has a smaller CCD imaging chip than the colour camera. It has a dimension of 1/3inch instead of 2/3inch. Therefore the image produced by the monochrome camera will suffer less from pin-cushion distortion than will the colour one. The reason for this is that in order to produce an image of identical size on both the larger colour and the smaller monochrome chip, a larger central portion of the lens is used in the case of the colour camera. The amount of distortion per unit area produced by a lens increases as the distance from its centre increases. Hence there is more distortion in the case of the colour camera.

Although a neural network can be trained to take into account the second problem, it is better to solve it by improving the hardware since, as will be seen later, a software approach would mean that a neural network with more inputs would be needed. This would increase volume of data and the processing time required to produce an optimal solution.

The illumination that is used with the camera is an important factor that must be considered. The lighting employed consisted of a circular fluorescent lamp that enclosed the camera as shown in **figure 5.2**. The image was also illuminated with both laboratory lighting and sunlight without producing significant errors in the readings taken that would prevented meaningful results from being acquired. The reason for this is that the camera is capable of automatic adjustment to fluctuations in lighting intensity. Therefore a special lighting chamber was not needed because the lighting conditions required no monitoring. Software corrections for uneven lighting across the image were also unnecessary. These last two points were found to be very important later in the work since it simplified data collection from the thousands of measurements that were made. The evidence that supports the conclusions in this paragraph are in the results of **section 5.10** and **section 6.2-6.7**. All these results were obtained using the lighting conditions described here.

5.2 Frame capture hardware and the personal computer

The frame capture board that was initially used was a Matrox Comet board. This board was employed when making the feasibility tests on commercial image analysis software and

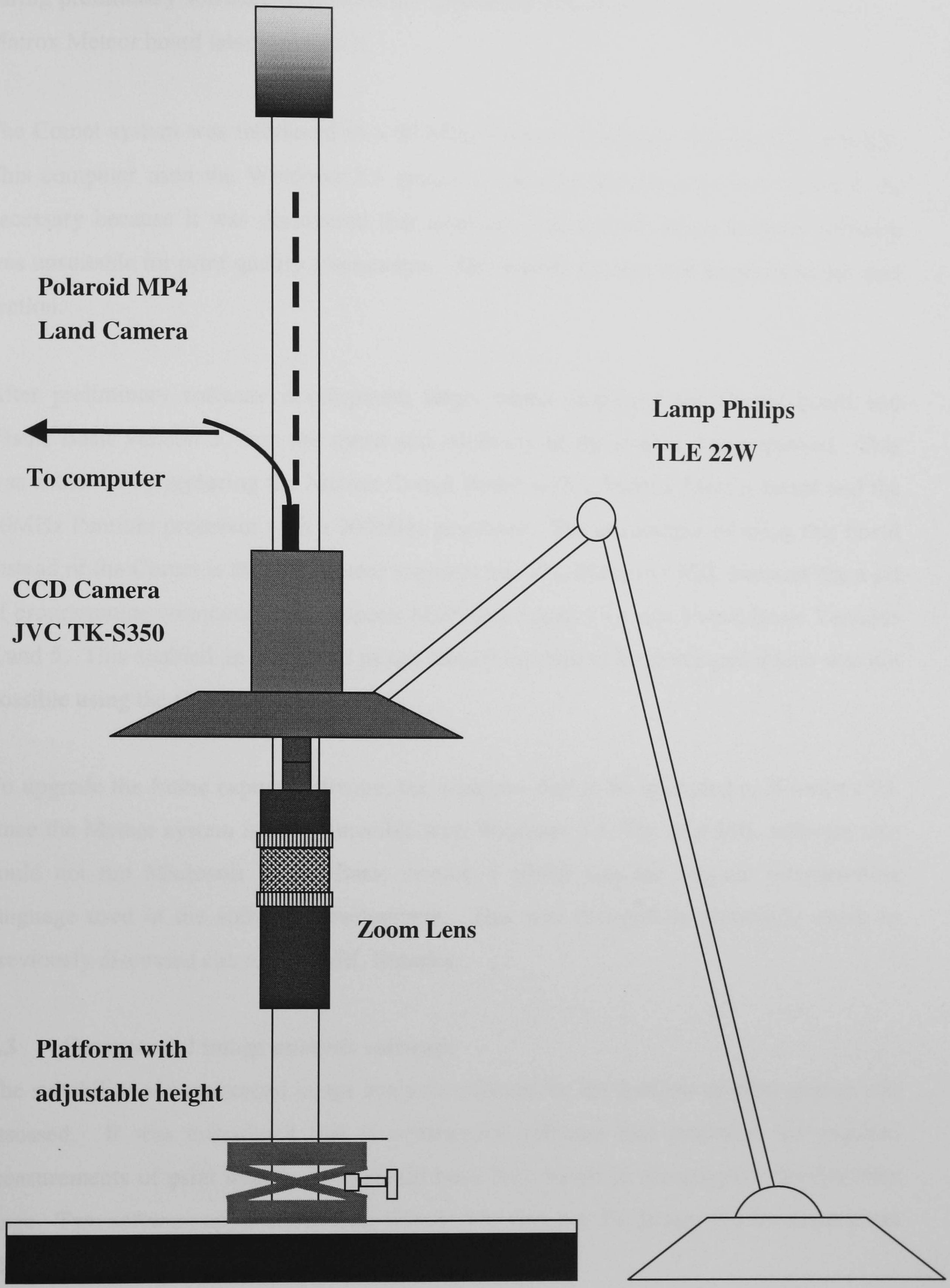


Figure 5.2. The experimental apparatus

during preliminary software development. Due to its slowness and practical limitations a Matrox Meteor board later replaced it.

The Comet system was interfaced to a 90 MHz Pentium computer with 64 Mb of RAM. This computer used the Windows 3.1 system. Software development was found to be necessary because it was discovered that available commercial image analysis software was unsuitable for print quality assessments. The reasons for this will be given in the next section.

After preliminary software development stage, which employed the Comet board and Visual Basic version 3, both the speed and reliability of the system was improved. This was achieved by replacing the Matrox Comet Board with a Matrox Meteor board and the 90MHz Pentium processor with a 200MHz processor. The advantages of using this board instead of the Comet is that the Meteor supports the MIL libraries. MIL libraries are a set of programming commands that supports Microsoft Visual C++ and Visual Basic Versions 4 and 5. This enabled an integrated image analysis system to be developed which was not possible using the Comet board.

To upgrade the frame capture software, the computer had to be upgraded to Windows 95, since the Meteor system is not compatible with Windows 3.1. The new MIL software also could not run Microsoft Visual Basic version 3 which was the original programming language used in the software development. This was changed to version 5, which as previously discussed can run the MIL libraries.

5.3 Commercial image analysis software

The suitability of commercial image analysis software for the analysis of print quality was assessed. It was considered that if commercial software had produced the required measurements of print quality, time would have been saved in the program development stage. Two software applications were tested. The first was PC Image and the second was Matrox Inspector.

PC Image is a Windows application which was developed by Foster Finlay that can analyse and process colour images. Measurements that this program can make which could have been of use in this investigation included the grey level classification and particle size detection of objects in an image. However, there is no facility for the

measurement of edge defects and noise in printed images. The automation of the image analysis system using this software proved to be difficult. Microsoft Recorder was used to automatically run the Matrox and PC Image software in sequence but the chain of computer commands required to do this often broke down.

Matrox Inspector is more advanced than PC Image in that it can capture live images. However, the Matrox Inspector application, like PC Image cannot make all the necessary measurements required for this project. Neither PC Image nor Matrox Inspector can measure halftone frequency patterns or quantify the noise and edge uniformity of an image. These quantities, as the previous research described in **section 3.3** indicates, influence perceived print quality and therefore must be determined if human print quality perception is to be simulated.

Section 5.5 describes an algorithm that has been designed and developed specifically for this project which can measure both noise and edge uniformity of a simple shape or text character. It will be shown later that the algorithm can do this in different ways. The algorithm also measures these properties over an entire image. It also determines the contrast and halftone frequency patterns within images. The next section describes how the dimensions of the printed images that were to be investigated were determined

5.4 The dimensions of the test images used in the investigation

The first part of this section calculates the dimensions of the images that were used in the assessments. The dimensions were based on matching, a magnification factor that was to be used by the image analysis system, to the resolving power of the human visual system.

In **sections 2.1-2.2** and **section 4.4**, the physiological processes concerning visual perception and the CCD camera were respectively discussed. From this discussion it can be concluded that the rod and cone density for the eye is non-uniform while the pixel distribution for the camera forms a regular pattern. In normal daylight the cones which are 7×10^6 in number, are used for vision. However these cones are unevenly distributed. The highest concentration of these cones is in an area approximately 8×10^{-5} cm^2 called the fovea. It is the fovea that is responsible for high-resolution vision. The maximum resolution of the human eye due to fovea vision is 0.017^o **[51]**. At a reading distance of 30 cm this corresponds to a minimum resolution separation of 9×10^{-3} cm for two neighbouring objects. In the JVC TK - S350 CCD camera the CCD array dimensions are

753 x 582 pixels. If a 1cm image of a square is completely on the CCD sensor so that 582 pixels corresponds to a length of 1 cm the separation distance of 9 x 10^{-3} cm would correspond to a length of 5 pixels rounded to the nearest integer. It can be deduced that the resolving power of the image analysis system can match that of human perception if the following two criteria can be met. Firstly, the dimensions of the image assessed by the image analysis system is 1.3 cm x 1 cm. Secondly, effects due to noise caused by the hardware are negligible.

Based on the preceding information it was decided to use image sizes of up to $1cm^2$ for the investigation. A Computar M6Z 1212 zoom lens was used to enlarge the image to the required dimensions. This is illustrated in **figure 5.3** for a 300 dpi halftone image of a 1 cm square. The image in this figure was produced using the image analysis system described in **sections 5.0-5.3**, **section 5.5** and the pre-processing program in **appendix 1**, which will be discussed later. The image dimensions calculated above are theoretically estimated quantities. It is the role of the neural networks to show whether the analysis is correct. The calculation can be shown to be correct if the neural networks can produce the correct input-output mappings for the simulation of human print quality perception using the image analysis system.

In the preceding section it has been concluded that available commercial image analysis software was unsuitable for the measurement of print quality variables in this investigation. In this section the dimensions of the image that was to be investigated were determined. The remainder of this section describes the development of a program that can take meaningful print quality measurements of images using the image analysis system.

5.5 The pre-processing algorithm

This algorithm evaluates the image quality properties by taking measurements of the spatial distances frequency pattern and the gradient values between the background and the darker image areas. This is achieved in the following way. A printed image can take the forms shown in **figure 5.4**. They are only a few examples of halftone images. The halftone patterns can take any form. The halftone patterns can also be transformed into a frequency pattern, the length of which is given by the number of pixels in the active image matrix on the computer screen. This is shown in **figure 5.5** and **5.6** for a regular squared pattern. This is an ideal pattern that has no noise. It also has only two gradient changes per cycle which means that the edges are clearly defined. Real camera images do not follow this

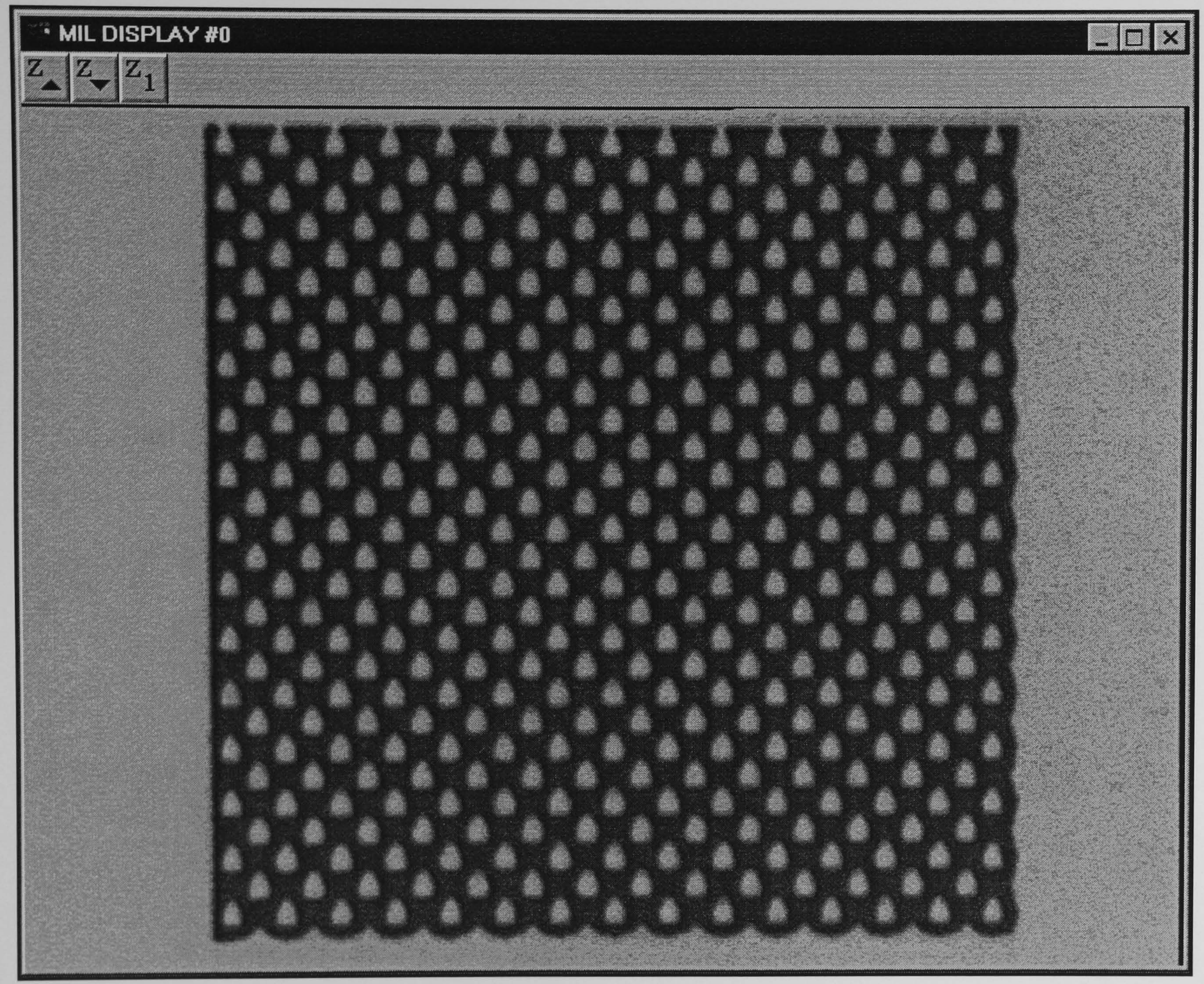


Figure 5.3. An image of a 1 cm square used in the investigation.

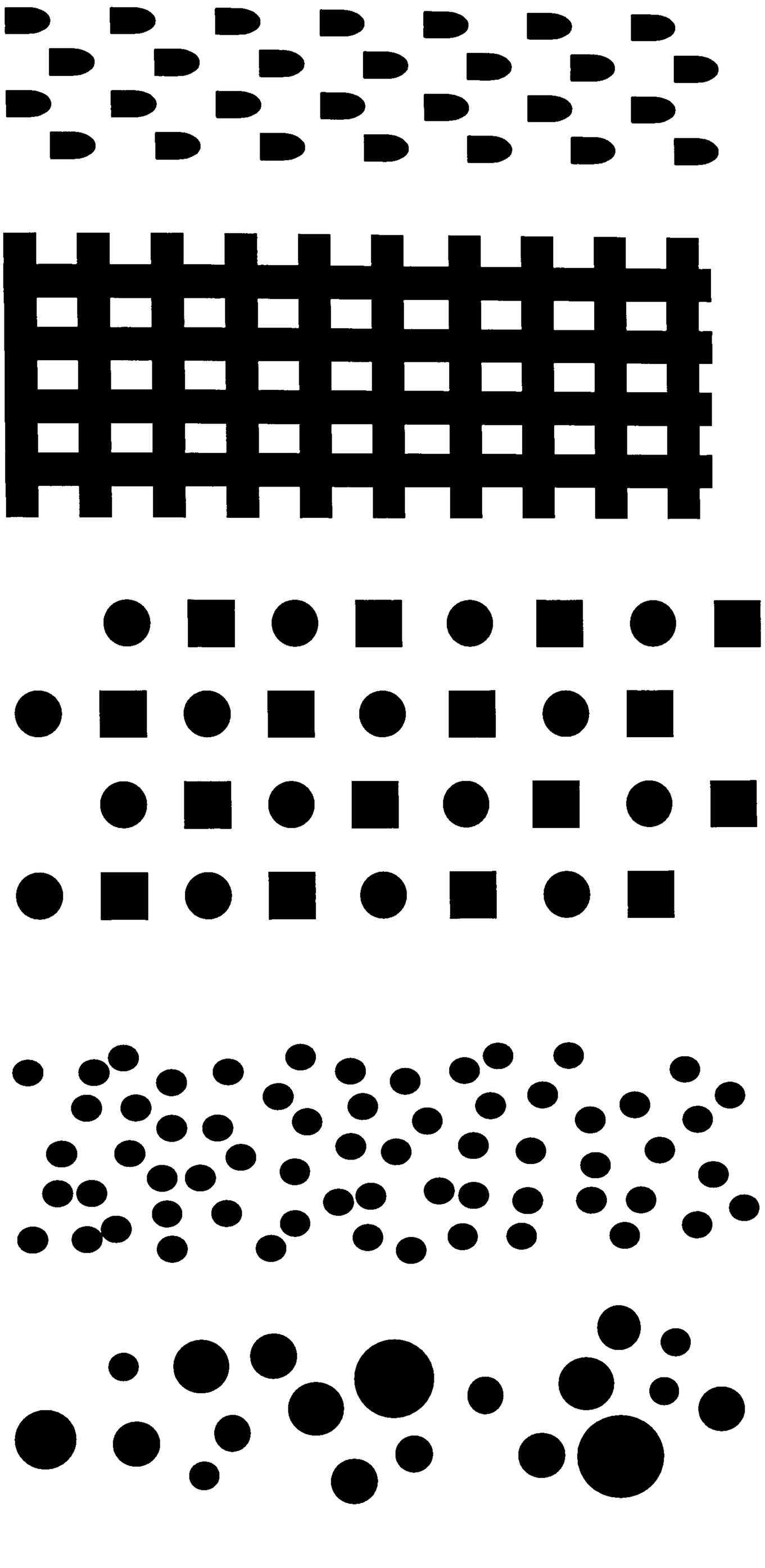


Figure 5.4. Possible halftone patterns of printed images.

ideal. Noise and poorly defined edges make taking meaningful measurements more difficult.

In this investigation the image is defined as the printed regions of a print sample. The background is defined as non-printed regions. The display window is the area of print and background that is processed by the image analysis system and is also visible to the observer via the display window on the computer screen. This display window is provided by the Matrox software and has the dimensions 768 x 576 pixels. An illustration of this can be seen in **figure 5.3**. These definitions will be used frequently in the sections involving pre-processing software development.

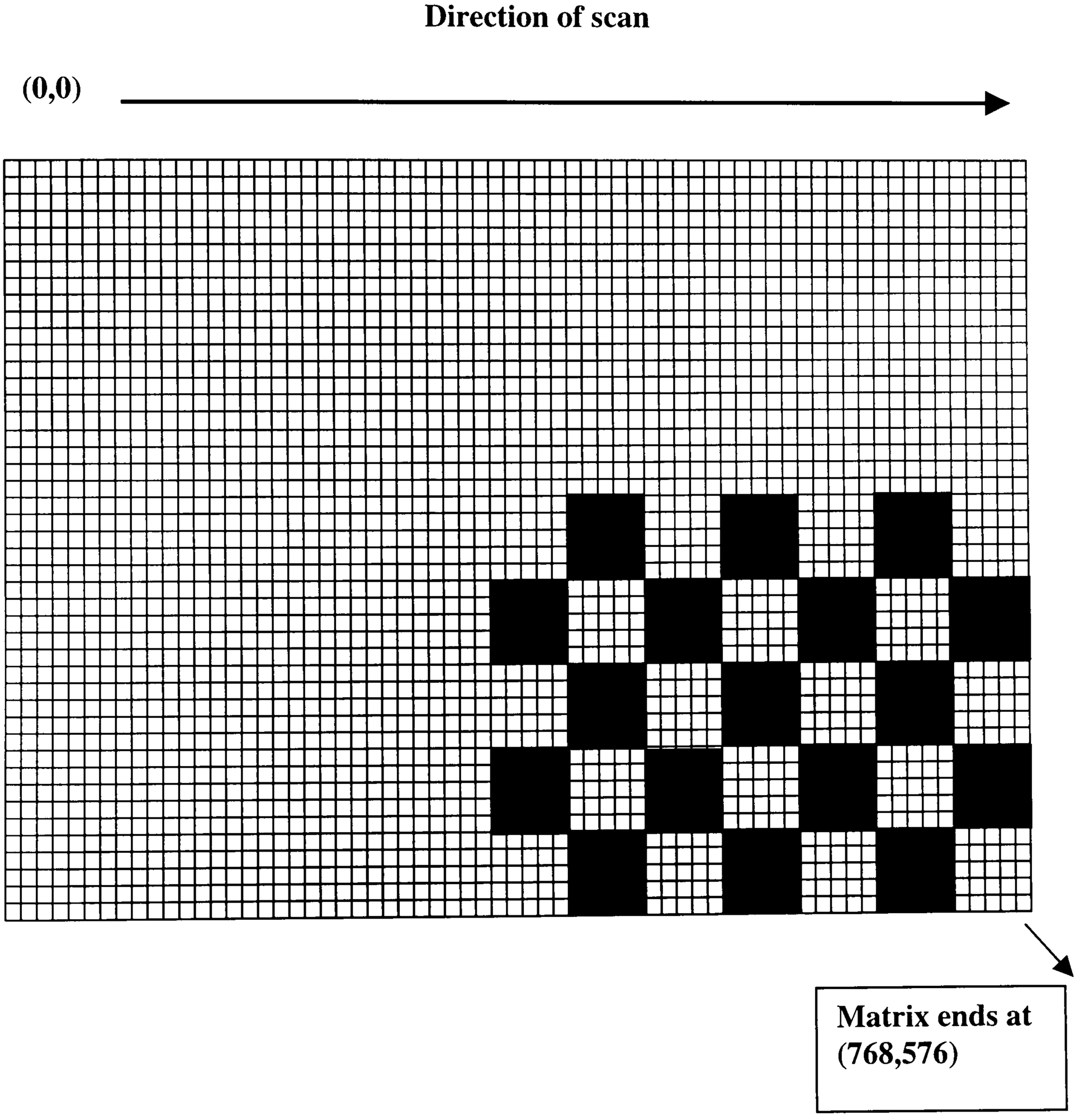


Figure 5.5. A magnified view of part of the Matrox display window. The scan moves in the x direction. When the scan reaches the end of the image window the y co-ordinate is incremented by 1 in a descending vertical direction.

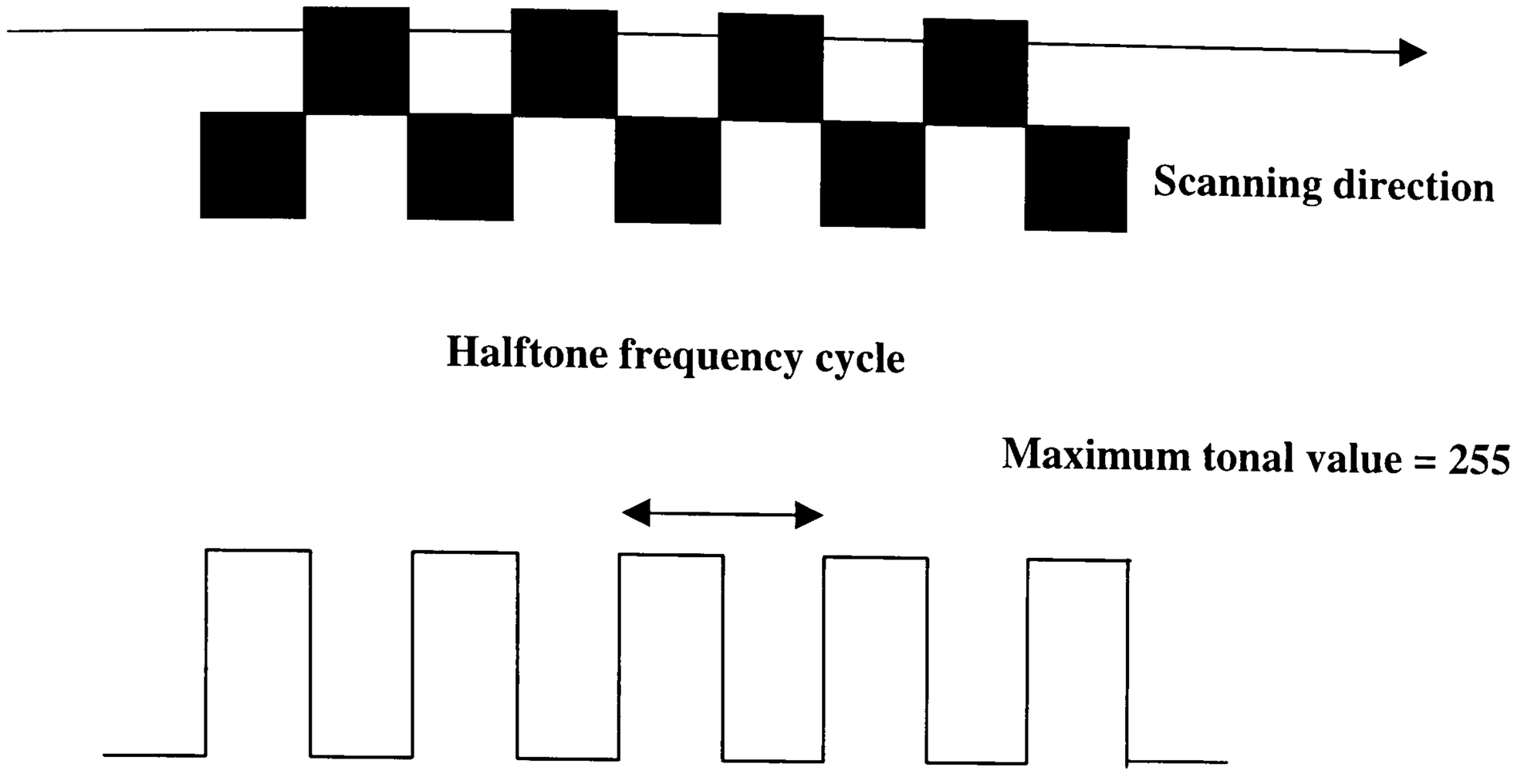


Figure 5.6. The extraction of the frequency pattern from the image shown in figure 5.5.

The pre-processing algorithm performs mathematical operations on an image matrix. This matrix which represents the image is divided into 768 x 576 pixel locations. The matrix dimensions are determined by the Matrox Meteor hardware. The pixel values are processed sequentially line by line. It does not use 2-dimensional mask operators like other image analysis applications **[13]**. Also, the minimum background or white intensity equals 0 and the maximum image or black intensity equals 255. The 0 and 255 intensities are saturation values that are due to the limitations of the hardware.

The convention that is adopted by commercial software such as PC image, Matrox and Paintshop Pro is to use 0 for the maximum image intensity and 255 for the background. The opposite is used here because the author is of the opinion that it is easier to conceptualise the maximum tonal intensity as 255. This would aid the development of the algorithm.

The software developed for this project scans the image from location (0,0) to (768,576) as illustrated in **figure 5.5**. The scan moves in the x-direction. When the scan reaches the end of the window, the y co-ordinate is incremented by 1. Measurements are taken from neighbouring pixels for the quantification of image noise and tonal gradients. The algorithm works on the principle of averaging these measurements over the entire image. This reduces the number of dimensions in the feature extraction process. This is necessary for the effective use of the neural networks.

So far only ideal images with no noise have been discussed. Real images of print samples captured using CCD cameras suffer from noise. Also the edges are not clearly defined. The next paragraph describes how the algorithm solves these problems to measure the frequency pattern.

Figure 5.7 illustrates the process of threshold detection for a non-ideal square halftone pattern that is used in the algorithm. To detect the beginning of a printed region in the display window, the tonal gradient at a location must exceed a preset threshold value. If this happens, this location is defined as an edge. This threshold value must be high enough to eliminate noise, but low enough to detect the image areas. When a rising image edge is detected by this process, the subsequent pixels are counted until a falling edge is detected. The falling edge stops the count and the value of the count and its location are stored in the memory awaiting further processing. The count is reset to zero and restarts when the next rising image edge is detected and the process is repeated.

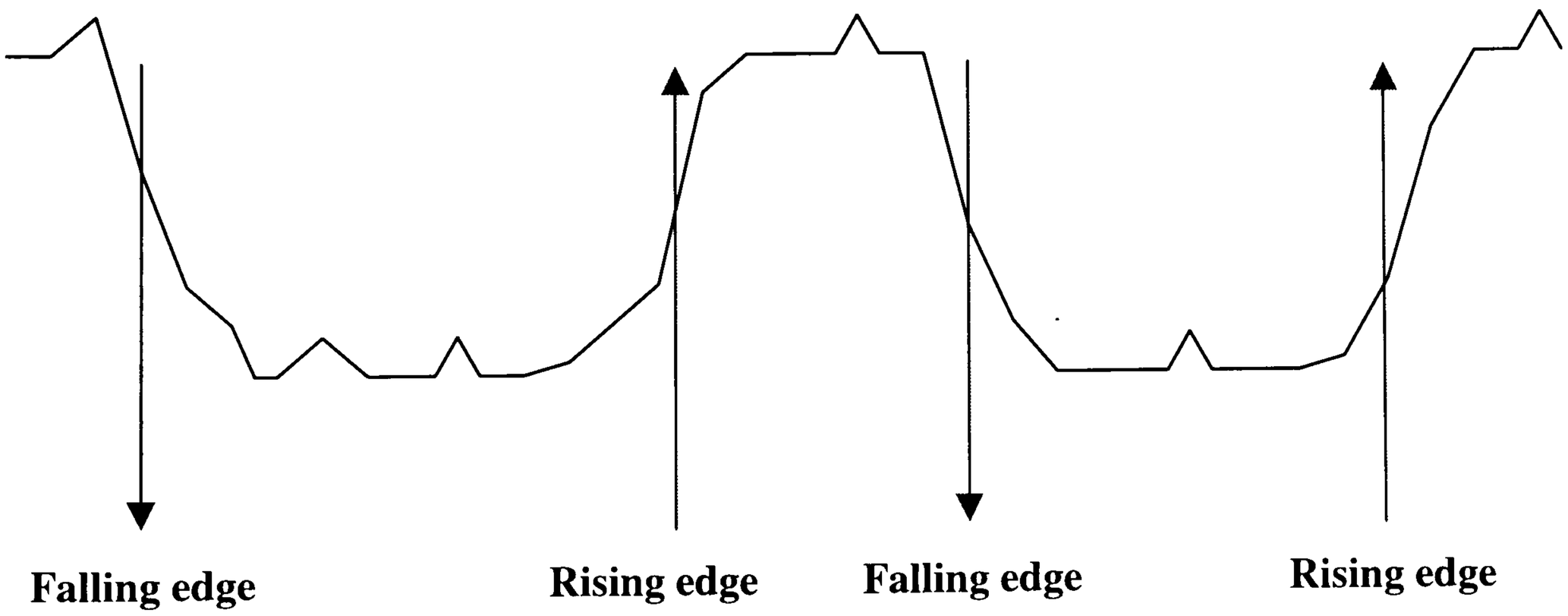


Figure 5.7. Diagram showing how setting the correct value for the threshold gradient can filter out the noise yet still detect the image.

The preceding paragraph describes the basic principle of the pre-processing algorithm. By locating the edges of the image, information that can quantify image quality can be extracted from it. This is demonstrated by the example given above, which describes how the halftone frequency can be determined.

The algorithm considers four print quality variables. These are contrast, resolution, noise and edge raggedness. However the algorithm uses more than one parameter to quantify any of these variables. For example to determine edge raggedness, the rising and falling edge gradients as image areas can be measured in perpendicular directions. The full program produces 52 measurements that can quantify the four print quality variables stated above. A list of these are given in **table 5.1** with an indication of the variable that each is associated with. A full description of the pre-processing program outputs is given after this table. The final program was implemented using the Meteor hardware, the MIL software library and the Visual Basic version 5 computer language. A description of the software development stages leading up to this final pre-processing program will be given in **section 5.7**. The code for the program is listed in **appendix 1**.

The algorithm and the program was developed and tested in stages. For example, the program code for noise and contrast given by the parameters 1 to 5 and 17 respectively was written first. Not all the output parameters in the program have been fully investigated. The potential of these to improve the performance of the system will be discussed in the concluding part of this investigation. The program code was written initially in Visual Basic version 3 and later using the upgraded version 5 of this language. The reasons for using this computer language will be explained in **section 5.6**.

The algorithm that was used to produce the outputs shown in **table 5.1** is an original concept. The usefulness of some of these outputs in producing print quality models was evaluated using real printed images. This will be demonstrated later in the thesis. The next part of this section describes the function of each output.

Output Number	Brief description of the output function
1	**Noise measurement by comparing the widths of neighbouring peaks**
2	Noise measurement by comparing the widths of neighbouring peaks
3	Noise measurement by comparing the widths of neighbouring peaks
4	Noise measurement by comparing the widths of neighbouring peaks
5	Noise measurement by comparing the widths of neighbouring peaks
6	Noise measurement by comparing the widths of neighbouring peaks
7	The sum total of the noise measurements from outputs 1 to 6
8	Length of the modal halftone cycle
9	Size of the peak of the modal halftone cycle
10	Halftone frequency sharpness
11	Spread of halftone frequencies in the image
12	The mean of the detected ascending gradients in the X direction two pixels either side of a detected edge
13	The mean of the detected ascending gradients in the Y direction two pixels either side of a detected edge
14	The mean of the difference of the ascending gradients two pixels either side of a detected edge
15	The position of the image on the screen determined by the mean X co-ordinate of the ascending gradients
16	The mean of the detected descending gradients in the X direction two pixels either side of a detected edge
17	The mean of the detected descending gradients in the Y direction two pixels either side of a detected edge
18	The mean of the difference of the descending gradients two pixels either side of a detected edge
19	The position of the image on the screen determined by the mean positions of the descending gradients
20	Total number of descending edges detected
21	The mode of the measured lengths of the tonal regions in the frequency cycles which represent the halftone image areas
22	The mode of the measured lengths of the background areas between the tonal regions
23	Total image intensity

Output Number	Brief description of the output function
24	The peak tonal value of the image area
25	The greyscale location of the peak tonal value of output 24
26	The second highest tonal value in the image area
27	The greyscale location of the value of output 26
28	The third highest tonal value in the image area
29	The greyscale location of the value of output 28
30	The peak tonal value of the background area
31	The greyscale location of the peak of output 30
32	The second highest tonal value in the background area
33	The greyscale location of the value of output 32
34	The third highest tonal value in the background area
35	The greyscale location of the value of output 34
36	A count of the number of turning points on the edge of an image
37	A measure of the tonal gradient non-uniformity on the edge of the image
38	Total number of gradient changes on the edge of an image
39	Number of 1 pixel spikes on the edge of an image
40	Number of 2 pixel spikes on the edge of an image
41	Number of 3 pixel spikes on the edge of an image
42	Number of 2 pixel spikes on the edge of an image
43	1 pixel width frequency filter
44	2 pixel width frequency filter
45	3 pixel width frequency filter
46	4 pixel width frequency filter
47	5 pixel width frequency filter
48	6 pixel width frequency filter
49	7 pixel width frequency filter
50	8 pixel width frequency filter
51	9 pixel width frequency filter
52	10 pixel width frequency filter

Table 5.1. The list of measurements produced by the pre-processing program.

Outputs 1 to 6

If $L_n = L_{n+2}$ then $Noise_1 = Noise_1 + 1$

If $L_n = L_{n+2} + 1$ pixel then $Noise_2 = Noise_2 + 1$

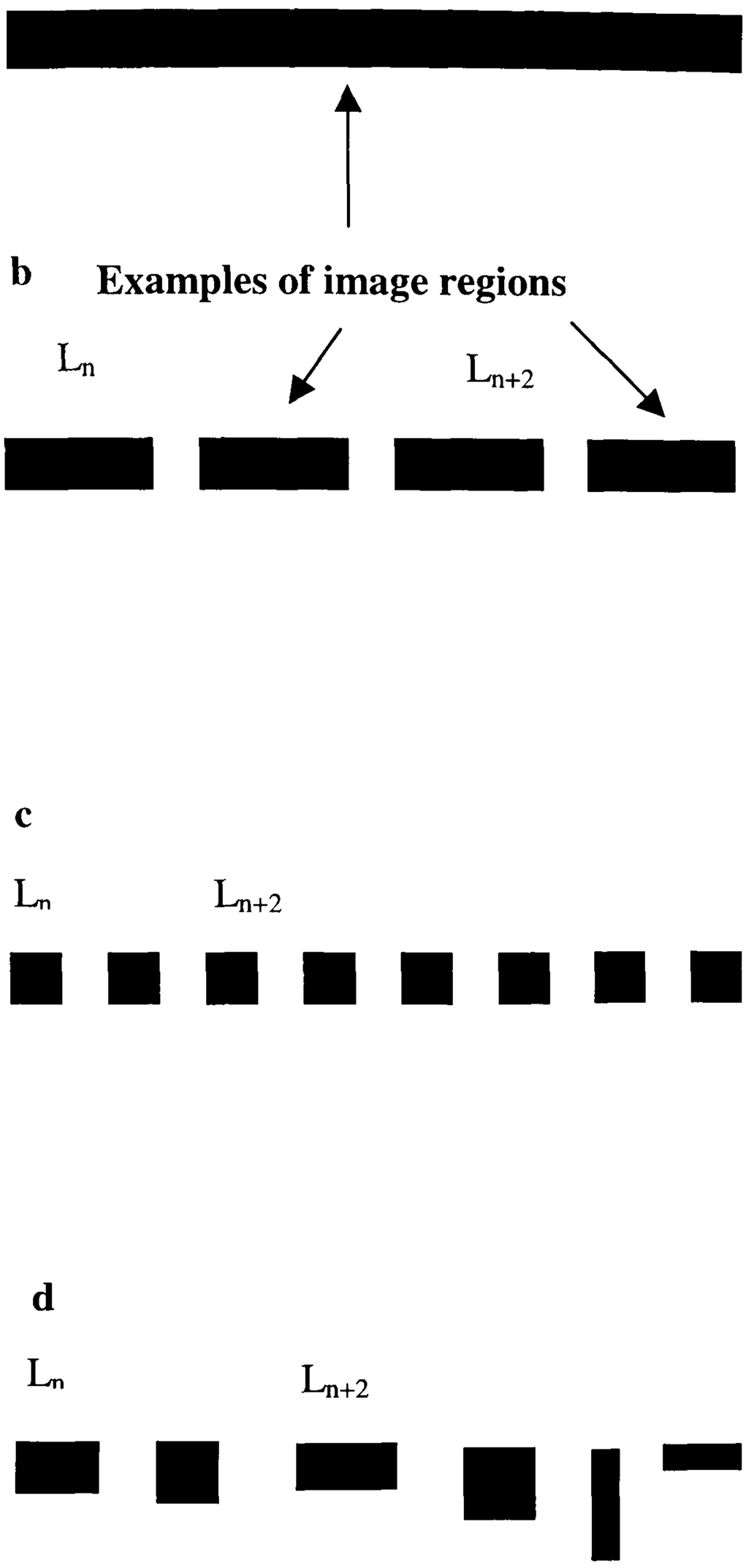
If $L_n = L_{n+2} + 2$ pixels then $Noise_3 = Noise_3 + 1$

If $L_n = L_{n+2} + 3$ pixels then $Noise_4 = Noise_4 + 1$

If $L_n = L_{n+2} + 4$ pixels then $Noise_5 = Noise_5 + 1$

If $L_n = L_{n+2} + 5$ pixels then $Noise_6 = Noise_6 + 1$

The expressions above can measure the noise of an image. They are used to quantify the noise in the body of the image and ignore the affect on the edge of the image. This is achieved by determining the lengths L_n (n = 1,2,3,4,....) of all the halftone image areas as they are scanned. These lengths are compared with neighbouring lengths as shown in **figure 5.8**. If a line consists of only one rising and one falling edge then this is defined by the algorithm as a solid. Therefore no noise is detected in this region of the image. If more than a pair of rising and falling edges are produced, then a halftone pattern is detected. The equations given by outputs 1 to 5 in the previous list apply to a line scan with five cycles. This is because the first and last halftone dots are ignored by the algorithm and only alternate tonal areas are assessed. This is necessary for two reasons. Firstly, the first and last cycles of halftone images are sometimes produced differently from the rest of the image. This difference is due to the software that produces the image and is not due to the physical printing process which can create noise on the image. Secondly, alternating patterns are sometimes produced as illustrated in **figure 5.4**. This means that five cycles need to be present on a scanned lined before any comparison can be made. The algorithm sums the $Noise_n$ (n = 1 to 6) individually. The higher the value of $Noise_1$ is, the less noise there is in the image. This is the opposite for $Noise_2$ to $Noise_6$. **Figure 5.8** below illustrates the description given by the preceding text on how the noise measurements are produced for outputs 1 to 6.



In a, there is no halftone pattern. There is only one rising and one falling edge. The program treats this as a solid and cannot determine noise value.

In b, there is a halftone pattern, but there is an insufficient number of cycles to produce an output value since the first and last dot are ignored by this part of the program. This sample will also produce no value for the noise.

In c, the pattern is completely uniform. There are also more than four dots. The noise reading will be defined as zero.

In d, there are more than four dots but the pattern is irregular. Therefore a noise reading greater than zero will be recorded.

Figure 5.8. Four different cases that demonstrate how outputs 1 to 5 from the pre-processing algorithm measure noise. Each sample represents a complete section of an image across the image window.

Output 7

This is the total count for all the peaks but excluding the first and last in each line detected in a complete image scan.

Output 8

This gives the value of the modal length of the image region of the halftone cycle for the image. The length of an image region of the halftone cycle is shown in **figure 5.8**. Output 8 calculates the count against length distribution and outputs the length with the maximum count.

Output 9

This produces the size of the count for the modal halftone frequency. This output is illustrated by **figure 5.9** below. The relative sharpness of this peak, which is calculated for output 10 below, measures the noise on an image.

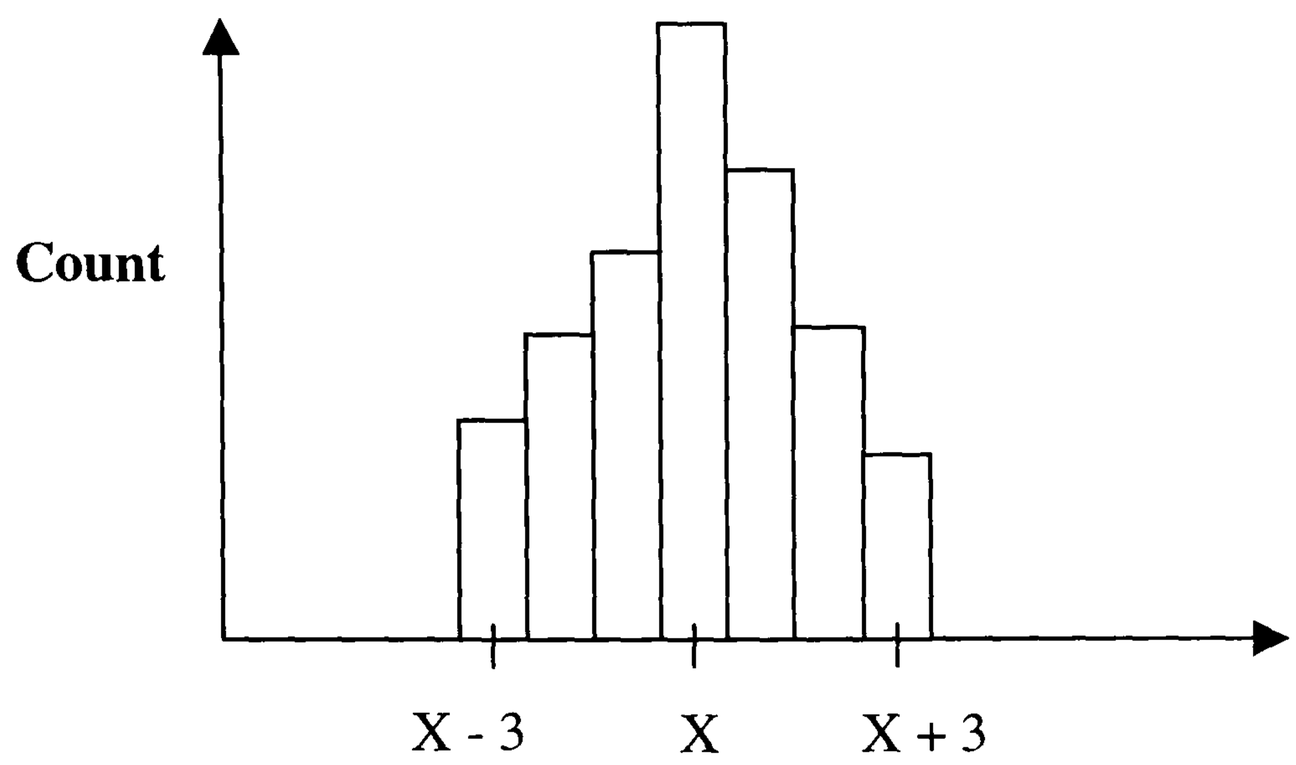


Figure 5.9 An illustration that shows the modal count and how the relative sharpness of this peak is calculated.

Output 10

This is a modal frequency factor and is defined as:

$$\text{Output } 10 = \frac{\text{Modal frequency count}}{\text{Frequency count over a width of } \pm 3 \text{ pixels from the modal position}}$$

Equation 5.1

This is illustrated in **figure 5.9** above. For an ideal image with a regular halftone pattern and some stochastic images that are generated on a computer and directly processed from

the computer file this quantity will be equal to 1. The smaller this value is, the greater the noise.

Output 11

This calculates the size distribution of the halftone cycles in an image. It does this by computing the number of halftone cycles for all the different cycle lengths. If there are more than five counts for a given length of cycle, a counter in the program is incremented by one. For an ideal computer generated image with a regular pattern and more than five cycles, the value of this counter will be equal to 1. The noisier an image is the higher the counter value will be.

Output 12

This calculates the average value of the ascending gradients in the X direction at the locations where the peaks have been detected. The method of determining the gradient value is illustrated below in **figures 5.10** and **5.11.**

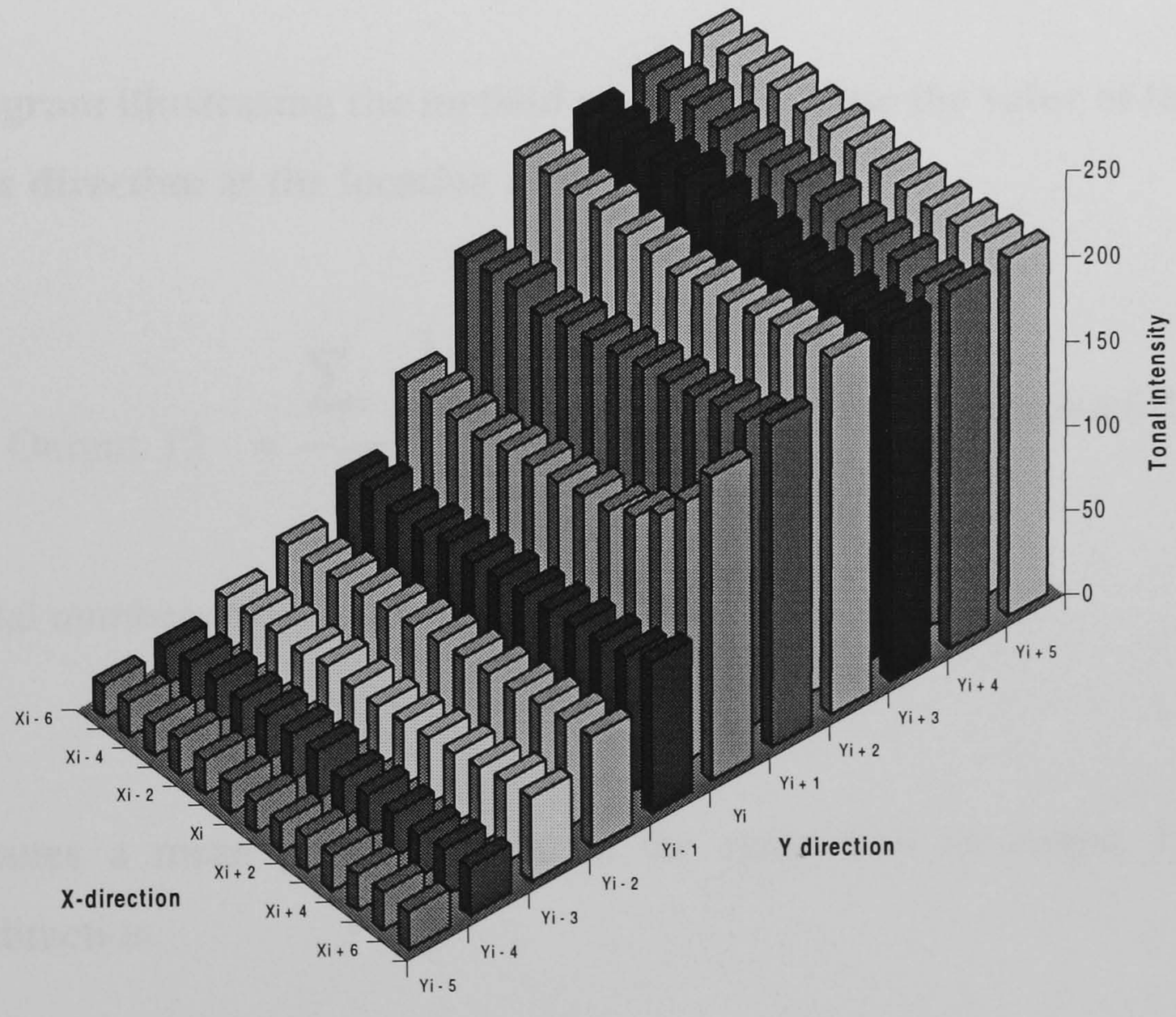


Figure 5.10. A three dimensional representation of the tonal intensity distribution near the edge of an image object.

Figure 5.10 is an illustration of a three dimensional intensity map generated by part of an image. As the intensity rises, the image moves from the background to the image object.

This transition can be detected by measuring the gradients as the image is scanned. If the gradient at a location exceeds a critical threshold value, then an edge exists at this location as shown in **figure 5.11.** Output 12 uses the calculated gradient values to compute a mean gradient value for the entire image. This equation is given in **equation 5.2** below.

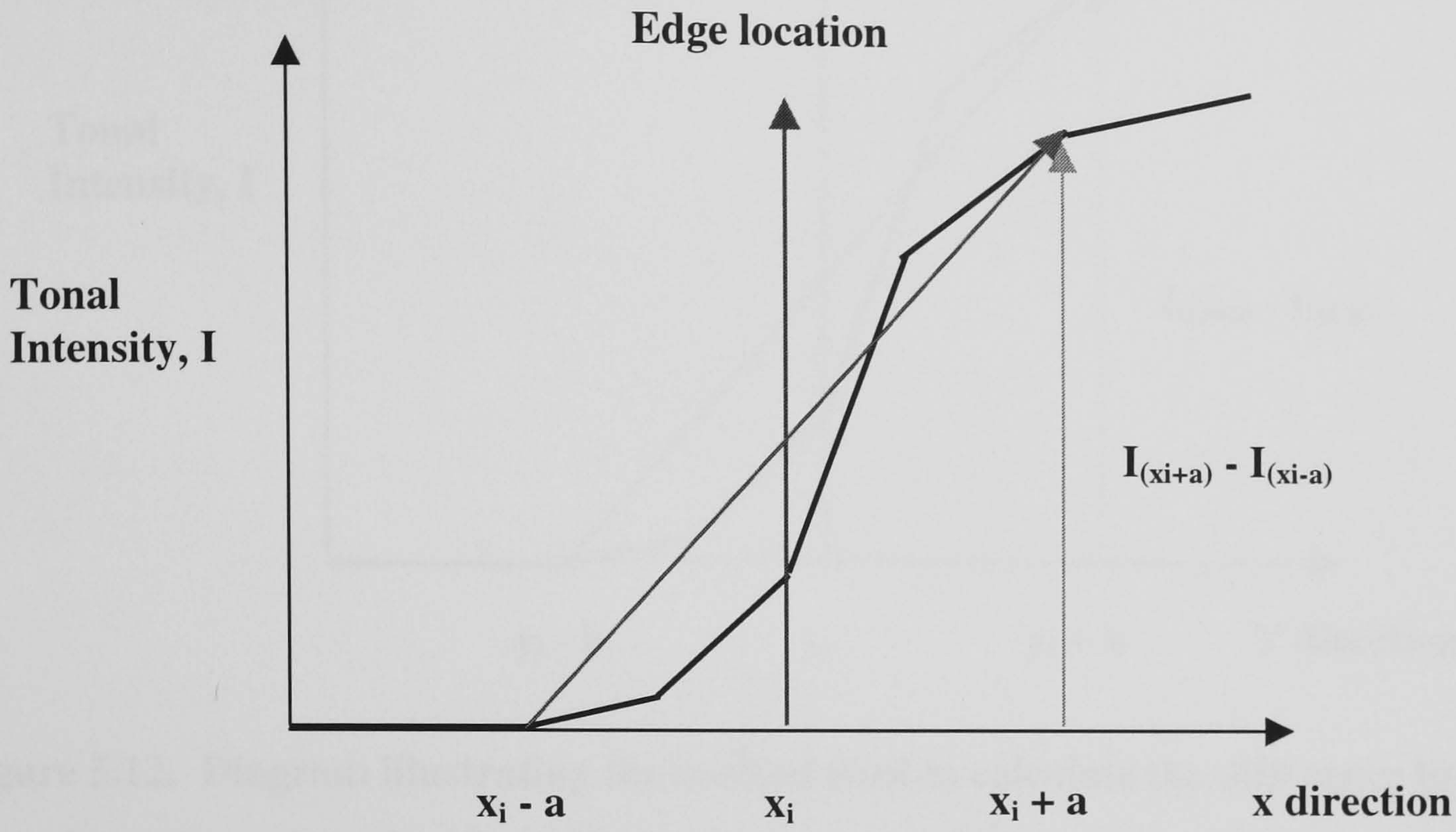


Figure 5.11. Diagram illustrating the method used to calculate the value of the gradients in the x direction at the location of the edges.

$$\text{Output } 12 = \frac{\sum_i \dfrac{I_{(x_i + a)} - I_{(x_i - a)}}{(x_i + a) - (x_i - a)}}{N} \qquad \textbf{Equation 5.2}$$

where N is the total number of detected edges.

Output 13

Output 13 computes a mean gradient value in the same way as output 12 for the perpendicular Y direction.

$$\text{Output } 13 = \frac{\sum_i \dfrac{I_{(y_i + a)} - I_{(y_i - a)}}{(y_i + a) - (y_i - a)}}{N} \qquad \textbf{Equation 5.3}$$

Output 14

This is a different method from the preceding output for calculating a mean gradient value in the Y direction. It calculates the difference in gradient values of two equal lengths

either side of a detected peak. This is illustrated in **figure 5.12** and given by **equation 5.4** below.

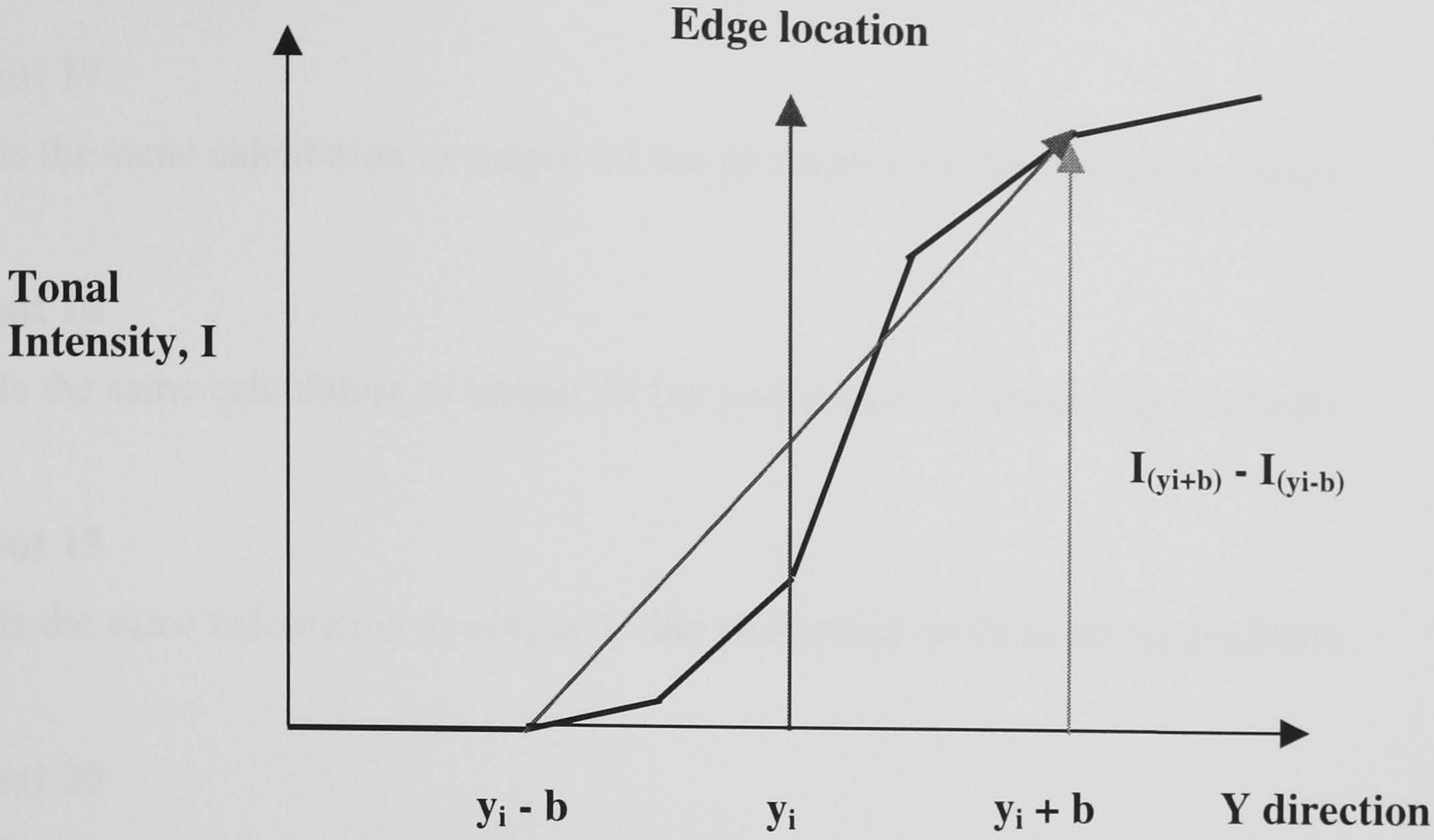


Figure 5.12. Diagram illustrating the method used to calculate the difference between the values of $y_i + b$ and $y_i - b$, at the location of the edges.

$$\text{Output } 14 = \frac{\sum_i \dfrac{\bar{I}(yi+b) - \bar{I}yi}{(yi+b) - yi} - \dfrac{\bar{I}yi - \bar{I}(yi-b)}{yi - (yi-b)}}{N} \qquad \textbf{Equation 5.4}$$

Output 15

This measures the position of the image on the screen. It achieves this by calculating the mean of all the x co-ordinates where a tonal gradient greater than the threshold value has been detected. This gives a vertical cursor position where an image has been detected. This is illustrated in **figure 5.13**.

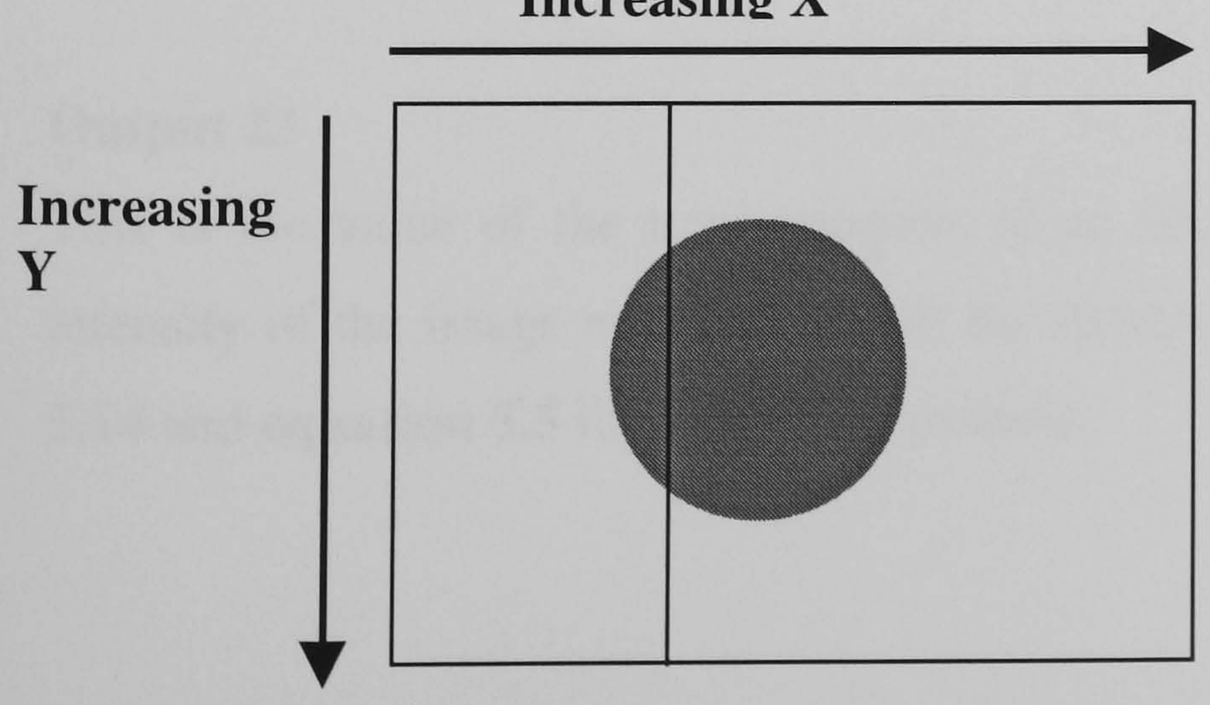


Figure 5.13. Output 15 is a calculation of the mean x co-ordinate of the rising edges. The approximate position of this line for an image of a circle is shown in this figure.

Output 16

This is the same calculation as output 12 but performed on descending gradients.

Output 17

This is the same calculation as output 13 but performed on descending gradients.

Output 18

This is the same calculation as output 14 but performed on descending gradients.

Output 19

This is the same calculation as output 15 but performed on descending gradients.

Output 20

This is the total number of detected descending edges as the scan moves from an image to a background region.

Output 21

This is the modal length of the detected image areas of the halftone cycles. This gives an indication of the darkness of an image. A greater peak to trough ratio usually indicates darker images.

Output 22

This is the modal length of the troughs of the halftone cycles. This gives an indication of the lightness of an image. A greater trough to detected image area markspace ratio usually indicates lighter images.

Output 23

This is the value of the total intensity of an image. It is calculated by integrating the intensity of the image regions over all the detected cycles in the image window. **Figure 5.14** and **equation 5.5** illustrates this process.

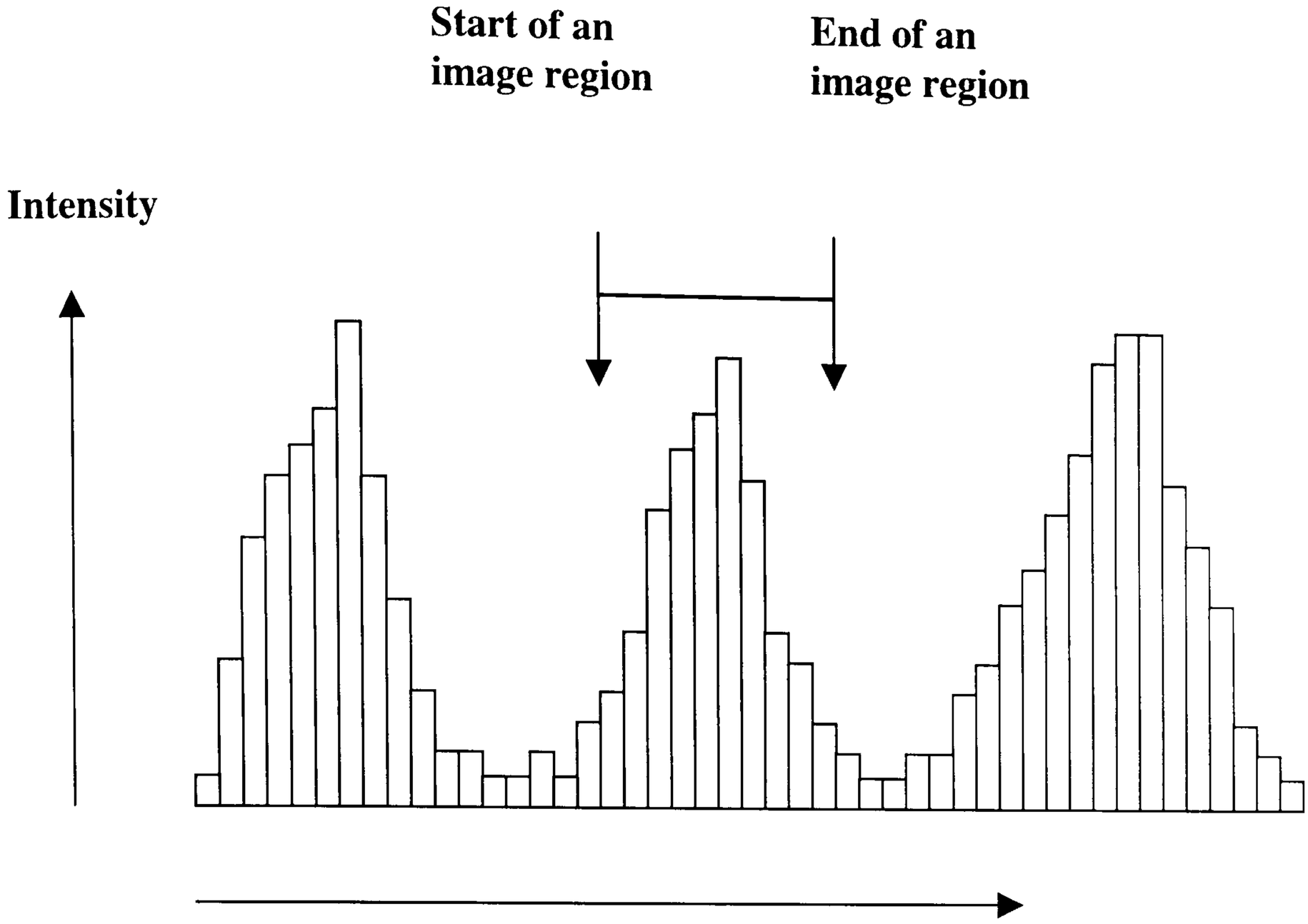


Figure 5.14. The diagram above with equation 5.5 below illustrates and explains how the total image intensity is computed from a halftone.

$$\text{Total intensity} = \sum_{\text{for all cycles}} \sum_{\text{end of Im}}^{\text{start of Im}} \text{tonal value} \qquad \textbf{Equation 5.5}$$

Where Im is the image region

Output 24

This outputs the number of pixels counted for the tone of an image with the largest population of pixels. It is found by firstly determining the tonal distribution for an image. This is the pixel count against tonal value. The program then searches for the maximum. This is illustrated in a histogram in **figure 5.15** below. If two or more tones fulfil this requirement, then the program returns the value of the pixel count nearest zero, which is the lightest tone.

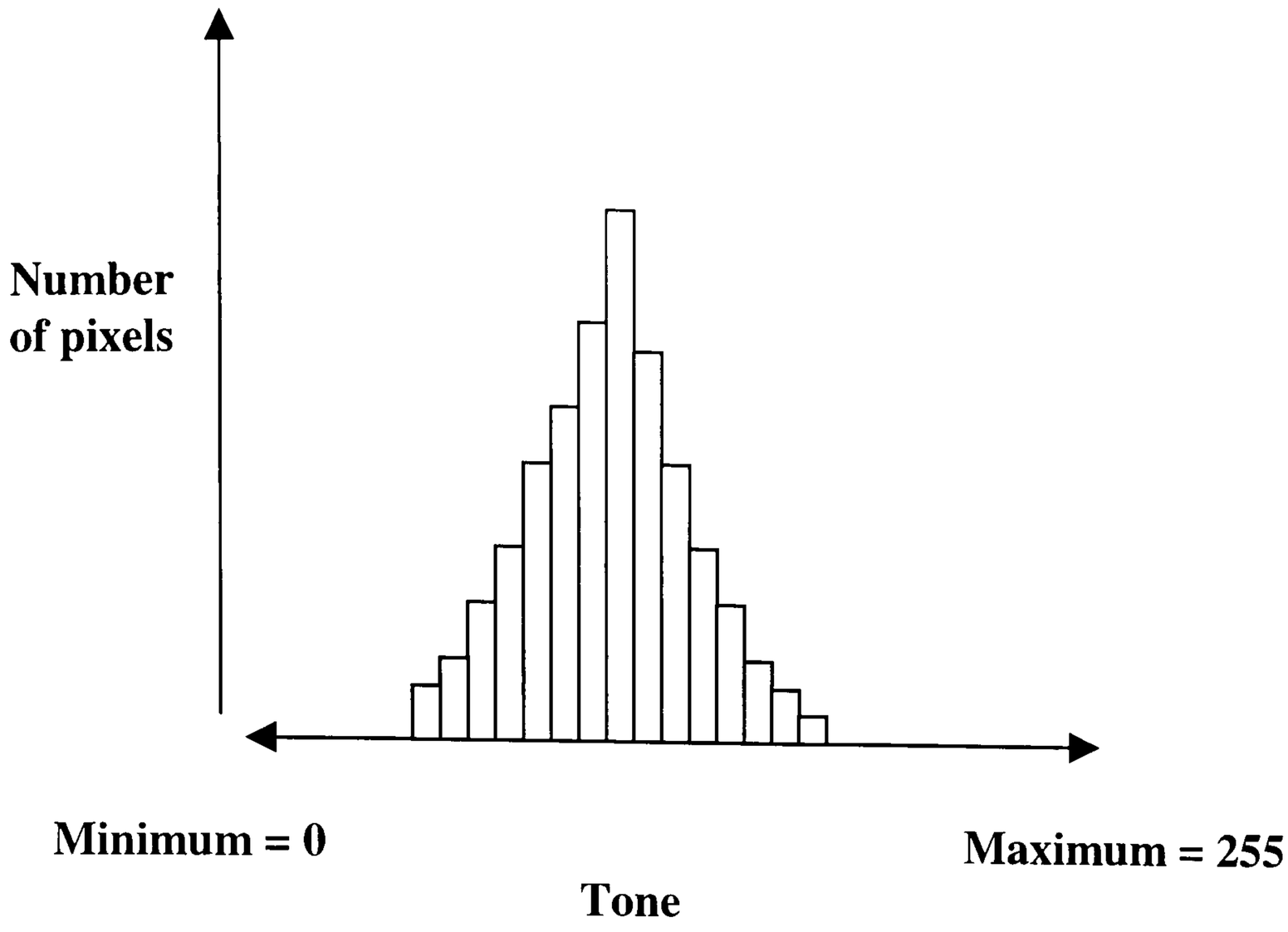


Figure 5.15. This is a graphical representation of tone against pixel population for an image. Output 24 gives the value for the size of the peak.

Output 25

This is the value of the tone where the largest population of pixels is situated. If two different tones fulfil this requirement, the tone nearest zero is recorded.

Output 26

This outputs the number of pixels counted for the tone of an image which has the second largest population of pixels. If two or more tones fulfil this requirement, then the program returns the value nearest zero, which is the lightest tone.

Output 27

This is the value of the tone where the second largest population of pixels is situated. If two different tones fulfil this requirement, the tone nearest zero is recorded.

Output 28

This outputs the number of pixels counted for the tone of an image which has the third largest population of pixels. If two or more tones fulfil this requirement, then the program returns the value of the pixel count nearest zero, which is the lightest tone.

Output 29

This is the value of the tone where the third largest population of pixels is situated. If two different tones fulfil this requirement, the tone nearest zero is recorded.

Output 30 –35

These outputs investigate the background areas of the image window in the same way and sequence as for the image areas in the preceding outputs 24-29.

Output 36

This analyses the smoothness of the edge of an image. It does this by producing the number of turning points on the edge of an image. This can be explained by using the following example illustrated in **figure 5.16 a** and **b**. These are diagrams of two edges that have been magnified. The edge in **figure 5.16 a** on the left can be considered more likely to be straight than that of **figure 5.16 b** on the right. This is because the number of gradient sign changes or turning points is less for the edge of the image in **figure 5.16 a**..

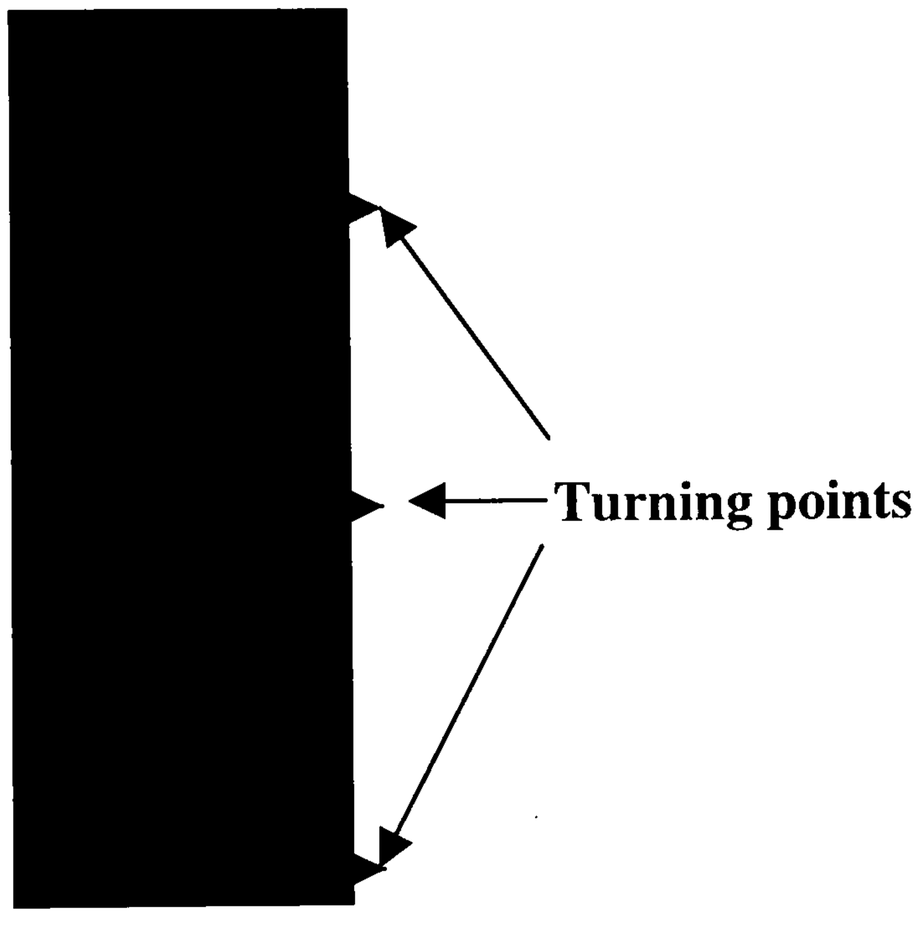


Figure 5.16 a. An edge with fewer turning points is more likely to be perceived as less ragged when compared with an edge with more turning points.

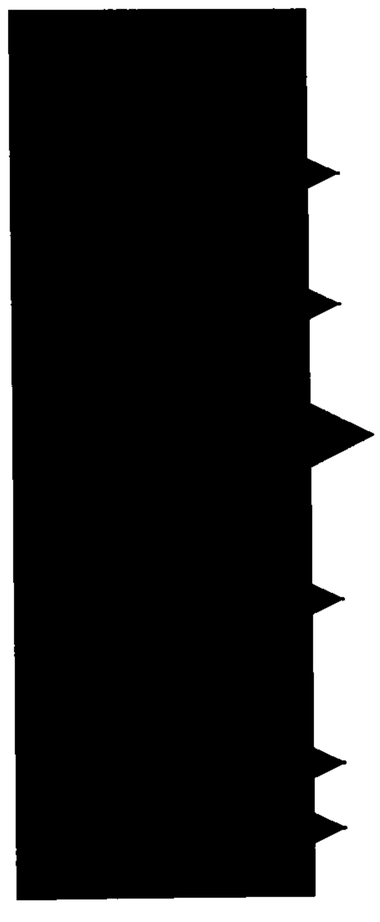


Figure 5.16 b. An edge with more turning points is more likely to be perceived as more ragged when compared with an edge with fewer turning points.

Output 37

This analyses the differences in the values of adjacent gradients. This can be seen using **figure 5.10**. The tonal gradient Gx_i at position x_i is given by the equation:

$$G_{xi} = \frac{I_{(xi + a)} - I_{(xi - a)}}{(x_i + a) - (x_i - a)} \qquad \textbf{Equation 5.6}$$

Output 37 uses **equation 5.6** to calculate ΔG_{mean} in **equation 5.7**. This is a mean of the sum of the difference of tonal gradients along the y axis for the final descending edge detected in a line scan. This is illustrated in the example shown in **figure 5.17** for a text character.

$$\Delta G_{mean} = \frac{\sum_{j-1} | G_{last,\ j} - G_{last,\ j\ +\ 1} |}{N - 1} \qquad \textbf{Equation 5.7}$$

where $G_{last,\ j}$ and $G_{last,\ j+1}$ are tonal gradients for the final descending edge of adjacent linescans.

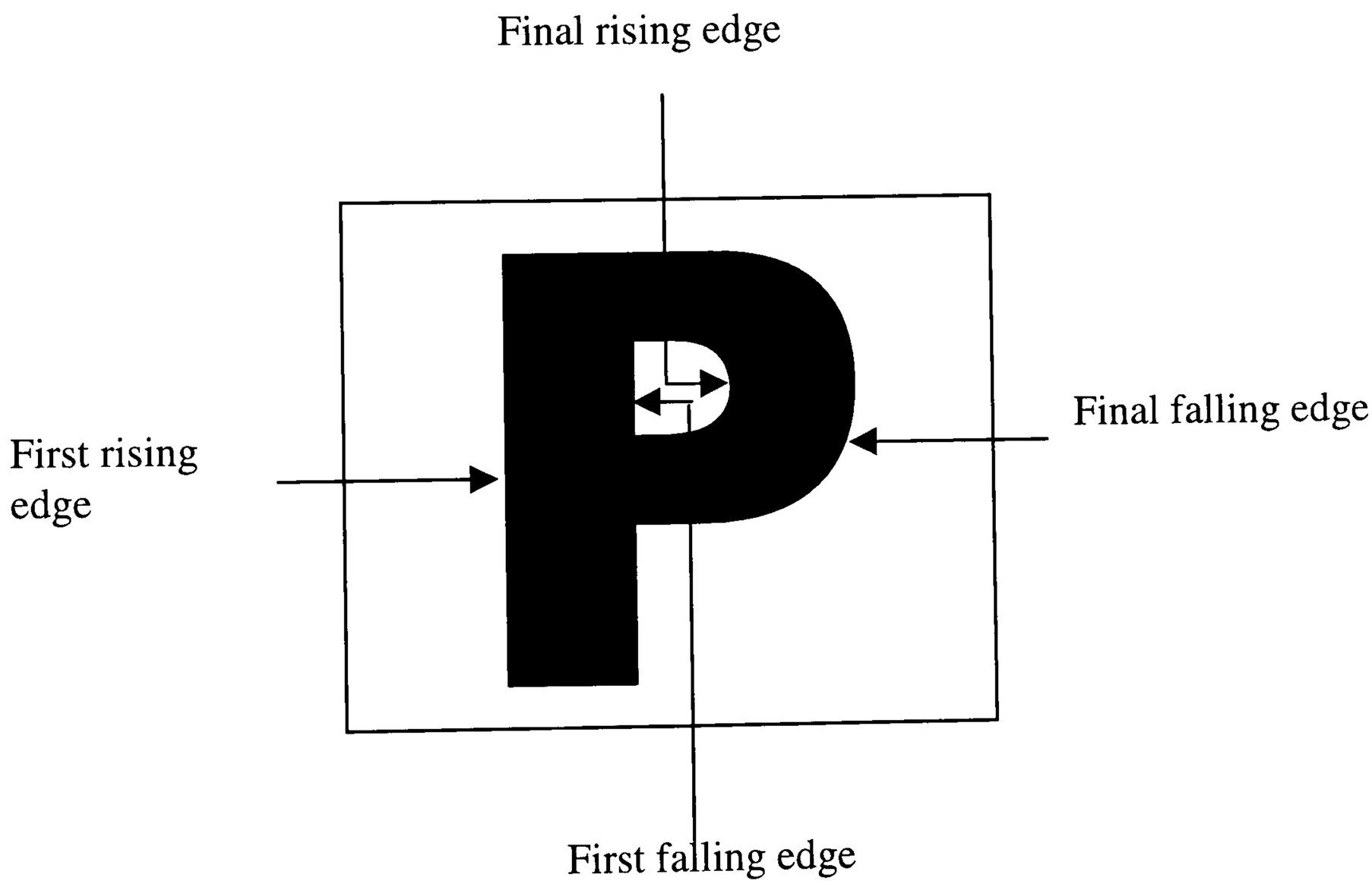


Figure 5.17. Output 37 computes ΔG_{mean} for the final falling edge.

Output 38

This is a count of positive gradients. A procedure detects the co-ordinates where the final descending tonal edge is located. This is carried out for every line of the display window that is occupied by the image. If the following statement is true

$$X_{Last\ ,\ j+1} > X_{Last\ ,\ j}$$

where $X_{Last\ ,\ j+1}$ and $X_{Last\ ,\ j}$ are the positions of the final falling tonal edges produced by adjacent lines, a counter is incremented by 1. The total of this count is divided by the number of lines of the display window occupied by the image.

Output 38 can be understood by considering the following example in **figure 5.18a and b.**

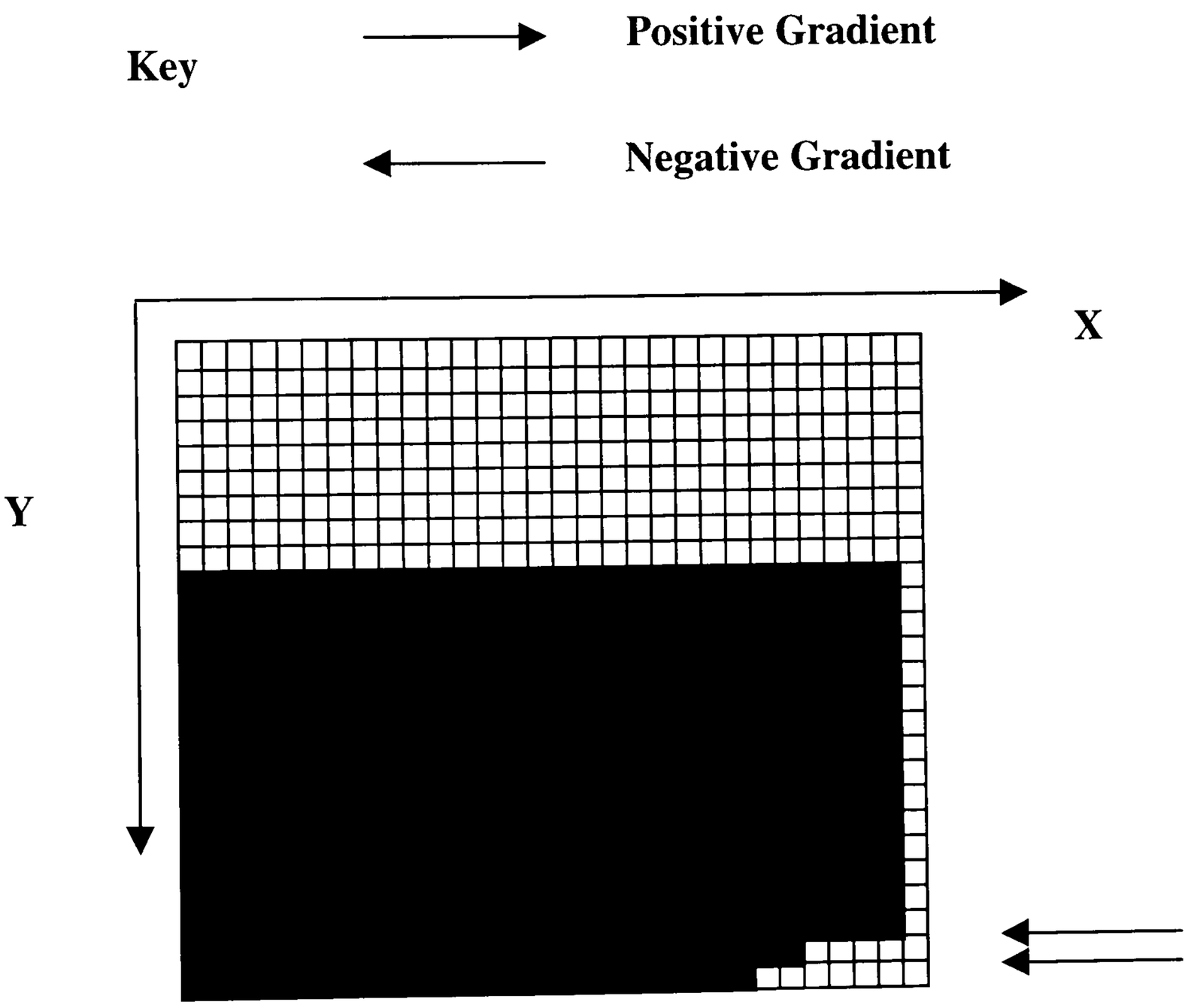


Figure 5.18 a. The edge has no gradients larger than 0. Therefore output 38 returns a value of 0.

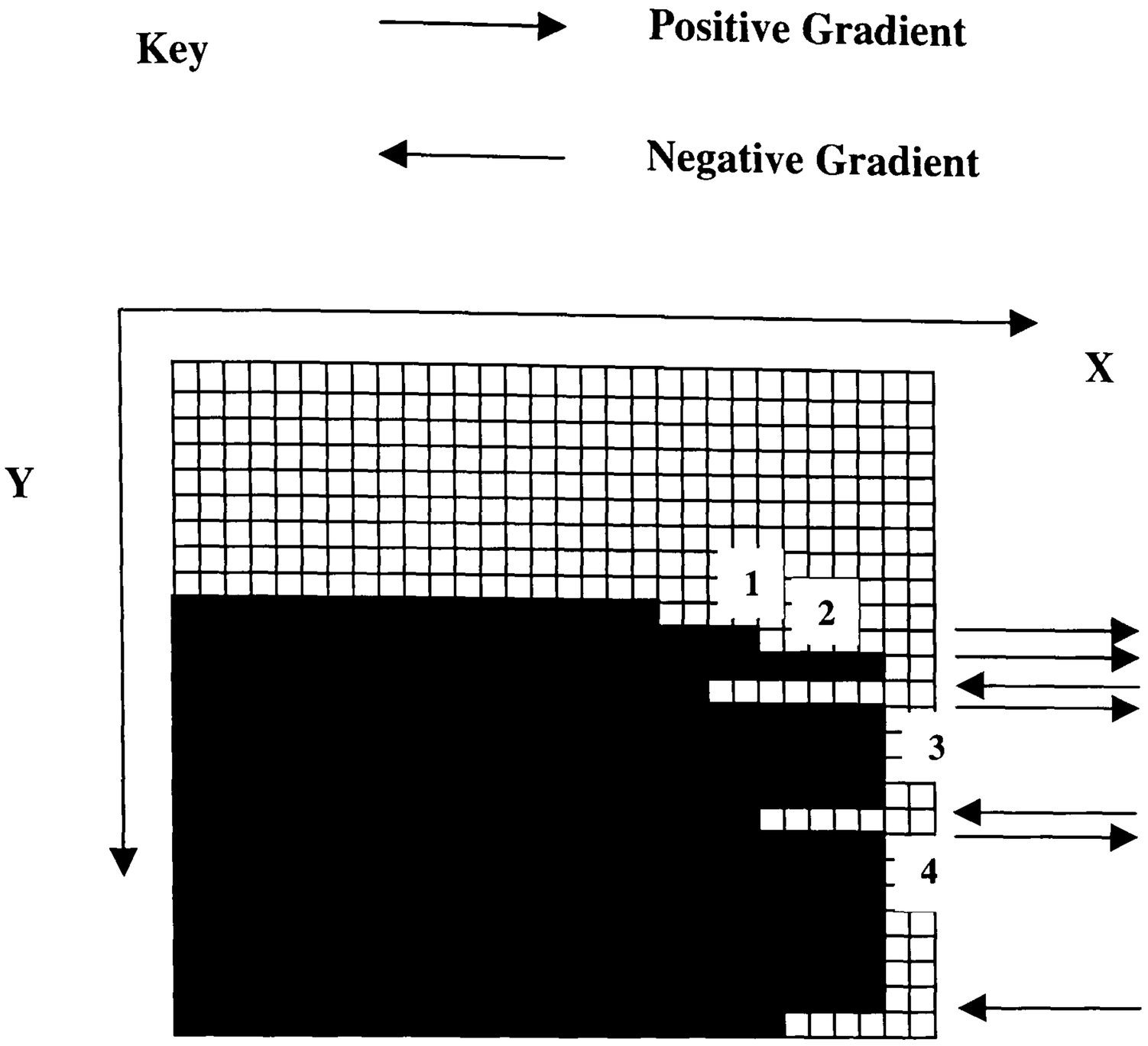


Figure 5.18 b. The edge has both positive and negative gradients. The positive gradients are added together. This result is divided by the total number of lines in the part of the display that contains the image.

Output 39

This gives the result of counting the number of gradients detected on the final descending edge of an image which has the value of 1 pixel. The gradients are determined in the same way as the preceding output. The difference is that the sign of the gradient is ignored and the result is not divided by the number of lines that contain the image. This procedure is described by the conditional statement below and by **figure 5.19**.

$$\text{If } |X_{\text{Last , j+1}} - X_{\text{Last , j}}| = 1 \quad \text{Then count} = \text{count} + 1$$

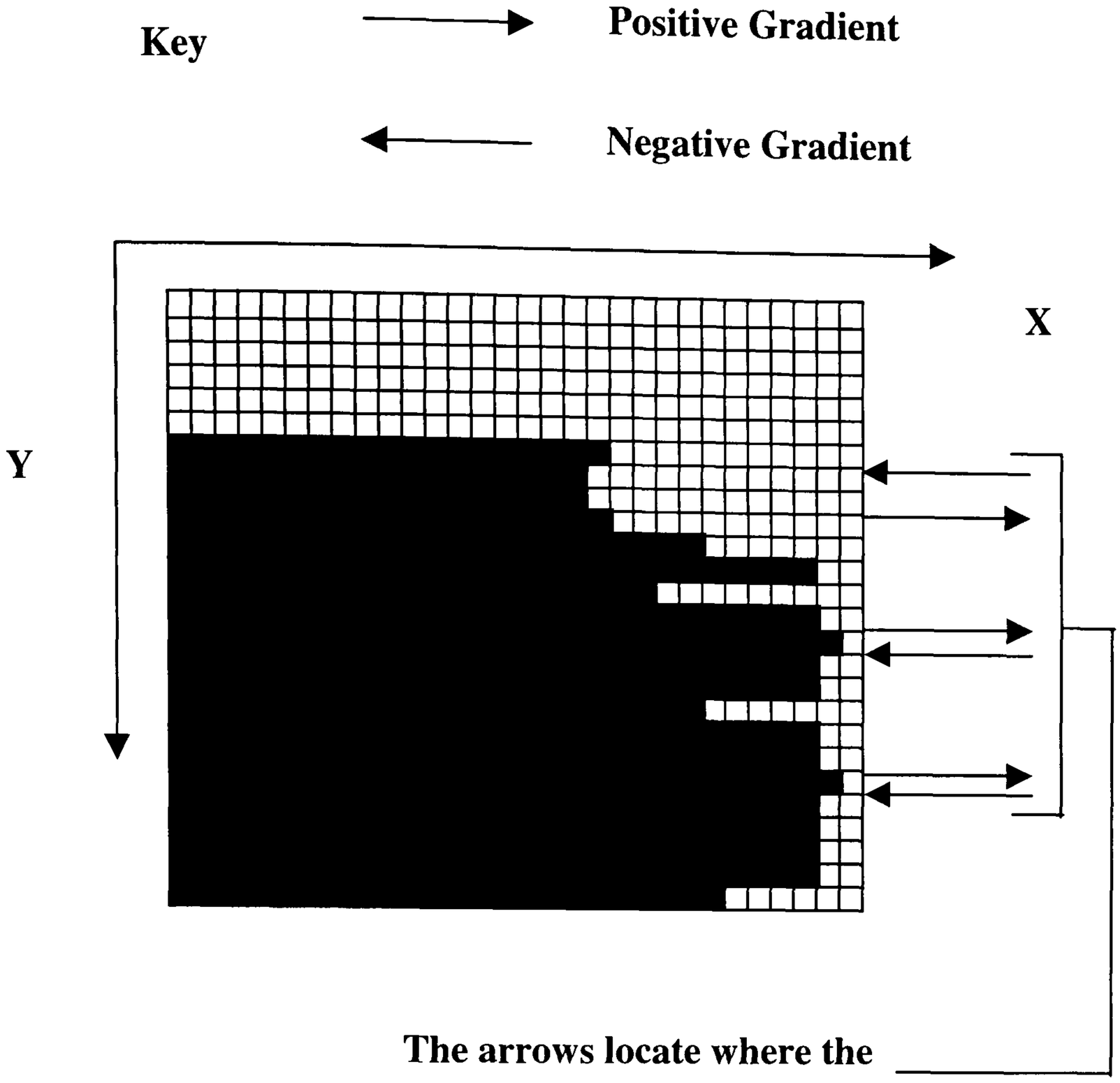


Figure 5.19. Output 39 is a count of all the gradients = 1 for the final descending edge of an image computed using the conditional statement immediately above the diagram.

Output 40

The computational method is the same as for output 39 but the gradient value required for incrementing the counter is 2 pixels. Therefore the conditional statement for output 39 changes to:

$$\text{If } |X_{\text{Last}, j+1} - X_{\text{Last}, j}| = 2 \text{ Then count} = \text{count} + 1$$

Output 41

The computational method is the same as for output 39 but the gradient value required for incrementing the counter is 3 pixels. Therefore the conditional statement for output 39 changes to:

$$\text{If } |X_{\text{Last}, j+1} - X_{\text{Last}, j}| = 3 \text{ Then count} = \text{count} + 1$$

Output 42

The computational method is the same as for output 39 but the gradient value required for incrementing the counter is 4 pixels. Therefore the conditional statement for output 39 changes to:

$$\text{If } |X_{\text{Last , j+1}} - X_{\text{Last , j}}| = 4 \quad \text{Then count} = \text{count} + 1$$

Output 43 – 52

These are frequency filters which will be discussed in the conclusion of this thesis.

5.6 Computer Programming Languages

The algorithm described in the last section was developed experimentally. To do this, requires a computer programming language and other software tools. The next two sections describe these and how they were used to produce the program from the algorithm in **section 5.5**. The first stage explains how the computer language used for the development of the pre-processing program was chosen.

Computer languages can be separated into three categories. These are listed below:

1 Low level assembler languages
2 Mid level languages
3 High level languages

Low level assembler uses short abbreviations and symbols as commands that the microprocessor can execute. These codes go directly to the microprocessor. This means that the programs run very fast. To program in assembler requires a detailed knowledge of the microprocessor which it uses. This, coupled with the symbolic nature of the code means that it is difficult to write programs easily and quickly in this type of language.

C can be regarded as a mid-level language. It uses short commands based on English and symbols. Parts of this language are similar to assembler and they both can be used together in the Dos operating system. C is a very compact language; a few short instructions can perform a large amount of computing. As it is a compiled language it can execute code very quickly and is more simple than assembler and is widely used for writing scientific software. A problem with C is that the programs can be difficult to

debug since the error messages can be difficult for inexperienced programmers to understand and decipher. C++ is an extension of C. It can do everything that C can do. The main advantage of using C++ is that it is an object-orientated language which enables more structured programs to be written than can be achieved with C.

The main advantage of using BASIC is that its instruction set is in English. This makes it easy to learn, understand, write code and to debug. It was for these reasons that Visual Basic was chosen for translating the algorithm into a program that could be implemented on a personal computer. Other advantages of this computer language include:

1 It enables the Microsoft graphical user interface to be exploited which aids data visualisation.

2 Other Microsoft applications such as EXCEL, RECORDER and ACCESS can be integrated into the Visual Basic application.

3 Visual Basic has visual tools which simplifies programming. A tool can be a command button that executes a section of BASIC code or a text box that displays data.

5 Visual Basic also uses the concept of forms. A form can be a complete program or part of a program. The program can therefore be written in a modular way. Each module or form can be individually tested. Testing a large program in a modular way is simpler than testing the entire program as a single unit.

A major drawback in using Visual Basic that was observed in the early stages of this investigation was that it executes the code much more slowly than for example visual C. C code can run approximately ten times faster than Visual Basic 3. However, it was decided to implement the algorithm in Visual Basic since the benefits of points 1-5 above were felt to outweigh the disadvantage of its slowness. Visual Basic 3 was at a later stage upgraded to version 5. This speeded up programming by a factor of 1.6. Data acquisition from the image capture board also governs the data processing speed. Data acquisition from the image capture board is faster using Visual Basic version 5 than using version 3. This is because version 5 is compatible with the Matrox MIL software libraries that enables direct

data transfer from the frame capture board. This and the other details associated with implementing the pre-processing algorithm will now be discussed in greater depth.

5.7 The development and implementation of the pre-processing algorithm in Visual Basic

The first step was to learn how to use the features of Visual Basic version 3 required to develop the pre-processing algorithm. This consisted of the syntax of the language, the Visual Basic toolbox that simplifies the coding and the use of forms which enable the coding to be produced in a modular form. The initial goal in the development of the pre-processing program was to interface the Matrox imaging system to Visual Basic

The first problem that had to be solved was the acquisition of image data from the Matrox Comet image capture board. This data had to be in a format that could be recognised by Visual Basic. This is either a data (dat) or comma separated value (CSV) file format. It was discovered that Visual Basic version 3 was unable to do this. The problem was solved using a commercial application called Ximage developed by Foster Finlay Limited. Ximage can convert a monochrome tagged image file (TIFF) into a CSV file. The Comet software can recognise the TIFF format. Therefore the data can now be read using Visual Basic. The method of changing the data format for Visual Basic compatibility is shown in **figure 5.20** below. The Paintshop Pro application is necessary in order to convert the TIFF colour or binary monochrome data format of the Comet software to a greyscale format. The binary format represents data in steps of 8, while a Paintshop Pro greyscale represents data in steps of 2. Therefore the greyscale format has a resolution which is a factor of 4 greater than the Comet monochrome binary file. There was also a software fault with Ximage that had to be corrected using Microsoft Word. The CSV file that the Ximage program produced had a missing comma in its header which produced an error message when Visual Basic was used to read it. Microsoft Word was employed to place the required comma in the file.

The entire system was automated using Microsoft Recorder and the Windows 3.1 operating system. This system was unreliable, slow and unsatisfactory for the processing of the thousands of images required for this investigation. Faults occurred when the file settings of any of the applications were adjusted at all or their icons on the screen were moved, which happened when the computer was used for other tasks. This problem could not be solved even if the computer was dedicated solely to the print quality assessment

task. Also for unknown reasons the Recorder Program could malfunction. Therefore no remedy was found for this problem.

It was decided that the development of more sophisticated data acquisition systems was time consuming with no guarantee of success. For example, a method exists for converting an image file directly into a data file that uses the Matrox Cometlib software library provided with the Comet hardware. However, to implement this, requires extensive programming in the C programming language. The priority was to obtain meaningful results before upgrading the software. This meant that the pre-processing algorithm would be able to measure print quality variables and further software development might improve the results. The first stage in the creation of the pre-processing software was to develop the outputs that could measured the intensity, halftone frequency and noise content of images.

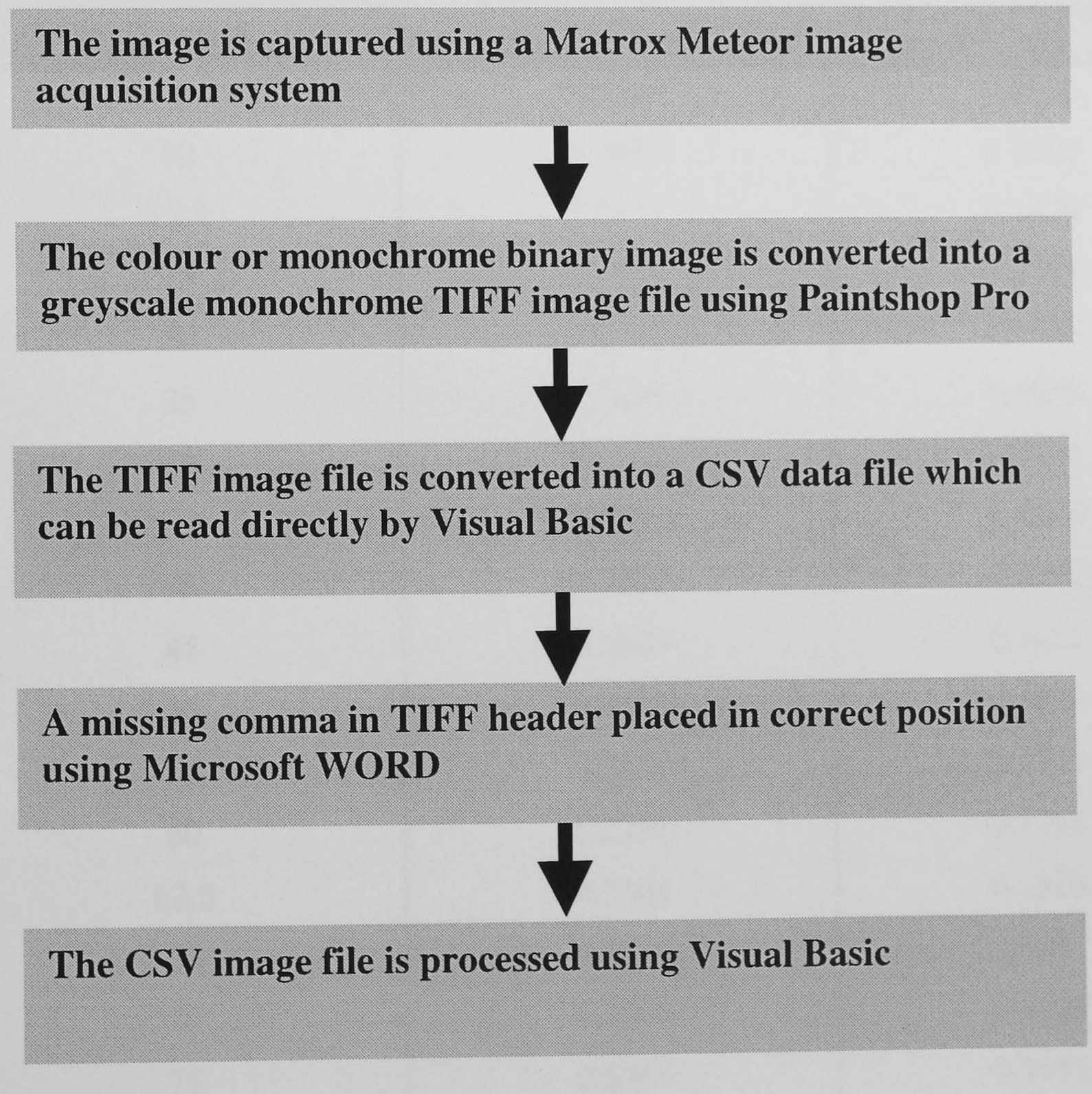


Figure 5.20. A flowchart showing the program to read data into Visual Basic that was used in the initial investigation.

The code that read the CSV data file was written into a Visual Basic form. The data was allocated into a (768,576) matrix array. The data was transferred into another form where

all the image analysis operations were developed and executed. The other forms were used to output graphical information on the data or used as an output software port for Matlab.

The first image analysis operations that were developed were those that measured the the noise(output1). A test was devised for an early version of the pre-processing program to determine whether it could differentiate between a set of 1cm squares of different halftones produced on a Hewlett Packard laser printer, and photocopies of these images produced on a Sharp SF 2022 photocopier. It performed this task successfully using the output 1 variable shown in **table 5.1**. **Table 5.2** shows this result. The tonal values in the left-hand column originate from the Word 6 software used to produce the image file for the

Relative Tonal Intensity %	Hewlett Packard 4M Laser Printer	Sharp SF - 2022 photocopier
5	0.2766	0.0836
10	0.2491	0.1506
12.5	0.2378	0.1428
15	0.2526	0.1587
20	0.2575	0.1523
25	0.2638	0.1658
30	0.2704	0.1601
35	0.2571	0.1764
40	0.2621	0.1976
45	0.2824	0.1682
50	0.2518	0.1636
55	0.2642	0.1223
60	0.2221	0.1161
62.5	0.2264	0.1583
65	0.2311	0.1687
70	0.2568	0.1092
75	0.2464	0.1616
80	0.2201	0.1249
85	0.2511	0.0604

Table 5.2. The data shows that the noise on the photocopy is greater than on the laser original.

squares. These tonal values are used in the instructions from the software to the printer to produce a particular density of print. Therefore, they cannot be taken as the true densities, since different printing sources can produce different densities of print from identical Word 6 tonal values. The middle and right-hand columns measure the noise for the laser print and its photocopy respectively. A high value for output 1 indicates a low noise. The photocopier therefore, was shown by the image analysis system, to introduce noise over the entire tonal range of the images from the laser printer used in this test. These were early results that demonstrated that it was possible to take print quality measurements using the initial pre-processing program. However, the time needed to take the measurements was long, about 15 minutes per image. It was decided that the development of the final algorithm would be hindered because of this, since it was estimated that thousands of measurements would be required. As the concepts behind the algorithm were shown to produce meaningful print quality measurements, a more sophisticated, faster version of the system was developed in order to fulfil the objectives of the project. This was achieved using the Matrox Meteor hardware, its accompanying MIL software and Visual Basic Version 5 to develop the program described in **section 5.5**.

5.8 The final pre-processing algorithm

The preceding section gave a history of the hardware and software development for the data pre-processing stages of the image analysis system. In this section a description of the procedures used to initially test the full, final pre-processing program will be given. The code for the program is listed in **appendix 1**. This includes the results of a full calibration and analysis of the software using a computer generated halftone image and live halftone images of a 1 cm square.

The structure of the Visual Basic pre-processing program is given in **figure 5.21.** The entire program is enclosed in a container MDI form. Each of the other forms are known as child forms. The role of the MDI form is to open the different child forms at the same time and to keep them in an unique folder. More child forms can be added to the program or some may be removed at anytime. This last point, highlights the modular nature of the program and hence the modular approach to the entire project.

MIL LIBRARIES

MIL BAS FILE
This contains the MIL library variables.

FORM 1
Data is captured and allocated into a 2D data buffer using the Matrox MIL libraries. The data into a format that can be processed by Visual Basic 5. This is in a 768 x 576 matrix.

BAS FILE
This contains all the variables common to all the forms in the program.

FORM 2
A data file is created. Image data is converted into a CSV data file that can be read by Microsoft EXCEL ™. This form can be bypassed. It is only needed if the raw data from the buffer was needed to be inspected during the development of the program.

FORM 3
Features are extracted from the image data matrix. These measurements include the following measurements:
Halftone frequency, contrast, edge defects, image noise, and image sharpness.

FORM 4
A tool plots the image component of the halftone frequency pattern.

FORM 5
A tool plots the background component of the halftone frequency pattern.

Continued on the next page

Continued on the next page

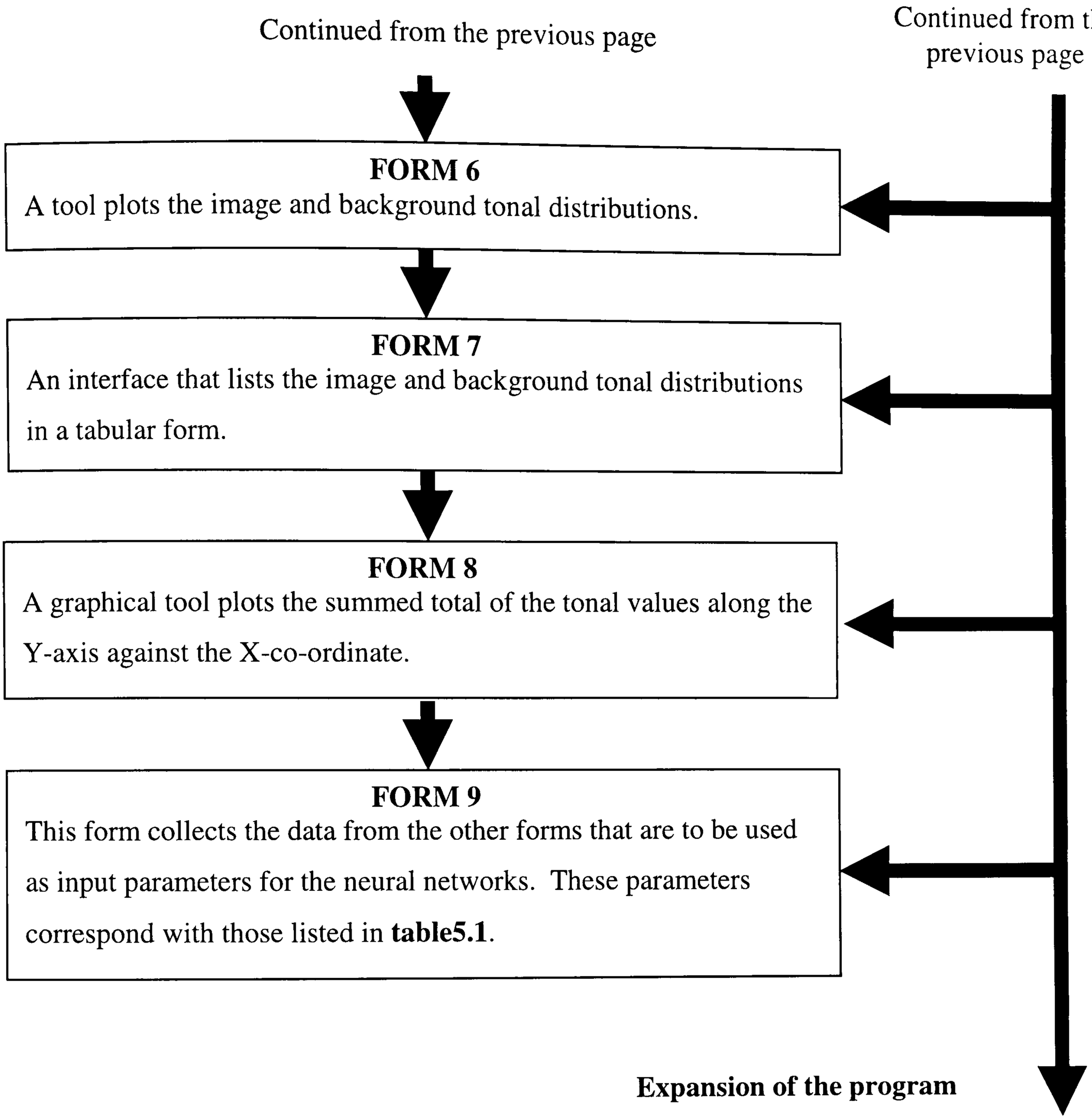


Figure 5.21. A flowchart which describes the functions of the different Visual Basic forms in the pre-processing program.

5.9 The initial evaluation of the pre-processing program using a computer generated test grid

The first test for the pre-processing program used a computer generated test grid. This was used to see if the output values of the pre-processing program were correct in the absence of camera and lighting noise. The grid is shown below in **figure 5.22** inside the Matrox image window. It was produced by drawing a square with a length dimension of 20 pixels in Word 7 and pasting a series of these squares in Paintshop Pro. The squares were then arranged into a regular square pattern. The subsequent image was stored as a TIFF file

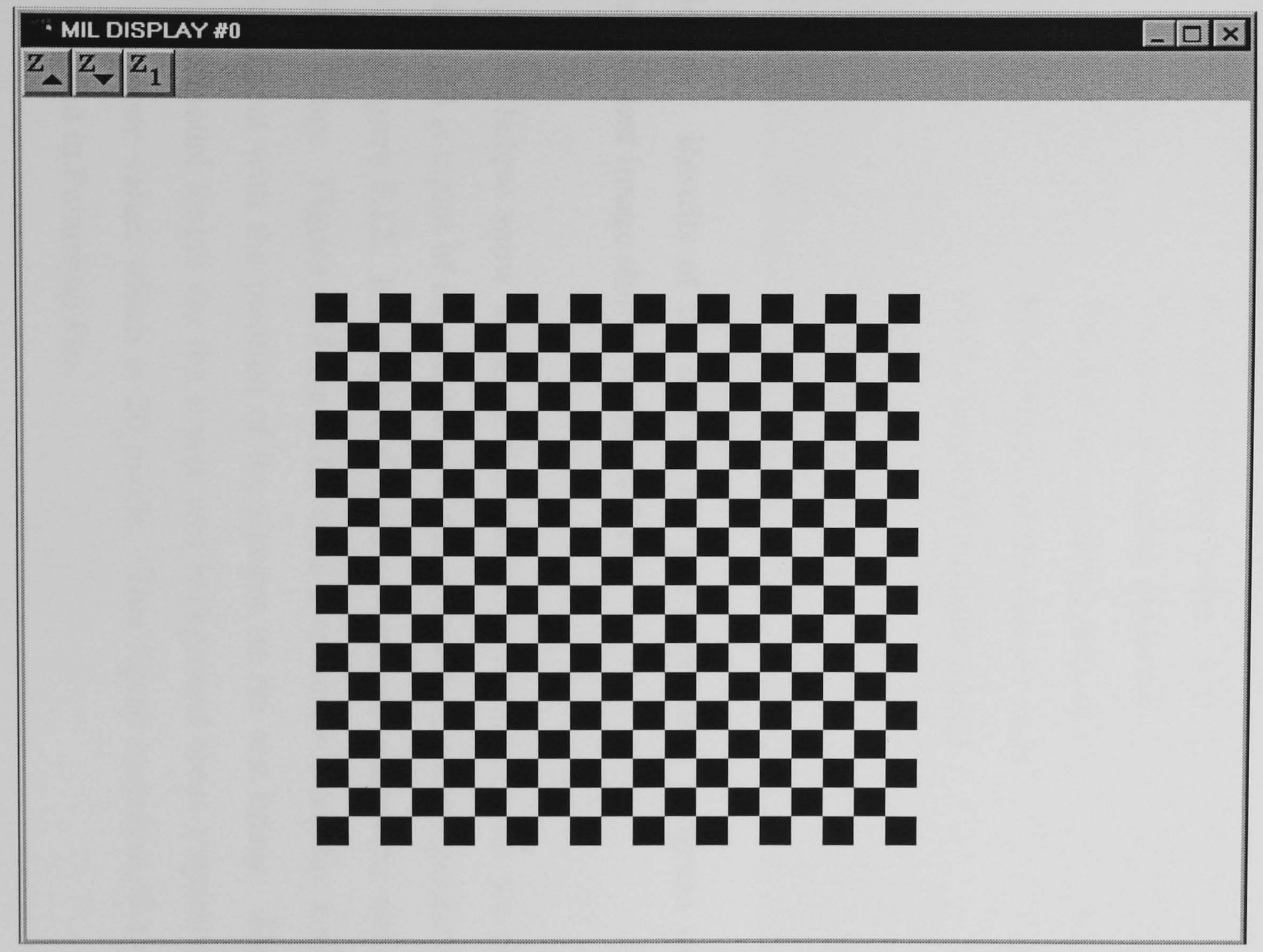


Figure 5.22. The test grid used to calibrate the image analysis system

which is recognised by the Matrox Mil imaging software. The dark regions have a value of 255 and the light regions a value of 0. Results from the test are also shown below in **table 5.3**.

Output number	Description of function	Output value in pixels
1	Noise measurement	1
8	Length of the modal halftone cycle	40
9	Number of image regions detected	3620
10	Modal frequency factor	1
12	Value of ascending gradients	255
16	Value of descending gradients	-255
21	Modal length of the image region	20
22	Modal length of the background region	20
23	Total image intensity	1.76×10^6

Table 5.3. Results of the test on the pre-processing program using the computer generated test image shown in figure 5.18.

The charts below show some of the results generated by the pre-processing program. **Figure 5.23** is a plot of the positions of the detected ascending gradients on the test image shown in **figure 5.22**. It is a plot of the x co-ordinate against the number of edges in the detected image. **Figure 5.24** shows the tonal distribution along the x-axis. These two plots are consistent with the position of the squares on the test image. **Figure 5.25** and **5.26** show the modal length for the image and background areas respectively. Both of these have only one value, which is 20 pixels. This figure corresponds to the length used to draw the grid in Paintshop Pro.

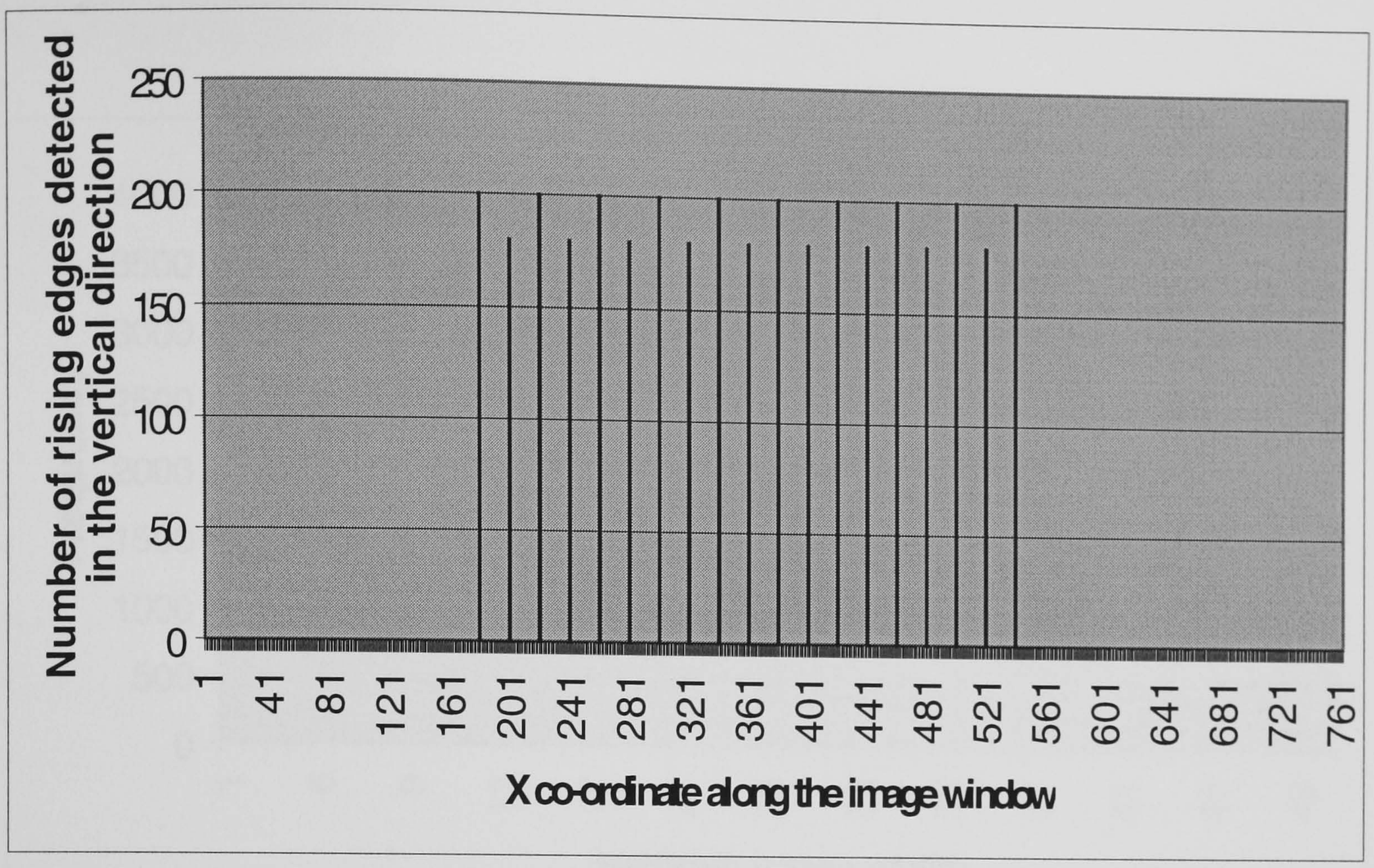


Figure 5.23. The positions of the ascending tonal gradients which were located by the pre-processing program. This plot shows that the program is correctly calibrated for the test image.

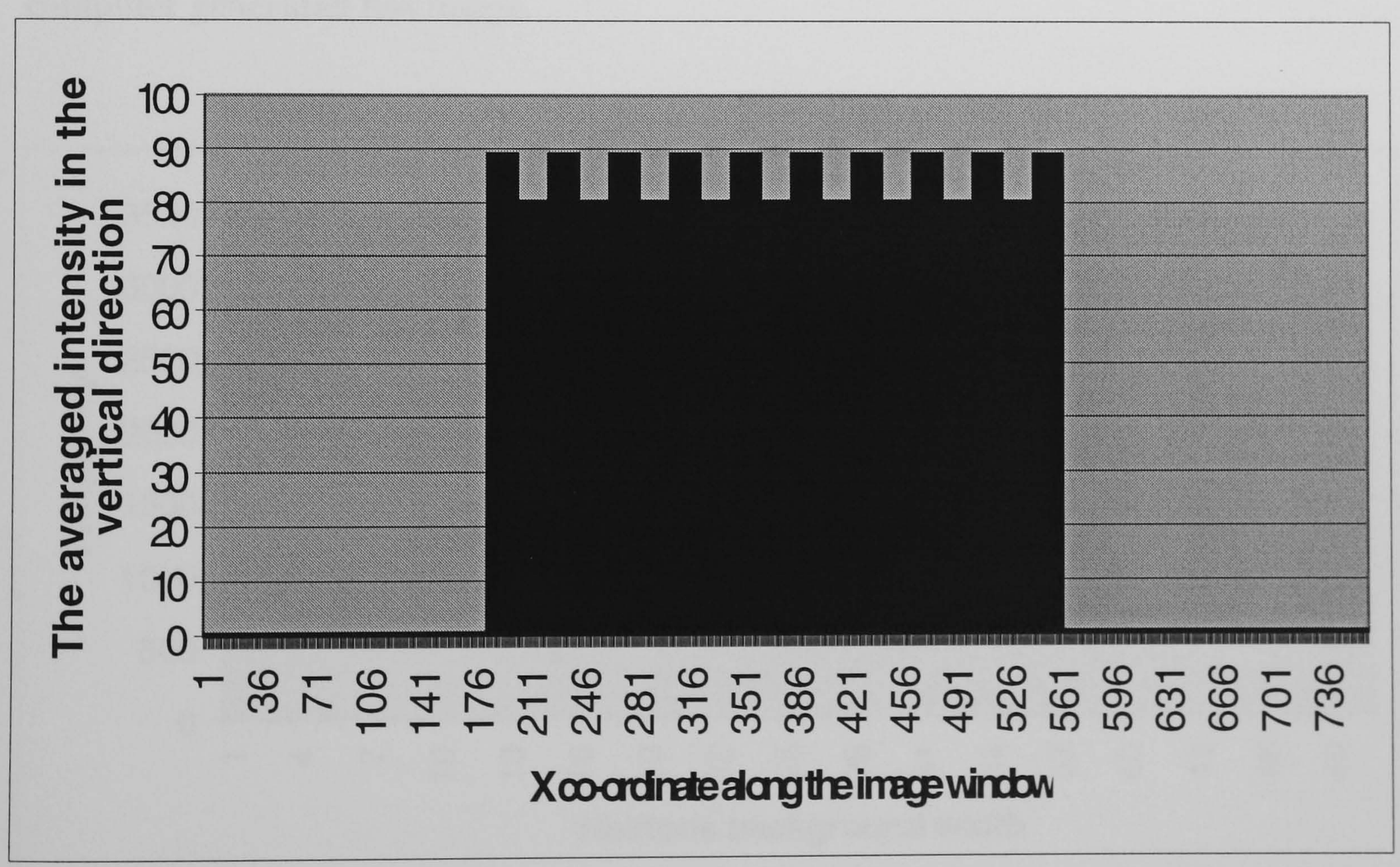


Figure 5.24. The mean tonal distribution along the x-axis. This plot corresponds with the preceding plot and the test image.

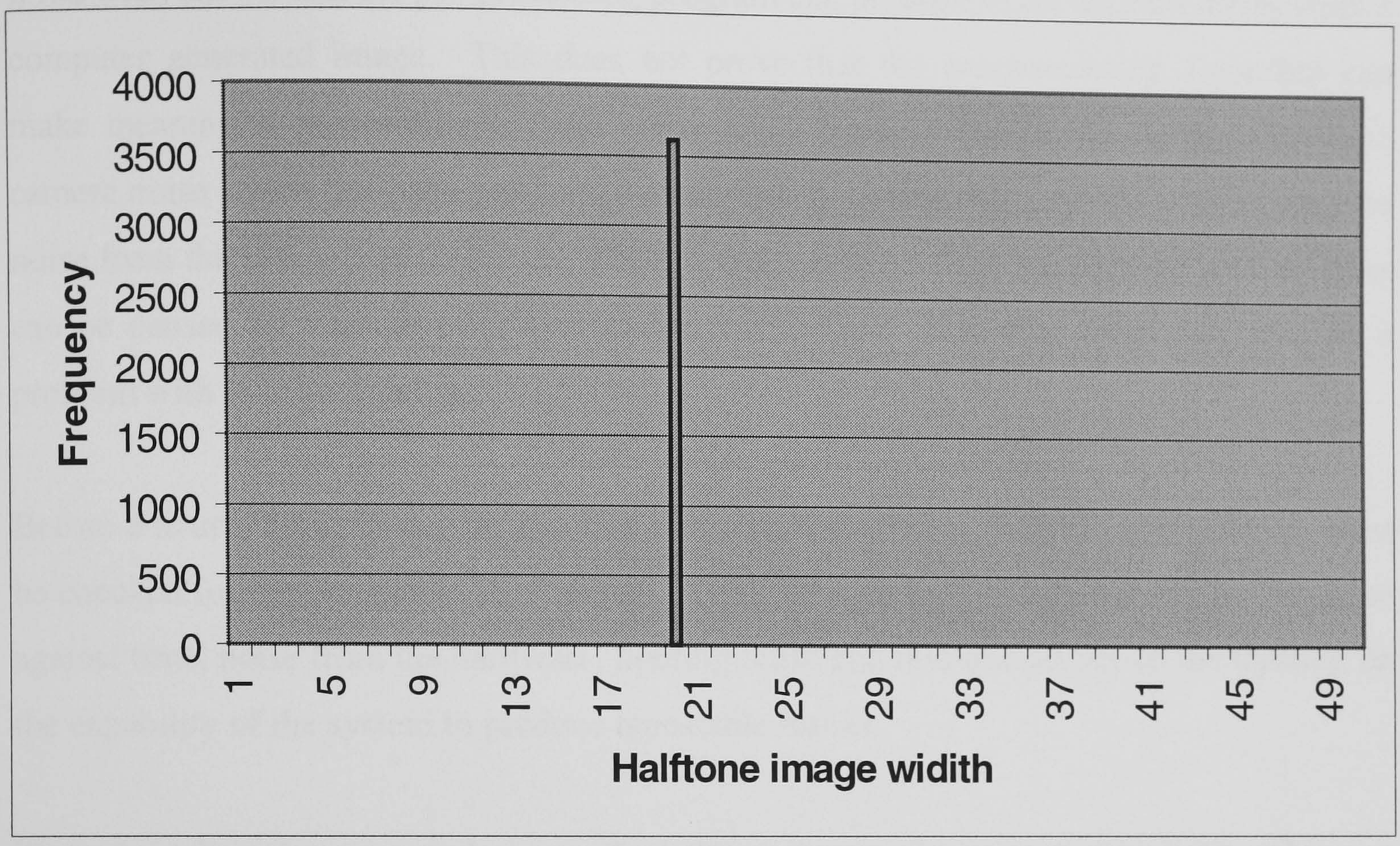


Figure 5.25. The modal length of the image areas for the halftone cycles in the computer generated test image.

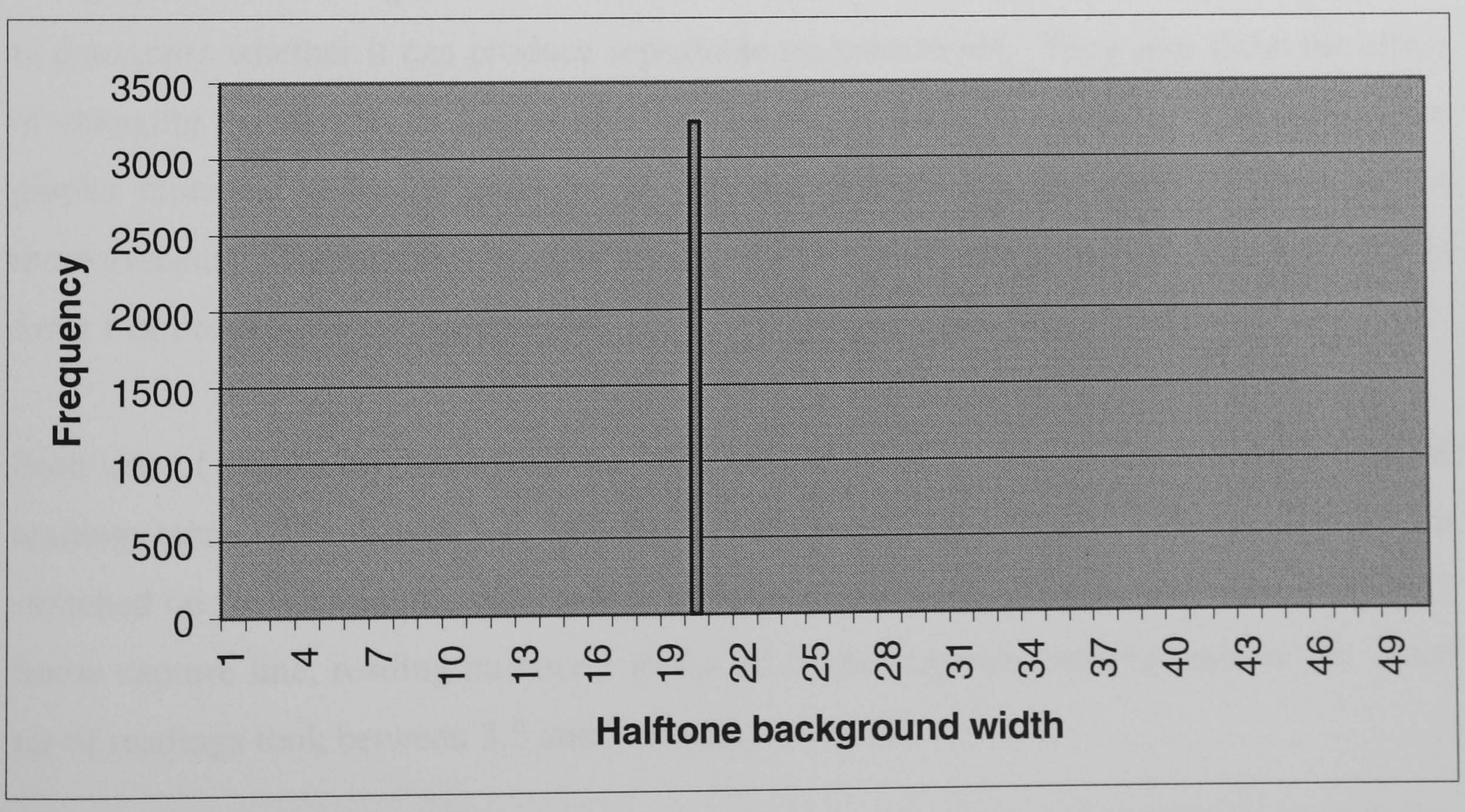


Figure 5.26. The modal length of the background areas for the halftone cycles in the computer generated test image.

5.10 Testing the image using a live camera image

It has been shown that the pre-processing program can produce exact measurements from a computer generated image. This does not prove that the pre-processing algorithm can make meaningful measurements from a live CCD image. This could be due firstly, to camera noise which has been previously discussed in **section 4.4**. Secondly, there may be noise from the frame capture board. Thirdly, there is noise from the lighting source. This can be caused by thermal effects or mains fluctuations. Fourthly, there may also be a problem with uneven lighting.

Before a neural network can be attached to the pre-processing program, the program must be checked for consistency in its readings. These tests include testing the camera response against time, noise from the hardware, lighting noise and non-uniformity of the lighting on the capability of the system to produce repeatable results.

These tests have been carried out on the halftone image shown in **figure 5.27**. This is a magnified image of a halftone square that has been produced on a Hewlett Packard 4M laser printer set in 600dpi mode. The actual dimension of this square is 1 cm. The cursor shown in the diagram was used to align the image to the horizontal.

The graphs shown in **figures 5.28** and **5.29** are the results of tests on the system hardware to determine whether it can produce repeatable measurements. They also show the effect of changing the number of frames used to capture the data. The three lines on each of the graphs represent different numbers of captured frames that are used for each set of measurements. The number of captured frames can be altered using the MIL software in form 1 of the pre-processing program

Both sets of graphs originate from the same sets of measurements. There were 30 sets of readings taken. The camera was switched off overnight, for a total of 15 hours. It was then switched on for 15 minutes, after which measurements were taken sequentially from the 1 frame capture line, reading number 1 to the 20 frames capture, reading number 10. Each set of readings took between 3.5 and 5 minutes to record.

In **figure 5.28**, the tonal intensity corresponding to output 23, the intensity reading, is plotted against the reading number for 1, 5, 20 frames. As the number of frames increases

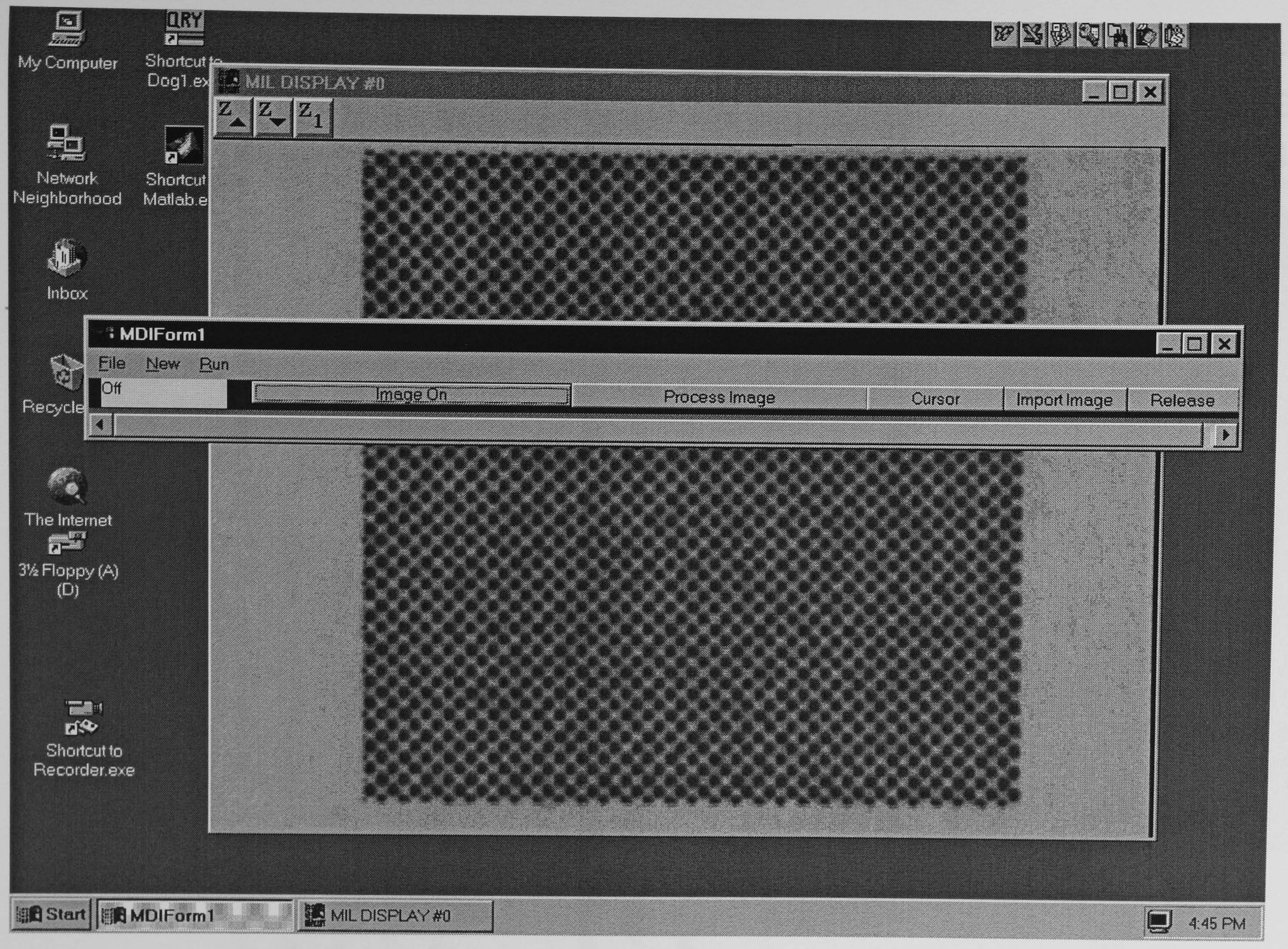


Figure 5.27. A halftone image used to test the image analysis system. Also shown is the cursor used for alignment of the image with the horizontal.

from 1 to 20 it can be seen for this sample that the size of the fluctuations are reduced. This is due to the random fluctuations in the noise from the different frames averaging out.

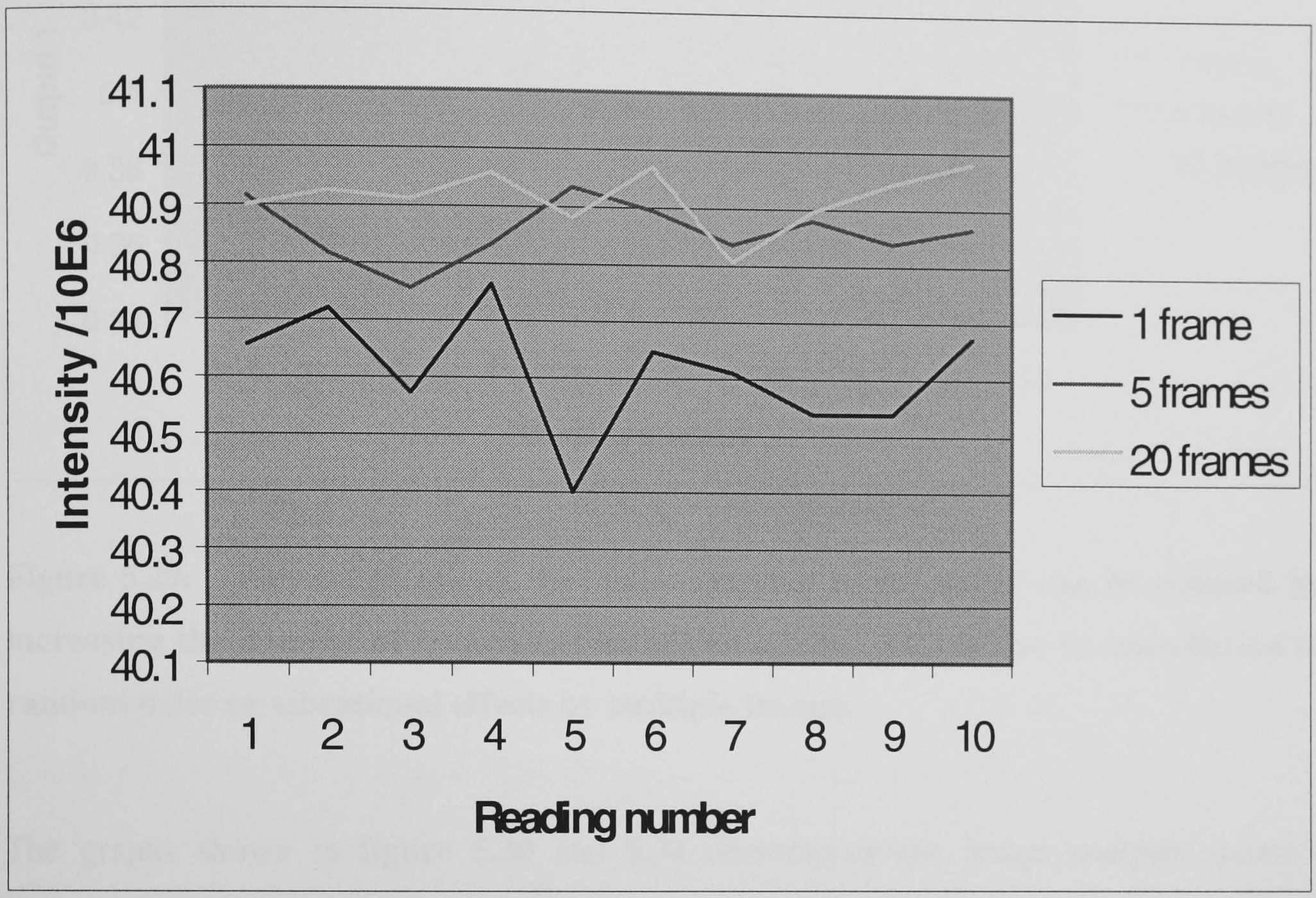


Figure 5.28 This graph demonstrates the effect of increasing the number of frames used in a data sample on the accuracy of the intensity readings. In this experiment changing the number of frames used from 1 to 20 has a noticeable affect on the reduction of noise.

In **figure 5.29** output 1 which measures noise is plotted for the 30 sets of readings. It can be seen that the value of output 1 increases with the number of frames used in a sample. A higher value for output 1 means that the halftone pattern is interpreted as more ordered. The increase in the value of output 1 for a higher number of frames used to capture the data means that the noise from the electronic hardware or vibration effects from the camera stand had been reduced.

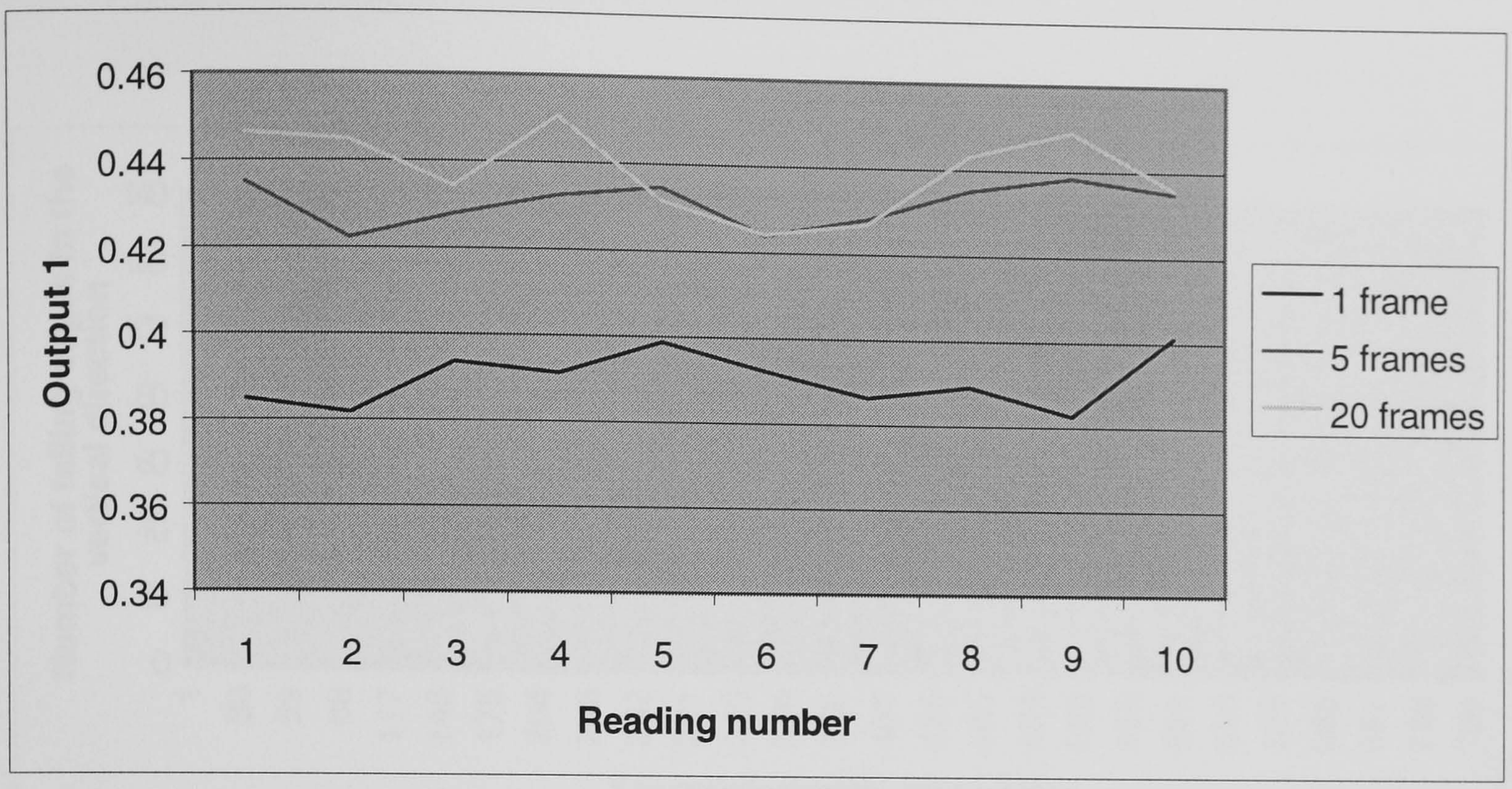


Figure 5.29. This graph shows that noise external to the image can be reduced by increasing the number of frames to capture an image. This is due to cancellation of random noise or vibrational effects by multiple frames.

The graphs shown in **figure 5.30** and **5.31** demonstrate the image analysis system's immunity from external noise. These results were achieved by setting the threshold to a high enough level that could filter out most of the noise. However this level had to be low enough that the image under investigation was not also filtered out.

Figure 5.31 also shows that uneven lighting was present. This can be seen in the slope in the tonal intensity of the background. It will be shown in the next section that meaningful print quality measurements can still be made even if this effect is present. It was for this reason that a background lighting correction was not made. **Figure 5.32** and **5.33** demonstrates the program measuring the halftone frequency. The figures show the image and background area size distributions for the halftone pattern respectively.

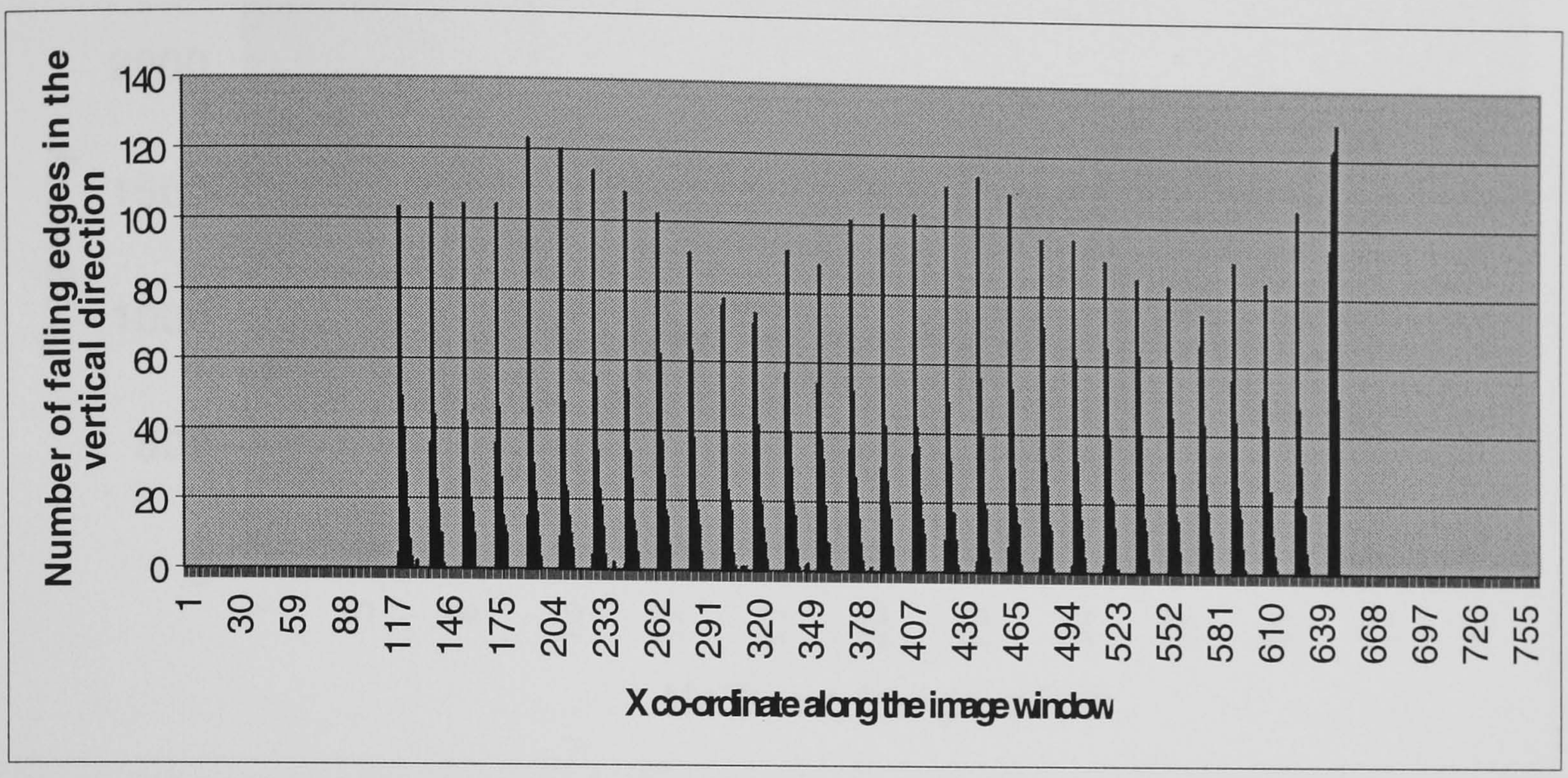


Figure 5.30. This is a graph of the falling edges for the halftone image shown in figure 5.21. The plot shows that the background can be differentiated from the image.

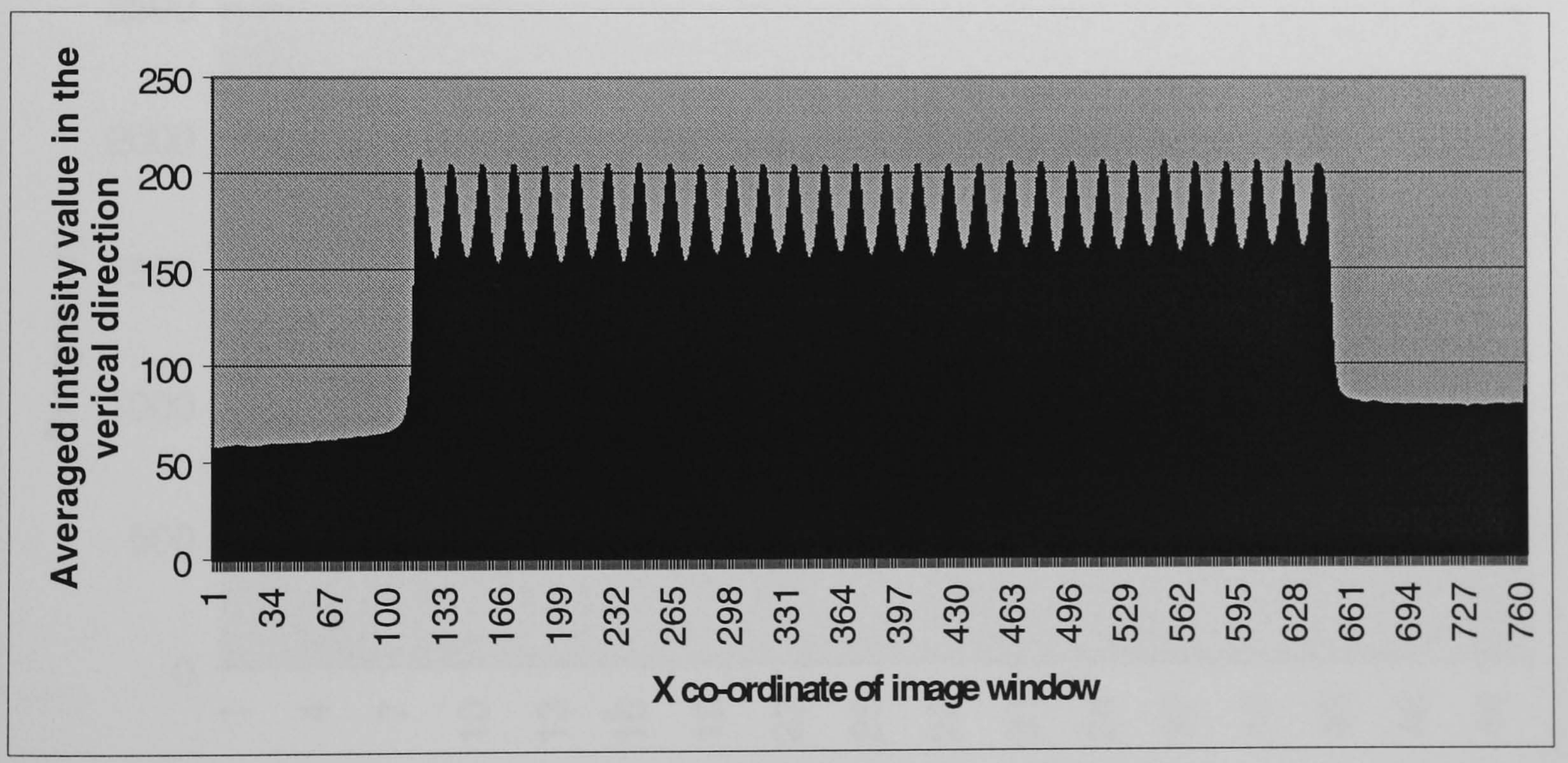


Figure 5.31. This graph shows that the image can be differentiated from the background by measuring the average image intensity as the window is scanned.

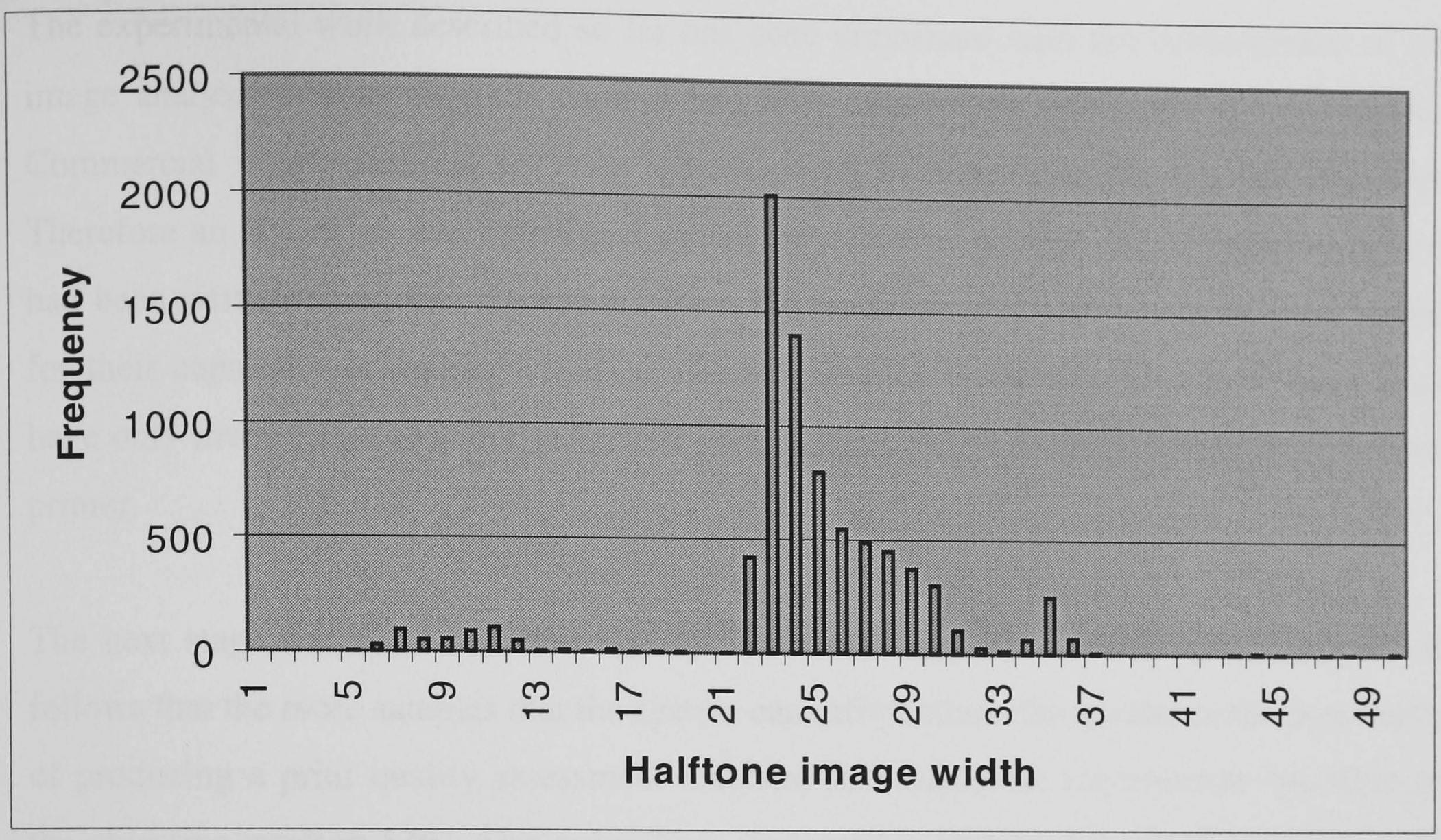


Figure 5.32. This graph shows the halftone image width distribution for the image shown in figure 5.27.

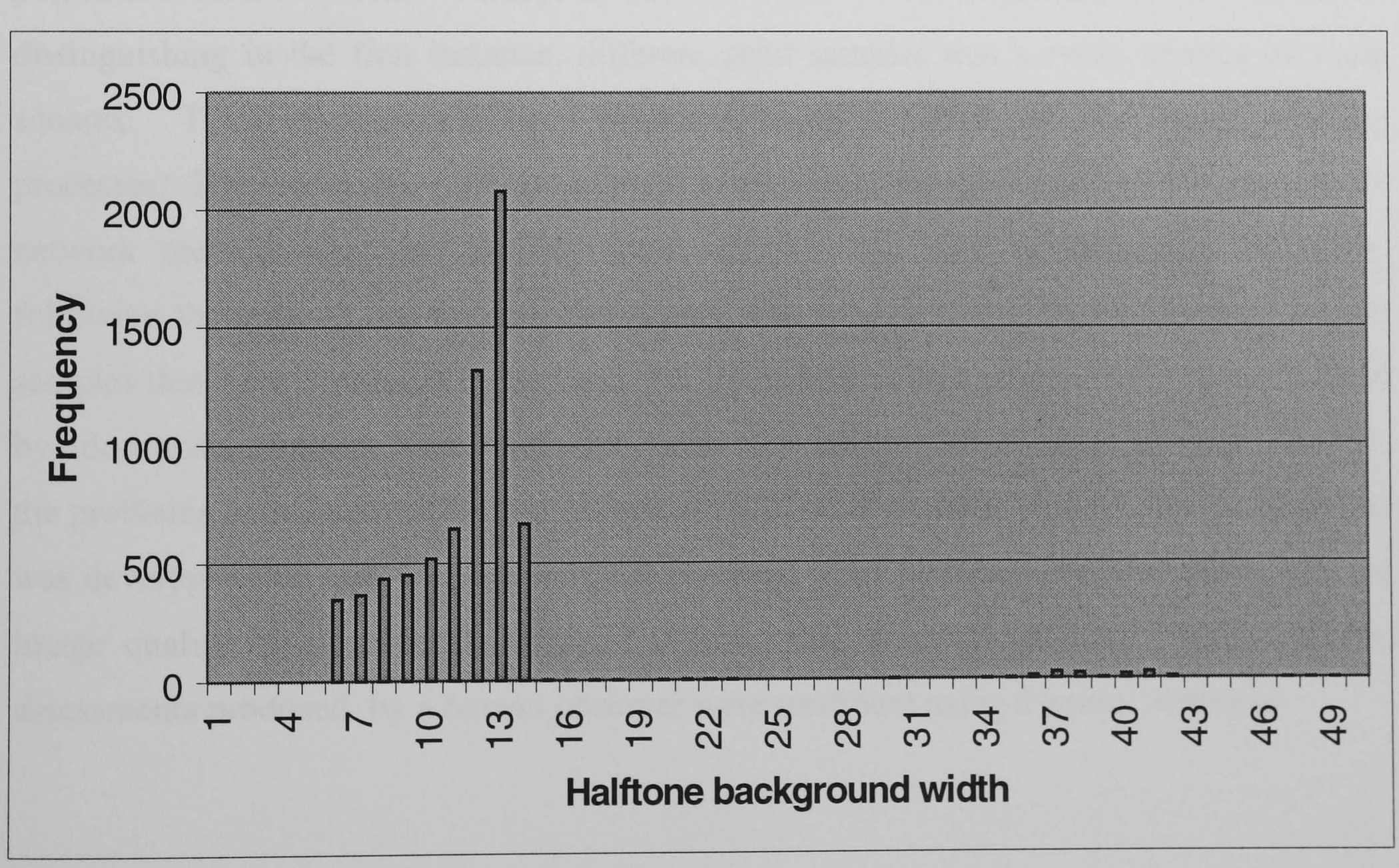


Figure 5.33. This graph shows the halftone background width distribution for the image shown in figure 5.27.

The experimental work described so far has been concerned with the development of an image analysis system. A CCD camera had been sucessfully interfaced to a computer. Commercial image analysis software was assessed as unsuitable for the investigation. Therefore an algorithm was developed specifically for this project. A computer program had been written using this algorithm. Both the hardware and software have been tested for their capability in making accurate and precise measurements. However these tests have only involved a computer generated pattern and a single halftone image from a laser printer.

The next stage was to see whether the system can distinguish different print samples. It follows that the more samples that the system can differentiate, the greater is the possibility of producing a print quality assessment machine that fulfils the requirement specified in the objective (**section 1.0**). Also as has been discussed in **section 4.5** neural networks need many different data samples if they are to develop accurate print quality models.

The next section describes tests using many different print samples that have been performed on the system. Firstly, these tests explored the capability of the system to distinguishing in the first instance, different print samples that contain images of 1 cm squares. These squares had been produced using a variety of non-impact printing processes. They were also different in tonal intensities. The application of different neural network methods that can classify these samples will also be described. Secondly, following the tests on the squares, the system was used to assess the differences in print samples that were images of circles and text characters. These images were also produced by non-impact printing. Neural networks were also used to classify these images. Thirdly, the problems associated with image alignment will be discussed. A neural network model was developed that took into account this problem. The final test involved modelling the image quality of a graphical image of a face using a neural network. Image quality assessments produced by a human observer were modelled using a neural network.

6.0 The measurement of print quality using the pre-processing program and neural networks

This section summarises the experimental work. The summary is divided into four parts which are listed below.

1 The selection of print samples and processes that were to be investigated.

2 The use of the pre-processing program and graphics to differentiate between print samples.

3 The application of neural networks to model the results of part 2.

4 The application of neural networks to discover models that are more complex than those in part 2. These more complex models can be of four or more dimensions.

Part 1 was used to restrict the number of print quality samples that had been investigated. This procedure simplifies the analysis of the results produced by the image analysis system. Studying a small number of processes still enables the methodology applied in this project to be investigated. If the method proved to be practicable, then it could be extended to analyse other printing samples and other variables.

Part 2 was a continuation of the testing procedure described in **sections 5.0-5.10** for the pre-processing algorithm. In **sections 5.0-5.10** the tests concerned whether the algorithm had been written correctly and if it could be used to make meaningful measurements using the image analysis hardware. This stage showed that the image analysis system could recognise different print samples. This was achieved by using the pre-processing program to classify print samples that originated from different printing processes. The results were viewed using the Matlab graphical toolbox.

Part 3 determined whether the Matlab neural network application could model the results that were produced in part 2 above. As will be demonstrated later, parts 1, 2 and 3 are interlinked and are not performed sequentially.

Part 4 concluded the experimental work in this investigation. It applied neural networks to find patterns in the pre-processing data that were not previously seen by visual inspection.

The purpose of the experimental work in this stage was threefold. Firstly, to determine whether different printing processes could be identified solely using neural networks. Secondly, to simulate human print quality perception using the image analyser. This was achieved by training neural networks with observer assessments. These trained networks were subsequently used to predict the outcomes of further observer assessments. Thirdly, the problem of automatic image alignment to the camera was investigated.

6.1 The selection of print samples used for the investigation.

The samples used were approximately 1 cm^2 in size. The choice of this size of image has already been discussed in **section 5.4**. Examples of these shown in **figure 6.1**, have already been used in **sections 5.7** and **5.10** to test the noise characteristics of the image analysis system. Non–impact printing processes have been chosen for the investigation. They were decided upon because small quantities of print samples can be obtained quickly from them.

The first tests involved 1 cm squares of different shades of grey. These squares were produced using the drawing tool in Word 6 or 7. The different shades were created using the colour tool, which could also produce a series of greyscale tones. This process has been previously used in **sections 5.7** and **5.10**. These squares were then printed using laser and inkjet printers, photocopiers and a thermal printer. Each printer produced a series of squares that differed in tone. An example of one of these series is given in **figure 6.1**. Squares were chosen for the initial tests because they were simpler to analyse than other shapes, thus any potential mistakes in the program could be rectified more easily.

One of the aims of the project was to find out whether the program could determine the print quality of other bi-level images. Therefore after the investigation with the square, more complex shapes such as circles, text characters and a simple tri-level monchrome pictorial image were used. The following parts of this section discusses these results sequentially from the square to the simple pictorial image. In doing this it also demonstrates how the image analysis system can be used to assess print quality.

6.2 The initial investigation of the square using the image analysis system.

To show that the image analysis system is capable of measuring print quality, it is necessary to demonstrate that it can differentiate between different print samples produced

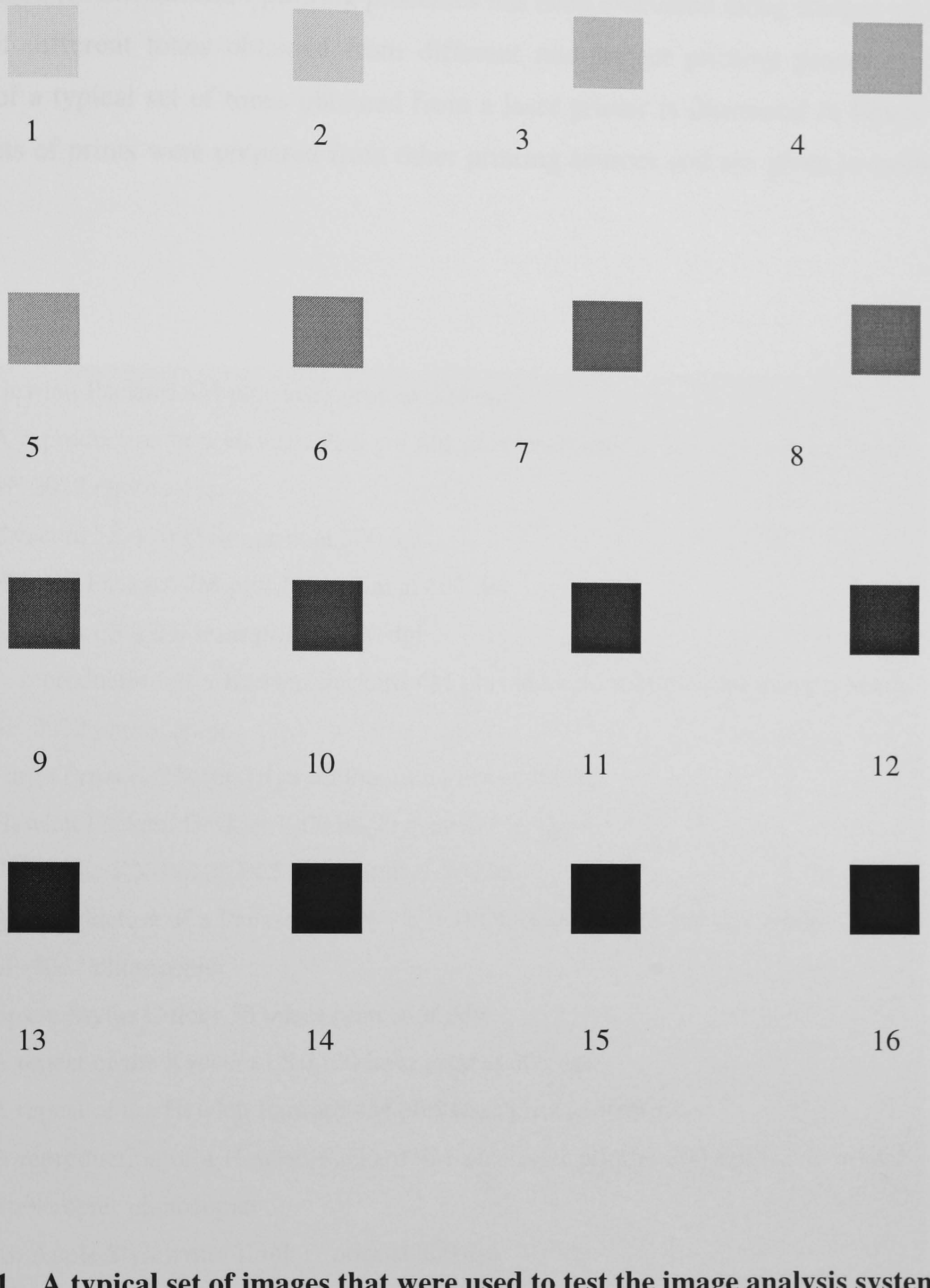


Figure 6.1. A typical set of images that were used to test the image analysis system.

by different processes. The more samples the system can differentiate between, the more realistic the simulation of the observer assessments will be. The capability of the system to differentiate between different printing processes has been evaluated using images of 1 cm squares of different tones obtained from different non-impact printing processes. An example of a typical set of tones obtained from a laser printer is illustrated in **figure 6.1**. Similar sets of prints were prepared from other printing sources and are given in **table 6.1** below:

1	Hewlett Packard 4M plus laser print at 300 dpi
2	A reproduction of a Hewlett Packard 4M plus laser print at 300 dpi using a Sharp SF-2022 photocopier
3	Kyocera FS-1700 laser print at 300 dpi
4	Hewlett Packard 4M plus laser print at 600 dpi
5	Kyocera FS-1700 laser print at 600 dpi
6	A reproduction of a Hewlett Packard 4M plus laser print at 600 dpi using a Sharp SF-2022 photocopier
7	Fargo Primera Thermal dye sublimation print at 300 dpi
8	Hewlett Packard Deskjet 1200 inkjet print at 300 dpi
9	Panasonic KX-P6100 PCL laser print at 300dpi
10	A reproduction of a Panasonic KX-P6100 PCL laser print at 300 dpi using a Sharp SF-2022 photocopier
11	Epson Stylus Colour 50 inkjet print at 300dpi
12	A repeat of the Kyocera FS-1700 laser print at 300 dpi
13	A repeat of the Hewlett Packard 4M plus laser print at 300dpi
14	A reproduction of a Hewlett Packard 4M plus laser print at 300 dpi using an Oce Bookcopier photocopier
15	An Apple Stylewriter II inkjet print at 300 dpi
16	A reproduction of a Hewlett Packard 4M plus laser print at 300 dpi using a CanonNP 6030 photocopier.

Table 6.1. The sets of prints of 1 cm squares used to test the image analysis system.

6.2.1 Test for halftone frequency measurement

Each set of images consisted of a series of 15 different tones and a group of 8 readings for the same solid. Using **figure 6.1** as an example, a set of readings was taken for each of the tones and eight sets of readings taken for the solid labelled 16. After the 23 sets of readings were taken for the entire group of squares, two sets of readings were taken from a calibration square. The set of squares from the next printing process was then measured. The calibration test was made to check for electronic drift in the camera. It also determined whether the zoom lens moved out of focus due to vibration caused by heavy machinery operating in the building, and traffic. The data acquisition sequence is illustrated below in **figure 6.2**.

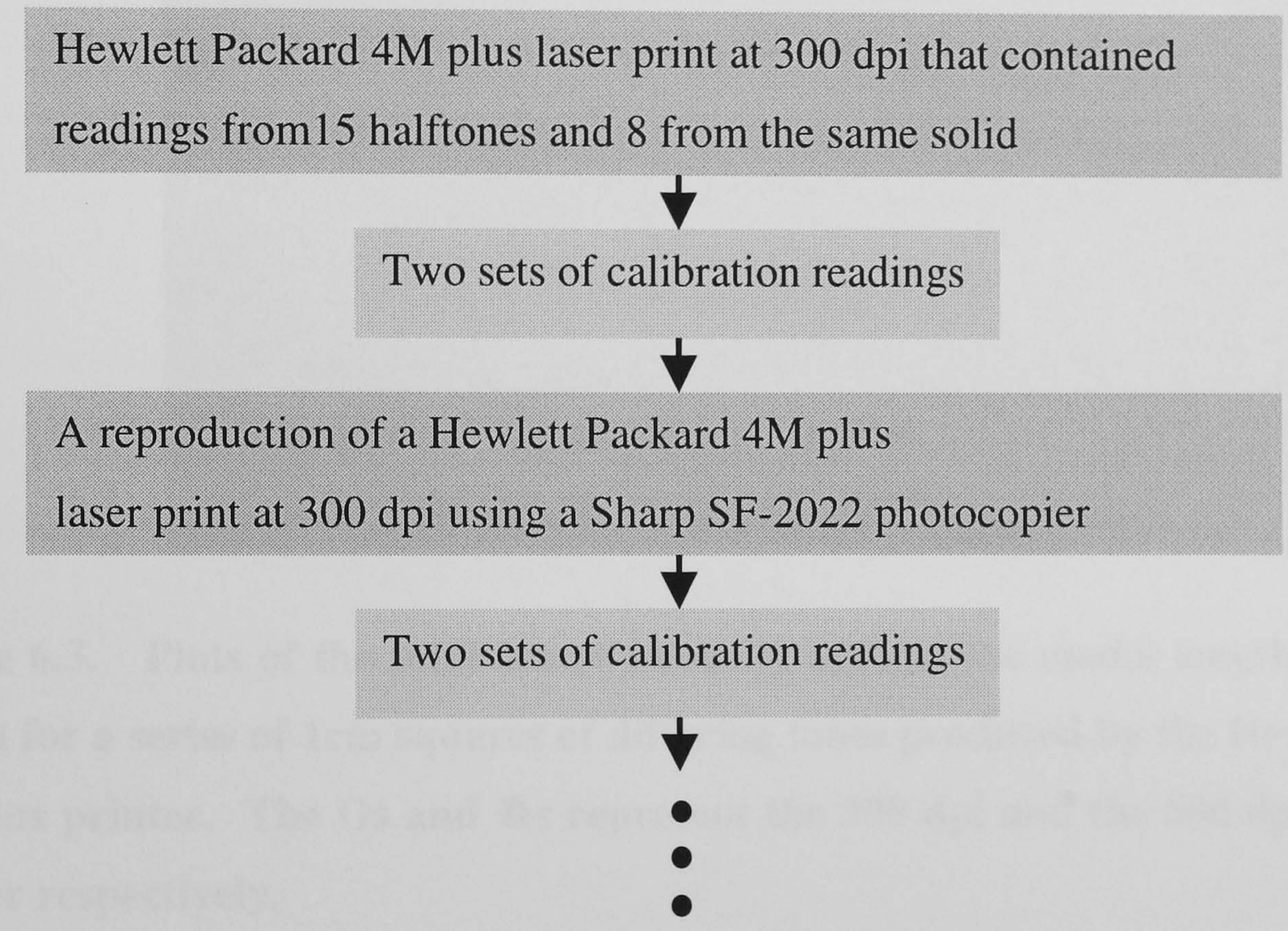


Figure 6.2. A block diagram that illustrates the sequence that the data was recorded in.

The first test involved measuring the halftone frequency from two sets of tones produced from a laser printer, one of which printed at 600dpi and the other at 300 dpi. These samples were taken from **table 6.1**. The example below demonstrates this for the Hewlett Packard laser printer in 300 and 600 dpi mode.

The graph of total image intensity versus the modal length of the image region in the halftone frequency cycle for the Hewlett Packard example is given in **figure 6.3**. The relaionship shown in **figure 6.3** is linear. This is to be expected, since, in theory, the length

of the image region is proportional to the intensity. This relationship gave evidence that the system was producing meaningful results. The total halftone frequency values given by output 21 of the pre-processing program for the images produced at 600 dpi was 18 pixels. For the images produced at 300dpi it was in the range 36-37 pixels. This demonstrated that the 600 dpi halftone frequency was half the size of that in the 300 dpi print.

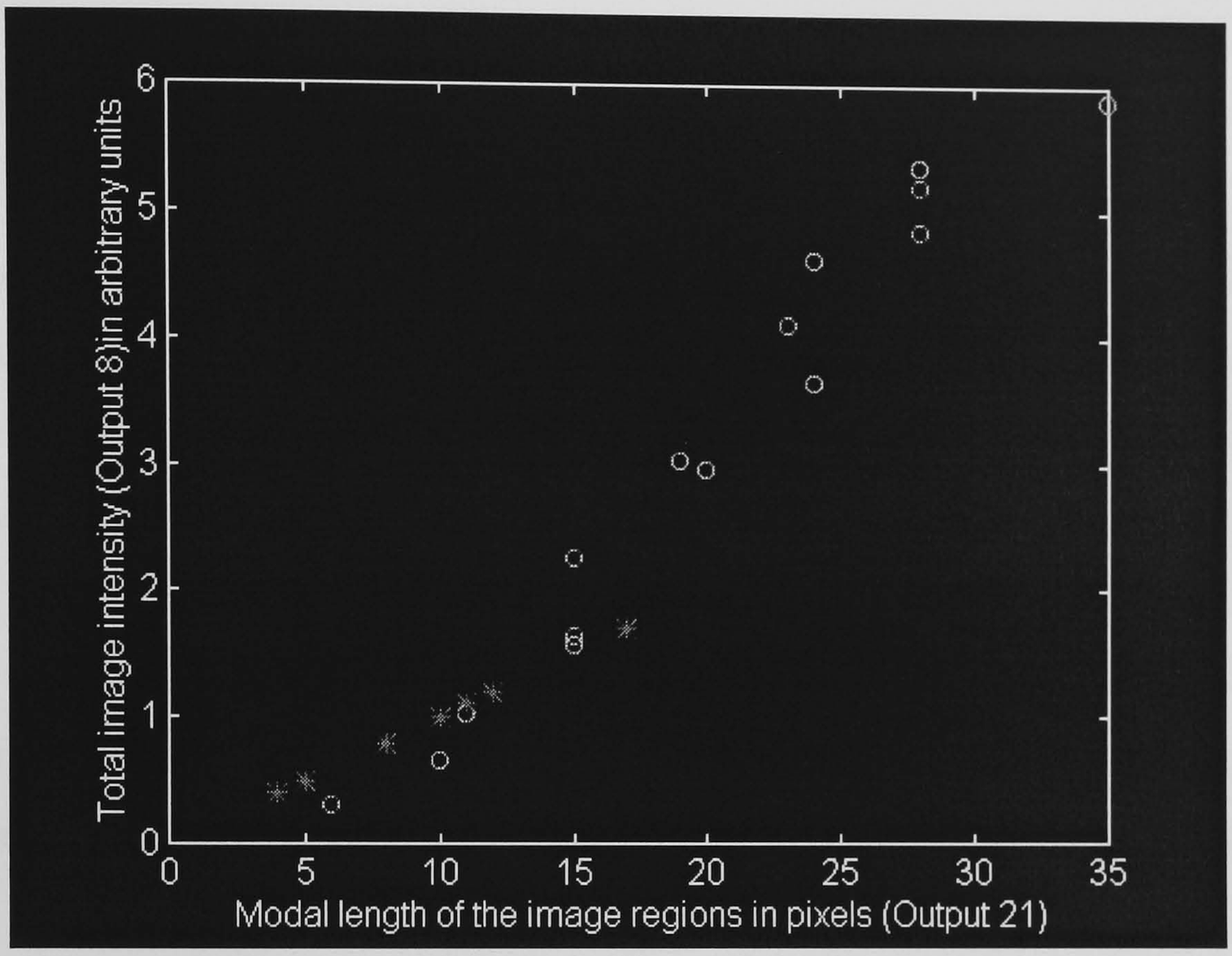


Figure 6.3. Plots of the total image intensity against the modal length of the image region for a series of 1cm squares of differing tones produced by the Hewlett Packard 4M plus printer. The Os and ✳s represent the 300 dpi and the 600 dpi mode of the printer respectively.

These test were carried out using the lighting conditions specified in **section 5.1**. This meant that any fluctuations in these lighting conditions did not mask the overall trend shown in the relationship between the total image intensity versus the modal length of the image region. Also the lighting conditions did not affect the precision of the measurement of the halftone frequency pattern significantly, since the halftone frequency measurements recorded for a range of 14 tones in both the 600dpi and 300dpi modes were exactly 18 pixels and 36-37 pixels respectively. The preceding results also show that any vibration of the camera caused by external sources such as traffic did not interfere with the consistency of the measurements.

6.2.2 The classification of laser prints from their photocopies

Linearly separable relationships were discovered by visual inspection of the experimental data from the output of the pre-processing program. For example, a linearly separable relationship was found for halftones by plotting output 1, a noise measurement, against output 17 the total image intensity. This graph is shown below in **figure 6.4**. The O set was from the laser printer and the ✳ set from the photocopier. A higher value for output 1 means less noise, as explained in **section 5.5**. The significance of this result is that two printing processes have been differentiated using the image analysis system.

Figure 6.5 shows another linearly separable classification made from the same set of pre-processing data that was used to produce **figure 6.4**. In this graph, output 4, a measurement of noise is plotted against output 39, the number of gradients equal to 1 detected on the edges of the image regions. The graph shows another linearly separable classification. It is possible that the combination of the two linearly separable classifications will produce a more accurate result than either of the individual results. The role of the neural networks was to model multi-dimensional functions from combinations of the 52 outputs of the pre-processing program. These functions were measurements of print quality.

The neural networks produced results to show whether an image had been correctly or incorrectly classified without continual reference to the validation set's sum squared error value during network training training. The procedure adopted was to train the neural networks until they reached an error value that was ≤ 0.01 was obtained. If it was seen that a successful classification had been made by visual inspection, the search for other neural network solutions was stopped. This type of assessment was used to achieve all the following neural network results. This procedure was justified by the fact that the emphasis of the investigation was to see whether in principle the image analysis system can simulate print quality perception. Once this fact was established the accuracy could be improved by improving the image analysis hardware and different neural network procedures.

However before neural networks could be applied to evaluate the pre-processing data, a suitable neural network paradigm had to be selected. This is dealt with in the next section.

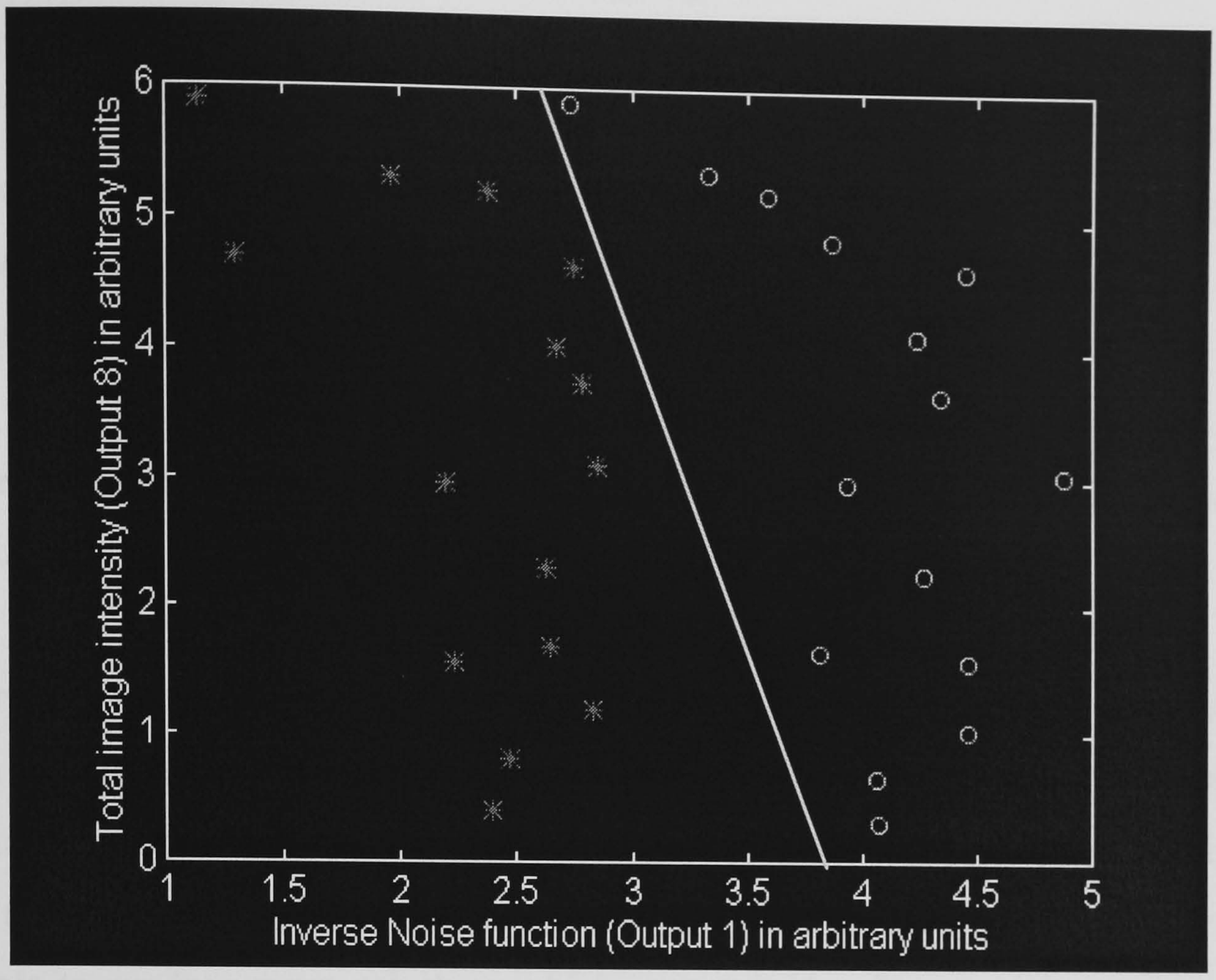


Figure 6.4. The classification of two different sets of halftones. The images represented by the Os originated from a Hewlett Packard laser printer. The images represented by the ✳s were reproductions of the Hewlett Packard prints using the Sharp SF-2022 photocopier.

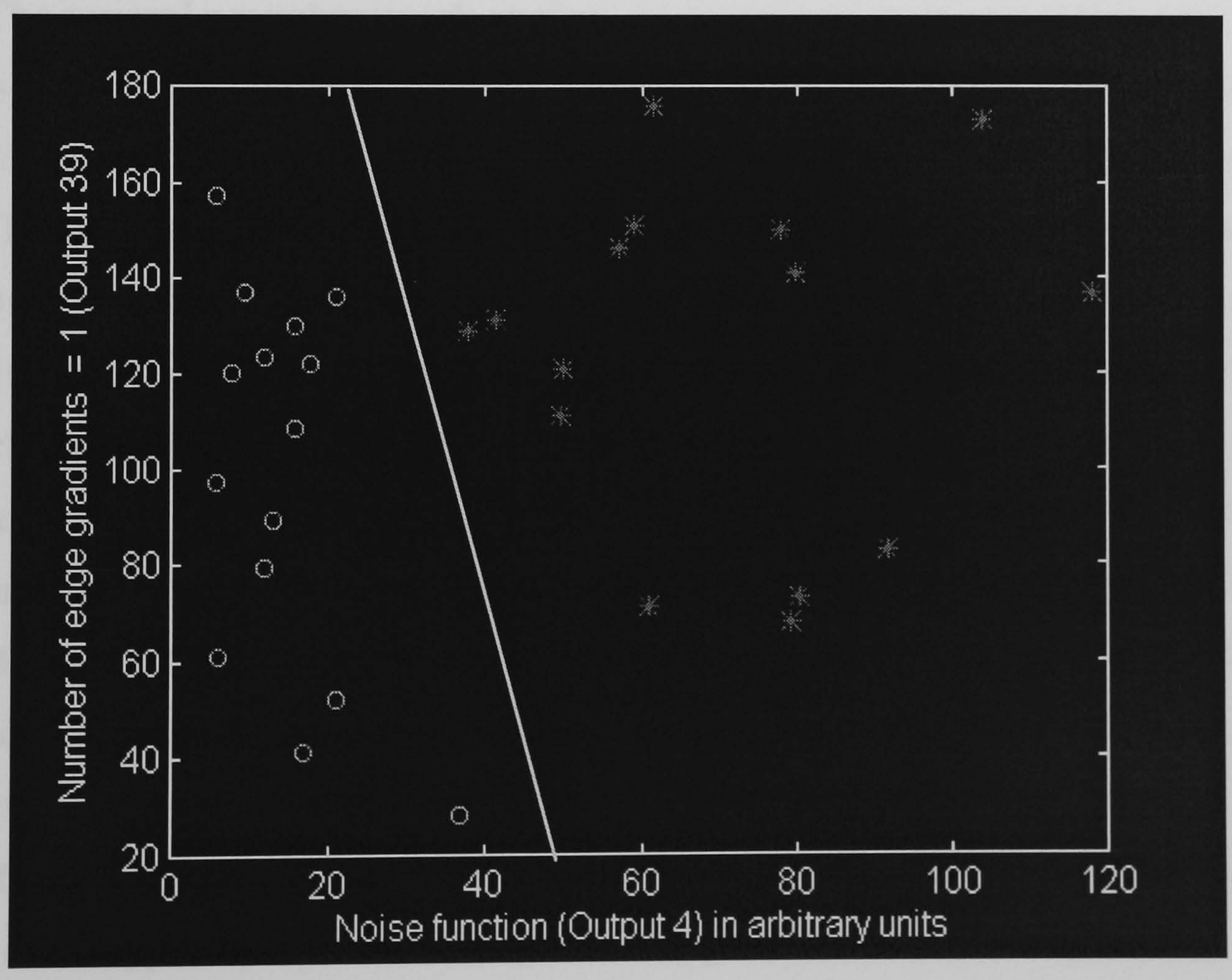


Figure 6.5. A different linearly separable classification using the same set of results that produced figure 6.4 above. This graph uses output 4 and output 39.

6.3 The selection of a neural network paradigm

In the introductory explanation given in **sections 4.5-4.9**, it was explained that the purpose of neural networks in this investigation was to recognise patterns in pre-processed image data that could be used to differentiate the print quality of different print samples. It was also explained in detail that this could be achieved using different types of neural network algorithms that can be converted to computer programs. These programs perform iterations which can converge to a solution. These solutions can be represented by minima in an energy landscape. The three methods that were discussed were backpropagation, the radial basis function and neurofuzzy logic.

The criterion that was used to judge the suitability of a neural network for this project was the speed of convergence to a minimum. The reason for this is that a neural network has many parameters that can be adjusted to give different solutions. Each solution has an accuracy that may or may not be satisfactory. Therefore in a given time, a faster network will be able to produce more solutions which improves the chance of discovering a good solution to the problem in a given timescale . The next section describes the Matlab neural network application used in this investigation. This is followed by initial evaluation of the suitability of the different neural network methods in Matlab for this investigation.

6.3.1 The general neural network engine

This was built using Matlab version 4.2c.1 equipped with the Neural Network and Fuzzy Logic Toolboxes. Matlab can operate in Windows and is a fully integrated modular system. This means that an interface can be built between the pre-processing program in Visual Basic and Matlab. In this interface, data that contains the features extracted from the raw image data is transferred to a Comma Separated (CSV) file, the contents of which can be checked in Excel. Matlab can also read this type of file, and through its own programming language, can manipulate the data to the required form for the plotting of graphs and neural network modelling. All the graphical examples in this section were produced using the Matlab functions. Approximately 5 lines of code are needed for the application of a neural network in Matlab. These lines of code are necessary for the adjustment of variables such as the learning rate, the level of accuracy required and the transfer functions used. The code is also needed to specify the network topology. An example of the type of program used is given in **appendix 2**.

Figure 6.6 below is a block diagram of the complete software system. The Matlab neural network can be connected to any combination of the 52 pre-processing outputs. All that is required is to specify the outputs using the Matlab program. No extra code is needed. This was found to be an efficient way to experiment with the outputs and different neural networks.

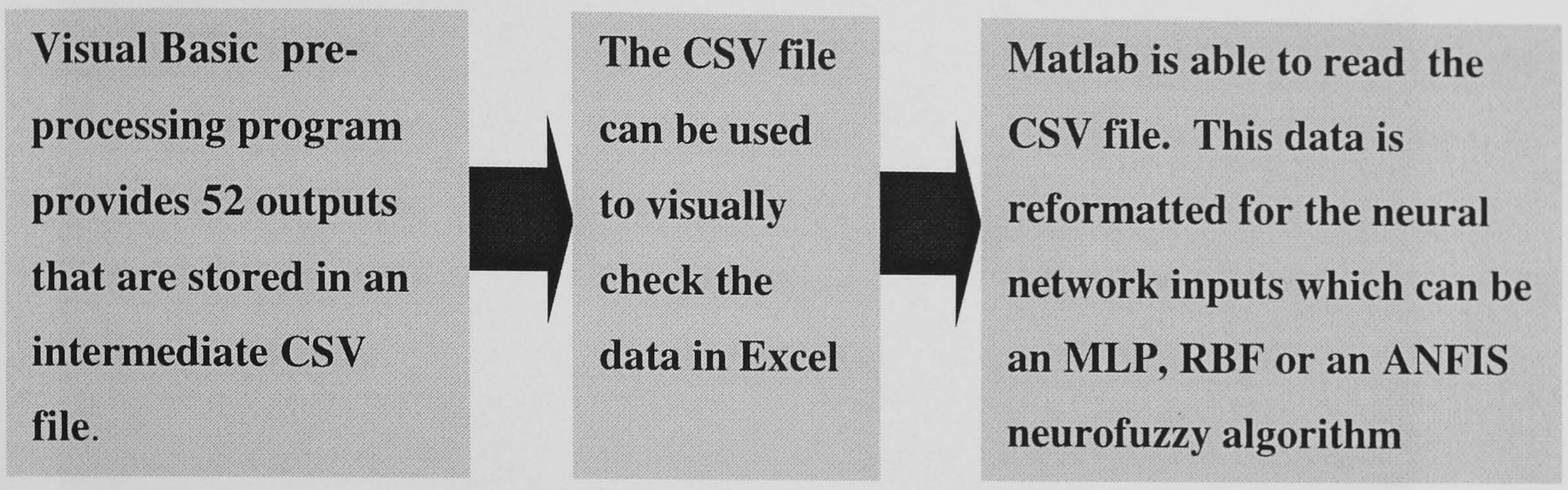


Figure 6.6. A flowchart describing how Matlab reads the data from the pre-processor via a CSV data file.

6.3.2 Neural network selection

As mentioned previously, three different neural network paradigms were initially tested for their suitability. The first task was to see whether the programs were functioning correctly by giving them a problem with a known linearly separable problem to solve. The second task was to reduce this number by assessing their convergence speeds. All three networks were given the identical task of producing a solution in the form of a plane that separates the two sets of points for the graph shown in **figure 6.7**. The data came from the set that was previously described and used in **sections 6.2-6.2.2**. Outputs 1 to 3 of the pre-processing program, which measured noise in the images, were employed in this test. It should be noted at this point that the units of the measurements are of no importance provided that the pre-processing program remains unadjusted, which it did throughout the experimental work. The next three sections describe the test results from the three neural network methods.

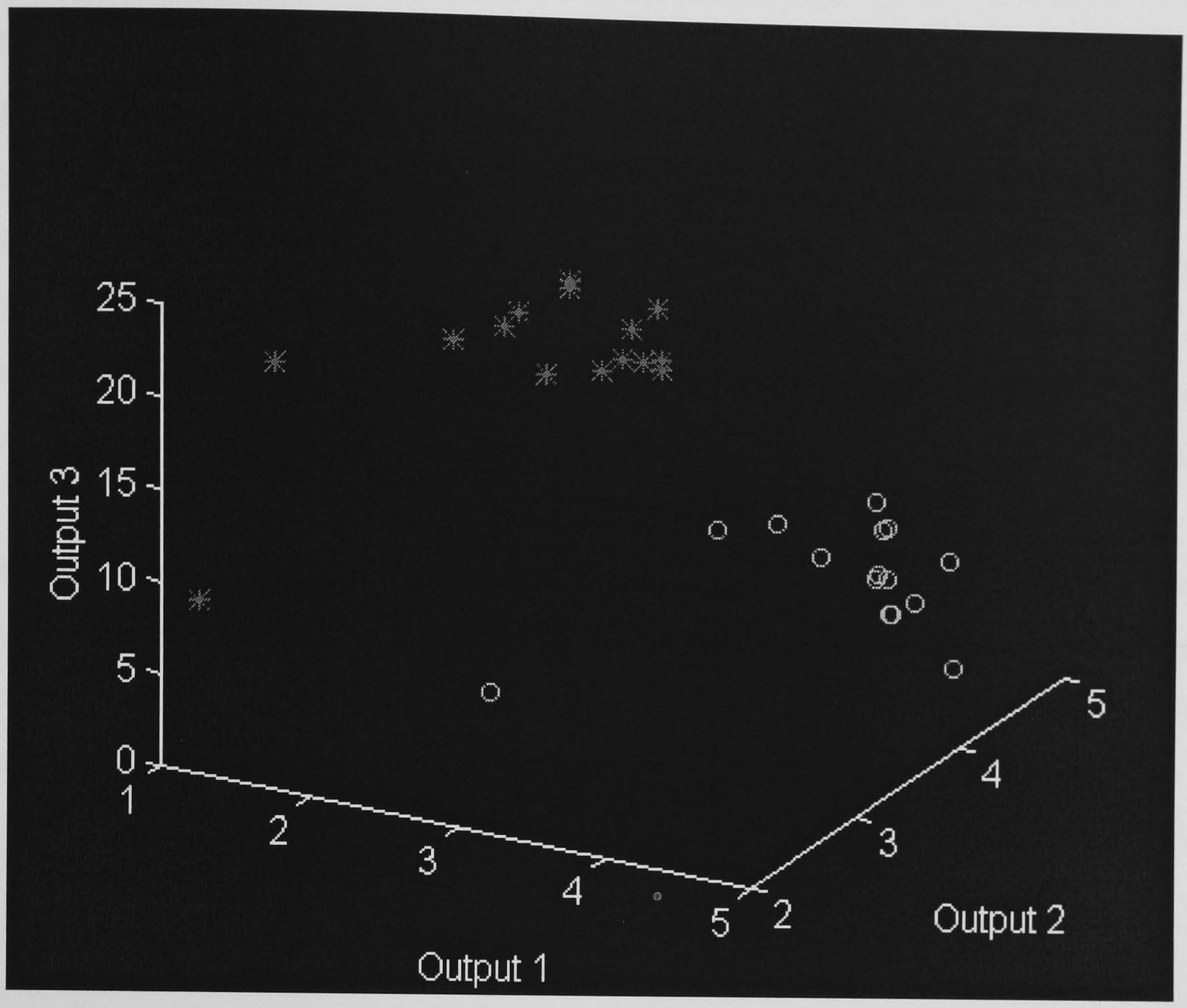


Figure 6.7. The data shown graphically used to measure the speed of convergence for different neural networks.

6.3.3 Backpropagation

Matlab 4.2c can execute this method in three ways. The first way is standard gradient descent. This has been extensively discussed in **sections 4.5.2** and **4.5.3** and also forms the basis for understanding the next two methods. The second method is known as backpropagation with momentum. **Equations 6.1 and 6.2** below, which describe this method, are the weight adjustment equations that are applied to the output to hidden and the hidden to input layers respectively.

$$\Delta W_{jk} = W_{jk}(t+1) - W_{jk}(t) = -\eta \delta_k y_j + m(W_{jk}(t) - W_{jk}(t-1)) \quad \textbf{Equation 6.1}$$

$$\Delta W_{ij} = W_{ij}(t+1) - W_{ij}(t) = -\eta \delta_j y_i + m(W_{ij}(t) - W_{ij}(t-1)) \quad \textbf{Equation 6.2}$$

Where $0 < m < 1$

Equations 6.1 and **6.2** are the same as **equations 4.8** and **4.10** in **section 4.5.2** except that in the equations above a momentum term has been added, $m(W_{jk}(t) - W_{jk}(t-1))$, for the output to hidden layer and $m(W_{ij}(t) - W_{ij}(t-1))$, for the hidden to input layer . This momentum term helps backpropagation converge faster because it adds a fraction of the previous weight change to the current one. This fraction is proportional to the previous weight change. Therefore the effect of the momentum term decreases as it approaches a minimum to prevent an overshoot. Another advantage is that the momentum term can push the weight value over shallow minima in the error landscape. It is a feature of standard gradient descent that the network can be trapped in these local shallow minima.

Matlab also possesses the Levenberg-Marquart algorithm. This algorithm is a method that is used specifically is to minimise a sum squared error function. This is **equation 4.4** in **section 4.5.2** used in the error minimisation procedure for standard backpropagation. This equation can also be expressed as:

$$E = \frac{1}{2}\sum_{n}(e^{n})^{2} \qquad \textbf{Equation 6.3}$$

where $e^{n} = T^{n} - O^{n}$, T is the target vector and O is the actual output vector.

The Levenberg-Marquart[**41**] method adopts standard backpropagation and also the Newton method to find minima. Referring to **figure 4.13** and **section 4.5.3**, backpropagation reduces the error by moving in the direction of steepest descent. This process does not guarantee that after an iteration in the algorithm the new position will have the lowest error value for a given step size. The Newton method is different in the sense that it does not move the error in the direction of steepest descent but locates the lowest error value for a given step size. However the Newton method can only be used when the error position is close to a minimum **[41]**.

The Levenberg-Marquart algorithm looks for an error minimum in the following way. It updates the weights using standard gradient descent for regions that are far away from a minimum in the error landscape. In these regions the step size should be large enough for rapid convergence. In regions near a minimum the step size should be small to prevent overshooting the minimum. Near a minimum where the error gradients are small the Newton method converges faster than standard gradient descent for small step sizes,

therefore the Newton method is used in the shallow regions close to a minimum. The Levenberg - Marquart update rule is:

$$\Delta \mathbf{W} = \frac{\mathbf{Z}^{\mathbf{T}}\mathbf{e}}{(\mathbf{Z}^{\mathbf{T}}\mathbf{Z} + \mu\mathbf{I})} \qquad \textbf{Equation 6.4}$$

where $\mathbf{Z}$ is a matrix and $\mathbf{Z}^{\mathbf{T}}$ is the transposed matrix of $\mathbf{Z}$ that contains the elements:

$$(\mathbf{Z})_{ni} = \frac{\partial e^{n}}{\partial w_{i}} \qquad \textbf{Equation 6.5}$$

and $\mathbf{e}$ is a vector that contains the elements e^n.

μ is a constant that determines whether **equation 6.5** is in the Newton or standard gradient descent mode. If μ is small the equation approximates to the Newton method. If μ is large then the equation approximates to standard gradient descent. The Levenberg-Marquart algorithm can change the value of μ after every weight update. If the error decreases after an update then the new weight is kept, μ is increased, which moves the process towards the backpropagation method. If the error increases then the old weight value is kept and μ is decreased moving the computation towards the Newton method. This process is repeated until the error is reduced. The Levenberg-Marquart algorithm converges faster than standard gradient descent but requires more computer memory.

6.3.4 The backpropagation network architecture and the Matlab commands used for the convergence test

This was the first stage in the development of neural network solutions in the investigation. This stage tests firstly that the programs under development had no unforeseen bugs and therefore could solve classification problems with known solutions. Secondly, the convergence speed of networks was monitored for the initial selection procedure for the networks that were to be used. Although it is known from the literature the order of the speeds of the networks, the speeds have not been quantified. This is an important fact that needs to be considered if the project is to be practicably achievable.

A plane can separate the two classes of data points in **figure 6.7**. This is a linearly separable relationship that can be solved using a single perceptron. Therefore it can also be

solved using the multilayered perceptron shown below in **figure 6.8.** All the three error minimisation algorithms above were carried out on this network.

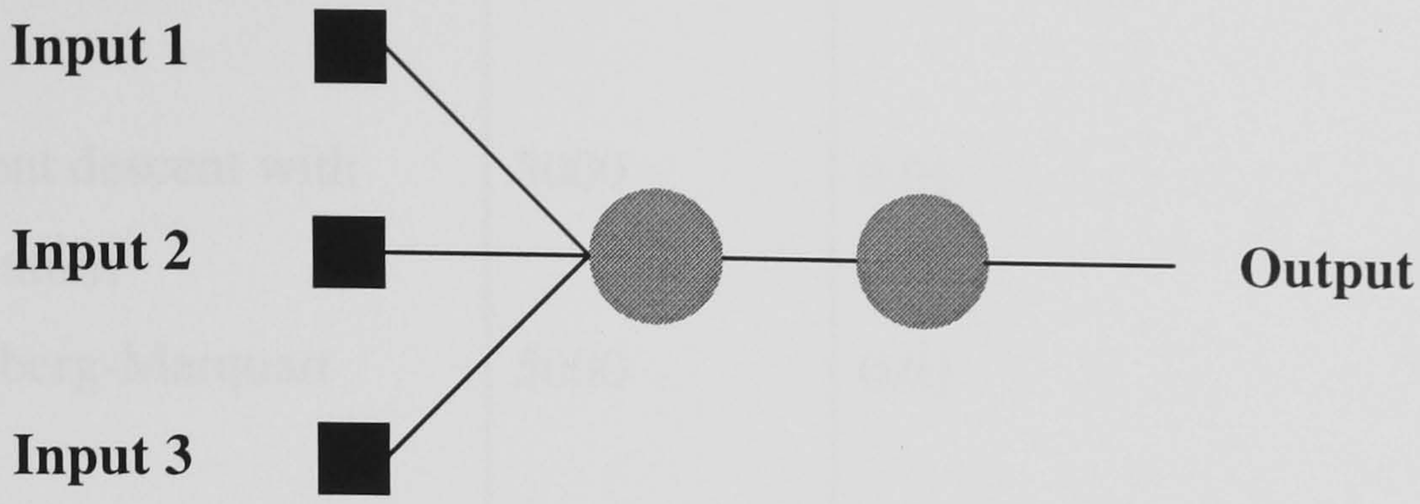


Figure 6.8. The network employed to separate linearly the two classes of print samples shown in figure 6.7.

The Matlab commands and the parameters that were adjusted were:

for standard gradient descent:

Command: trainbp

Adjustable parameters: epoch limit, sum squared error goal, learning rate

for gradient descent with momentum:

Command: trainbpx

Adjustable parameters: epoch limit, sum squared error goal, learning rate

for the Levenberg-Marquart algorithm:

Command: trainlm

Adjustable parameters: epoch limit, sum squared error goal, learning rate

The parameters were experimentally adjusted during the training process. The epoch number is the number of iterations the program performs before it stops regardless of whether a solution has been found. The sum squared error goal is the error value the program must reach before it stops. The learning rate is the step size taken by the program after each loop and corresponds to η in equations **4.8** and **4.10**. The following values for these parameters used in the test are given below in **table 6.2**.

Algorithm	Epoch limit	Sum squared error goal	Learning rate
Standard gradient descent	5000	0.1	0.002
Gradient descent with momentum	5000	0.01	0.002
Levenberg-Marquart	5000	0.01	0.002

Table 6.2. The settings for the parameters in the Matlab backpropagation commands used in the convergence tests.

6.3.5 The radial basis function

The theory needed to understand the radial basis function for this test has been described in **section 4.6**. From a practical viewpoint, to produce comparable results, radial basis functions require many more neurons than does backpropagation. They also need more training data since they work on the principle of exact interpolation. The neuron architecture used for the rate of convergence investigation is given below in **figure 6.9**.

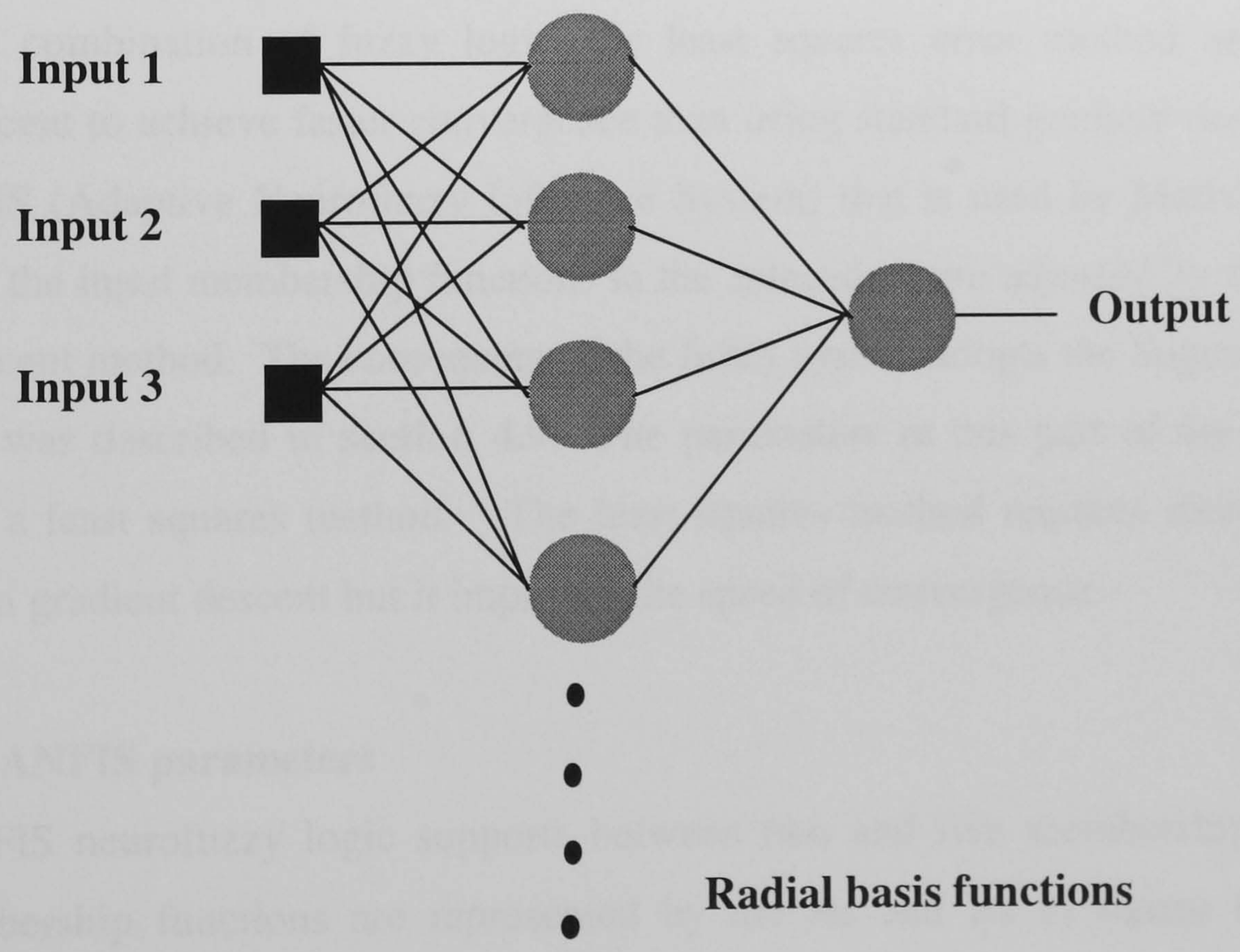


Figure 6.9. The radial basis function architecture with three inputs for the three pre-processor outputs shown in figure 6.7.

The Matlab radial basis function that was used to partition the data in **figure 6.7** and the parameters that were used in this function are given below:

Command: solverb

Adjustable parameters: Neuron limit, sum squared error goal, spread function

The parameters can be altered to optimise the accuracy of the network. The number of neurons increases in the hidden layer as the program runs. This will reduce the error. The program will stop if either the neuron limit or error goal is reached. The spread represents the width of the radial basis functions. This can change the response of the radial basis function with distance from a small localised centre. Therefore, the accuracy of the network can be tuned by adjusting this parameter. The parameters used to obtain the results in **table 6.3** were:

Neuron limit =1000

Error goal = 10^{-2}

Radial basis function spread function =5

6.3.6 ANFIS neurofuzzy logic

This uses a combination of fuzzy logic, the least squares error method and standard gradient descent to achieve faster convergence than using standard gradient descent alone. In the ANFIS (Adaptive Neurofuzzy Inference System) that is used by Matlab shown in **figure 6.10**, the input membership functions in the antecedent are adjusted by the standard gradient descent method. The consequent of the fuzzy system adopts the Sugeno inference system that was described in **section 4.9**. The parameters in this part of the system are adjusted by a least squares method. The least squares method requires more computer memory than gradient descent but it improves the speed of convergence.

6.3.7 The ANFIS parameters

Matlab ANFIS neurofuzzy logic supports between two and five membership functions. These membership functions are represented by the As and Bs in **figure 6.10**. Two triangular membership functions were used in this test. Since gradient descent is also involved in this method an error goal has to be set. An error value of 0.01 was used.

6.3.8 The results of the convergence test

The procedure used was to present the data to the network and assign them with either a 1 or 0 depending on which class they belonged to. These are the target vectors. Each minimisation method was used to produce a function that could map the data to the target values. The time taken for each method to achieve this was recorded. **Table 6.3** gives the results of these test. The convergence times refer to the 200MHz Pentium processor that was used in the test. From these results it is seen that the Levenberg-Marquart algorithm, the radial basis function and ANFIS converged in 1s and are much faster than standard gradient descent and gradient descent with momentum which converged in 12mins and 20s respectively. All the techniques assigned all of their input vectors to the target values without error. Using the reason described in **section 6.3.4** the use of standard gradient descent and gradient descent with momentum was discontinued. The only reason for their possible reintroduction would be if firstly, the faster techniques failed to reach a sum squared error of 0.1 for training set data for a given problem. Secondly, the faster techniques applied here proved to be excessively slower for other problems. A time limit of 30 mins for convergence for all the networks had been set, otherwise the network parameters, inputs or network type was changed.

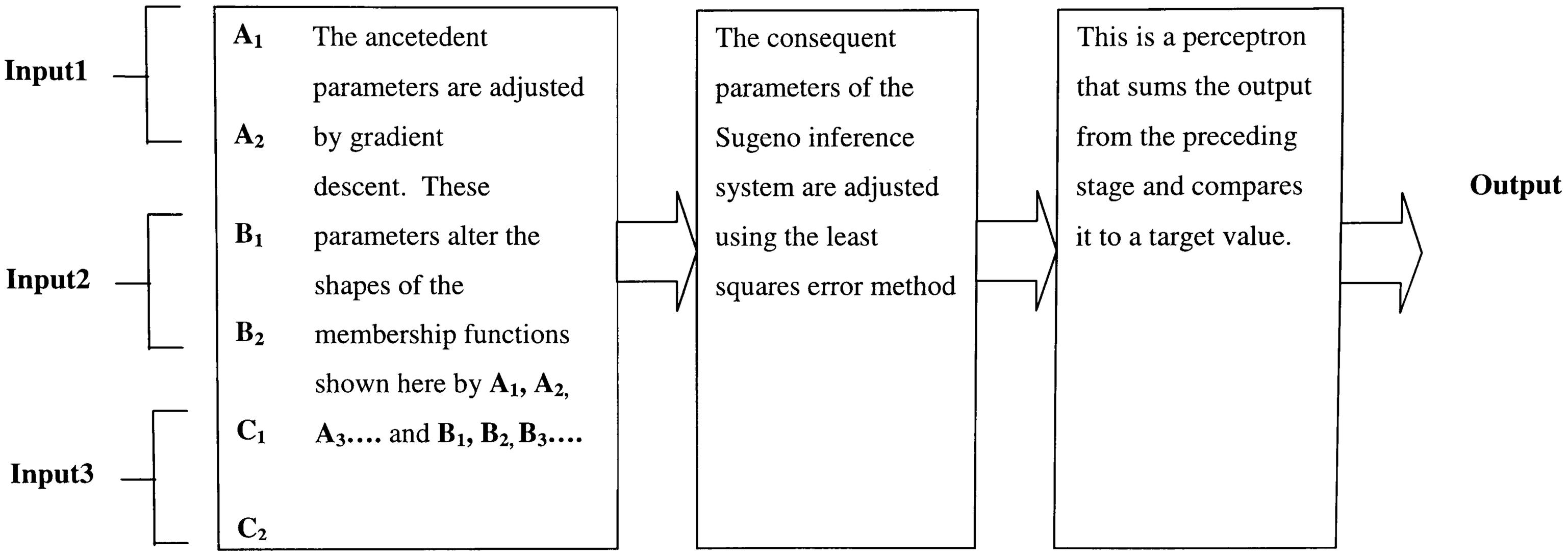


Figure 6.10. A flow chart that describes the principles of ANFIS neurofuzzy logic that is found in Matlab.

Target value	Standard gradient descent response Time Taken = 12 mins	Gradient descent with momentum response Time taken = 20 s	Levenberg-Marquart response Time taken = 1 s	Radial Basis Function Time taken = 1s	ANFIS neuro-fuzzy logic Time taken = 1s
1	0.9614	0.9847	1.0000	1.0157	1.0000
1	0.9617	0.9847	1.0000	1.0128	0.9999
1	0.9635	0.9848	1.0000	0.9513	1.0000
1	0.9621	0.9847	1.0000	1.0121	1.0000
1	0.9635	0.9848	1.0000	0.9811	1.0001
1	0.9628	0.9848	1.0000	1.0046	1.0000
1	0.9639	0.9848	1.0000	0.9984	1.0001
1	0.9638	0.9848	1.0000	0.9999	0.9999
1	0.9636	0.9848	1.0000	1.0076	0.9994
1	0.9638	0.9848	1.0000	1.0192	1.0002
1	0.9559	0.9844	1.0000	1.0020	1.0000
1	0.9596	0.9846	1.0000	1.0395	1.0005
1	0.9464	0.9830	1.0000	0.9967	0.9997
1	0.9213	0.9791	1.0000	0.9583	1.0000
1	0.9212	0.9816	0.9514	0.9921	1.0000
0	0.0627	0.0191	0.0013	0.0003	0.0000

0	0.0625	0.0191	0.0013	-0.0102	0.0000
0	0.0636	0.0191	0.0013	-0.0018	0.0000
0	0.0626	0.0191	0.0013	0.0074	-0.0001
0	0.0664	0.0194	0.0013	-0.0079	0.0001
0	0.0660	0.0193	0.0013	-0.0105	0.0000
0	0.0762	0.0206	0.0013	0.0156	0.0007
0	0.0626	0.0191	0.0013	0.0080	0.0001
0	0.0701	0.0198	0.0013	-0.0011	-0.0013
0	0.0678	0.0195	0.0013	-0.0071	0.0007
0	0.0816	0.0213	0.0013	0.0188	-0.0001
0	0.0624	0.0191	0.0013	0.0062	0.0000
0	0.0646	0.0192	0.0013	-0.0185	-0.0001
0	0.0626	0.0191	0.0013	0.0035	0.0000
0	0.0707	0.0206	0.0013	0.0060	0.0000

Table 6.3. The results of the classification of the data shown in figure 6.5 using standard gradient descent, gradient descent with momentum and the Levenberg Marquart algorithm.

6.4 The classification of printing techniques for images of 1 cm squares using neural networks

In the first three introductory sections the difficulties in defining and measuring print quality perception were discussed. The problems not only come from the quantity of variables that need to be measured but also from the fact that print quality perception is subjective. These problems can be tackled initially by identifying different printing processes. Prints from different printing processes are sometimes percieved to be of differing quality. If different printing processes can be identified then a form of print quality measurement will have been produced. Another reason for measuring print quality in this way is that it provides an objective way to test the precision of the image analysis system.

An initial investigation to test the feasibility of this technique was made by using some of the measurements taken in **table 6.1**. In this test the neural networks were used to differentiate between the printing processes shown in **figure 6.11**. Seven different processes were used. Each process contributed a series of eight sets of measurements from an image of a 1 cm solid square. A training set was produced for the neural networks using measurements from the 4M plus laser printer, reproductions of the 4M print using the Sharp SF-2022 photocopier and the Hewlett Packard Deskjet inkjet printer. The resulting neural network models were validated using the Panasonic KX -P6100 laser printer, reproductions of the Panasonic print using the Sharp SF-2022 photocopier, the Epson Stylus Colour inkjet, and tested using a different set of measurements from the 4M laser printer, an Oce bookcopier photocopier and an Apple Stylewriter II inkjet printer.

In the last section it was shown that the Levenberg-Marquart backpropagation, radial basis function and ANFIS neurofuzzy logic converged most quickly. These methods were used to see whether the models they produced could distinguish between the processes described above and those in **figure 6.11**. Different experimental combinations of pre-processing outputs were used to produce network models. This procedure was also adopted in all subsequent development of neural network models. These experimental models started with two variables and were increased in size. These models varied in accuracy. The best results produced by the three different network paradigms in 1 day of experimentation are shown in **table 6.4**. These results were produced with identical pre-processing data. All three techniques used outputs 17,18, 24, 25, 37, 38 and 39. The target

Training Set	Validation Set	Test Set
Hewlett Packard 4M plus laser print at 300 dpi	Panasonic KX-P6100 PCL laser print at 300dpi	Hewlett Packard 4M plus laser print at 300 dpi
A reproduction of a Hewlett Packard 4M plus laser print at 300 dpi using a Sharp SF – 2022 photocopier	A reproduction of a Panasonic KX-P6100 PCL laser print at 300dpi using a Sharp SF – 2022 photocopier	A reproduction of a Hewlett Packard 4M plus laser print at 300 dpi using an Oce Bookcopier photocopier
Hewlett Packard Deskjet 1200 inkjet print at 300dpi	Epson Stylus Colour 50 inkjet print at 300dpi	An Apple Stylewriter II inkjet print at 300 dpi

Figure 6.11. Block diagram of the arrangement of the datasets used to investigate whether the image analysis system could identify different printing processes.

Levenberg-Marquart			Radial basis function			ANFIS Neurofuzzy logic		
Set1	Set2	Set3	Set1	Set2	Set3	Set1	Set2	Set3
0.9759	0.7021	0.9754	0.9974	0.6536	0.9866	1.0000	-3.5956	0.9803
0.9759	0.6400	0.9746	1.0018	0.5875	0.9820	1.0000	-2.6084	0.9828
0.9754	0.8144	0.9741	1.0027	0.6038	0.9890	1.0000	-1.3693	0.9841
0.9750	0.7528	0.9733	1.0007	0.5953	0.9889	1.0000	-1.4395	0.9843
0.9749	0.8277	0.9754	0.9956	0.6097	0.9530	1.0000	-1.0592	0.9472
0.9740	0.7822	0.9705	1.0008	0.6001	0.9942	1.0000	-0.7938	0.9711
0.9728	0.8235	0.9732	0.9996	0.6089	0.9940	1.0000	-0.8212	0.9960
0.9759	0.7504	0.9746	1.0011	0.5969	0.9814	1.0000	-0.7181	0.9829
0.4889	0.5269	0.2004	0.5001	0.5200	0.1412	0.0000	-1.2108	1.1716
0.5030	0.5129	0.2179	0.5014	0.5071	0.2848	0.0000	0.0783	1.1910
0.4825	0.6029	0.1997	0.4972	0.5086	0.1871	0.0000	-0.2457	1.3273
0.5056	0.5412	0.2192	0.5031	0.5201	0.4005	0.0000	0.5092	1.5504
0.5038	0.5243	0.1961	0.4953	0.5214	0.3070	0.0000	0.4858	2.5176
0.4970	0.4925	0.1815	0.50050	0.5144	0.2586	0.0000	0.1895	2.1227
0.5036	0.5579	0.1851	0.5018	0.5001	0.2585	0.0000	0.4816	2.3586
0.4919	0.5198	0.0706	0.5009	0.5155	0.1195	0.0000	0.3563	1.6945
0.0069	0.0454	0.0045	0.0000	0.0000	0.0000	1.0000	0.8165	0.5061
0.0045	0.0045	0.0073	0.0000	0.0000	0.0000	1.0000	0.8098	0.4775
0.0045	0.0045	0.0045	0.0000	0.0000	0.0000	1.0000	0.8231	0.4907
0.0045	0.0045	0.0049	0.0000	0.0000	0.0000	1.0000	0.7490	0.4816
0.0045	0.0045	0.0074	0.0000	0.0000	0.0000	1.0000	0.7659	0.4485
0.0051	0.0045	0.0045	0.0000	0.0000	0.0000	1.0000	0.7366	0.5064
0.0045	0.0045	0.0048	0.0000	0.0000	0.0000	1.0000	0.6897	0.4738
0.0073	0.0045	0.0045	0.0000	0.0000	0.0000	1.0000	0.7950	0.5263

Table 6.4. Results of the printing process classification experiment described above. The target values assigned to the laser print, photocopies and inkjets were 1, 0.5 and 0 respectively.

values assigned to the laser print, photocopies and inkjets were 1,0.5, and 0 respectively. The same 4M laser printer was used in the training and the second validation set to check that the image analysis system was working consistently. The parameters for the networks were:

Levenberg-Marquart(1 hidden layer, 1 neuron, learning rate = 0.002, error goal = 0.01, transfer functions were for the hidden layer: tansig and for the output layer logsig)

Radial basis function (neuron limit =1000, error goal = 10^{-4}, radial basis function spread function =15)

ANFIS (Error goal =0.01, number of membership functions = 2, membership function type = triangular)

For all three methods given above, the other parameters were at their default values. **Table 6.4** demonstrates that the printing processes used in this test were distinguished better using the Levenberg-Marquart algorithm and the radial basis function than with ANFIS neurofuzzy logic. This is because for the Levenberg-Marquart and radial basis function all three inkjet print samples produced near zero output responses, the two photocopiers produced the middle values, and the two laser printers produced the highest values. Also equal amounts of time were allocated to finding a solution for each of the three methods. This was 1 hour for each method.

These results established that the classification of different printing processes can be made using the pre-processing program and Matlab neural networks. This of course does not mean that all non-impact printing processes can be distinguished using it. However it does mean that it does have the potential to do so, since the system is modular and can be expanded to measure more variables. This last point and its potential practical application will be explored in the concluding part of this thesis.

6.5 The simultaneous modelling of halftones and solids

So far models that can differentiate between printing processes have been produced. These models have considered halftones and solids separately. To build an automated print quality system the models for solids and halftones need to be combined. **Figure 6.12** is an example where, in two datasets from two different printing processes, there are measurements both from halftones and solids. These measurements originate from the two previous problems. The **O**s are from the 4M laser printer. The ✱ points are from reproductions of the 4M samples using the Sharp-2022 photocopier. It can be seen that there exists a non-linear function that can classify the two sets of data. This type problem can be modelled using a neural network having a hidden layer.

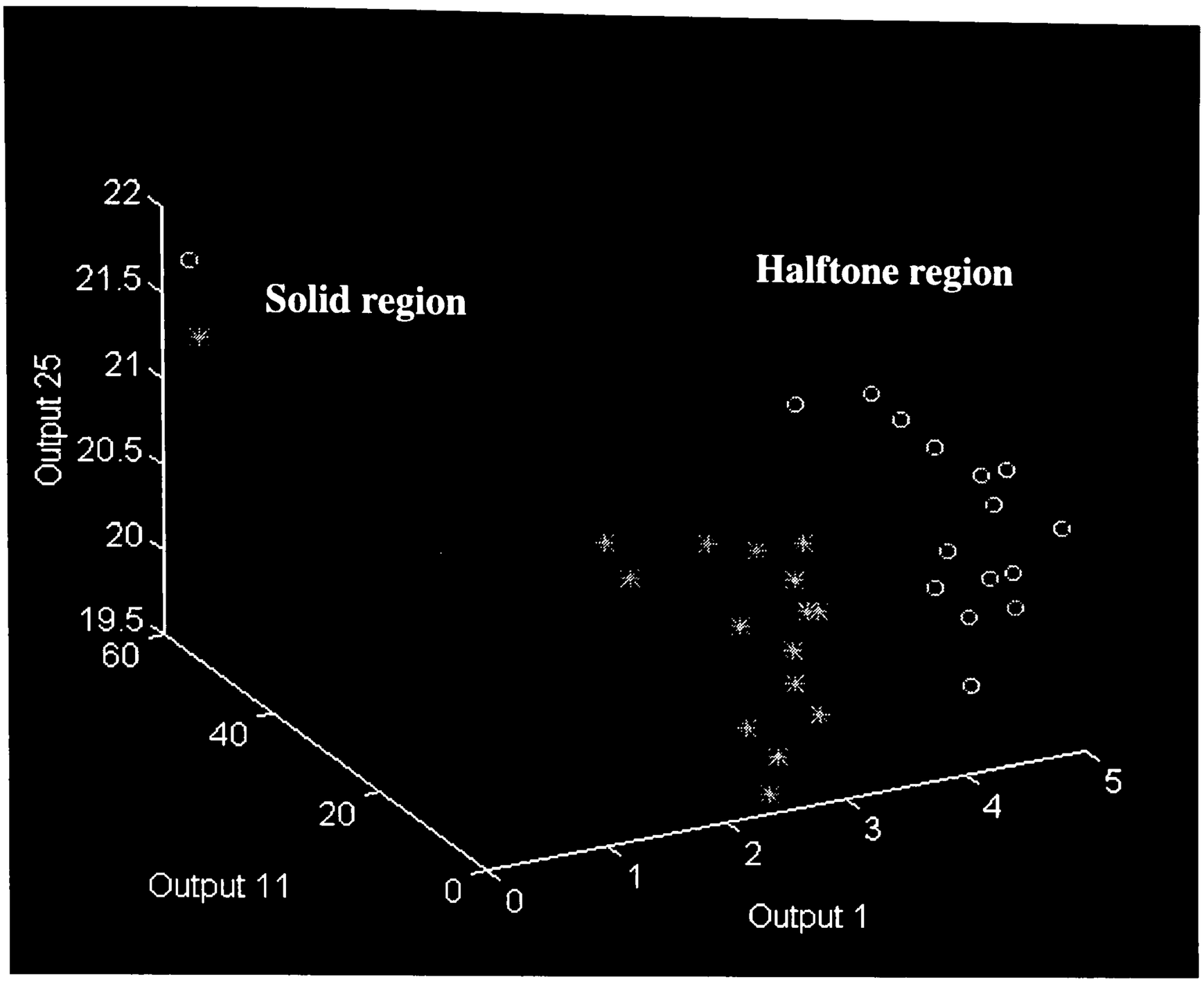


Figure 6.12. This graph shows a non-linearly separable relationship between halftones and solids. The data from the Os originate from the 4M laser printer and the ✳s from a reproduction of this print using a Sharp S-2022 photocopier.

Solutions to this problem were found using the Levenberg-Marquart algorithm and an example of the results of this are given below in **table 6.5**. The validation results for this network came from prints produced from the Hewlett Packard 4M laser printer and a set of readings from a photocopy of the Hewlett Packard 4M print using a Canon 6030 photocopier.

Training Set		Validation Set	
4M laser printer (Set 1)	**Sharp – 2020 photocopier**	**4M laser printer (Set2)**	**Canon 6030 photocopier**
0.9780	0.0135	0.9780	0.0132
0.9780	0.0117	0.9780	0.0127
0.9780	0.0108	0.9780	0.0108
0.9777	0.0111	0.9780	0.0108
0.9780	0.0112	0.9780	0.0110
0.9780	0.0108	0.9780	0.0108
0.9780	0.0108	0.9780	0.0108
0.9780	0.0108	0.9780	0.0108
0.9780	0.0108	0.9780	0.0108
0.9780	0.0113	0.9780	0.0108
0.9780	0.0108	0.9780	0.0108
0.9780	0.0108	0.9780	0.0108
0.9780	0.0108	0.9780	0.0108
0.9777	0.0108	0.6591	0.0108
0.9264	0.0514	0.9612	0.0161
0.9271	0.0587	0.9618	0.0167
0.9341	0.0543	0.9632	0.0166
0.9541	0.0732	0.9612	0.0170
0.9532	0.0557	0.9635	0.0184
0.9524	0.0457	0.9625	0.0175
0.9559	0.0547	0.9622	0.0169
0.9475	0.0584	0.9629	0.0166

Table 6.5. These results show that it is possible to differentiate between a laser print and its photocopy for both halftones and solids using a single network model.

The results above show that the image analysis system differentiated between a series of tones and solids from the 4M laser printer and photocopies of these tones and solids from a Canon 6030. These results were obtained by training the neural network with another set of data from the 4M laser printer and data from a photocopied version of the 4M laser print using a Sharp SF-2020.

6.6 The production of print quality models for images of circles

All the print quality measurements that have been carried out up to this point have been concerned with the image of a 1 cm square. It was demonstrated in **sections 6.4**and **6.5**, that print quality models using the image analysis system could be produced for this shape. These models were multi-dimensional and used features that were extracted from the entire image.

However, one of the aims of this project was to produce a system that can automatically measure the print quality of an image irrespective of its shape. The next stage of this investigation was to see whether the system could produce similar models for images of other shapes. A circle of 1cm diameter was used next, to test the image analysis system. This section describes the investigation that was carried out on the image of a circle. The procedure which was used for producing the images of the square was also used here, except that fewer print samples were made. **Table 6.6** below states the printing processes that were used to produce the images of the circle.

1	Hewlett Packard 4M plus laser print at 600 dpi
2	A reproduction of a Hewlett Packard 4M plus laser print at 600dpi using a Sharp SF-2022 photocopier
3	Epson Stylus Colour 50 inkjet print at 300dpi
4	A reproduction of an Epson Stylus Colour 50 inkjet print at 300dpi using a Sharp SF – 2022 photocopier
5	Kyocera FS-1700 laser print at 600 dpi
6	A reproduction of a Kyocera FS-1700 laser print at 600 dpi using a Sharp SF – 2022 photocopier
7	Hewlett Packard Deskjet 1200 inkjet print at 300dpi
8	A reproduction of a Hewlett Packard Deskjet 1200 inkjet print at 300dpi using a Sharp SF-2022 photocopier

Table 6.6. The sets of prints of 1 cm diameter circles used to test the image analysis system.

The first four processes shown in **table 6.6** were used as the training set and the last four as the validation set. To demonstrate the ability of the image analysis system to differentiate

between the images of the circle produced by the processes in the validation set four different neural networks each with a single output were constructed. A simpler alternative would be to build a single network with four outputs. This method has disadvantages, firstly, each output has to converge to the desired solution in one training run. If one or more fails to do so, then the entire program has to be rerun. Secondly, if one network with four outputs is employed, all the outputs must use the same inputs. If four single networks are used then each output can have its own set of different inputs. Therefore, using single networks is more versatile than using a single network.

1 The images from the 4M laser printer, target value = 1, and the photocopies of these images, target value = 0 were used as the training set. The resulting model was validated using the images from the Kyocera FS-700 laser printer and the photocopies of these images. A Levenberg-Marquart algorithm was used to classify the data. The neural network parameters, transfer functions and pre-processing outputs that were used were:

Number of neurons in the hidden layer = 4

Transfer functions: tansig for the hidden layer and logsig for the output layer

Network learning rate = 0.005

Error Goal = 0.1

Pre-processing outputs used: 1,9, 11,16, 23,

2 The images from the Epson inkjet printer, target value = 1, and the photocopies of these images, target value = 0 were used as the training set. The resulting model was validated using the images from the Hewlett Packard inkjet printer, target value = 1 and the photocopies of these images target value = 0. A Levenberg-Marquart algorithm was used to classify the data. The neural network parameters, transfer functions and pre-processing outputs that were used were:

Number of neurons in the hidden layer = 3

Transfer functions: tansig for the hidden layer and logsig for the output layer

Network learning rate = 0. 05

Error Goal = 0.1

Pre-processing outputs used: 1, 10, 16, 37, 39

3 The images from the 4M laser printer, target value = 1, and the images from the Epson inkjet printer, target value = 0 were used as the training set. The resulting model was validated using the images from the Kyocera FS-1700 laser printer target value =1 and the Hewlett Packard inkjet printer target value = 0. A Levenberg-Marquart algorithm was used to classify the data. The neural network parameters, transfer functions and pre-processing outputs that were used were

Number of neurons in the hidden layer = 3

Transfer functions: tansig for the hidden layer and logsig for the output layer

Network learning rate = 0.05

Error Goal = 0.1

Pre-processing outputs used: 1, 10, 16, 37, 39

Each set of images consisted of 15 halftones and solids. They were placed in random positions in the image window, but the halftone patterns were aligned parallel to the horizontal axis. The solids could not be aligned since the images were circular and were placed completely at random in the image window. The results of these tests are given in **tables 6.7-6.9** below.

The results demonstrate that the image analysis system can extract features that can be used to differentiate a laser print from its photocopy. If this were not true, the neural network would not converge to produce the training set results shown in **tables 6.7-6.9**.

There is an approximately 4% discrepancy in the classification for the validation results in **tables 6.7-6.9**. These may not be errors since the printing processes in the validation sets are only of the same class and are not identical. For example, the Hewlett Packard and Kyocera laser printers produced different halftone patterns. Another reason for this discrepancy might be that the toners they use are chemically different.

The next section describes the use of images of text characters to test the characteristics of the image analysis system. These tests verified that the system could extract features from arbitrary bi-level images.

Training Set		Validation Set	
Hewlett Packard 4M Laser Printer	**Sharp – SF2022 Photocopier**	**Kyocera FS – 1700 Laser Printer**	**Sharp – SF2022 Photocopier**
0.9872	0.0108	0.0129	0.0106
0.9874	0.0106	0.9869	0.0106
0.9874	0.0106	0.9833	0.0106
0.9868	0.0106	0.0125	0.0106
0.9874	0.0106	0.9866	0.0106
0.9874	0.0106	0.9870	0.0106
0.9874	0.0106	0.9874	0.0107
0.9874	0.0106	0.9864	0.0474
0.9874	0.0107	0.8832	0.0820
0.9874	0.0122	0.9836	0.1041
0.9874	0.0900	0.9332	0.2000
0.9874	0.0496	0.9862	0.1115
0.9874	0.1050	0.9873	0.4226
0.9137	0.1372	0.9604	0.0800
0.9503	0.1627	0.9798	0.1863

Table 6.7. Results showing the differentiation between the images of a circle produced on a laser printer and their photocopies.

Training Set		Validation Set	
Epson Stylus Colour 50 Inkjet Printer	**Sharp – SF2022 Photocopier**	**Hewlett Packard Deskjet 1200 Inkjet Printer**	**Sharp – SF2022 Photocopier**
0.9876	0.0017	1.0000	0.0004
0.9947	0.0002	0.9995	0.0000
1.0000	0.0338	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
1.0000	0.0970	1.0000	0.0000
1.0000	0.1007	1.0000	1.0000
1.0000	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0506
0.9748	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
0.9924	0.0000	0.0000	0.0000

Table 6.8. Results showing the differentiation between the images of a circle produced on an inkjet printer and their photocopies.

Training Set		Validation Set	
Hewlett Packard 4M Laser Printer	**Epson Stylus Colour 50 Inkjet Printer**	**Kyocera FS - 1700 Laser Printer**	**Hewlett Packard Deskjet 1200 Inkjet Printer**
0.9761	0.0145	**0.0178**	0.0313
0.9782	0.0233	0.9262	0.0231
0.9869	0.0648	0.9809	0.0404
0.9880	0.0747	0.9851	0.0434
0.9872	0.0732	0.9871	0.1212
0.9879	0.0572	0.9852	0.0741
0.9896	0.0450	0.9837	0.0691
0.9883	0.0424	0.9796	0.0879
0.9849	0.0504	0.9074	0.0833
0.9842	0.0421	0.9035	0.0699
0.9772	0.0185	0.6852	0.0646
0.9748	0.0121	**0.4933**	0.0417
0.9341	0.0711	**0.4337**	0.0214
0.9101	0.0554	0.8878	0.0225
0.9564	0.0674	0.9474	0.1370

Table 6.9. Results showing the differentiation between the images of a circle produced on a laser printer and their photocopies. The classification errors are in bold.

6.7 The use of images of text characters to test the image analysis system

The two experiments on the image analysis system performed in this section showed that features from images of text characters could be extracted using the image analysis system. The experiments also demonstrated that neural networks models could be constructed using these features.

Two text characters were chosen for this test. One was a 24pt letter 'R'. The other was a 36pt letter 'G'. Both of these text characters were produced using a Swiss "San Serif" font. The letter 'R' was printed on the 4M laser printer and the Epson inkjet printer. The letter G was printed on the 4M laser printer and the Apple inkjet printer. Each printing process produced a series of 15 of their assigned characters. These characters differed in print density.

In the first experiment, two sets of measurements were taken using the image analysis system for each of the images of the letter 'R'. The first and second sets were used as the training and validation sets respectively. A Levenberg-Marquart algorithm was used to solve this problem. The neural network parameters, transfer functions and pre-processing outputs that were used were:

Number of neurons in the hidden layer = 1
Transfer functions: tansig for the hidden layer and logsig for the output layer
Network learning rate = 0.005
Error Goal = 0.1
Pre-processing outputs used: 1, 16, 10, 7, 18

The results of this test shown in **table 6.10** contain no misclassifications. This demonstrates that the output results from the image analysis system are repeatable using the image analysis system for text characters.

In the second experiment, the preceding network system and results were applied to the problem of classifying the images of the Gs that were produced on the 4M laser printer and Apple inkjet printer. It can be seen from the data shown in **table 6.11**, that the network that was used to train and validate the results for the letter R can be used to classify the letter G. It is also observed that there are no misclassifications.

The work described in **section 6** so far has concentrated on using the image analysis system to identify printing processes. This has been done for the following reasons. Firstly, it provided an objective test for the system. Secondly, if the tests were successful, then the system would be able to measure at least one print quality variable, because a factor that determines print quality is the printing process. However none of the results so far have involved observers. That is to say, observers have not been used to see whether

they could differentiate between the printing processes used in the tests. The next stage assesses whether the image analysis system can classify and predict print quality assessments that have been made by a human observer.

Training Set		Validation Set	
Set 1 of the 4M laser print of the letter R	**Set1 of the Epson inkjet print of the letter R**	**Set 2 of the 4M laser print of the letter R**	**Set 2 of the Epson inkjet print of the letter R**
0.9729	0.0227	0.9731	0.0227
0.9773	0.0227	0.9773	0.0227
0.9750	0.0227	0.9765	0.0227
0.9763	0.0227	0.9772	0.0227
0.9772	0.0344	0.9772	0.0407
0.9759	0.0357	0.9763	0.0344
0.9687	0.0667	0.9654	0.0466
0.9662	0.0254	0.9636	0.0253
0.9764	0.0227	0.9772	0.0227
0.9767	0.0611	0.9765	0.0227
0.9070	0.0228	0.9687	0.0228
0.9769	0.0228	0.9773	0.0229
0.9773	0.0247	0.9773	0.0250
0.9773	0.0442	0.9773	0.0658
0.9772	0.0228	0.9766	0.0228

Table 6.10. The differentiation of Hewlett Packard 4M laser print from Epson Colour 50 inkjet print for the 24pt sized 'Rs'. No classification errors were recorded for this solution.

Training Set		Test Set	
Set 1 of the 4M laser print of the letter R	**Set1 of the Epson inkjet print of the letter R**	**4M laser print of the letter G**	**Apple Stylewriter II inkjet print of the letter G**
0.9729	0.0227	0.9761	0.0293
0.9773	0.0227	0.8541	0.0227
0.9750	0.0227	0.9773	0.0227
0.9763	0.0227	0.9773	0.0239
0.9772	0.0344	0.9773	0.2564
0.9759	0.0357	0.9772	0.0346
0.9687	0.0667	0.9768	0.0308
0.9662	0.0254	0.9738	0.0334
0.9764	0.0227	0.9773	0.0586
0.9767	0.0611	0.9772	0.0238
0.9070	0.0228	0.9772	0.0227
0.9769	0.0228	0.9772	0.0229
0.9773	0.0247	0.9773	0.0332
0.9773	0.0442	0.9773	0.0366
0.9772	0.0228	0.9772	0.0284

Table 6.11. The differentiation of the Hewlett Packard 4M laser and Apple Stylewriter inkjet print for the 36pt sized Gs. No classification errors were recorded for this solution.

6.8 The simulation of print quality perception using the image analysis system

The previous sections of this chapter described an investigation performed on the image system to see whether it could differentiate between different non-impact printing processes. Although this was successful, observers have not been employed for the same task to corroborate measurements. A test of this kind would be limited in the sense that the number of assessments that could be made would be small. It would be limited in this case to the 14 processes in **table 6.1**. To classify the 16 processes would also have been time consuming since each process had to be compared with all the others individually. Also no information would be available about how print quality varies within a process itself.

A more accurate model can be produced more quickly by reducing the importance of the role that the printing process has been given so far in this investigation. This can be achieved since the outputs of the pre-processing program can be used to assess directly the print quality of an image instead of the process it originated from. This also means that the quality of different images within a process can also be modelled. This would give a greater variety of assessments and a better chance for the system to pass a Turing type test.

It has been shown that the image analysis system can differentiate between printing processes for a square, circle and two text characters. For the observer assessment, a more complex image object was chosen. The object was a 1cm diameter graphical representation of a face. A few examples of these images are shown in **figure 6.13** below.

Figure 6.13. Examples of the face images used for image analysis and subjective assessments.

This image is more complex than a text character since it contains more than one component. It has the eyes and a mouth within a solid circular border. Therefore, using this image would test the generalising capabilities of the image analysis further.

Two hundred different images of the face were produced using different laser and inkjet printers. These images were varied in tone and resolution. The images were photocopied using a variety of photocopiers and the copies thoroughly shuffled up. They were then presented for an observer to classify into five groups according to the perceived quality. This was carried out in a lighting booth using D6500 lighting. These groups were given the labels very good, good, average, bad and very bad. The classification procedure was continued until there were at least thirty images in each group. The observer assessments took approximately 45 minutes to carry out. The images were checked by another observer for possible discrepancies and when the first observer rechecked these; eight corrections were made. The checking procedure was important, as any contradictions in the data would prevent the neural network finding a solution.

The images in the average group were ignored. This increases the error margin within which the image analysis system was required to operate, since only the principles of this method were investigated here. The two groups of thirty images labelled very good and good were combined into a single group and the groups labelled bad and very bad into another single group of sixty images. Forty-eight images from each of the two groups of images were randomly selected but not mixed. These were divided into two further sub-groups of thirty-six and twelve images. The sub-groups with thirty-six images were used as a test set for the neural network. The very good and good images were assigned the target value 1 while the bad and very bad images were assigned the target value 0. The other groups with twelve images each were used as the validation set for the backpropagation network and as a test set for the radial basis network. **Figure 6.14** shows how the data was applied to the network.

A Levenberg- Marquart and a radial basis function neural network were used to model the assessment results. The network functions, parameters and pre-processing outputs used are given below.

For the Levenberg-Marquart algorithm

Number of neurons in the hidden layer = 3

Transfer functions: tansig for the hidden layer and tansig for the output layer

Network learning rate = 0. 005

Error Goal = 0.1

Pre-processing outputs used: 1, 8, 10, 16, 17, 18, 22, 25,

For the radial basis function

Number of neurons = 10000

Error Goal = 0.1

Pre-processing outputs used: 1, 16, 10, 25

Spread function = 0.5

The results in **tables 6.12 and 6.13** show that the very good and very bad images in the validation set for the Levenberg-Marquart network and the test set for the radial basis function were classified correctly. However both network paradigms misclassified one

image. Some possible reasons that can explain these errors will now be given. Firstly, the vectors of the images in the good and bad group are closer to the decision boundary than the vectors of the images in the very good and bad group. Secondly, there was insufficient training data for the neural networks. Thirdly, the assessments made by the observer were sometimes inconsistent. Fourthly, the random error inherent when aligning the images in the display window, was large enough to cause the misclassification. Fifthly, solutions that contain no errors exist, but were not discovered.

The work described so far in this section has shown that the image analysis system can perform two tasks. Firstly, it can recognise different non-impact printing processes. Secondly, it can predict the response of observer assessments of images from non-impact printing processes. These tasks have also been successful for a variety of images, including simple shapes, text characters and a simple picture of a face. Also the tones and resolution of these images were varied. This work represents but a small step in the development of a fully artificial intelligent system that can simulate print quality perception. The reason for this is that there are many problems that are still unresolved, for example, that of rotational alignment.

It was shown in the investigation of the images of the circle, the text characters and the face in **sections 6.6 - 6.9**, that the system can measure print quality variables for an image regardless of its translational position in the image window. However the images needed manual rotational alignment with the exception of circles that do not possess a halftone pattern. If human interaction for this procedure is introduced then the system cannot be artificially intelligent. In the next section an investigation into part of this problem will be described.

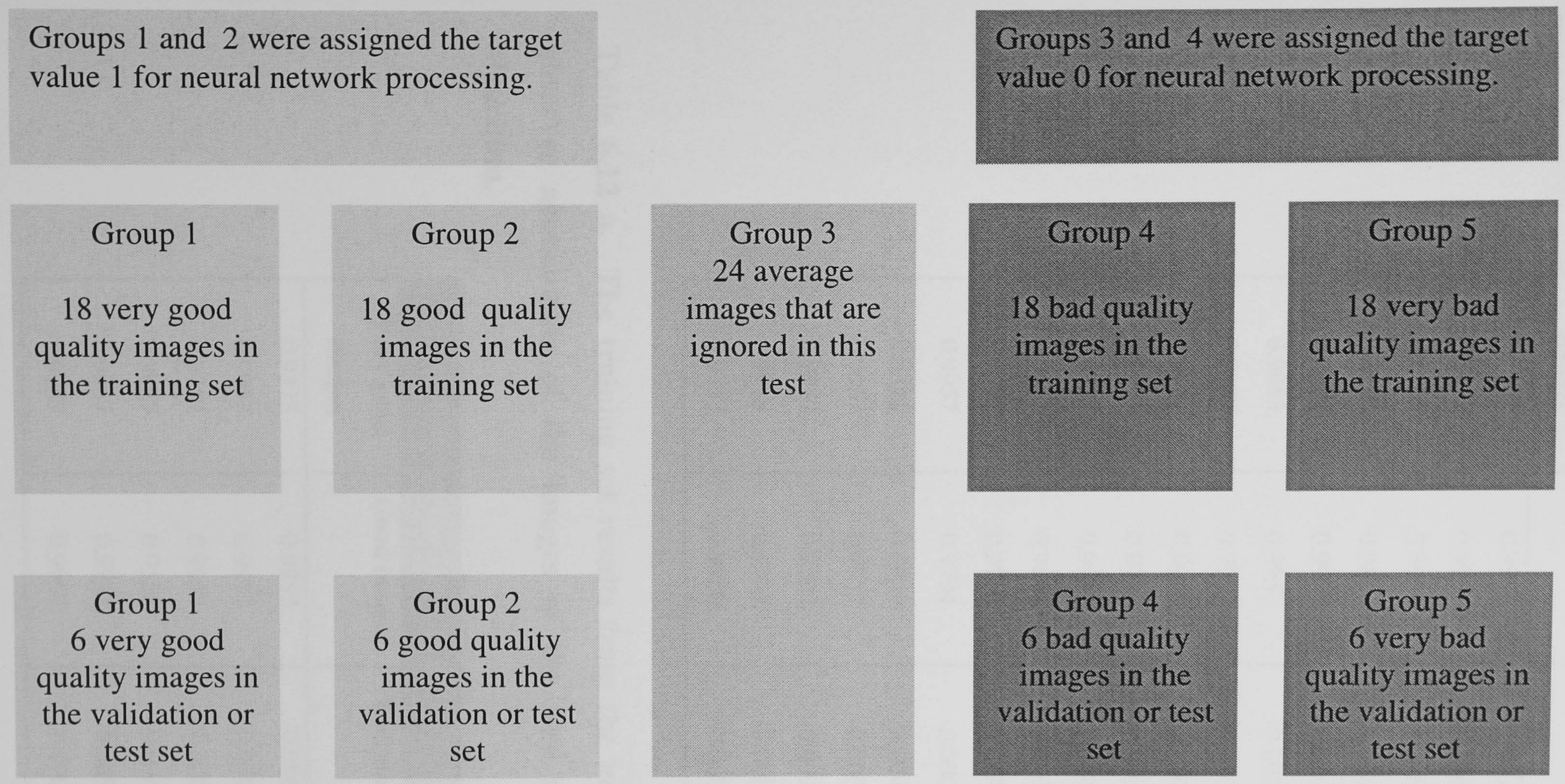


Figure 6.14. Block diagram showing how the data from the images of the face was applied to the neural networks.

Training set results for the Levenberg-Marquart network			
Very good images	**Good images**	**Bad images**	**Very bad images**
0.9957	0.9934	0.0067	-0.0004
0.9911	0.9957	-0.0004	-0.0004
0.9957	0.9957	0.0272	-0.0004
0.9935	0.9834	-0.0002	-0.0002
0.9957	0.9957	-0.0002	-0.0002
0.9853	0.9957	-0.0004	-0.0168
0.9957	0.9957	-0.0002	-0.0002
0.9857	0.9957	0.0080	0.0286
0.9957	0.9957	-0.0002	-0.0002
0.9957	0.9957	-0.0004	0.0095
0.9957	0.9881	-0.0004	-0.0002
0.9957	0.9956	-0.0002	-0.0120
0.9957	0.9794	-0.0002	-0.0002
0.9795	0.9536	-0.0002	-0.0004
0.9957	0.9957	-0.0002	-0.0002
0.9957	0.9302	-0.0002	0.0042
0.9918	0.9957	-0.0002	-0.0002
0.9957	0.9950	-0.0002	-0.0001

Table 6.12 a. The training set results from the image analysis simulation of the observer assessments of the images of the face using the Levenberg-Marquart algorithm.

Validation set for the Levenberg-Marquart network			
Very good images	**Good images**	**Bad images**	**Very bad images**
0.9875	0.9957	-0.0002	0.0005
0.9957	0.9917	0.0592	-0.0004
0.9957	0.9957	-0.0004	-0.0002
0.9957	0.9558	-0.0004	-0.0002
0.9957	0.9957	**0.8694**	-0.0002
0.9957	0.9957	-0.0003	-0.0002

Table 6.12 b. The validation results from the image analysis simulation of the observer assessments of the images of the face using the Levenberg-Marquart algorithm. The classification errors are in bold

Training set results for the radial basis function network			
Very good images	**Good images**	**Bad images**	**Very bad images**
0.9814	1.0236	0.0095	0.0027
0.8875	1.0425	0.0106	0.0026
0.9800	0.9646	0.0347	0.0026
0.9756	1.0408	0.0091	-0.0197
1.0375	0.9040	0.0861	0.0769
1.0029	1.0106	0.0026	0.0026
1.0909	1.0099	-0.0057	-0.0099
0.9358	0.9478	0.0109	0.0032
0.9797	1.0195	-0.0447	0.0179
1.0118	1.0005	-0.0015	-0.0372
0.9949	1.0017	0.0026	-0.0455
1.0310	1.0803	0.0247	0.0378
0.9894	1.0124	0.0003	-0.0248
0.9596	0.9919	-0.0456	0.0026
0.9944	1.0422	-0.0113	-0.0023
1.0094	1.0043	-0.0010	0.0027
0.9997	0.9526	-0.0131	0.0027
1.0033	1.0001	0.0001	0.0027

Table 6.13 a. The training results from the image analysis simulation of the observer assessments of the images of the face using a radial basis function.

Test set for the radial basis network			
Very good images	**Good images**	**Bad images**	**Very bad images**
1.1675	0.7569	-0.0488	0.0028
0.8417	0.9596	0.1413	0.0026
0.9129	**0.0788**	0.0026	0.0097
0.7698	0.9398	0.0027	0.0026
0.8771	1.3037	0.2468	-0.0048
0.9440	0.7398	0.0029	-0.0006

Table 6.13 b. The test results from the image analysis simulation of the observer assessments of the images of the face using a radial basis function. The classification errors are in bold.

6.9 The automatic measurement of the print quality of a text character using the image analysis system

The measurements that have been carried out up to this point using the image analysis system required manual alignment of the images. Although the system can make print quality assessments using this procedure, the system is not fully automated. The results can depend on variables associated with the human operator, such as motivation, concentration and tiredness. Ideally, the image analysis system should be able to measure the print quality of an image regardless of its position and orientation on the screen.

The positional problem has been experimentally investigated in **sections 6.6-6.9**. This was necessary to determine the effect of spatially non-uniform distortion of the image caused by the CCD and the zoom lens. However, it was found that random positional placements of the images did not prevent the classification of the printing processes using images of the circle and the text characters or the print quality assessment predictions using the images of the face. These were relatively large images that filled most of the screen. Therefore any differences in positional placement of these images on the screen produces an error that was small enough to be neglected. Also, if the position of the image had been important or a greater accuracy in the measurements had been required, the appropriate outputs of the pre-processing program could have been used. These are outputs 15 and 19 which measure the relative position of the image on the horizontal axis.

The rotational alignment problem does not require an experimental investigation to establish its existence. This is because part of the problem is independent of the instrumentation and can be shown to exist by the geometric analysis of a halftone pattern. This is illustrated in **figure 6.15 a** and **b**. The diagram shows a halftone pattern parallel to the horizontal, and another inclined at an angle to the horizontal. Both are the same halftone pattern but the pre-processing program used in this investigation will produce different output results for each alignment. For example, output 1 of the pre-processing program will produce a reading that will be equal to 1 for **figure 6.15 a**. This means that no noise is detected. For **figure 6.15 b** this reading will be less than 1. Therefore this will be interpreted as a noisy image by the program.

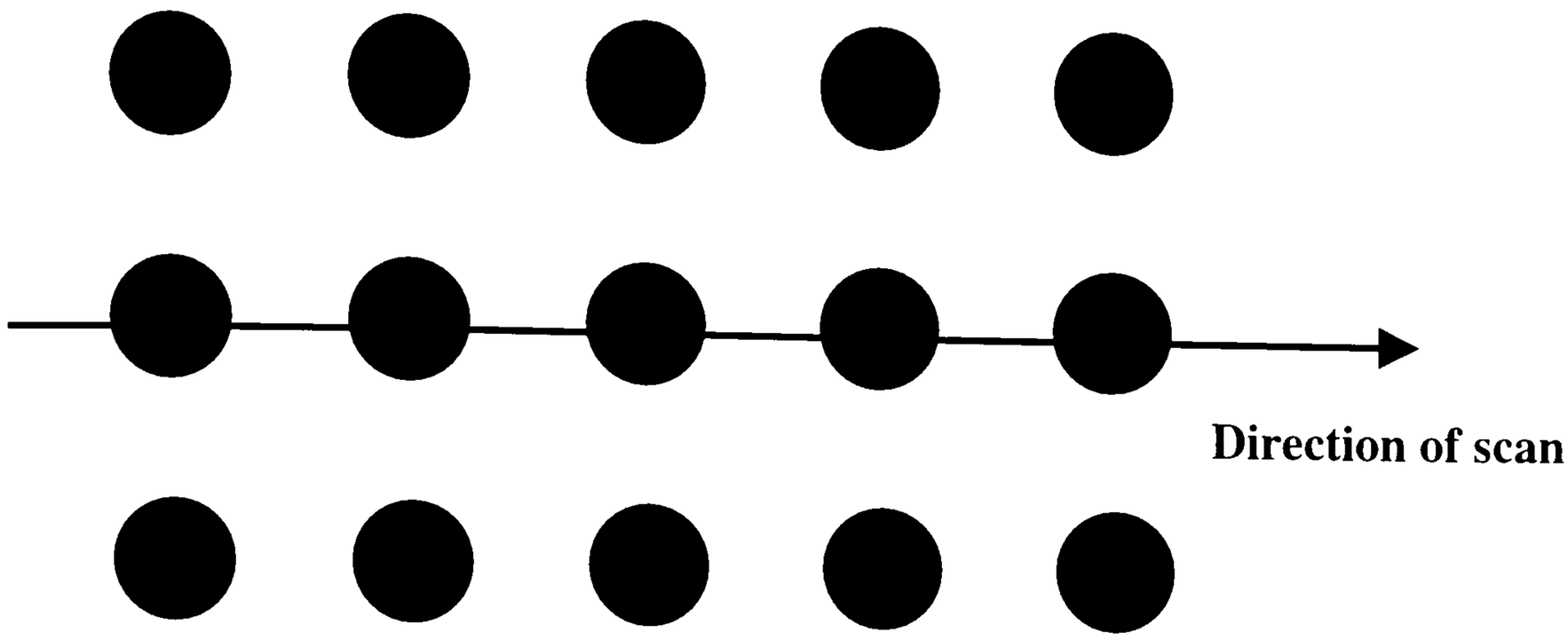


Figure 6.15 a. The halftone pattern is parallel to the horizontal. Therefore a constant frequency halftone pattern is recorded by the pre-processing program.

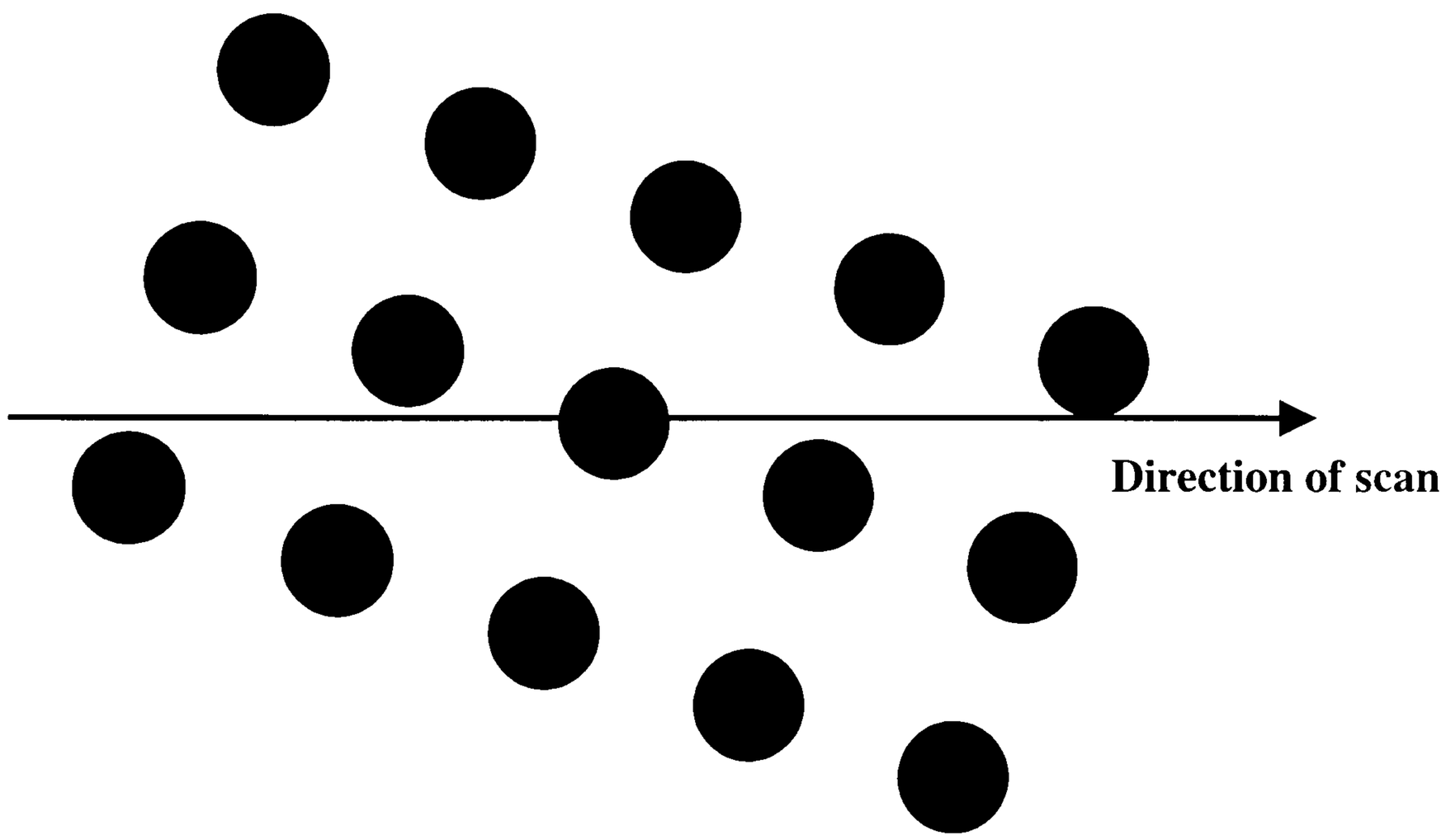


Figure 6.15 b. The halftone pattern is at an angle to the horizontal. Therefore the pre-processing program records a variable frequency halftone pattern as the display is scanned in the x direction.

Rotating an image will also change the values of the tonal gradients at the edges. To illustrate this point this is proved for the case of a vertical edge that lies exactly at the boundary in **figure 6.16 a** and **b**. These diagrams are magnified representations of the

screen image and show two border regions between the image and the background. The straight edge that is perpendicular to the horizontal in **figure 6.16 a** has gradients:

$$| \partial I / \partial X_{straight} | > 0 \text{ and } \partial I / \partial Y_{straight} = 0$$

between the pixels (x, y) and (x +1, y). The slanted straight edge in **figure 6.16 b** has gradients

$$| \partial I / \partial X_{slant} | < | \partial I / \partial X_{straight} | \text{ and } | \partial I / \partial Y_{slant} | > 0$$

between the pixels (x, y) and (x + 1, y). The reason for this transformation is that when the perpendicular edge is tilted it must cut across pixels. This means that the intensity of the border pixel on the image side is also a function of the background intensity. This will decrease the intensity values of the border pixels on the image side.

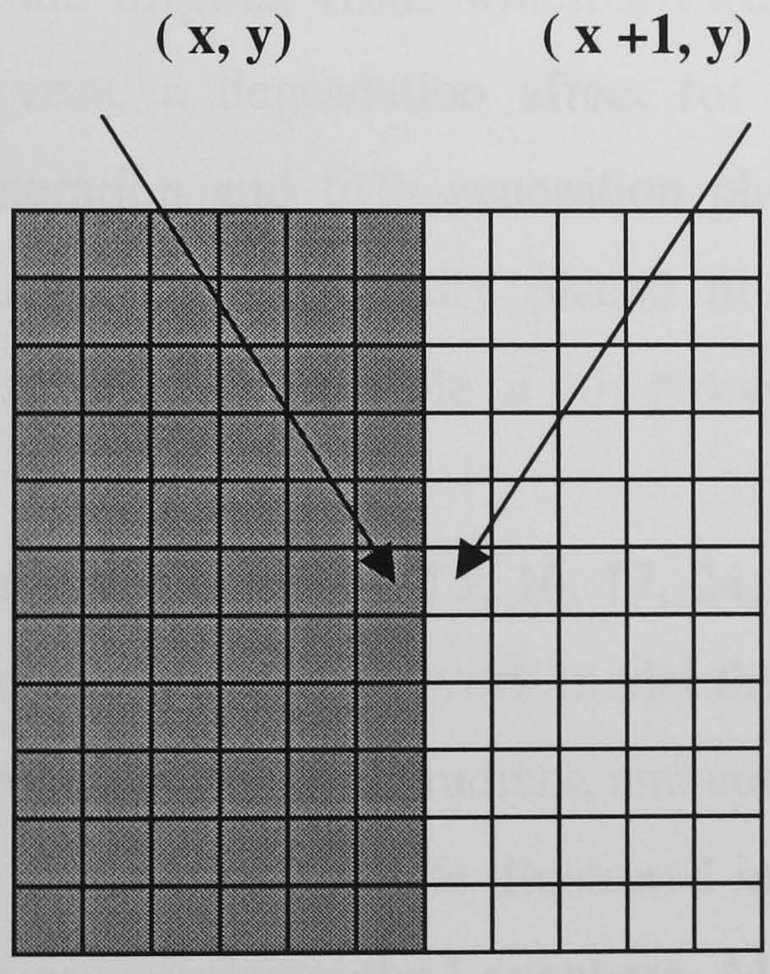


Figure 6.16 a. A representation of a border region of a straight edge perpendicular to the scanning direction.

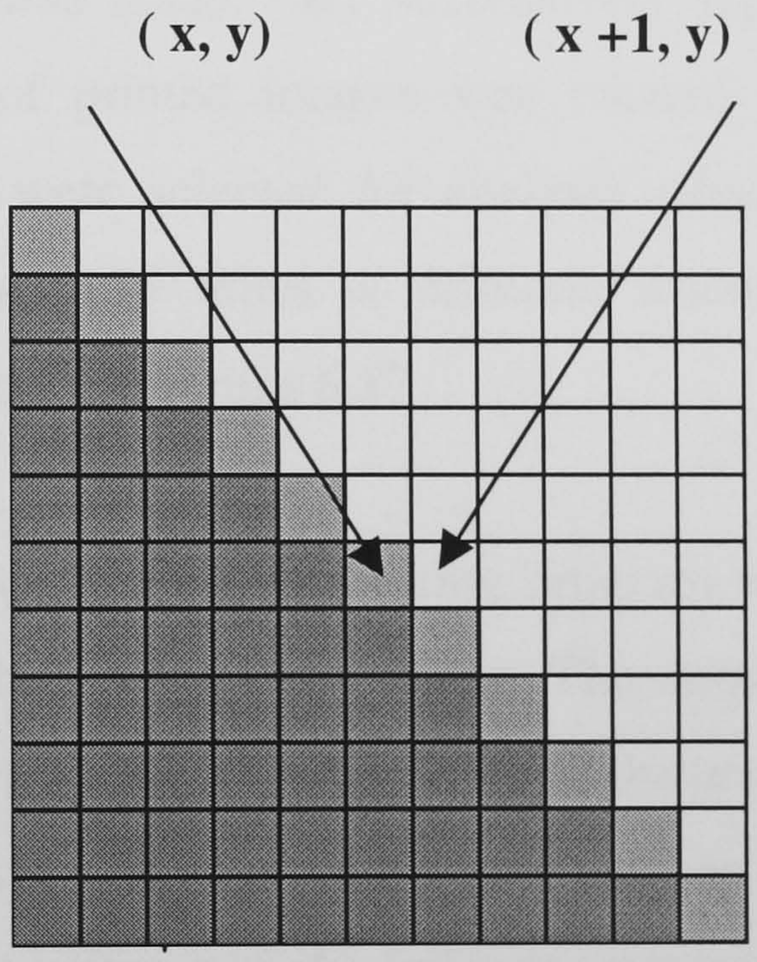


Figure 6.16 b. A representation of a border region of an edge at an angle to the scanning direction.

These two problems must be solved if the image analysis system is to be able to classify all bi -level images irrespective of the orientation of the image on the screen. The first problem, which involves the measurements of halftone patterns at different angles, cannot be solved using the pre-processing program. The outputs that measure the characteristics

of halftone patterns were designed only for images that can be aligned to the horizontal axis.

The pre-processing program can, in theory solve the second problem involving the measurements of the tonal gradients. This is because the program is able to measure the tonal gradients in both the horizontal and vertical directions. This means that it may be possible to use the system to measure the print quality of solids. The investigation in this section was designed to achieve this. The next paragraph describes the procedure for the experimental work.

The image analysis system was given the task of differentiating between two images of the letter 'a' produced in the Swiss "san serif" font and 16 pt in size. This text character was printed using a Hewlett Packard 4M printer in 600dpi mode. This laser print was then photocopied using a Sharp SF2022 photocopier. The photocopied image was designated as the original from which a further copy was made. By successively repeating this process, a degradation affect for a series of printed images was created. The first generation and fifth generation photocopies were selected for analysis. Each of these images was repeatedly placed in the display 156 times at different orientations and positions. An example of this procedure is shown in **figure 6.17**.

Outputs 12, 13, 14, 15, 16, 17, 24, and 25 from the pre-processing program were able to produce a neural network model that could classify the two images. The outputs selected measured the tonal gradient, and contrast characteristics, and position of the text characters on the screen. This is illustrated in **figure 6.17**. The neural network used to classify the images employed the Levenberg- Marquart algorithm with the following parameters.

Number of neurons in the hidden layer = 8

Transfer functions: logsig for the hidden layer and logsig for the output layer

Network learning rate = 0. 1

Error Goal = 0.001

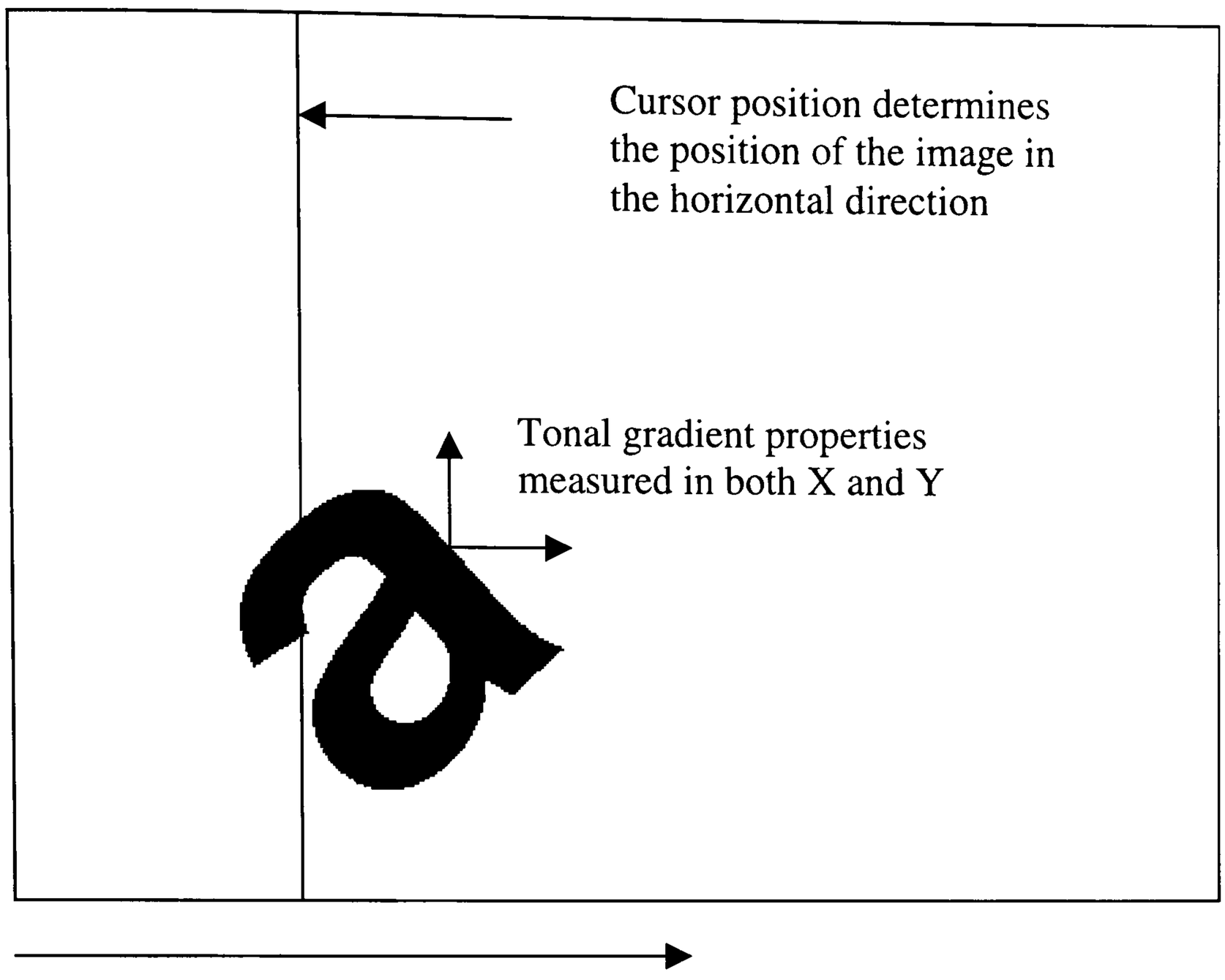


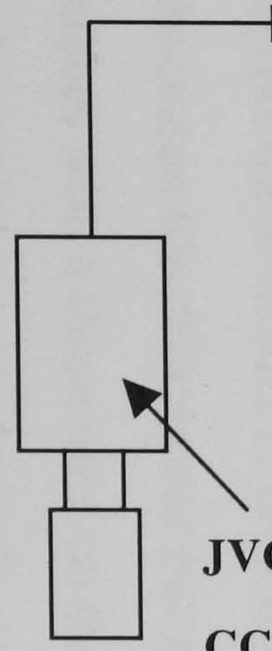
Figure 6.17. An image of a text character was repeatedly placed at random in the screen and the measurements shown in the captions were taken. This process was repeated for another character that was reproduced from the original.

A set of results that have been produced with the network is shown in **table 6.14**. To reduce the classification errors shown in the table using the current image analysis system, other combinations of pre-processing outputs can be tried. Other possibilities include changing the parameters of the network or using different network paradigms. However these methods have already been used in acquiring the data shown in **table 6.14**. It took 20 different models and a time period of 2 days to produce these classifications and any further network adjustment may not yield better results.

Another way of increasing the accuracy would be to increase the number of measurements taken by the pre-processor for each image. If this has a positive effect on the classification success rate, it would mean that increasing the number of measurements taken improves the accuracy of the interpolation by the network. A problem with increasing the number of measurements is that it is also time consuming. It takes approximately 5 minutes to make each of the 156 sets of measurements in the test already carried out. In this and the preceding paragraph there has been an emphasis on time. This is seen as a critical factor in any further work and will be discussed in the concluding part of this thesis.

Validation result for photocopy version1	Validation result for photocopy version5	Test set results for photocopy version1	Test set results for photocopy version5
1.0000	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
1.0000	**0.9990**	1.0000	0.0000
1.0000	0.0001	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
1.0000	0.0000	0.9997	**1.0000**
0.9467	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0007
1.0000	0.0000	1.0000	0.0061
0.0000	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
1.0000	0.2059	0.9995	0.0000
0.0000	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
0.0000	0.0000	1.0000	0.0000
1.0000	0.0000	0.9994	0.0000
1.0000	0.0000	1.0000	**0.9994**
1.0000	0.0000	1.0000	0.0000
0.1141	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
0.0000	0.0032	**0.0021**	0.0000
0.9939	0.0000	1.0000	0.0000
1.0000	0.0000	1.0000	0.0000
0.0000	0.0000	1.0000	0.0000
1.0000	**1.0000**	1.0000	0.0141
1.0000	0.0000	1.0000	0.0000
1.0000	0.0164	1.0000	0.0000
0.0000	0.0001	1.0000	0.0000
7 errors out of 30 examples	2 errors out of 30 examples	1 error out of 30 examples	2 errors out of 30 examples

Table 6.14. Classification results for the two text characters that were placed at random orientations and positions on the screen during analysis. The classification errors are in bold.



The Matrox Meteor frame capture board produces an image matrix with the dimensions 768 x 576 pixels.

The Matrox MIL software acts as a buffer between the Meteor hardware and the CPU and produces visual display on the monitor The image data is directly assessable from MIL using Visual Basic version 5.

The pre-processing program written in Visual Basic version 5 is used to extract features from the image matrix. The program converts the background and the image in the display into a serial frequency pattern. It analyses each line of the display individually and obtains information about the image by detecting its edges using a process known as thresholding.

The pre-processing program uses the 768x576 pixels in the image matrix to produce 52 output variables that contain print quality information about the image.

The Matlab neural network toolbox is used to produce print quality models from the pre-processing data. Two procedures were selected from the toolbox for this task. They were the backpropagation method using the Levenberg-Marquart algorithm and the radial basis function.

Decisior

Figure 7.1. A flow chart showing the components of the image analysis system used in this investigation.

7.0 Conclusions and further work

In this investigation an image analysis system has been developed that can simulate print quality perception. A description of the development stages of this image analysis system, which is shown in **figure 7.1** on the preceding page, was given in **sections 5.0-5.10**. In **sections 6.0-6.9** a sequence of successful results was produced for the testing procedure of the image analysis system. The testing procedure started with the objective analysis of the 1 cm squares, using the image analysis system to differentiate between printing processes that produced the squares and finished with the image analysis system performing a simulation of print quality perception for a simple image of a face. **Figure 7.2** summarises the sequence of tests carried out in the preceding chapter.

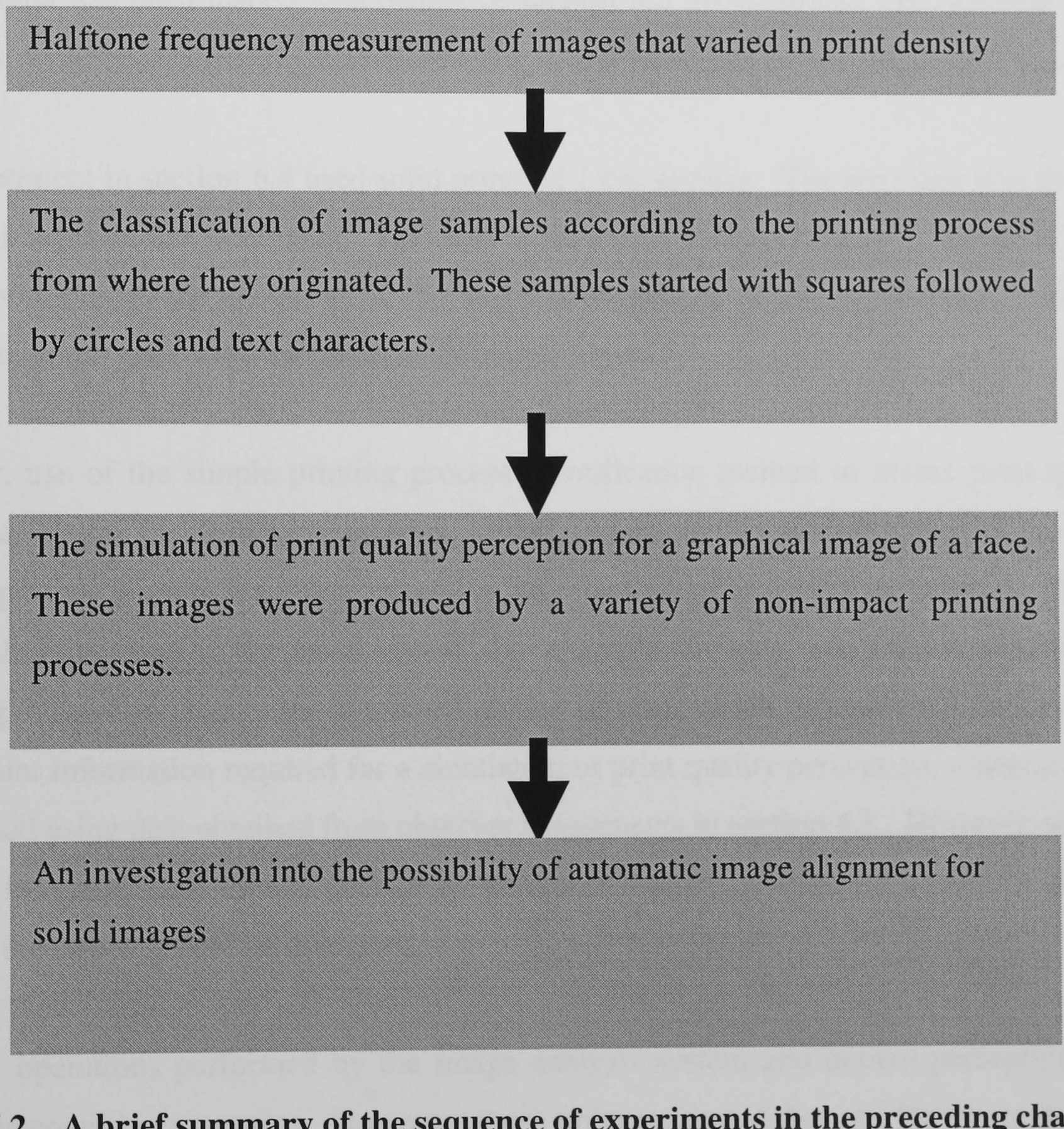


Figure 7.2. A brief summary of the sequence of experiments in the preceding chapter.

This section interprets these results in relation to the principles of artificial intelligence and the Turing Test. It discusses the differences between the image analysis system and human visual perception. It finishes with a discussion on the potential that the system has for producing new technological products in the area of printing. These products are devices that can measure the

degradation of print produced from photocopiers, and a system that can identify the process from which a print sample originates.

7.1 A comparison between human print quality perception and the image analysis system

The identification of printing processes is considered as an important prelude to the investigation into the simulation of print quality perception. The reason for this is that the identification of printing processes can provide an objective measurement of print quality. If prints from two different processes are perceived to differ in print quality and the image analysis system can differentiate between these two processes, it follows that a print quality measurement has been made. The results in **section 6.4** demonstrated that different printing processes could be successfully classified using neural networks for the image of a square.

The experiment in **section 6.4** used solid prints of 1 cm squares. The next step was to classify halftone images in the same way. This was achieved in the experiments of **sections 6.5-6.7**, which also included the images of circles and text characters. These experiments showed that the system could generalise and analyse different shapes.

However, use of the simple printing process identification method to assess print quality is unsatisfactory for the following reasons. Firstly, some of the variables that differentiate a printing process from one another may have no effect on print quality perception. Secondly, print quality may vary in the process itself. For example, the tones and resolution settings of a printing process can change the perceived quality of print which it produces. To extract only the relevant information required for a simulation of print quality perception, a neural network was trained using data obtained from observer assessments in **section 6.8**. However, the neural network interprets, and human perception perceives images in different ways. A difference between the two will now be discussed.

Both the operations performed by the image analysis system and human perception can be viewed as cognitive processes. However, these processes are different. For example, human cognition perceives **figure 6.1** as a series of sixteen squares of different densities but the neural network interprets it as a group of sixteen different objects. This can be understood using **figures 5.3** and **5.4** and a review of **sections 5.0-5.10** on the principles of the pre-processing algorithm. **Figure 5.3** is a magnified image of a 1 cm halftone square produced by a laser printer. Squares that are of a different density will possess a different pattern or the same pattern of different dimensions. The pre-processing algorithm extracts information from the

magnified image in the form of a frequency pattern. It was not primarily designed to recognise shapes. Therefore when different halftone squares produced by different printing processes are assessed by the image analysis system, they will be treated as two different sets of objects. Human perception alone cannot resolve the halftone frequency patterns in the same detail as the image analysis system, whereas human perception can recognise shapes better than the image analysis system.

This means that the print quality assessment rule defined in **section 4.2** adopted in this investigation has been broken by the image analysis system. This stated that images used for the assessment of print quality must be reproduced from the same original. This can be viewed as a paradox since the halftone pattern of reproductions are frequently different from one another.

The paradox described above can be resolved by considering the following hypothesis. Humans perceive print quality by recognising the objects on an image as well as assessing the quality of the print. Therefore, the definition of print quality used in **section 4.2** is valid if all the images in an assessment trial are assigned to a class of images. In the case of **figure 6.1** this class can be labelled as 1 cm squares. If the print quality attributes of these squares are different then the squares can be divided into sub-classes. For example, they can be divided into sub-classes that represent the printing process they originated from or whether they are good or bad images according to the definition given in **section 4.2**. It follows that human cognition can give each image more than one label. In this case, one label identifies the object as a square and another label classifies it according to print quality. Therefore it can be deduced that a group of squares also can be viewed as a group of different objects.

The image analysis system can be interpreted as a system that produces a print quality label for an image. It cannot however recognise other object features. For example it cannot recognise the image of a square as a geometric shape or an image of a letter q as a text character. Therefore the label that provides this classification must be supplied to the image analysis system by human intelligence before the system can make a print quality assessment. This can be illustrated by first considering the experimental procedure of **section 6.8**.

In this experiment an image of a round face was produced on a computer screen. The computer file for this image can be treated as an original from which hardcopy reproductions can be produced. An observer assessed the print quality of the images of the face. These results were used as training, validation and test sets for neural networks. Both the radial basis

function and backpropagation neural network used in this investigation produced one error out of twenty-four classification attempts.

A question that was not answered in the test was whether the network could generalise in the assessment of other image objects without further training. For example, if a set of images of a square face were used in another observer assessment trial, would the original neural network model be able to predict the results for the square face? If this answer is no, another neural network model would be necessary for this image.

Evidence for the requirement of different neural network models for different image objects exists in the experiment described in **section 6.9**. In this section it was theoretically demonstrated that the alignment of an image under the camera was an important factor in the measurement of print quality variables using the image analysis system. An interpretation of this problem is that if an image is rotated beneath the camera, the pre-processing algorithm treats this rotated image as a new image.

Neural networks can be used to recognise different objects so that an appropriate evaluation of the pre-processed image data can be made. This was achieved in **section 6.9**, when an image placed at different orientations was treated as different images. The function of the neural network here can be considered as an orientation recognition step and a print quality step. Further work should include improving the object recognition capability of the image analysis system by solving the orientation problem for the halftone patterns.

The processing of the data for all the experiments described in **section 6.2-6.9** was very slow when compared to human print quality perception. It takes the image analysis system about seven minutes to compare the two face images described in **section 6.8** which would take an observer a second. This is without taking into consideration the fact that the system is restricted to analysing the face at a fixed orientation.

Section 6.8 has demonstrated that the image analysis can assess images of the same object at a correctly aligned orientation to the camera lens. **Section 6.9** and this section have shown that the system cannot generalise and assess the print quality for all images with equal accuracy. This was` deduced from the fact that the orientation of the image in relation to the display window was an important variable. The orientation problem for solid images was investigated successfully in **section 6.9** for a text character. This can be viewed as a partial solution to the generalisation problem, since halftones were excluded.

If the neuronal structures in the brain operate in the same way as artificial neural networks, a possible explanation can be proposed as to why artificial neural networks are poorer at object recognition than human perception. A possible explanation is that the number of neurons in the brain for the task of object recognition is far greater than the number of neurons in the artificial neural network. A greater number of neurons would enable more labels for the images to be provided. If this were the case, then the neural network employed by the image analysis system would require many millions of neurons, each with tens of thousands of interconnections.

7.2 The Turing test for print quality

The results in **section 6.9** show that the neural network can produce the same print quality assessments as made by an observer. The error ratio was 1 in 24 assessments. This error ratio can be interpreted in the following ways.

One view is that adjusting the number of observer assessments used in the neural network training set or changing the network itself can reduce the error ratio. The problem with this approach is that if additional assessments are made by the observer there is no guarantee that the network predictions will be free of errors. This is because observers can also make decisions that sometimes require corrections. This has been demonstrated in **section 6.9** where adjustments were made to the network training set. This is essential since any contradictions in the training set will stop a neural network from converging to a solution. The image analysis output can also be used to correct any mistakes made by observers in any future assessments. The observer errors can in this case be viewed as human errors that have been corrected by a machine.

The evaluation given above for the experiment in **section 6.9** is highly deterministic in the sense that the objective is to eliminate all possible errors. Another way in which the results can be viewed is to accept errors made by the network within certain limits. These limits would be set by a Turing test for the print quality assessment domain. If the human and machine assessments for the same set of images contain results that do not match, both can be considered as producing errors. A second, different observer can be asked to classify these errors according to whether they originate from the image analysis system or the first observer. If the second observer cannot differentiate between the human and machine errors, to the second observer all the assessments are perceived as originating from the same source.

It can be concluded from the results of **section 6.9** that the image analysis system classified the print quality from an observer with an accuracy of 23 out of 24 for the face image. To produce a system that can simulate human print quality perception, the discrepancies must be evaluated. Therefore more assessments must be performed so that more errors are produced. If these errors cannot be recognised as machine or human errors by a second observer who is independent of the initial assessments then the second observer cannot differentiate between the two sets of assessments.

The overall approach in this section has been from an objective standpoint. It is a statistical evaluation of the effectiveness of the image analysis system's capability of simulating human print quality perception. An alternative view that can be adopted is the fuzzy inference paradigm. In this approach the observer who makes the print quality assessments also operates the image analysis system. The observer assesses the machine's capabilities from a subjective viewpoint using fuzzy labels. For example these labels could be weak, poor, good or excellent. The fuzzy inference approach may also assist in any further developments of the system. This concept was previously discussed in **section 4.9**. For example, the observer might notice that the system is excellent at determining tonal contrast. The observer may also notice features of the image analysis system that are incompatible with human perception. These could be the alignment problem that was previously discussed and the time the system requires to make an assessment. It takes three to four minutes for the system to make an assessment that an observer can make in less than ten seconds. **Section 7.3**, discusses how in theory the alignment problem can be solved. To achieve this solution in practice, the speed of the pre-processing program needs to be increased. **Section 7.4** discusses improvements that can be made to improve the computational efficiency of the software.

7.3 A possible solution to the alignment problem

The results of the test in **section 6.9** cannot be fully considered as the result of an artificially intelligent system that can pass the Turing test, since human cognitive skills are required to align the image in the display window. This problem was partially solved in **section 6.10** for a text character. A neural network found a relationship between features in the gradients in perpendicular directions and properties of the tonal distribution of the text character that was independent to its orientation in the display window. However, solving the problem for the text character did not take into account halftone images. A way to solve to the full problem will be suggested in this section.

In theory computing the values for output 1 as the image is scanned at different angles, can solve the alignment problem for halftone images. **Figure 7.3** illustrates this process. An image is placed at a random angle beneath the camera. The frame capture board converts the image into an m x n matrix array. The first scan moves in the direction 0 to n before m is incremented by 1. This process is repeated until the end of the matrix is reached. In **figure 7.3 a** the direction of the scan is not correctly aligned with the image. This misalignment can be detected by the image analysis system using output 1 of the pre-processing program. If the image is correctly aligned as in **figure 7.3 d**, the value of output 1 will be a maximum value. This can be understood using **figure 5.8** in **section 5.5**. A maximum can be found by rotating the scan of the matrix by small fixed increments of approximately 1 degree. This estimate was obtained by an assessment of the error that was produced by manual misalignment. A maximum must exist within 90 degrees. Output 1 may need to be computed 90 times before a maximum is discovered corresponding to the correct alignment. Therefore more data processing time is needed if the automatic alignment were to replace manual alignment. This is only one reason why it is important to make the pre-processing program more computationally efficient. The next section discusses a method that can improve the speed of the pre-processing program.

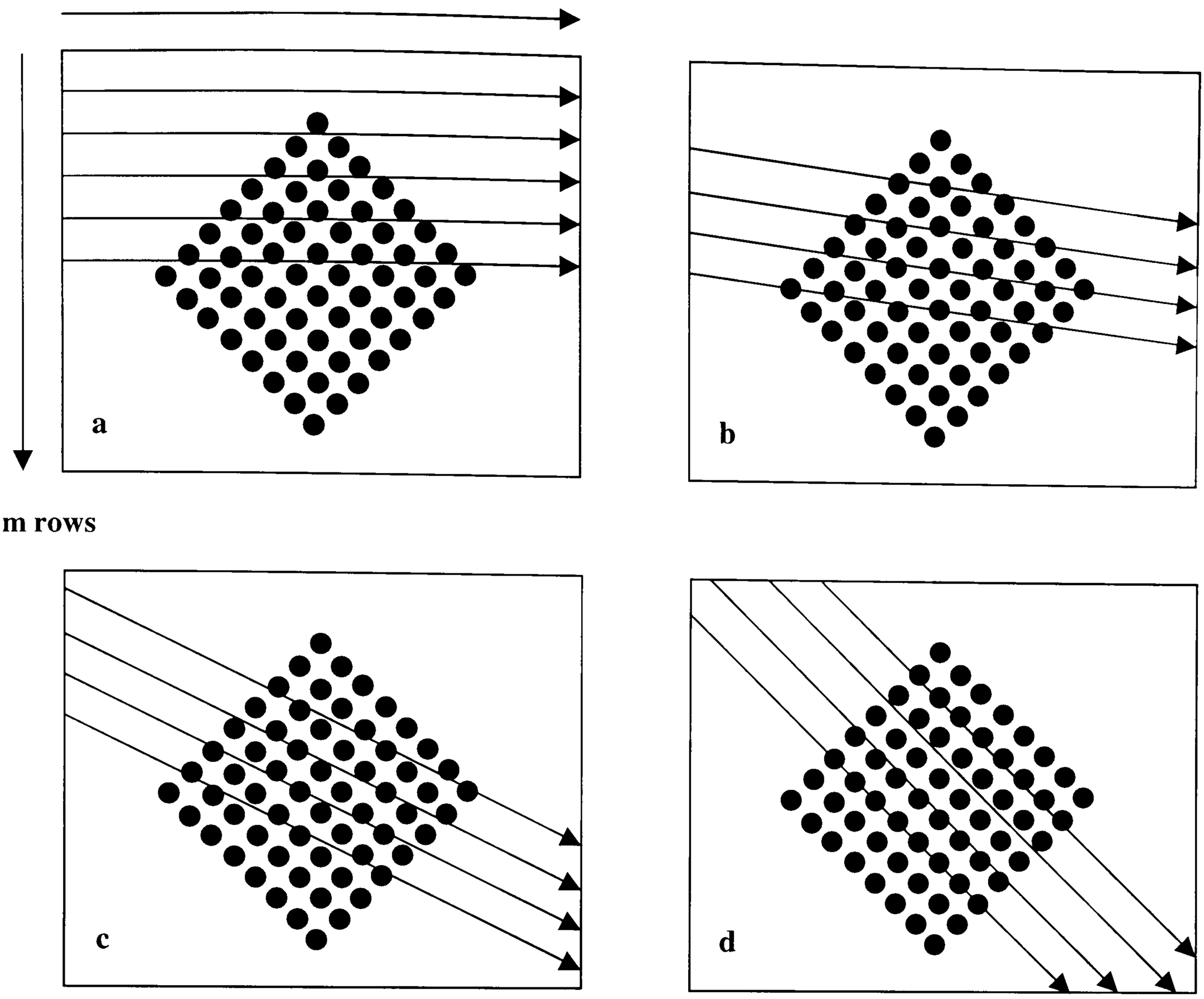


Figure 7.3. An illustration showing how a halftone frequency pattern can be measured automatically.

7.4 Improving the computational speed of the pre-processing program

The reason why Visual Basic was used for the pre-processing program has been discussed in **section 5.6**. The main reason for the selection of Visual Basic was the ease in which programs can be written using it. This means that different lines of thought can be implemented and tested quickly. However the computational speed of the language is slow. Once a program is developed the advantage stated above is lost. Microsoft Visual C++ can be used to increase the speed of the image analysis system. Microsoft Visual C++ can be directly implemented into the current image analysis system since it is compatible with the Matrox MIL libraries, the software that is provided with the Matrox Meteor frame capture hardware.

Appendix 1 is a hardcopy of the program used to obtain the results in **section 6**. It can be seen that the program consists of loops within loops that perform millions of calculations. The efficiency of Visual Basic version 5 can be directly compared to Microsoft Visual C++ version1 using the code below. The first program has been written in Basic and the second set

in C++. The code in both programs performs the same task. They increment a counter in integer steps of one, two hundred million times, and print out the result. Both programs were executed on a 233MHz Pentium processor and the time taken was recorded. The Visual Basic version 5 program used the p code compiler and the C code used the Microsoft C++ code compiler for the test. It was shown that the C code is eight times faster. Also the results that were obtained in **sections 6.0-6.9** used a 200MHz Pentium processor. This was the fastest processor available when the pre-processor was under development. By the time the results in **sections 6.0-6.9** were obtained 400 MHz Pentium processors were available. This will increase the program speed by a factor of sixteen in total. These are not the only programming considerations that can alter the execution time required for the pre-processing program. Running a C program in DOS instead of Windows will also increase the speed.

The first program in Basic code

```
Dim Increment, Number As Long
For Increment = 1 To 200000000
Number = Number + 1
Next Increment
Print Number
```

Run time required using the Microsoft Visual Basic version 5 p code compiler and 233MHz Pentium processor = 85 seconds

The second program in C code

```
#include <stdio.h>
void main()

{
long int increment;
long int number;
number=0;
for(increment=1; increment<=200000000;
increment= increment+1)
number= number+1;
printf("%li\n",number);

}
```

Run time required using the Microsoft Visual C++ compiler and 200MHz processor = 8 seconds

7.5 Enhancing the pre-processing program

Section 5.3 described the program that was used to obtain the results in **section 6.0-6.9.** It was previously stated in **section 5.0** that the software adopts a modular approach that allows the measurement of new image features to be integrated easily into the system. This is demonstrated by the fact that not all the outputs were used to obtain the results in **section 6.0-6.9**. An example of this are the outputs 43 to 52, these are frequency filters which were under development when the measurements were being taken.

7.5.1 The use of frequency filters in the pre-processing program

These outputs are frequency filters that measure the smoothness of the edge of an image. An image edge that is ragged will produce spatial irregularities that can be converted into a frequency pattern. Outputs 43 to 52 detect the changes in the direction of the gradients. This is illustrated in **figures 7.4 and 7.5**.

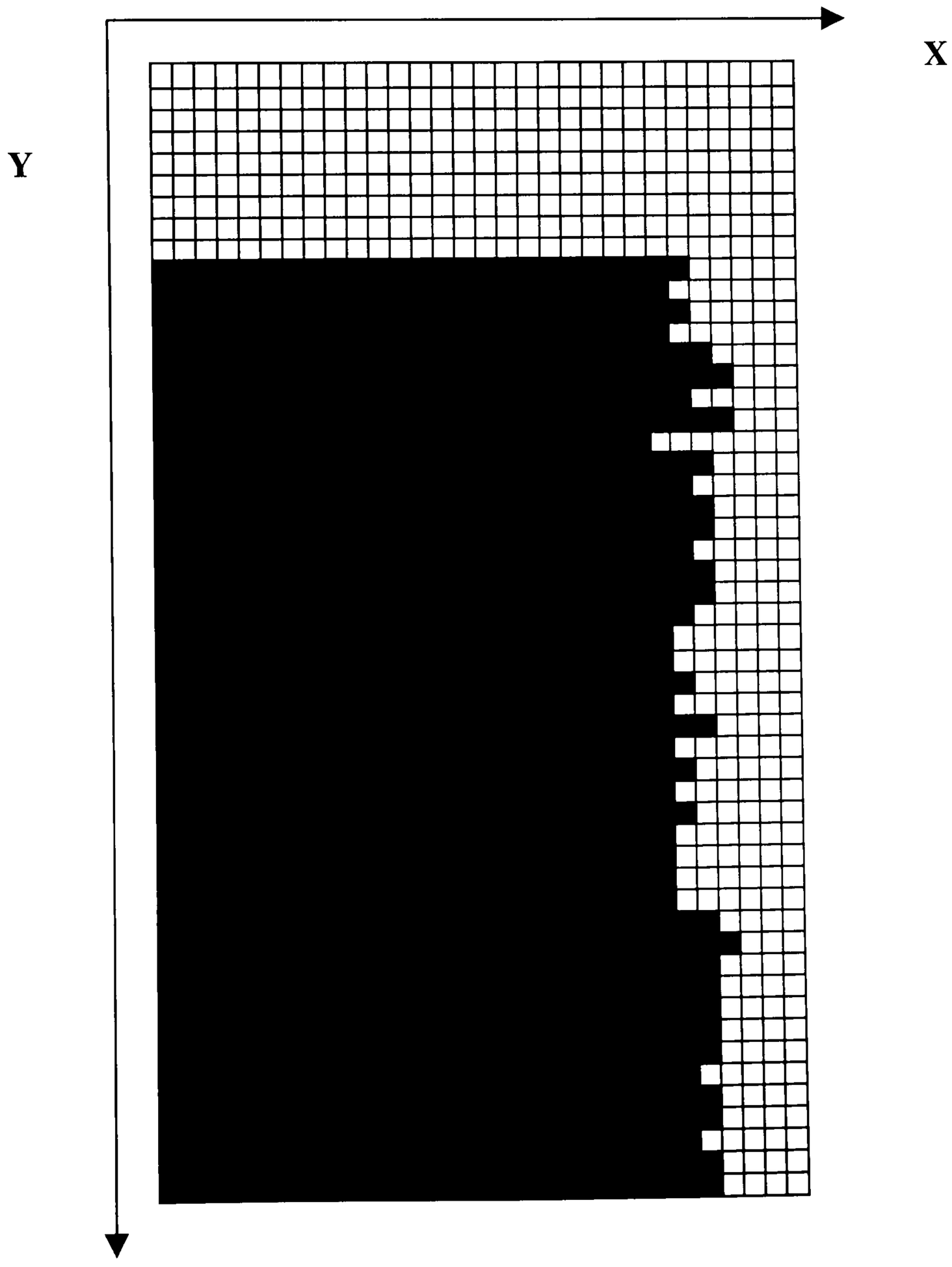


Figure 7.4. An illustration of part of an irregular edge.

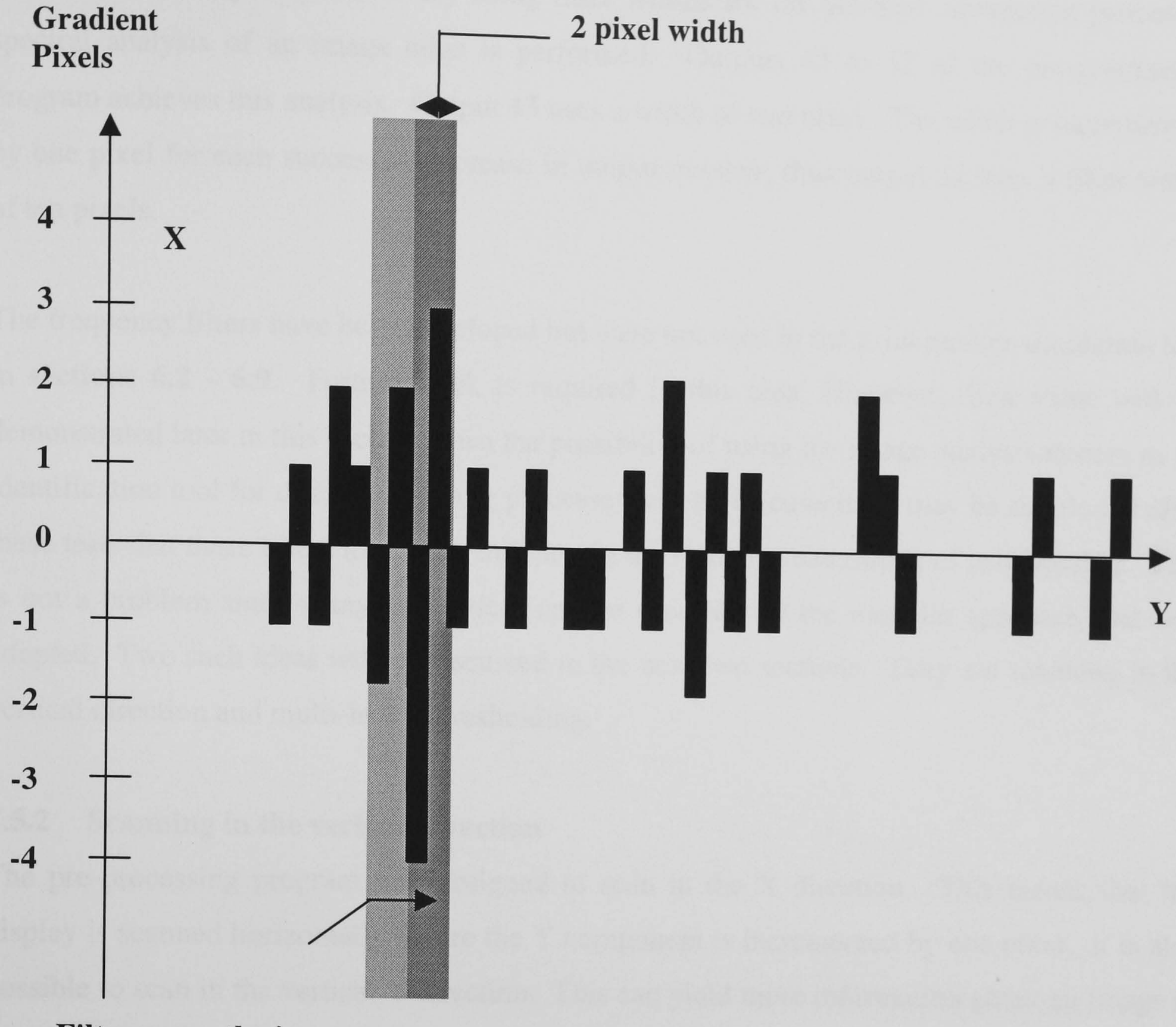


Figure 7.5. A plot of the magnitude of gradients along the irregular edge shown in figure 7.4.

Figure 7.4 is a representation of an edge of an image. **Figure 7.5** is a plot of the gradients against distance for the uneven edge that is shown in **figure 7.4**. It has been rotated by 90 degrees. It can be seen from both of these figures that both the amplitude and the frequency can have an affect on print quality perception. The following procedure was written to investigate these factors.

The two adjacent grey bars of different shades in **figure 7.5** represents the scan of the edge along the Y-axis. Each shaded region computes the sum of the gradient values in its region. In the diagram the shaded region is two pixels in width, but any value can be used. The total of the gradient values for the two shaded regions are compared with each other. If the difference from this comparison exceeds a pre-set threshold value, a counter is incremented by 1. The

grey bars are moved one pixel along the Y-axis and then the process is repeated. Thus the grey bars acts as a frequency filter. By using filter widths for the gradient summation process a spectral analysis of an image edge is performed. Outputs 43 to 52 of the pre-processing program achieves this analysis. Output 43 uses a width of one pixel. The width is incremented by one pixel for each successive increase in output number, thus output 52 uses a filter width of ten pixels.

The frequency filters have been developed but were not used in the print quality simulation test in **sections 6.2 - 6.9**. Further work is required in this area. However, their value will be demonstrated later in this section when the possibility of using the image analysis system as an identification tool for different printing processes will be discussed. It may be concluded after these tests that these filters have very little use in the machine assessment of print quality. This is not a problem since many other ideas can be tried due to the modular approach that was adopted. Two such ideas will be discussed in the next two sections. They are scanning in the vertical direction and multi-level thresholding.

7.5.2 Scanning in the vertical direction

The pre-processing program was designed to scan in the X direction. This means that the display is scanned horizontally before the Y component is incremented by one pixel. It is also possible to scan in the vertical Y direction. This can yield more information about an image as the following example demonstrates.

The graphs shown below in **figures 7.6 and 7.7** were produced from the analysis of an image of a solid 1cm square produced on the 4M laser printer using the pre-processing program. Two sets of readings were taken. In the first set the square was aligned to the horizontal and eight readings taken. Then the square was rotated 90 degrees and a further set of eight readings taken. Rotating the image 90 degrees has the same effect as changing the scanning direction. **Figures 7.6 and 7.7** show that additional information can be obtained from an image by changing the scanning direction. These differences may be shown to be too small to affect the overall print quality of an image. However even if this is the case, it will be shown later that by analysing the image in perpendicular directions may yield information about the origins of a printed image.

Changing the scanning direction requires that the X and Y variables in the pre-processing program shown in **appendix 1** be interchanged if automation of this process is required. It

would be simpler to use two pre-processing programs to analyse the images at the two different angles. To incorporate both sets of readings into one program would be more difficult.

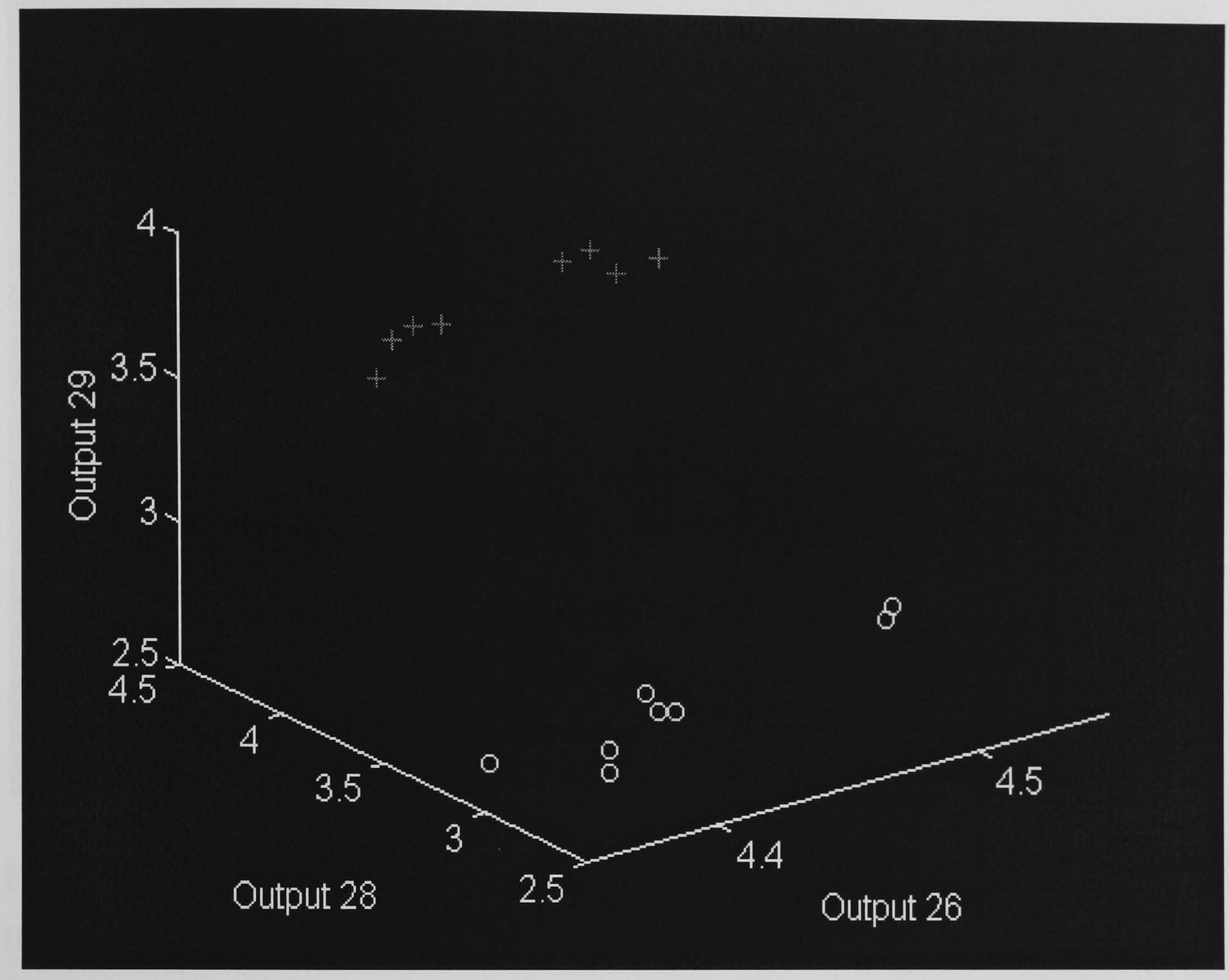


Figure 7.6. Graph showing the effect of rotating an image on the outputs 26, 28 and 29 of the pre-processing program. The Os represent data from the image of the square aligned to the horizontal. The +s represent readings that were taken when the image of the square was rotated 90 degrees.

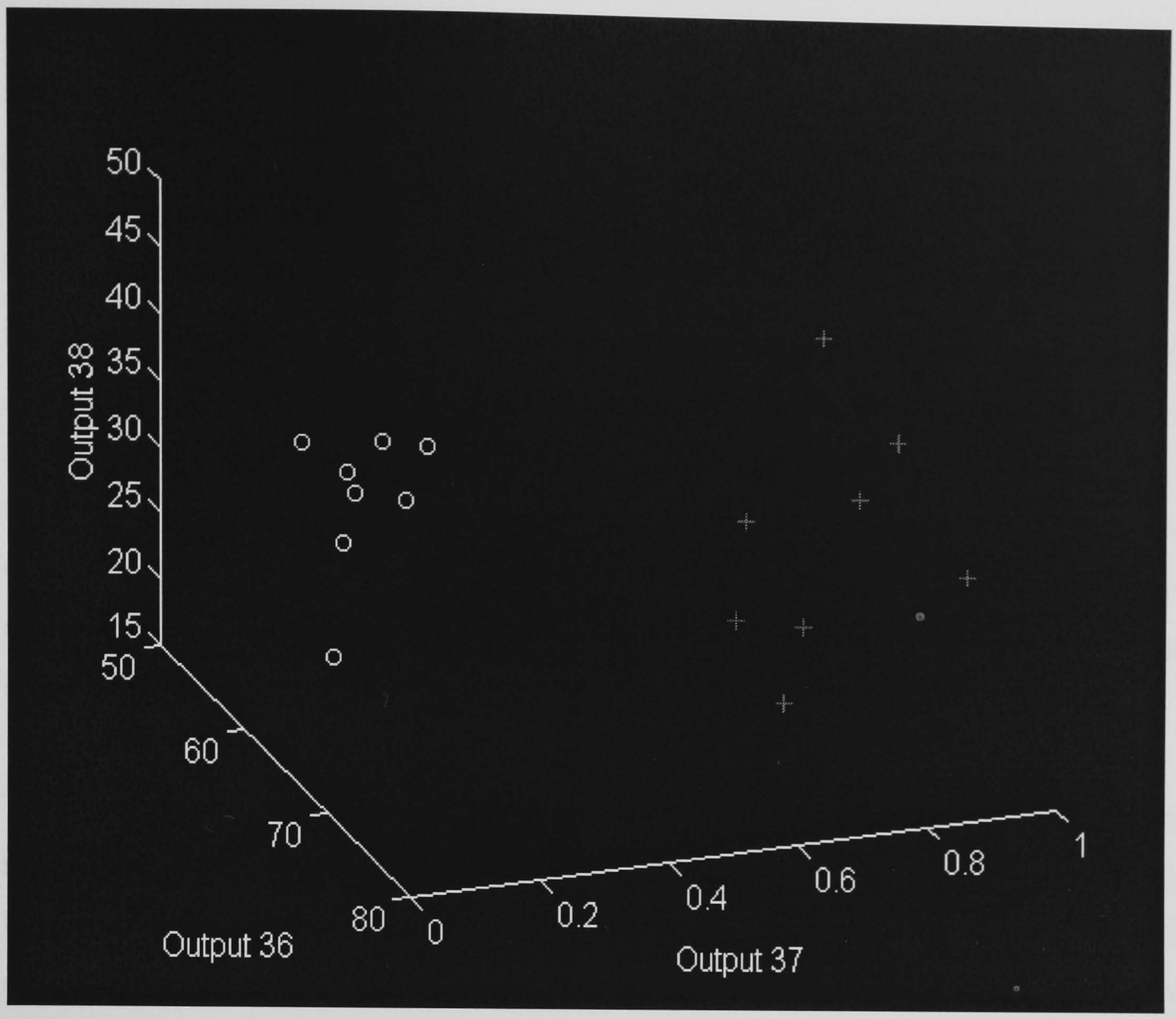


Figure 7.7. Graph showing the effect of rotating an image on outputs 36, 37 and 38 of the pre-processing program. The Os represent data from the image of the square aligned to the horizontal. The +s represent readings that were taken when the image of the square was rotated 90 degrees.

7.5.3 Multi-level thresholding

The principle that the pre-processing program uses to detect the presence of an image is edge detection. A process known as thresholding achieves this. When the gradient between the background and image exceeds a pre-set threshold value a marker is placed in the image matrix that denotes the presence of an image. The position of the image can be shifted if the pre-set threshold value is altered. This would also change the values of the pre-processor outputs that use the image position for their calculations. Examples of this are the gradient measurements from outputs 8, 10, 11, 26, 28 and 29 described in **section 5.5**. The pre-processing program was designed to multi-threshold. This means that it can automatically process the data for a range of threshold values. However, this option was not used here because of the time required for this operation.

So far, the discussion on improving the system has centred on developing and modifying the software. The next section deals with a possible hardware improvement that could be made.

7.5.4 The use of a camera that produces less noise and a higher resolution

The camera that was used to obtain the results in this investigation has demonstrated that an image analysis system utilising a CCD camera and a digital computer can in principle assess print quality and identify the printing process from where an image originates. However, these assessments have only been for small images with dimensions of approximately 1 cm^2 and a tonal range of 0-255. To increase the versatility of the system the hardware must be improved. One way in which to improve the hardware is to employ a camera with a higher spatial and tonal resolution and a higher signal to noise ratio. This would enable larger images to be analysed.

The camera used in this investigation was a JVC TK – S350 monochrome camera, with a resolution of 753 (horizontal) x 582(vertical) pixels and an 8 bit tonal resolution. An improvement could be made by using a camera such as the Hamamatsu C4880 – 31. This camera has a resolution of 1024 x 1024 pixels and an16 bit tonal resolution. To reduce thermal noise, it uses a water cooled Peltier system to reduce the temperature of the CCD to –50 ^{o}C.

Another improvement would be the replacement of the zoom lens with a fixed focal length lens since the magnification required has now been established. Using a fixed focal length lens will reduce the amount of image distortion caused by the optical system. The importance in acquiring more precise measurements will be highlighted in **sections 7.6 and 7.7**, which deal with more practical applications of the image analysis system.

7.6 Using the image analysis system to test the print quality of photocopiers

An automated print quality assessment machine for photocopiers can be developed on the lines described in **section 6.8**. This is possible since print quality variables of images of text characters can be measured using the image analysis system, although there are practical problems that still need to be solved. These problems are outlined below.

Section 6.8 demonstrated that the degradation effect that was produced by the successive photocopying of a photocopied image could be measured. This is illustrated in **figure 7.8** below. However only photocopies versions 1 and 5 were compared. The reasons why this

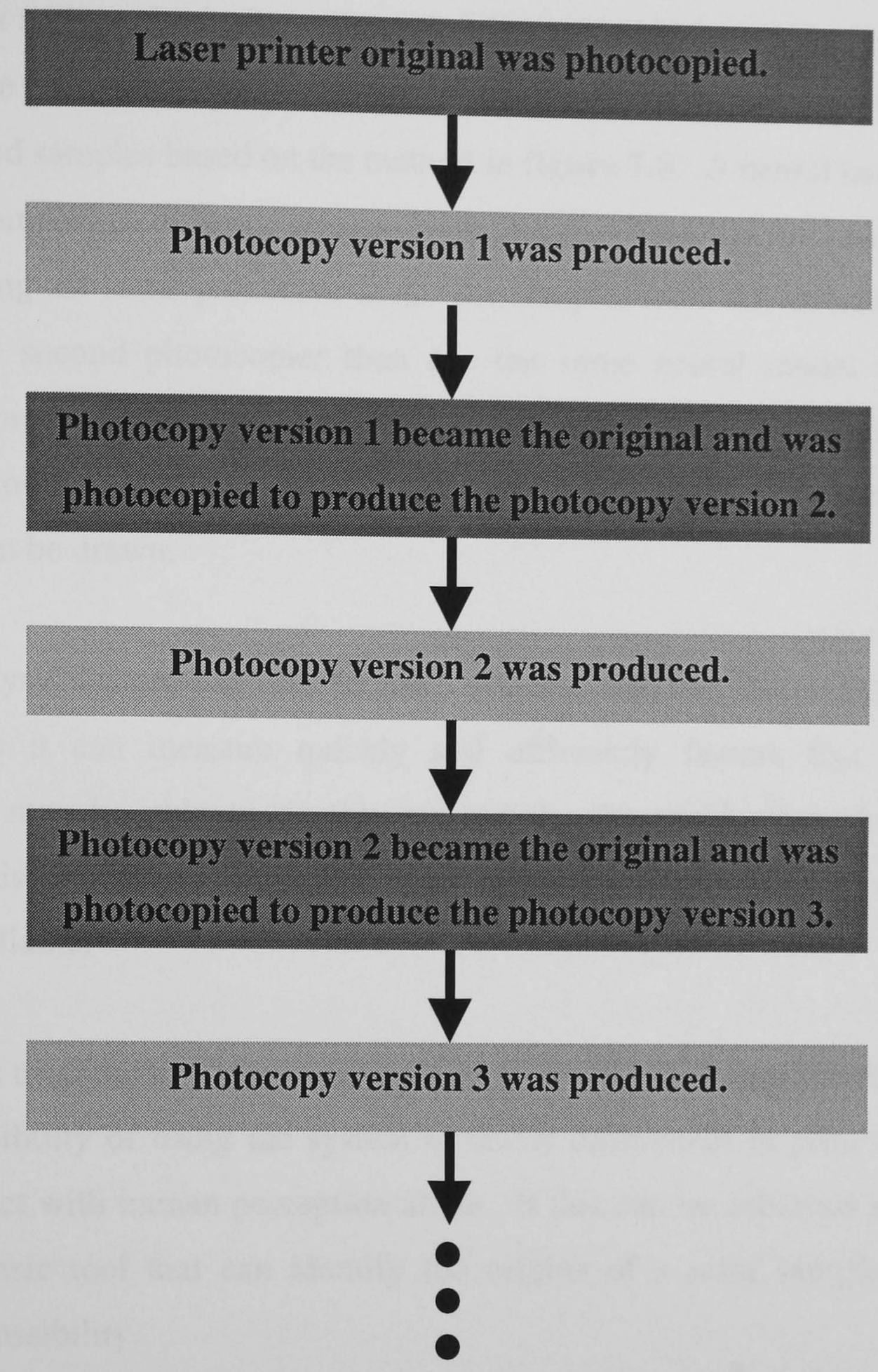


Figure 7.8. Flowchart showing how the print quality was purposely degraded in the experiment in section 6.9.

procedure was chosen were that firstly, it was the initial test to establish that the methodology would work, the accuracy and the sensitivity could be increased at a later stage. Secondly, the main aim of the test was to see whether the two images could be distinguished regardless of their orientation to the camera. Therefore the sensitivity needs to be improved so that smaller variations in the degradation of print quality can be detected and without producing the errors displayed in **table 6.14**.

A definition of print degradation must also be produced that can be agreed upon. This would be difficult if different makes of photocopier were to be compared since manufacturers that received a poor rating would dispute the claims of the image analysis system. The assessments should be independent of any subjective analysis, since subjectivity is where the dispute would

be centred. It is suggested to base an objective measurement of reproduction faithfulness on edge raggedness and tonal contrast. This can be achieved by using the procedure described in **section 6.9**. The principle employed here is to use a reference photocopier to produce, say, the first ten degraded samples based on the method in **figure 7.8**. A neural network model that can classify these samples as different images is employed. Samples are produced from a second photocopier using the same procedure as for the samples from the reference photocopier. The prints from the second photocopier then use the same neural model as for the reference photocopier to produce a second set of classification data. Then by comparing the two sets of neural network output data conclusions about the relative fidelity of reproduction for the two photocopiers can be drawn.

The image analysis system can also be used as a research and development tool for printing processes since it can measure quickly and efficiently factors that affect print quality perception. It may be able to do this beyond the threshold where humans can perceive differences. This last factor is important if the printing process needs to be over-engineered to make it more reliable.

This project has used the image analysis system to identify different printing processes. There is also the possibility of using the system to detect differences in print samples that are not possible to detect with human perception alone. If this can be achieved it may be possible to develop a forensic tool that can identify the origins of a print sample. The next section discusses this possibility.

7.7 Using the image analysis system to identify the origins of print samples

It has been shown that the system can identify different printing processes. However these test have been restricted to identifying two or three processes. For the print identification system to be of practical use it must be able to differentiate between tens of different machines that use the same process. The potential of the system to achieve this task is demonstrated in **figures 7.9-7.12**. The Os, +s, ✳s and Xs represent data from the three laser, three photocopiers, three inkjet and a thermal printing machine respectively. The data was obtained by analysing the images of the squares used in **section 6.2**.

Figures 7.9-7.11 show that the image analysis system can distinguish between the laser, photocopy and inkjet print samples. The different processes occupy different regions of space. The thermal process is placed in its own space between the laser and inkjet processes. The data in **figures 7.9-7.11** originate from direct spatial measurements taken from the images.

The data in **figure 7.12** was processed using the frequency filter outputs of the pre-processing algorithm described earlier in **section 7.5.1**. The results are similar to those shown in **figures 7.9-7.11** except that there is overlap between the classes and the data points for the thermal dye sublimation printer have moved position. The overlap of data points between the printing process classes is likely to be due to random errors in the measurements. A camera that produces less noise with a greater spatial and tonal resolution can reduce the chance of misclassification. Such a camera was described in **section 7.5.4**. The fact that the data points for the thermal dye sublimation has moved position enables it to be distinguished from those produced by the photocopiers shown in **figures 7.9-7.11**.

It has been demonstrated that the system has the potential to identify the source of a print sample by a visual inspection of the pre-processing data. The analysis of a greater number of printing machines may require a more sophisticated technique such as a neural network since more than three dimensions might be required to separate the classes.

The practical application of this technique is in the fact that it may be possible for the image analysis to identify differences in print, which are not possible using humans. For example, human perception may be able to differentiate between an inkjet and a laser print, but it may not be able to detect the differences between two laser prints produced on different machines. The results in **figures 7.9-7.11** demonstrate that it may be possible not only to distinguish between two different machines using the same process, but also to identify exactly the origins of a print from hundreds of different possibilities.

To prove that the system can identify printing processes as described in the paragraph above requires collecting print samples from many printing machines of the same make. This is another advantage of using non-impact printing over other printing methods, which require a much greater effort to produce a print sample. The image analysis system may also need improvements to increase the number of printing machines it can recognise. The number of dimensions in the model may have to be increased to separate the different classes. This increase complexity in the model can be achieved by using a greater spatial and a greater tonal resolution, multilevel thresholding or using a colour camera to obtain spectral reflectance information from the image.

A practical area in which print identification using the image analysis can be applied, is forensic science. A problem that must be addressed is the fact that whether evidence using the

image analysis would be valid in a court of law for the following reasons. Firstly, the results produced by a neural network cannot be fully explained. Secondly, the images require manual alignment when the measurements are taken using the image analysis system. However, even if the results using the image analysis system are shown to be invalid in a court of law, the results can still produce other evidence that can be used for this purpose.

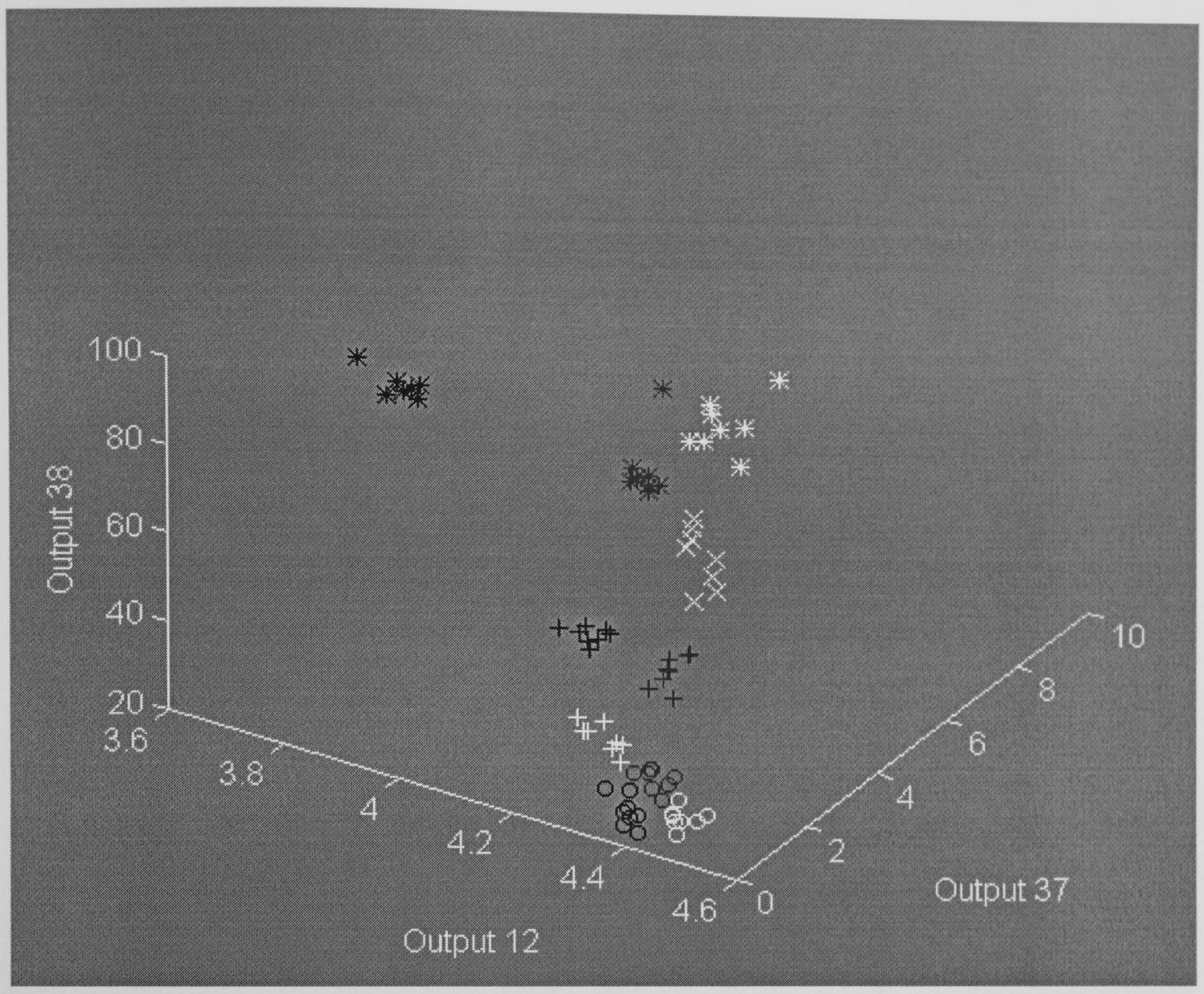


Figure 7.9. A plot of Outputs 12, 37 and 38 for the image of a solid 1cm square. The following processes produced the data points:

Hewlett Packard 4M laser printer	O	**Hewlett Packard Deskjet 1200 inkjet printer**	*
Panasonic KX-P6100 PCL laser printer	O	**Epson Stylus Colour 50 inkjet printer**	*
Kyocera FS-1700 laser printer	O	**Apple Stylewriter II inkjet printer**	*
Sharp SF 2020 photocopier	+	**Fargo Primera dye sublimation printer**	X
Canon photocopier	+		
Oce Bookcopier photocopier	+		

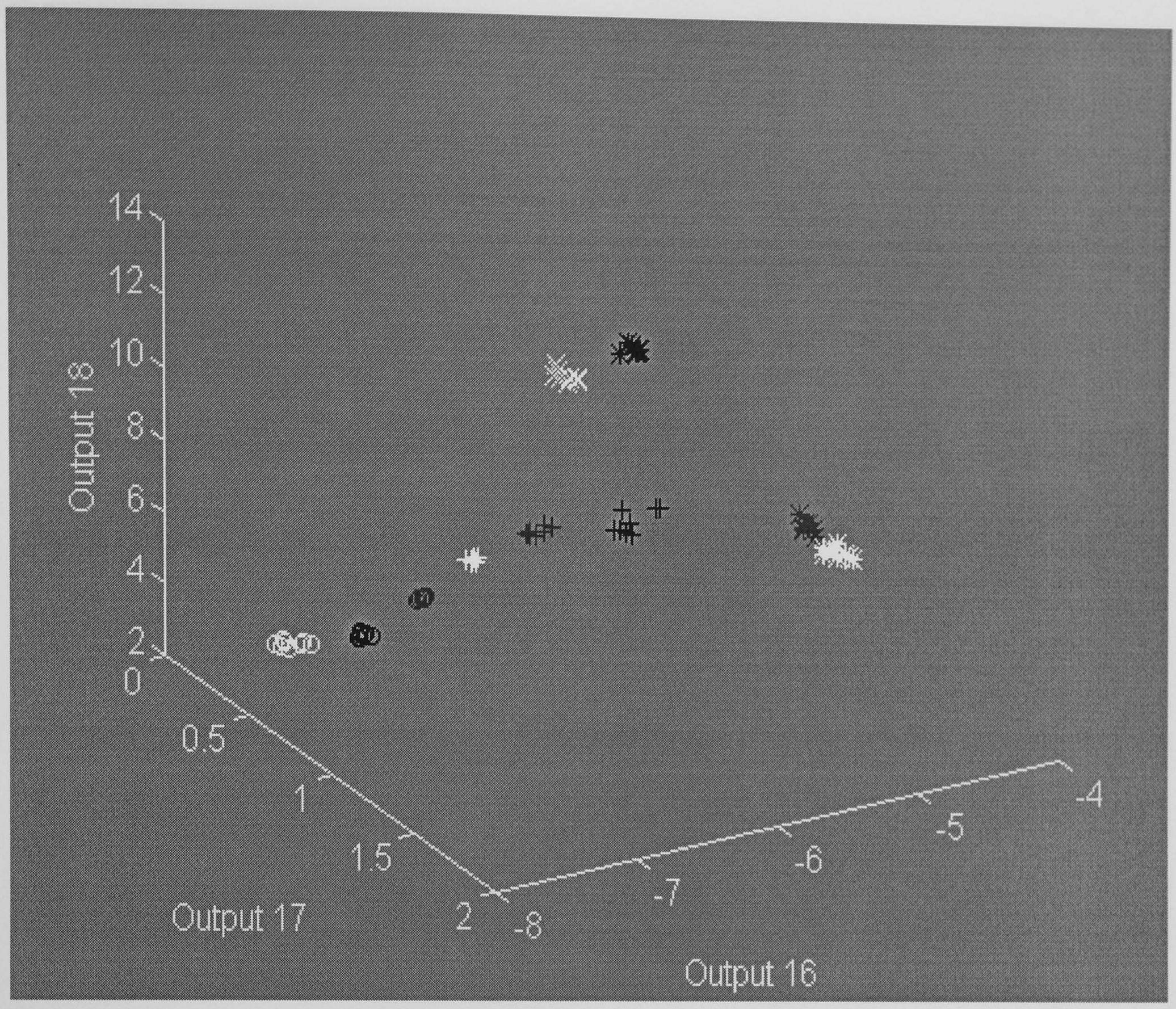


Figure 7.10. A plot of Outputs 16, 17 and 18 for the image of a solid 1cm square. The following processes produced the data points:

Hewlett Packard 4M laser printer	O	**Hewlett Packard Deskjet 1200 inkjet printer**	*
Panasonic KX-P6100 PCL laser printer	O	**Epson Stylus Colour 50 inkjet printer**	*
Kyocera FS-1700 laser printer	O	**Apple Stylewriter II inkjet printer**	*
Sharp SF 2020 photocopier	+	**Fargo Primera dye sublimation printer**	x
Canon photocopier	+		
Oce Bookcopier photocopier	+		

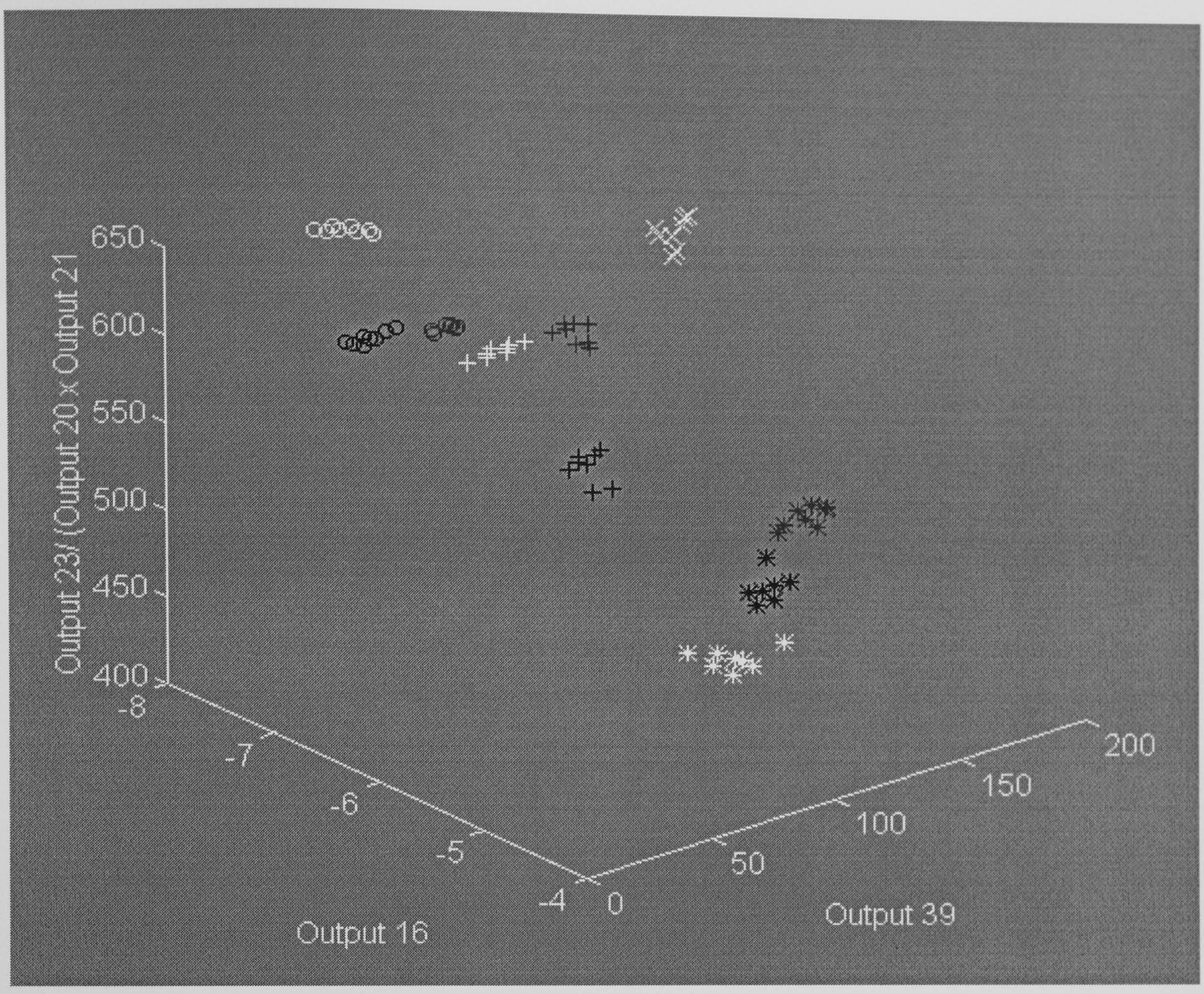


Figure 7.11. A plot of Outputs 16, 39 and Output 23/(Output 20 x Output 21) for the image of a solid 1cm square. The following processes produced the data points:

Hewlett Packard 4M laser printer	O	**Hewlett Packard Deskjet 1200 inkjet printer**	*
Panasonic KX-P6100 PCL laser printer	O	**Epson Stylus Colour 50 inkjet printer**	*
Kyocera FS-1700 laser printer	O	**Apple Stylewriter II inkjet printer**	*
Sharp SF 2020 photocopier	+	**Fargo Primera dye sublimation printer**	x
Canon photocopier	+		
Oce Bookcopier photocopier	+		

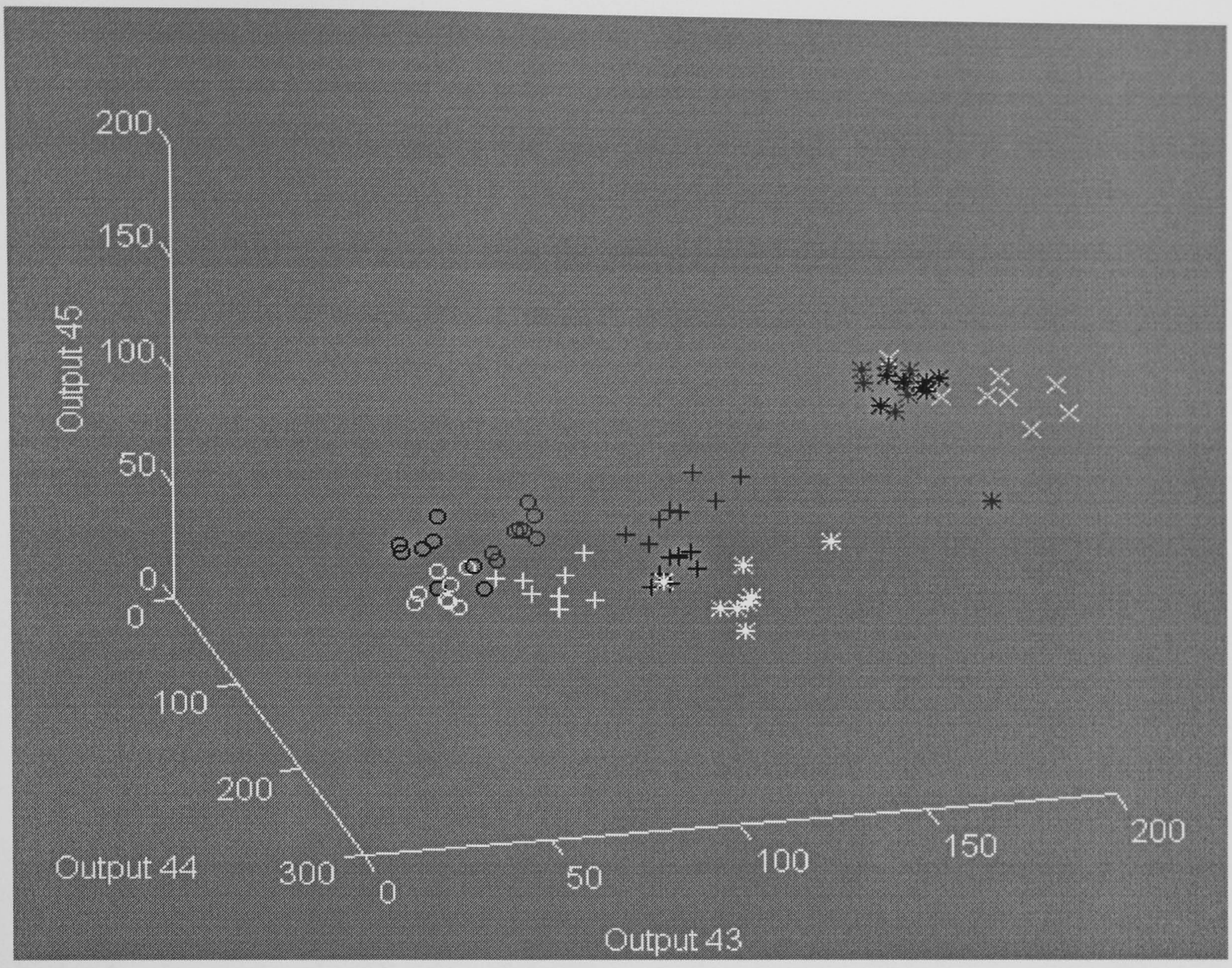


Figure 7.12. A plot of Outputs 43, 44 and 45 for the image of a solid 1cm square. The following processes produced the data points:

Hewlett Packard 4M laser printer	O	**Hewlett Packard Deskjet 1200 inkjet printer**	*
Panasonic KX-P6100 PCL laser printer	O	**Epson Stylus Colour 50 inkjet printer**	*
Kyocera FS-1700 laser printer	O	**Apple Stylewriter II inkjet printer**	*
Sharp SF 2020 photocopier	+	**Fargo Primera dye sublimation printer**	x
Canon photocopier	+		
Oce Bookcopier photocopier	+		

7.8 Improving the neural network search technique

The project has also highlighted the effectiveness of using neural networks to produce multi-dimensional models for print quality that give the correct output response from a set of input variables. However, the production of these models have been searched for on a random heuristical basis and better models from the data may still be awaiting discovery. A better system to find the optimum neural network model is to write an algorithm that looks automatically for the best neural network model. This would work by say, changing the learning rates, the transfer functions and inputs in a systematic way. The information on the weights, biases, number of neurons and pre-processing outputs used for most accurate models would be stored in a data file. This approach requires that the networks used converge quickly. Such networks would employ backpropagation using the Levenberg-Marquart algorithm or radial basis functions.

7.9 Final concluding discussion

This investigation has demonstrated that the image analysis system can simulate human print quality perception of a simple image of a face for a range of non-impact printing processes. Also the investigation has shown that prior knowledge of the printing processes is unimportant in the development and application of the image analysis system. This fact is compatible with the observation that observers do not require specialist information about printing processes before they can make print quality assessments. This simplifies the development process for the image analysis system. Only the print quality variables need to be analysed and not the large number of different printing processes that exist. The complexity of the subject area of printing technology was illustrated in **section 3.2**, which was only a brief introduction to this topic.

However, the simulation requires manual alignment of the images before the measurements can be taken. It has also been shown that print quality variables can be measured for images that are independent of orientation using the image analysis system. The next step would be to increase the computing power of the image analysis system so that human print quality perception can be simulated without the manual alignment of the image. If this is shown to be possible, it would mean that print quality perception is computable since a neural network produces functions of the form $y = f(x)$. This means that print quality assessments using observers that were previously considered subjective can be shown to be objective using the image analysis system.

Although the initial investigation concerned the simulation of print quality, two potential practical applications have emerged from this investigation. One is an instrument that can objectively measure the print quality from a photocopier and the other an instrument that can be used in forensic work. To advance the research in all these areas requires more sensitive instrumentation so that more variables can be measured, for example, colour which has already been mentioned. Others include the texture of the paper substrate and print mottle, which can also be viewed as print texture. Both require carefully directed lighting. This is needed since texture effects are dependent on the direction of the lighting. In fact pre-processing outputs that can measure the texture of the paper and print have already been developed. These are labelled outputs 24-35 in the pre-processing program.

It can be finally concluded that the image analysis can measure a wide range of print quality variables. Neural networks can be used to produce empirical models that can differentiate between different printing processes and model the human perception of a simple monochrome image. Non-impact printing was used in this investigation. However, the investigation can easily be expanded to include other printing processes, since the same variables would be measured. These variables include contrast, edge uniformity, noise and resolution.

References

1 Bruce V, Green P, **Visual Perception,** Lawrence Erlbaum Associates Ltd, Hove, East Sussex, p.1-104, 1993

2 Humphreys G W, Bruce V, **Visual Cognition,** Lawrence Erlbaum Associates Ltd, Hove, East Sussex, p.1-101, 1995

3 Firschein O, Fischler M, **The Eye, the Brain and the Computer**, Addison Wesley Publishing Co Ltd, Reading, Massachusetts, p.12, 1987

4 Green J, Hicks C, **Basic Cognitive Processes**, Open University Press, Milton Keynes, p.1-95, 1993

5 Galotti K, **Cognitive Psychology In and Out of the Laboratory**, Brooks/Cole Publishing Co, Pacific Grove, California, p.35-68, 1994

6 Johnson-Laird P, **The Computer and the Mind**, Fontana Press, London, p.174-178, 1993

7 Chalmers D J, **Scientific American**, Vol.273 No.6 p.62-68, 1995

8 Gillies D, **Artificial Intelligence and Scientific Method**, Oxford University Press, Oxford, p.118-119, 1996

9 Cost F, **Pocket Guide to Digital Printing**, Delmar Publishers Inc, Albany, New York, USA, p.3-33, p.57-110, 1997

10 Baron C J, Prichard E J, **Pira visual aid kit No 18 Print Recognition**, Pira International, Leatherhead, Surrey, 1971

11 Martin G, **Non-impact printing Pira reviews of printing**, Pira International, Leatherhead, Surrey, p.7-9, p.13-21, 1992

12 Nothman G A, **Non-impact printing**, Graphic Arts Technical Foundation, Pittsburgh, Pennsylvania, USA, ch.1-ch.8, 1989

13 J L Automation, **Image Analysis Principle and Practice**, J L Automation, Joyce Loebl Ltd, Gateshead, p.3-57, 1989

14 Lyne M, Jordan B, **Tappi Journal,** Vol.62 No.12 p.95-96, 1979

15 McCool M A, **Tappi Journal,** Vol.66 No.8 p.69-71, 1983

16 Amand F J A, Perrin B J, Sabater J A, **Tappi Journal**, Vol.76 No.5 p.139-146, 1993

17 Johansson P, **Advances in Printing Sciences and Technology**, No.17 p.269-277, 1983

18 Humphrey K, **Taga Proceedings**, Rochester, New York, USA, p.264-267, 1991

19 Safadi T, **Professional Printer**, Vol.38 No.1 p.10-14, 1994

20 Tanaka Y, Abe T, **Journal of Imaging Technology** Vol.13 No.6 p.202-206, 1987

21 Kubo S, Inui M, Miyake Y, **Journal of Imaging Science** Vol.29 p.213-215, 1985

22 Edinger J, **The Journal of Imaging Science,** Vol.31 No.4 p.177-183, 1987

23 Edinger J, **The Journal of Imaging Science,** Vol.32 No.2 p.216-220, 1988

24 Edinger J, **The Journal of Imaging Science** Vol.39 No 2 p.142-147, 1995

25 Visa A, Laaginmaa A, **Advances in printing Science and Technology**, No.21 p.168 – 173, 1992

26 Fraser G, **Print and Image Quality Conference,** San Diego, California, USA, May 6-7, 1996

27 Bristow J A, **Advances in Printing Science and Technology**, Vol.20 p.193-217, 1990

28 Paul A, **Professional Printer**, Vol.38 No.2, p.13-17, 1994

29 Bartleson C J, **The Journal of Photographic Science**, Vol.32 p.33-38, 1982

30 Engeldrum P, **Print and Image Quality Conference,** San Diego, California, USA, 6-7 May 1996

31 Engledrum P, McNeill G, **Journal of Imaging Science**, Vol.29 p.18-p.23, p.207, 1985

32 **HTTP://www.Subtechnique.com/app note app n 05.htm**, 25 Sept. 1997

33 **HTTP//www.acuity imaging. Com/ix.htm**, 27 June. 1997

34 Kipman Y**, Print and Image Quality Conference**, San Diego, California, USA, 6-7 May 1996

35 Daloia G, Dusel P, Grovanz J, Brown B, **Xerox Disclosure Journal**, Vol.21 No.2, p.139-141, 1996

36 Davidson D, **Modelling the Mind**, Clarendon Press, Oxford, p.1-9, 1990

37 Aleksander I, Morton H, **Neurons and Symbols**, Chapman and Hall, London, p.52-56, 1993

38 van der Heijden F, **Image Based Measurement Systems**, John Wiley and Sons, p.9-129, 1994

39 Beale R, **Neural Computing: An Introduction**, Institute of Physics Publishing Ltd, Bristol, p.1-105, 1994

40 Aleksander I, Morton H. **An Introduction to Neural Computing**, Thomson Computer Press, London, p.223-226, 1995

41 Bishop C M, **Neural Networks for Pattern Recognition**, Oxford University Press, Oxford, ch.1-ch.5, 1995

42 Haykin S, **Neural Networks**, Macmillan College Publishing, Ontario, Canada, ch.1, ch.4, ch.6, ch.7, 1994

43 Tarassenko L, **A Guide To Neural Computing Applications**, Arnold, London, ch.1-ch.9, 1998

44 Borggaard C, Madsen N T, Thodberg H H, **Neural Computing Applications Forum**, The University of Oxford, 27-28 June 1996

45 Murray A F, **Applications of Neural Networks**, Kluwer Academic Publishers, Dordrecht, The Netherlands, ch.2-13, 1995

46 Hassoun M, **Fundamentals of Artificial Neural Networks**, The MIT Press, Cambridge, Massachusetts, p.285-295, 1995

47 Ruck D, Rogers S, Karbriski M, Oxley M, Suter B, **IEEE Transactions on Neural Networks,** Vol.1 No.4, p296, 1990

48 Terano T, Asai K, Sugeno M, **Applied Fuzzy Systems,** Academic Press Ltd, London, ch1, ch.3, 1989

49 Zadeh L, **Fuzzy Sets and Their Applications to Cognitive Decision Processes**, Academic Press, New York, USA, p.1-3, 1975

50 Jang R; **IEEE Transactions on Systems Man and Cybernetics**, Vol. 23 No.3 p655 - 683, 1993

51 McDonald R, **Colour Physics for Industry**, Society of Dyers and Colourist, Bradford, West Yorkshire, p. 429, 1997

Appendix. 1. The pre-processing program.

This is the code that was used to produce the results in chapter 6 of this thesis. It has not been modified in any way except for the introduction of some comment statements.

This is the bas form that is used to declare the variables as global so that they can be used by all the forms.

```
Type Nextline
    Next1 As String
    End Type
Global Test As Nextline
Global Matrix(770, 580) As Single
Global Contrast(260) As Long
Global Imag(20, 260), Backgrd(20, 260) As Long
Global Black(20, 800), White(20, 800) As Integer
Global BKFreq
Global Greyscale1, Greyscale2, Grad
Global Thres, LastThres As Integer
Global Gradient(30) As Integer
Global ClkB, ClkW, ColChB, ColChW, Colchange
Global TotalImag(100), TotalDiffBackGround(100) As Long
Global TotalGrey, TotalBackGround
Global WhiteSep(40), BlackSep(40), MxB(40), MxW(40)
Global IntSwitch
Global DivB(30), DivW(30), DivG
Global XLimit, YStart, YFinish, XStart, XFinish
Global ScanLine(20, 800), LineCount(20, 800)
Global BCountLine(800, 800) As Integer
Global BNormit, Diff0(30), Diff1(30), Diff2(30), Diff3(30), Diff4(30), Diff5(30), Diff6(30),
Diff7(30), Diff8(30), Diff9(30), Diff10(30), Diff11(30), Diff12(30), Diff13(30), Diff14(30),
Diff15(30)
Global Diff16(30), Diff17(30), Diff18(30), Diff19(30), Diff20(30), Diff21(30), Diff22(30),
Diff23(30), Diff24(30), Diff25(30), Diff26(30), Diff27(30), Diff28(30), Diff29(30), Diff30(30),
Diff31(30), Diff32(30)
Global Diff33(30), Diff34(30), Diff35(30), Diff36(30), Diff37(30), Diff38(30), Diff39(30),
Diff40(30), Diff41(30), Diff42(30), Diff43(30), Diff44(30), Diff45(30), Diff46(30), Diff47(30),
Diff48(30), Diff49(30)
Global BCulFrqy(100)
Global BLineFind(20, 800), WLineFind(20, 800)
Global RagBlack(20, 800), RagWhite(20, 800)
Global DYDownLine(20, 800), DYUpLine(20, 800), BFLast(800), WFLast(800), BTot(30, 1000),
TotStoch(20), BLineFreq(800, 800) As Long
Global FuzzB(30), FuzzB1(30), FuzzB2(30) As Single
Global FuzzB3(30), FuzzB4(30), FuzzB5(30) As Single
Global FuzzB6(30) As Single
Global Fuzzw(20), Fuzzw1(20), Fuzzw2(20) As Single
Global Fuzzw3(20), Fuzzw4(20), Fuzzw5(20) As Single
Global Fuzzw6(20) As Single
Global MkSpace(20, 2000) As Single
```

This form imports the image data from the Matrox hardware into a Visual Basic Matrix array. It also draws two perpendicular cursor for the alignment of the 1cm squares.

```
Dim MilApplication As Long      ' Application identifier.
Dim MilSystem As Long           ' System identifier.
Dim MilDisplay As Long          ' Display identifier.
Dim MilDigitizer As Long        ' Camera identifier.
Dim MilImage As Long            ' Image buffer identifier.
Dim MilImage1 As Long
Dim SubImage1 As Long
Dim MonochromeArray(768, 576) As Byte
Dim MonochromeArrayTotal(768, 576) As Single
Const X_STARTPOINT = 0&
Const Y_STARTPOINT = 0&
Const X_ARRAY = 768&
Const Y_ARRAY = 576&
Dim Value, ImageNo, Chr(13) As Long

Private Sub Command1_Click()

Call MappAllocDefault(M_SETUP, MilApplication, MilSystem, MilDisplay, MilDigitizer,
MilImage)

  ' Grab an image.
Call MdigGrabContinuous(MilDigitizer, MilImage)

End Sub

Private Sub Command2_Click()

Call MappFreeDefault(MilApplication, MilSystem, MilDisplay, MilDigitizer, MilImage)
Rem Call MappFreeDefault(MilApplication, MilSystem, MilDisplay, MilDigitizer, MilImage1)

End Sub

Private Sub Command3_Click()

Text1.Text = "On"
Dim NoIm, c, d As Integer
NoIm = 20
c = 1
d = 0
Rem Open "c:\cmtutil\M4_17.csv" For Output As #1
Rem Call MbufClear(MilImage, 2&)
Call MbufInquire(MilImage, M_SIZE_Y, Value)
For T = 1 To NoIm
Call MdigHalt(MilDigitizer)
Call MdigHalt(MilDigitizer)
Rem Call MbufImport("c:\cmtutil\vidpres\square_cal.tif", M_TIFF, M_LOAD, M_NULL,
MilImage)
Call MbufGet2d(MilImage, X_STARTPOINT, YSTARTPOINT, X_ARRAY, Y_ARRAY,
MonochromeArray(0, 0))
For Y = Y_STARTPOINT To Y_ARRAY
For X = X_STARTPOINT To X_ARRAY
MonochromeArrayTotal(X, Y) = MonochromeArray(X, Y) + MonochromeArrayTotal(X, Y)
If T = NoIm Then MonochromeArrayTotal(X, Y) = MonochromeArrayTotal(X, Y) / NoIm
Next X
Next Y
Call MdigGrabContinuous(MilDigitizer, MilImage)
Next T
```

```
For YY = Y_STARTPOINT To Y_ARRAY
For XX = X_STARTPOINT To X_ARRAY
c = c + 1
If c = X_ARRAY - 1 Then d = d + 1
Matrix(c, d) = MonochromeArrayTotal(XX, YY)
If c = X_ARRAY - 1 Then c = -1
Next XX
Next YY
Text1.Text = "Finish"
Call MappFreeDefault(MilApplication, MilSystem, MilDisplay, MilDigitizer, MilImage)
Form3.Command1.Value = True

End Sub

Private Sub Command4_Click()

Call MdigHalt(MilDigitizer)
Call MbufImport("c:\cmtutil\vidpres\square_cal.tif", M_TIFF, M_LOAD, M_NULL, MilImage)

End Sub

Private Sub Command5_Click()

Rem Call MbufAlloc2d(MilSystem, X_ARRAY, Y_ARRAY, 8, M_IMAGE + M_DISP +
M_PROC + M_OVR, MilImage1)
Rem Call MdispSelect(MilImage, MilImage1)

For I = 1 To 200
Call MdispControl(MilDisplay, M_OVR_SHOW, M_DISABLE)
Call MdispControl(MilDisplay, M_OVR_WRITE, M_ENABLE)
Rem Call MdispInquire(MilDisplay, M_WINDOW_OVR_BUF_ID, MilImage1)
Rem Call MgraLine(M_DEFAULT, MilImage, 0, 0, 500, 500)
Call MdigHalt(MilDigitizer)
Rem Call MgraLine(M_DEFAULT, MilImage, 0, 0, 500, 500)
Call MdigGrabContinuous(MilDigitizer, MilImage)
Call MgraLine(M_DEFAULT, MilImage, 650, 0, 650, 576)
Call MgraLine(M_DEFAULT, MilImage, 0, 15, 768, 15)

Next I

End Sub
```

This form exports the raw image data into a CSV data file that can be displayed in Excel.

```
Private Sub Command1_Click()

Dim J, Xp, L, M, Yp As Integer
Dim Number As Integer
Xp = 1
J = 0
Open "c:\cmtutil\test2.csv" For Input As #1
Do While Not EOF(1)
Input #1, Test.Next1
Rem Select data
J = J + 1
If Yp <= 767 Then Yp = Yp + 1
If Yp <= 767 And J > 2 Then Modulus = Abs(Val(Test.Next1) - 256)
If Yp <= 767 And J > 2 Then Number = Val(Test.Next1)
If J = 3 Then Yp = 1
Matrix(Xp, Yp - 1) = Modulus
If Yp = 767 Then Xp = Xp + 1
If Yp = 767 Then Yp = 0
If Yp = 766 Then Dog = Number
```

```
If Yp = 765 Then Cat = Number
If Yp = 1 Then Comma = Number
Loop
Close #1

Form3.Command1 = True

End Sub
```

This form extracts features from the the raw image data.

```
Private Sub Command1_Click()
Dim BFrequency(770, 580), WFrequency(770, 580) As Integer
Dim J, M, N, I, P, Threshold, Count, BWhite, Switch1 As Integer
Dim PkPos(770, 580), Rg(1000), Recordit(1000) As Integer
Dim Peakreference As Integer
Dim RGPk(1000), RGPKvert(1000), RagDiff1(30), RagDiff2(30) As Long
Dim Graddev1(1000), Graddev2(1000) As Integer
Dim RgFreq(800), Rg1(800), Rg2(800), Rg3(800), Rg4(800), Rg5(800), Rg6(800), Rg7(800),
Rg8(800), Rg9(800), Rg10(800), Rg11(800), Rg12(800), RgX(800), RgY(800) As Single
Dim Errordev11, Errordev12, NTotalvert As Integer

NTotal = 1
NTotalvert = 1
Increment = 1
YStart = 2
YFinish = 576
XStart = 2
XFinish = 768
Nflt = 4
RagLev = 200
```

Setting the minimum background level to 0 and the maximum image level to 255.
Measuring the total contrast valus of the image.

```
For Y = YStart To YFinish
For X = XStart To XFinish

Matrix(X, Y) = Abs(255 - Matrix(X, Y))
Contrast(Matrix(X, Y)) = Contrast(Matrix(X, Y)) + 1

Next X
Next Y
```

Detecting the locations of the image.

```
Grad = 24
LastThres = 1
Noiseft = 20000
For Thres = 1 To LastThres
Diff6(Thres) = 0
Grad = Grad + 1
CursorX = 0
CursorwX = 0
Gradient(Thres) = Grad
Peakreference = 1


For Y = YStart To YFinish
For X = XStart To XFinish - G

G = 2
If (Matrix(X + G, Y) - Matrix(X, Y)) > Grad Then PkPos(X, Y) = (Matrix(X + G, Y) - Matrix(X,
Y))
```

```
If (Matrix(X + G, Y) - Matrix(X, Y)) < -Grad Then PkPos(X + 1, Y) = (Matrix(X + G, Y) -
Matrix(X, Y)) Else PkPos(X + 1, Y) = 0
If X = XStart Then PkPos(X, Y) = -255
If X = XFinish Then PkPos(X, Y) = 255

Next X
Next Y

Switch1 = 1
Count = 0

For Y = YStart + G To YFinish - G

LineCount(Thres, Y) = 1

BF = 0
WF = 0

For X = XStart + G To XFinish - G
MoveBack = 0
If PkPos(X, Y) > Grad Then Filter1 = PkPos(X, Y)
If PkPos(X, Y) < -Grad Then Filter1 = -PkPos(X, Y)
If PkPos(X, Y) > Grad Then MKGrey = PkPos(X, Y)
If PkPos(X, Y) < -Grad Then MKGrey = PkPos(X, Y)
```

Measuring the halftone frequency distribution for the background regions.

```
If Switch1 = 1 Then Count = Count + 1
If Switch1 = 1 And MKGrey > Grad And Matrix(X, Y) > 150 Then Imag(Thres, Matrix(X, Y)) =
Imag(Thres, Matrix(X, Y)) + 1
If Switch1 = 1 And MKGrey > Grad And Matrix(X, Y) > 150 And Count = 1 Then Imag(Thres,
Matrix(X - 1, Y)) = Imag(Thres, Matrix(X - 1, Y)) - 1
If Switch1 = 1 And MKGrey > Grad And Count = 1 Then Backgrd(Thres, Matrix(X - 1, Y)) =
Backgrd(Thres, Matrix(X - 1, Y)) + 1
If Switch1 = 1 And MKGrey < -Grad Then Backgrd(Thres, Matrix(X, Y)) = Backgrd(Thres,
Matrix(X, Y)) + 1
If X = XFinish Then Imag(Thres, Matrix(X - 1, Y)) = Imag(Thres, Matrix(X - 1, Y)) - 1
If X = XFinish Then Backgrd(Thres, Matrix(X, Y)) = Backgrd(Thres, Matrix(X, Y)) + 1
Rem If Switch1 = 1 And MKGrey > Grad And Count = 1 Then Backgrd(Thres, Matrix(X - 1, Y)) =
Backgrd(Thres, Matrix(X - 1, Y)) + 1
If BWhite > 2 Then LineCount(Thres, Y) = LineCount(Thres, Y) + 1
If Abs(PkPos(X, Y)) > Grad And PkPos(X - 1, Y) = 0 And Count > Nflt Then Switch1 = 0
```

Rem < is read black

```
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) Then
White(Thres, Count + 1) = White(Thres, Count + 1) + 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) Then WF =
WF + 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) And WF =
1 Then RagWhite(Thres, X) = Y
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) Then
WFrequency(WF, Y) = Count
```

Calculating the mean value of the descending gradients.

```
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) Then
DYUpLine(Thres, X) = DYUpLine(Thres, X) + 1If Switch1 = 0 And Count > Nflt And
Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) And X > 2 And X < 700 Then Fuzzw1(Thres) =
Fuzzw1(Thres) + (Matrix(X + G, Y) - Matrix(X - G, Y))
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) And Y > 2
And X < 700 Then Fuzzw2(Thres) = Fuzzw2(Thres) + Abs(Matrix(X, Y + 2) - Matrix(X, Y - 2))
```

```
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) And Y > 2
And X < 700 Then Fuzzw3(Thres) = Fuzzw3(Thres) + Abs((Matrix(X, Y + 2) - Matrix(X, Y)) -
(Matrix(X, Y) - Matrix(X, Y - 2)))
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) And X <
700 Then Fuzzcountw = Fuzzcountw + 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) And X > 2
And X < 700 Then Fuzzcountw1 = Fuzzcountw1 + 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) And Y > 2
And X < 700 And Abs(Matrix(X + G, Y) - Matrix(X - G, Y)) < RagLev Then Fuzzcountw2 =
Fuzzcountw2 + 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) And Y > 2
And X < 700 And Abs(Matrix(X + G, Y) - Matrix(X - G, Y)) < RagLev Then Fuzzcountw3 =
Fuzzcountw3 + 1
If DYUpLine(Thres, X) > Noiseft Then DYUpLine(Thres, X) = Noiseft
```

Detecting the position of an image using the mean of the locations of the detected rising gradients.

```
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) > Abs(PkPos(X, Y)) And X > 2
And X < 700 Then CursorwX = CursorwX + X
```

Measuring the halftone frequency distribution for the image regions.

```
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) Then
Black(Thres, Count - 1) = Black(Thres, Count - 1) + 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) Then BF =
BF + 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) Then
RagBlack(Thres, X) = Y
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) Then
BFrequency(BF, Y) = Count
```

Calculating the mean value of the descending gradients.

```
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) Then
DYDownLine(Thres, X) = DYDownLine(Thres, X) + 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2
Then Rg1(Y) = Count
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2
Then RgX(Y) = X
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2
Then RgY(Y) = Y

If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And X > 2
Then FuzzB1(Thres) = FuzzB1(Thres) + (Matrix(X + G, Y) - Matrix(X - G, Y))
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2
Then FuzzB2(Thres) = FuzzB2(Thres) + Abs(Matrix(X, Y + 2) - Matrix(X, Y - 2))

If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And X > 2
And Abs(Matrix(X + G, Y) - Matrix(X - G, Y)) < RagLev / 4 Then RagDiff1(Thres) =
RagDiff1(Thres) + Abs(Matrix(X + G, Y) - Matrix(X - G, Y))
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2
And Abs(Matrix(X, Y + G) - Matrix(X, Y - G)) < RagLev / 4 Then RagDiff2(Thres) =
RagDiff2(Thres) + Abs(Matrix(X, Y + G) - Matrix(X, Y - G))

If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And X > 2
And Abs(Matrix(X + G, Y) - Matrix(X - G, Y)) < RagLev / 4 Then Graddev1(Y) = Abs(Matrix(X
+ G, Y) - Matrix(X - G, Y))
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2
And Abs(Matrix(X, Y + G) - Matrix(X, Y - G)) < RagLev / 4 Then Graddev2(Y) = Abs(Matrix(X,
Y + G) - Matrix(X, Y - G))
```

If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2 And Abs(Matrix(X, Y + G) - Matrix(X, Y - G)) < RagLev / 4 And Graddev2(Y - 1) > 0 Then NTotalvert = NTotalvert + 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2 Then FuzzB3(Thres) = FuzzB3(Thres) + Abs((Matrix(X, Y + 2) - Matrix(X, Y)) - (Matrix(X, Y) - Matrix(X, Y - 2)))
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2 Then Rg(Y) = X
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2 Then RGPk(Y) = (Matrix(X + G, Y) - Matrix(X - G, Y))
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2 Then RGPKvert(Y) = (Matrix(X, Y + G) - Matrix(X, Y - G))
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2 Then Recordit(Y) = 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) Then Fuzzcount = Fuzzcount + 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And X > 2 Then Fuzzcount1 = Fuzzcount1 + 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And X > 2 Then CursorX = CursorX + X
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2 And Abs(Matrix(X + G, Y) - Matrix(X - G, Y)) < RagLev Then Fuzzcount2 = Fuzzcount2 + 1
If Switch1 = 0 And Count > Nflt And Abs(PkPos(X, Y) + Filter1) < Abs(PkPos(X, Y)) And Y > 2 And Abs(Matrix(X + G, Y) - Matrix(X - G, Y)) < RagLev Then Fuzzcount3 = Fuzzcount3 + 1
If DYDownLine(Thres, X) > Noiseft Then DYDownLine(Thres, X) = Noiseft
If Count > Nflt And Switch1 = 0 Then Count = 0
If Abs(PkPos(X, Y)) > Grad And Switch1 = 0 Then Switch1 = 1
If MoveBack = 1 Then X = X - 1

Next X

BFLast(Y) = BF
WFLast(Y) = WF

Next Y

A Noise calculation.

For Y = YStart To YFinish
For R = 2 To BFLast(Y) - 1

MkSpace(Thres, BFrequency(R, Y) + WFrequency(R, Y)) = MkSpace(Thres, BFrequency(R, Y) + WFrequency(R, Y)) + 1

Next R
Next Y

Peak detection.

For Peakval = 1 To 100

If MkSpace(Thres, Peakval) > 5 Then MKRegister = MKRegister + 1
If Peakval = 1 Then Peakreference = MkSpace(Thres, Peakval)
If Peakval > 1 And MkSpace(Thres, Peakval) > Peakreference Then P = Peakval
If Peakval > 1 And MkSpace(Thres, Peakval) > Peakreference Then Peakreference = MkSpace(Thres, Peakval)

Next Peakval
If P > 2 Then Diff6(Thres) = (Peakreference + 0.00001) / (Peakreference + 0.00001 + (3 * MkSpace(Thres, P + 2)) + (2 * MkSpace(Thres, P + 1)) + (3 * MkSpace(Thres, P - 2)) + (2 * MkSpace(Thres, P - 1)))

```
BNormit = 0

For X = XStart To XFinish

BCountLim = 0
Modl = 0
For Z = YStart To YFinish
If BCountLine(X, Z) >= BCountLim Then BCountLim = BCountLine(X, Z)
If BCountLine(X, Z) = BCountLim Then BModl = Z
Next Z
```

Noise calculations.

```
For R = 2 To BFLast(X)
If R > 2 And R < BFLast(X) - 1 Then BNormit = BNormit + 1
If Abs(BFrequency(R, X) - BFrequency(R - 2, X)) = 0 And R > 2 And R < BFLast(X) - 1 Then
Diff0(Thres) = Diff0(Thres) + 1
If Abs(BFrequency(R, X) - BFrequency(R - 2, X)) = 1 And R > 2 And R < BFLast(X) - 1 Then
Diff1(Thres) = Diff1(Thres) + 1
If Abs(BFrequency(R, X) - BFrequency(R - 2, X)) = 2 And R > 2 And R < BFLast(X) - 1 Then
Diff2(Thres) = Diff2(Thres) + 1
If Abs(BFrequency(R, X) - BFrequency(R - 2, X)) = 3 And R > 2 And R < BFLast(X) - 1 Then
Diff3(Thres) = Diff3(Thres) + 1
If Abs(BFrequency(R, X) - BFrequency(R - 2, X)) = 4 And R > 2 And R < BFLast(X) - 1 Then
Diff4(Thres) = Diff4(Thres) + 1
If Abs(BFrequency(R, X) - BFrequency(R - 2, X)) = 5 And R > 2 And R < BFLast(X) - 1 Then
Diff5(Thres) = Diff5(Thres) + 1
If Abs(BFrequency(X, R) - BFrequency(X, R - 1)) <= 8 And R > 2 And R < BFLast(X) - 2 Then
Diff7(Thres) = Diff7(Thres) + 1
If Abs(BFrequency(X, R) - BFrequency(X, R - 1)) <= 10 And R > 2 And R < BFLast(X) - 2 Then
Diff8(Thres) = Diff8(Thres) + 1
If Abs(BFrequency(X, R) - BFrequency(X, R - 1)) <= 12 And R > 2 And R < BFLast(X) - 2 Then
Diff9(Thres) = Diff9(Thres) + 1
If Abs(BFrequency(X, R) - BFrequency(X, R - 1)) <= 14 And R > 2 And R < BFLast(X) - 2 Then
Diff10(Thres) = Diff10(Thres) + 1
If Abs(BFrequency(X, R) - BFrequency(X, R - 1)) <= 16 And R > 2 And R < BFLast(X) - 2 Then
Diff11(Thres) = Diff11(Thres) + 1

If BFLast(X) > 4 And R < BFLast(X) And BFrequency(X, R) > WFrequency(X, R) Then
BLineFind(Thres, R) = BLineFind(Thres, R) + (WFrequency(X, R + 1) - BFrequency(X, R))
If BFLast(X) > 4 And BFrequency(X, R) < WFrequency(X, R) Then WLineFind(Thres, R) =
WLineFind(Thres, R) + (BFrequency(X, R) - WFrequency(X, R))

Next R
Next X

If Fuzzcount = 0 Then Fuzzcount = 1
If Fuzzcount1 = 0 Then Fuzzcount1 = 1
If Fuzzcount2 = 0 Then Fuzzcount2 = 1
If Fuzzcount3 = 0 Then Fuzzcount3 = 1

If Fuzzcountw = 0 Then Fuzzcountw = 1
If Fuzzcountw1 = 0 Then Fuzzcountw1 = 1
If Fuzzcountw2 = 0 Then Fuzzcountw2 = 1
If Fuzzcountw3 = 0 Then Fuzzcountw3 = 1
If BNormit = 0 Then BNormit = 1

Diff0(Thres) = Diff0(Thres) / BNormit
Diff1(Thres) = Diff1(Thres) / BNormit
Diff2(Thres) = Diff2(Thres) / BNormit
Diff3(Thres) = Diff3(Thres) / BNormit
Diff4(Thres) = Diff4(Thres) / BNormit
```

```
Diff5(Thres) = Diff5(Thres) / BNormit
Diff6(Thres) = Diff6(Thres)
Diff7(Thres) = FuzzB1(Thres) / Fuzzcount1
Diff8(Thres) = CursorX / Fuzzcount1
Diff9(Thres) = FuzzB2(Thres) / Fuzzcount2
Diff10(Thres) = FuzzB3(Thres) / Fuzzcount3
Diff11(Thres) = MKRegister
Diff12(Thres) = Peakreference
Diff13(Thres) = P
Diff14(Thres) = BNormit
Diff15(Thres) = Fuzzcount
Diff23(Thres) = Fuzzw1(Thres) / Fuzzcountw1
Diff24(Thres) = CursorwX / Fuzzcountw1
Diff25(Thres) = Fuzzw2(Thres) / Fuzzcountw2
Diff26(Thres) = Fuzzw3(Thres) / Fuzzcountw3
Rem List5.Clear
Rem For Y = 300 To 600
Rem List5.AddItem DYDownLine(Thres, Y) & "    " & DYUpLine(Thres, Y)
Rem Next Y

For X = XStart To XFinish
For Y = YStart To YFinish

ScanLine(Thres, X) = ScanLine(Thres, X) + Matrix(X, Y)
Next Y
Next X
RagDiff1(Thres) = RagDiff1(Thres) / NTotal
RagDiff2(Thres) = RagDiff2(Thres) / NTotalvert
Open "c:\cmtutil\vidpres\files\vital4.csv" For Output As #50
Testing = 14

For Testloop = 1 To YFinish
Testing = Testing + 1
If Graddev1(Testing - 1) > 1 Then Errordev11 = Abs(Abs(Graddev1(Testing)) -
Abs(Graddev1(Testing - 1))) + Errordev11
If Graddev2(Testing - 1) > 1 Then Errordev12 = Abs(Abs(Graddev2(Testing)) -
Abs(Graddev2(Testing - 1))) + Errordev12

If Graddev1(Testing - 1) > 1 Then NTotal = NTotal + 1
If Graddev2(Testing - 1) > 1 And Graddev2(Testing) > Graddev2(Testing - 1) And
Graddev2(Testing) > Graddev2(Testing + 1) Then PKLoc = PKLoc + 1
Errordev2
Rg6(Testing) + Rg7(Testing) + Rg8(Testing) + Rg9(Testing) + Rg10(Testing) + Rg11(Testing) +
Rg12(Testing)
RgTotal = Abs((Rg4(Testing) + Rg5(Testing)) - (Rg4(Testing - 1) + Rg5(Testing - 1)))
RgFreq(Testing) = Rg(Testing) - Rg(Testing - 1)
If RgFreq(Testing) < -5 Then RgFreq(Testing) = 0
If RgFreq(Testing) > 5 Then RgFreq(Testing) = 0
If RgFreq(Testing) > 0 Then NTotalNorm = NTotalNorm + 1
If Abs(RgFreq(Testing)) = 1 Then Lev1 = Lev1 + 1
If Abs(RgFreq(Testing)) = 2 Then Lev2 = Lev2 + 1
If Abs(RgFreq(Testing)) = 3 Then Lev3 = Lev3 + 1
If Abs(RgFreq(Testing)) = 4 Then Lev4 = Lev4 + 1
Pyth = Sqr(((RgX(Testing) - RgX(Testing - 1)) ^ 2) + ((RgY(Testing) - RgY(Testing - 1)) ^ 2))
If Pyth > 10 Then Pyth = 1
Pythtotal = Pyth + Pythtotal
Write #50, RgFreq(Testing), Rg1(Testing), RgX(Testing), RgY(Testing), Pythtotal,
Graddev1(Testing), Graddev2(Testing), NTotal, NTotalvert
Diff36(Thres) = RgTotal + Diff36(Thres)
Next Testloop

Close #50
Tlev = 1
```

Frequency filters.

```
For F = YStart + 15 To YFinish - 16

AdjMinus1 = RgFreq(F - 1)
AdjEqual1 = RgFreq(F)

AdjMinus2 = RgFreq(F - 1) + RgFreq(F - 2)
AdjEqual2 = RgFreq(F) + RgFreq(F + 1)

AdjMinus3 = RgFreq(F - 1) + RgFreq(F - 2) + RgFreq(F - 3)
AdjEqual3 = RgFreq(F) + RgFreq(F + 1) + RgFreq(F + 2)

AdjMinus4 = RgFreq(F - 1) + RgFreq(F - 2) + RgFreq(F - 3) + RgFreq(F - 4)
AdjEqual4 = RgFreq(F) + RgFreq(F + 1) + RgFreq(F + 2) + RgFreq(F + 3)

AdjMinus5 = RgFreq(F - 1) + RgFreq(F - 2) + RgFreq(F - 3) + RgFreq(F - 4) + RgFreq(F - 5)
AdjEqual5 = RgFreq(F) + RgFreq(F + 1) + RgFreq(F + 2) + RgFreq(F + 3) + RgFreq(F + 4)

AdjMinus6 = RgFreq(F - 1) + RgFreq(F - 2) + RgFreq(F - 3) + RgFreq(F - 4) + RgFreq(F - 5) +
RgFreq(F - 6)
AdjEqual6 = RgFreq(F) + RgFreq(F + 1) + RgFreq(F + 2) + RgFreq(F + 3) + RgFreq(F + 4) +
RgFreq(F + 5)

AdjMinus7 = RgFreq(F - 1) + RgFreq(F - 2) + RgFreq(F - 3) + RgFreq(F - 4) + RgFreq(F - 5) +
RgFreq(F - 6) + RgFreq(F - 7)
AdjEqual7 = RgFreq(F) + RgFreq(F + 1) + RgFreq(F + 2) + RgFreq(F + 3) + RgFreq(F + 4) +
RgFreq(F + 5) + RgFreq(F + 6)

AdjMinus8 = RgFreq(F - 1) + RgFreq(F - 2) + RgFreq(F - 3) + RgFreq(F - 4) + RgFreq(F - 5) +
RgFreq(F - 6) + RgFreq(F - 7) + RgFreq(F - 8)
AdjEqual8 = RgFreq(F) + RgFreq(F + 1) + RgFreq(F + 2) + RgFreq(F + 3) + RgFreq(F + 4) +
RgFreq(F + 5) + RgFreq(F + 6) + RgFreq(F + 7)

AdjMinus9 = RgFreq(F - 1) + RgFreq(F - 2) + RgFreq(F - 3) + RgFreq(F - 4) + RgFreq(F - 5) +
RgFreq(F - 6) + RgFreq(F - 7) + RgFreq(F - 8) + RgFreq(F - 9)
AdjEqual9 = RgFreq(F) + RgFreq(F + 1) + RgFreq(F + 2) + RgFreq(F + 3) + RgFreq(F + 4) +
RgFreq(F + 5) + RgFreq(F + 6) + RgFreq(F + 7) + RgFreq(F + 8)

AdjMinus10 = RgFreq(F - 1) + RgFreq(F - 2) + RgFreq(F - 3) + RgFreq(F - 4) + RgFreq(F - 5) +
RgFreq(F - 6) + RgFreq(F - 7) + RgFreq(F - 8) + RgFreq(F - 9) + RgFreq(F - 10)
AdjEqual10 = RgFreq(F) + RgFreq(F + 1) + RgFreq(F + 2) + RgFreq(F + 3) + RgFreq(F + 4) +
RgFreq(F + 5) + RgFreq(F + 6) + RgFreq(F + 7) + RgFreq(F + 8) + RgFreq(F + 9)

If AdjMinus1 > AdjEqual1 And Abs(AdjEqual1 - AdjMinus1) > 0 And AdjEqual1 >= 0 And
AdjMinus1 <= 0 Then Errordev1 = Errordev1 + Abs(AdjEqual1 - AdjMinus1)
If AdjMinus1 < AdjEqual1 And Abs(AdjEqual1 - AdjMinus1) > 0 And AdjEqual1 >= 0 And
AdjMinus1 <= 0 Then Errordev1 = Errordev1 + Abs(AdjEqual1 - AdjMinus1)

If AdjMinus2 > AdjEqual2 And Abs(AdjEqual2 - AdjMinus2) > 0 And AdjEqual2 >= 0 And
AdjMinus2 <= 0 Then Errordev2 = Errordev2 + Abs(AdjEqual2 - AdjMinus2)
If AdjMinus2 < AdjEqual2 And Abs(AdjEqual2 - AdjMinus2) > 0 And AdjEqual2 >= 0 And
AdjMinus2 <= 0 Then Errordev2 = Errordev2 + Abs(AdjEqual2 - AdjMinus2)

If AdjMinus3 > AdjEqual3 And Abs(AdjEqual3 - AdjMinus3) > 1 And AdjEqual3 >= 1 And
AdjMinus3 <= -1 Then Errordev3 = Errordev3 + Abs(AdjEqual3 - AdjMinus3)
If AdjMinus3 < AdjEqual3 And Abs(AdjEqual3 - AdjMinus3) > 1 And AdjEqual3 >= 1 And
AdjMinus3 <= -1 Then Errordev3 = Errordev3 + Abs(AdjEqual3 - AdjMinus3)
```

```
If AdjMinus4 > AdjEqual4 And Abs(AdjEqual4 - AdjMinus4) > 1 And AdjEqual4 >= 1 And
AdjMinus4 <= -1 Then Errordev4 = Errordev4 + Abs(AdjEqual4 - AdjMinus4)
If AdjMinus4 < AdjEqual4 And Abs(AdjEqual4 - AdjMinus4) > 1 And AdjEqual4 >= 1 And
AdjMinus4 <= -1 Then Errordev4 = Errordev4 + Abs(AdjEqual4 - AdjMinus4)

If AdjMinus5 > AdjEqual5 And Abs(AdjEqual5 - AdjMinus5) > 2 And AdjEqual5 >= 2 And
AdjMinus5 <= -2 Then Errordev5 = Errordev5 + Abs(AdjEqual5 - AdjMinus5)
If AdjMinus5 < AdjEqual5 And Abs(AdjEqual5 - AdjMinus5) > 2 And AdjEqual5 >= 2 And
AdjMinus5 <= -2 Then Errordev5 = Errordev5 + Abs(AdjEqual5 - AdjMinus5)

If AdjMinus6 > AdjEqual6 And Abs(AdjEqual6 - AdjMinus6) > 2 And AdjEqual6 >= 2 And
AdjMinus6 <= -2 Then Errordev6 = Errordev6 + Abs(AdjEqual6 - AdjMinus6)
If AdjMinus6 < AdjEqual6 And Abs(AdjEqual6 - AdjMinus6) > 2 And AdjEqual6 >= 2 And
AdjMinus6 <= -2 Then Errordev6 = Errordev6 + Abs(AdjEqual6 - AdjMinus6)

If AdjMinus7 > AdjEqual7 And Abs(AdjEqual7 - AdjMinus7) > 3 And AdjEqual7 >= 3 And
AdjMinus7 <= -3 Then Errordev7 = Errordev7 + Abs(AdjEqual7 - AdjMinus7)
If AdjMinus7 < AdjEqual7 And Abs(AdjEqual7 - AdjMinus7) > 3 And AdjEqual7 >= 3 And
AdjMinus7 <= -3 Then Errordev7 = Errordev7 + Abs(AdjEqual7 - AdjMinus7)

If AdjMinus8 > AdjEqual8 And Abs(AdjEqual8 - AdjMinus8) > 3 And AdjEqual8 >= 3 And
AdjMinus2 <= -3 Then Errordev8 = Errordev8 + Abs(AdjEqual8 - AdjMinus8)
If AdjMinus8 < AdjEqual8 And Abs(AdjEqual8 - AdjMinus8) > 3 And AdjEqual8 >= 3 And
AdjMinus2 <= -3 Then Errordev8 = Errordev8 + Abs(AdjEqual8 - AdjMinus8)

If AdjMinus9 > AdjEqual9 And Abs(AdjEqual9 - AdjMinus9) > 4 And AdjEqual9 >= 4 And
AdjMinus9 <= -4 Then Errordev9 = Errordev9 + Abs(AdjEqual9 - AdjMinus9)
If AdjMinus9 < AdjEqual9 And Abs(AdjEqual9 - AdjMinus9) > 4 And AdjEqual9 >= 4 And
AdjMinus9 <= -4 Then Errordev9 = Errordev9 + Abs(AdjEqual9 - AdjMinus9)

If AdjMinus10 > AdjEqual10 And Abs(AdjEqual10 - AdjMinus10) > 4 And AdjEqual10 >= 4 And
AdjMinus10 <= -4 Then Errordev10 = Errordev10 + Abs(AdjEqual10 - AdjMinus10)
If AdjMinus10 < AdjEqual10 And Abs(AdjEqual10 - AdjMinus10) > 4 And AdjEqual10 >= 4 And
AdjMinus10 <= -4 Then Errordev10 = Errordev10 + Abs(AdjEqual10 - AdjMinus10)

Next F

For Y = YStart To YFinish
For R = 2 To BFLast(Y) - 1
MkSpace(Thres, BFrequency(R, Y) + WFrequency(R, Y)) = 0
Next R
Next Y
Peakreference = 1
```

Data transfer into other forms for transfer into MATLAB.

```
Diff33(Thres) = Errordev1
Diff34(Thres) = Errordev2
Diff35(Thres) = Errordev3
Diff36(Thres) = Errordev4
Diff37(Thres) = Errordev5
Diff38(Thres) = Errordev6
Diff39(Thres) = Errordev7
Diff40(Thres) = Errordev8
Diff41(Thres) = Errordev9
Diff42(Thres) = Errordev10
Diff43(Thres) = PKLoc
Diff44(Thres) = Errordev11 / NTotalvert
Diff45(Thres) = NTotalNorm
Diff46(Thres) = Lev1
Diff47(Thres) = Lev2
Diff48(Thres) = Lev3
Diff49(Thres) = Lev4
```

```
Fuzzcount = 0
Fuzzcount1 = 0
Fuzzcount2 = 0
Fuzzcount3 = 0

Fuzzcountw = 0
Fuzzcountw1 = 0
Fuzzcountw2 = 0
Fuzzcountw3 = 0
Next Thres

Form4.Command1.Value = True
End Sub
```

Data check procedure for the measurement of gradients.

```
Private Sub ThresBlk_Click()
Thres = InputBox("Threshold = ", Thres)
List5.Clear
For R = 1 To 720
List5.AddItem DYDownLine(Thres, R) & "     " & DYUpLine(Thres, R)
Next R
End Sub
```

Plot of the halftone cycle size distribution for the image areas.

```
Private Sub Command1_Click()

ClkB = ClkB + 1

For Thres = 1 To LastThres

List1.AddItem Gradient(Thres)
XLimit = 720
For I = 1 To XLimit
If Black(Thres, I) > MxB(Thres) And I > 2 Then MxB(Thres) = Black(Thres, I)
If Black(Thres, I) = MxB(Thres) And I > 2 Then BlackSep(Thres) = I

Next I

Graph1.AutoInc = 0
Graph1.NumPoints = XLimit

For J = 1 To XLimit
Graph1.ThisPoint = J
Graph1.XPosData = J
Graph1.GraphData = Black(Thres, J)
Next J

Graph1.GraphType = 6
Graph1.GraphStyle = 2
Graph1.BottomTitle = "Pk to Pk Separation/Pixels"
Graph1.LeftTitle = "N"
Graph1.Background = 14
Graph1.DrawMode = 2

Next Thres

If ClkB = 1 Then Form5.Command1.Value = True

End Sub
```

```
Private Sub Form_Load()

ClkB = ClkB
ColChB = ColChB
Colchange = Colchange
End Sub

Private Sub List1_Click()

Icr = 0

For YesThres = 1 To LastThres
If Val(List1.Text) >= 0.99 * Gradient(YesThres) And Val(List1.Text) <= 1.01 *
Gradient(YesThres) Then Thres = YesThres
List2.Clear
Next YesThres
Graph1.DataReset = 9
For I = 1 To XLimit
If Black(Thres, I) > MxB(Thres) Then MxB(Thres) = Black(Thres, I)
If Black(Thres, I) = MxB(Thres) Then BlackSep(Thres) = I
Next I

Graph1.AutoInc = 0
Graph1.NumPoints = XLimit

For M = 1 To XLimit
If Black(Colchange, M) = Black(Thres, M) Then Icr = Icr + 1
Next M

For J = 1 To XLimit

If Icr > XLimit - 3 Then Graph1.ColorData = 12 Else Graph1.ColorData = 2
Graph1.ThisPoint = J
Graph1.XPosData = J
Graph1.GraphData = Black(Thres, J)
List2.AddItem "        " & Thres & "          " & J & "          " & Black(Thres, J)
Next J

Graph1.GraphType = 6
Graph1.GraphStyle = 2
Graph1.BottomTitle = "Pk to Pk Separation/Pixels"
Graph1.LeftTitle = "N"
Graph1.Background = 14
Graph1.DrawMode = 2

Colchange = Thres

End Sub
```

Plot of the halftone cycle size distribution for the background areas.

```
Private Sub Command1_Click()

ClkW = ClkW + 1

For Thres = 1 To LastThres

List1.AddItem Gradient(Thres)
XLimit = 710
For I = 1 To XLimit
If White(Thres, I) > MxW(Thres) Then MxW(Thres) = White(Thres, I)
If White(Thres, I) = MxW(Thres) Then WhiteSep(Thres) = I
Next I
```

```
Open "c:\cmtutil\vidpres\Files\FqDist1.csv" For Output As #2
Graph1.AutoInc = 0
Graph1.NumPoints = XLimit

For J = 4 To XLimit

Graph1.ThisPoint = J
Graph1.XPosData = J
Graph1.GraphData = White(Thres, J)
Write #2, Black(Thres, J), White(Thres, J)

Next J

Graph1.GraphType = 6
Graph1.GraphStyle = 2
Graph1.BottomTitle = "Pk to Pk Separation/Pixels"
Graph1.LeftTitle = "N"
Graph1.Background = 15
Graph1.DrawMode = 2
Close #2

Next Thres
If ClkW = 1 Then Form6.Command1.Value = True
End Sub

Private Sub Form_Load()
ClkW = ClkW
ColChW = ColChW
Colchange = Colchange
End Sub

Private Sub List1_Click()
Icr = 0

For YesThres = 1 To LastThres
If Val(List1.Text) >= 0.99 * Gradient(YesThres) And Val(List1.Text) <= 1.01 *
Gradient(YesThres) Then Thres = YesThres
List2.Clear
Next YesThres
Graph1.DataReset = 9
For I = 1 To XLimit
If White(Thres, I) > MxW(Thres) Then MxW(Thres) = White(Thres, I)
If White(Thres, I) = MxW(Thres) Then WhiteSep(Thres) = I
Next I
Graph1.AutoInc = 0
Graph1.NumPoints = XLimit

For M = 1 To XLimit

If White(Colchange, M) = White(Thres, M) Then Icr = Icr + 1

Next M

For J = 1 To XLimit

If Icr > XLimit - 3 Then Graph1.ColorData = 12 Else Graph1.ColorData = 2
Graph1.ThisPoint = J
Graph1.XPosData = J
Graph1.GraphData = White(Thres, J)
List2.AddItem "        " & Thres & "          " & J & "          " & White(Thres, J)

Next J
```

```
Graph1.GraphType = 6
Graph1.GraphStyle = 2
Graph1.BottomTitle = "Pk to Pk Separation/Pixels"
Graph1.LeftTitle = "N"
Graph1.Background = 15
Graph1.DrawMode = 2

Colchange = Thres

End Sub
```

Plot of the total greylevel distribution, the grey level distribution of the image areas and the grey level distribution of the background areas.

```
Private Sub Command1_Click()
IntSwitch = IntSwitch + 1
For Thres = 1 To LastThres
TotalGrey = 0
DivG = 1
DivB(Thres) = 1
DivW(Thres) = 1
Graph1.DataReset = 1
Graph1.DataReset = 8
Graph1.AutoInc = 0
Graph1.NumPoints = 256
For J = 1 To 256
Graph1.ThisPoint = J
Graph1.XPosData = J
Graph1.GraphData = Contrast(J)
Graph1.ColorData = 1

If IntSwitch = 1 Then TotalGrey = TotalGrey + (J * Contrast(J))
If IntSwitch = 1 Then DivG = DivG + Contrast(J)
If IntSwitch = 1 Then TotalImag(Thres) = TotalImag(Thres) + (J * Imag(Thres, J))
If IntSwitch = 1 Then DivB(Thres) = DivB(Thres) + Imag(Thres, J)
If IntSwitch = 1 Then TotalDiffBackGround(Thres) = TotalDiffBackGround(Thres) + (J *
Backgrd(Thres, J))
If IntSwitch = 1 Then DivW(Thres) = DivW(Thres) + Backgrd(Thres, J)
Next J

Graph1.GraphType = 6
Graph1.GraphStyle = 2
Graph1.BottomTitle = "Greylevel"
Graph1.LeftTitle = "N"
Graph1.Background = 15
Graph1.DrawMode = 2
Next Thres

If IntSwitch = 1 Then Form8.Command1.Value = True
End Sub

Private Sub Command2_Click()
Thres = InputBox("Threshold=  ", Thres)
Graph1.DataReset = 1
Graph1.DataReset = 8
Graph1.AutoInc = 0
Graph1.NumPoints = 256
For J = 1 To 256
Graph1.ThisPoint = J
Graph1.XPosData = J
Graph1.GraphData = Imag(Thres, J)
Graph1.ColorData = 0
```

```
Next J
Graph1.GraphType = 6
Graph1.GraphStyle = 2
Graph1.BottomTitle = "Greylevel"
Graph1.LeftTitle = "N"
Graph1.Background = 15
Graph1.DrawMode = 2

End Sub

Private Sub Command3_Click()
Thres = InputBox("Threshold=  ", Thres)
Graph1.DataReset = 1
Graph1.DataReset = 8
Graph1.AutoInc = 0
Graph1.NumPoints = 256
For J = 1 To 256
Graph1.ThisPoint = J
Graph1.XPosData = J
Graph1.GraphData = Backgrd(Thres, J)
Graph1.ColorData = 7

TotalDiffBackGround(Thres) = TotalDiffBackGround(Thres) + Backgrd(Thres, J)
Next J

Graph1.GraphType = 6
Graph1.GraphStyle = 2
Graph1.BottomTitle = "Greylevel"
Graph1.LeftTitle = "N"
Graph1.Background = 15
Graph1.DrawMode = 2


End Sub

Private Sub Form_Load()
IntSwitch = IntSwitch
End Sub
```

This forms outputs CSV datafiles of the extracted features that can be imported into MATLAB. It also can display the data values for the extracted features.

```
Private Sub Command1_Click()

For Thres = 1 To LastThres
List1.Clear
Open "c:\cmtutil\vidpres\files\Vital3.csv" For Output As #3
Open "c:\cmtutil\vidpres\files\vital2.csv" For Append As #30
Open "c:\cmtutil\vidpres\files\Vital1.csv" For Append As #40

List1.AddItem Thres & "   " & Gradient(Thres) & "   " & TotalGrey / DivG & "       " &
BlackSep(Thres) & "       " & MxB(Thres) & "       " & TotalImag(Thres) / DivB(Thres) & "
" & TotalImag(Thres) & "       " & WhiteSep(Thres) & "       " & MxW(Thres) & "       " &
TotalDiffBackGround(Thres) / DivW(Thres)
Write #40, Diff0(Thres), Diff1(Thres), Diff2(Thres), Diff3(Thres), Diff4(Thres), Diff5(Thres).
Diff6(Thres), Diff7(Thres), Diff8(Thres), Diff9(Thres), Diff10(Thres), Diff11(Thres),
Diff12(Thres), Diff13(Thres), Diff14(Thres), Diff15(Thres), TotalImag(Thres), BlackSep(Thres),
WhiteSep(Thres), Diff17(Thres), Diff18(Thres), Diff19(Thres), Diff20(Thres), Diff21(Thres),
Diff22(Thres), Diff23(Thres), Diff24(Thres), Diff25(Thres), Diff26(Thres), Diff27(Thres),
Diff28(Thres), Diff29(Thres), Diff30(Thres), Diff31(Thres), Diff32(Thres), Diff33(Thres),
Diff34(Thres), Diff35(Thres), Diff36(Thres), Diff37(Thres), Diff38(Thres), Diff39(Thres),
```

Diff40(Thres), Diff41(Thres), Diff42(Thres), Diff43(Thres), Diff44(Thres), Diff45(Thres), Diff46(Thres), Diff47(Thres), Diff48(Thres), Diff49(Thres)

Write #3, Diff0(Thres) * 10#
Write #3, Diff1(Thres) * 10#
Write #3, Diff2(Thres) * 100#
Write #3, Diff3(Thres) * 1000#
Write #3, Diff4(Thres) * 1000#
Write #3, Diff5(Thres) * 10000#
Write #3, Diff6(Thres) * 10#
Write #3, Diff7(Thres) / 10#
Write #3, Diff8(Thres) / 100#
Write #3, Diff9(Thres) / 10#
Write #3, Diff10(Thres)
Write #3, Diff11(Thres) / 10#
Write #3, Diff12(Thres) / 1000#
Write #3, Diff13(Thres) / 10#
Write #3, Diff14(Thres) / 1000#
Write #3, Diff15(Thres) / 1000#
Write #3, TotalImag(Thres) / 10000000#
Write #3, BlackSep(Thres) / 10#
Write #3, WhiteSep(Thres) / 10#
Write #3, Diff17(Thres) * 10#
Write #3, Diff18(Thres) / 10#
Write #3, Diff19(Thres) * 10#
Write #3, Diff20(Thres)
Write #3, Diff21(Thres) * 10#
Write #3, Diff22(Thres)
Write #3, Diff23(Thres) / 10#
Write #3, Diff24(Thres) / 100#
Write #3, Diff25(Thres)
Write #3, Diff26(Thres)
Write #3, Diff27(Thres)
Write #3, Diff28(Thres)
Write #3, Diff29(Thres)
Write #3, Diff30(Thres)
Write #3, Diff31(Thres)
Write #3, Diff32(Thres)
Write #3, Diff33(Thres)
Write #3, Diff34(Thres)
Write #3, Diff35(Thres)
Write #3, Diff36(Thres)
Write #3, Diff37(Thres)
Write #3, Diff38(Thres)
Write #3, Diff39(Thres)
Write #3, Diff40(Thres)
Write #3, Diff41(Thres)
Write #3, Diff42(Thres)
Write #3, Diff43(Thres)
Write #3, Diff44(Thres)
Write #3, Diff45(Thres)
Write #3, Diff46(Thres)
Write #3, Diff47(Thres)
Write #3, Diff48(Thres)
Write #3, Diff49(Thres)

Write #30, Diff0(Thres) * 10#
Write #30, Diff1(Thres) * 10#
Write #30, Diff2(Thres) * 100#
Write #30, Diff3(Thres) * 1000#
Write #30, Diff4(Thres) * 1000#
Write #30, Diff5(Thres) * 10000#
Write #30, Diff6(Thres) * 10#

```
Write #30, Diff7(Thres) / 10#
Write #30, Diff8(Thres) / 100#
Write #30, Diff9(Thres) / 10#
Write #30, Diff10(Thres)
Write #30, Diff11(Thres) / 10#
Write #30, Diff12(Thres) / 1000#
Write #30, Diff13(Thres) / 10#
Write #30, Diff14(Thres) / 1000#
Write #30, Diff15(Thres) / 1000#
Write #30, TotalImag(Thres) / 10000000#
Write #30, BlackSep(Thres) / 10#
Write #30, WhiteSep(Thres) / 10#
Write #30, Diff17(Thres) * 10#
Write #30, Diff18(Thres) / 10#
Write #30, Diff19(Thres) * 10#
Write #30, Diff20(Thres)
Write #30, Diff21(Thres) * 10#
Write #30, Diff22(Thres)
Write #30, Diff23(Thres) / 10#
Write #30, Diff24(Thres) / 100#
Write #30, Diff25(Thres)
Write #30, Diff26(Thres)
Write #30, Diff27(Thres)
Write #30, Diff28(Thres)
Write #30, Diff29(Thres)
Write #30, Diff30(Thres)
Write #30, Diff31(Thres)
Write #30, Diff32(Thres)
Write #30, Diff33(Thres)
Write #30, Diff34(Thres)
Write #30, Diff35(Thres)
Write #30, Diff36(Thres)
Write #30, Diff37(Thres)
Write #30, Diff38(Thres)
Write #30, Diff39(Thres)
Write #30, Diff40(Thres)
Write #30, Diff41(Thres)
Write #30, Diff42(Thres)
Write #30, Diff43(Thres)
Write #30, Diff44(Thres)
Write #30, Diff45(Thres)
Write #30, Diff46(Thres)
Write #30, Diff47(Thres)
Write #30, Diff48(Thres)
Write #30, Diff49(Thres)

Close #3
Close #30
Close #40
Next Thres
Text1.Text = "Grad"
Text2.Text = "Total Grey"
NL = Chr(13) + Chr(10)
Text3.Text = "Image" & NL & "Frequency"
Text4.Text = "Peak Count" & NL & "For Image"
Text5.Text = "Average Intensity" & NL & "For Image"
Text6.Text = "Total Image" & NL & "Count"
Text7.Text = "Background" & NL & "Frequency"
Text8.Text = "Peak Count" & NL & "Background"
Text9.Text = "Average Intensity" & NL & "Background"

End Sub
```

```
Rem Unload MDIForm1
```

This form displays the grey level values of the image and background in tabular form.

```
Private Sub Command1_Click()

SigmaImag = 1
SigmaBackgrd = 1

Open "C:\cmtutil\vidpres\Files\greys1.csv" For Output As #7
For Thres = 1 To LastThres
List1.AddItem "Grad=" & Gradient(Thres)
SigmaImag = 1
SigmaBackgrd = 1
BP1 = 0
BP2 = 0
BP3 = 0
WP1 = 0
WP2 = 0
WP3 = 0

For M = 1 To 256

SigmaImag = Imag(Thres, M) + SigmaImag
SigmaBackgrd = Backgrd(Thres, M) + SigmaBackgrd

Next M

For J = 1 To 256

Write #7, J, Contrast(J), Imag(Thres, J) / SigmaImag, Backgrd(Thres, J) / SigmaBackgrd
If Thres = Lev Then List1.AddItem Format(J, " 000") & Format(Contrast(J), "  0000000") &
Format(Imag(Thres, J), "   000000") & Format(Backgrd(Thres, J), "    000000") &
Format((Imag(Thres, J) + Backgrd(Thres, J)), "   000000")

Next J
B1 = 160

For bk1 = B1 To 256
If Imag(Thres, bk1) / SigmaImag > BP1 Then BP1 = Imag(Thres, bk1) / SigmaImag
If Imag(Thres, bk1) / SigmaImag = BP1 Then BLocation1 = bk1
Next bk1
Imag(Thres, BLocation1) = 0#
For bk2 = B1 To 256
If Imag(Thres, bk2) / SigmaImag > BP2 Then BP2 = Imag(Thres, bk2) / SigmaImag
If Imag(Thres, bk2) / SigmaImag = BP2 Then BLocation2 = bk2
Next bk2
Imag(Thres, BLocation2) = 0#
For bk3 = B1 To 256
If Imag(Thres, bk3) / SigmaImag > BP3 Then BP3 = Imag(Thres, bk3) / SigmaImag
If Imag(Thres, bk3) / SigmaImag = BP3 Then BLocation3 = bk3
Next bk3

Diff17(Thres) = BP1
Diff18(Thres) = BLocation1

Diff19(Thres) = BP2
Diff20(Thres) = BLocation2 - BLocation1
If Diff20(Thres) > 10 Then Diff20(Thres) = 10
If Diff20(Thres) < -10 Then Diff20(Thres) = -10
Diff21(Thres) = BP3
Diff22(Thres) = BLocation3 - BLocation1
If Diff22(Thres) > 10 Then Diff22(Thres) = 10
```

```
If Diff22(Thres) < -10 Then Diff22(Thres) = -10


W1 = 160

For wk1 = 1 To W1
If Backgrd(Thres, wk1) / SigmaBackgrd > WP1 Then WP1 = Backgrd(Thres, wk1) /
SigmaBackgrd
If Backgrd(Thres, wk1) / SigmaBackgrd = WP1 Then WLocation1 = wk1
Next wk1
Backgrd(Thres, WLocation1) = 0#
For wk2 = 1 To W1
If Backgrd(Thres, wk2) / SigmaBackgrd > WP2 Then WP2 = Backgrd(Thres, wk2) /
SigmaBackgrd
If Backgrd(Thres, wk2) / SigmaBackgrd = WP2 Then WLocation2 = wk2
Next wk2
Backgrd(Thres, WLocation2) = 0#
For wk3 = 1 To W1
If Backgrd(Thres, wk3) / SigmaBackgrd > WP3 Then WP3 = Backgrd(Thres, wk3) /
SigmaBackgrd
If Backgrd(Thres, wk3) / SigmaBackgrd = WP3 Then WLocation3 = wk3
Next wk3

Diff27(Thres) = WP1
Diff28(Thres) = WLocation1
Diff29(Thres) = WP2
Diff30(Thres) = WLocation2 - WLocation1
If Diff30(Thres) > 10 Then Diff30(Thres) = 10
If Diff30(Thres) < -10 Then Diff30(Thres) = -10
Diff31(Thres) = WP3
Diff32(Thres) = WLocation3 - WLocation1
If Diff32(Thres) > 10 Then Diff32(Thres) = 10
If Diff32(Thres) < -10 Then Diff32(Thres) = -10

Next Thres


Close #7
Form9.Command1.Value = True
End Sub
```

This form displays the grey level distribution and the total number of detected edges for each individual Y value across the display.

```
Private Sub Command1_Click()
Open "C:\cmtutil\vidpres\Files\scan1.csv" For Output As #6
For ThresScan = 1 To LastThres
Rem MinLine = ScanLine(ThresScan, XStart) / LineCount(ThresScan, XStart)
Rem For M = XStart To XFinish - 1
Rem If ScanLine(ThresScan, M) / LineCount(ThresScan, M) <= MinLine Then MinLine =
ScanLine(ThresScan, M) / LineCount(ThresScan, M)
Rem Next M

If MinLine < 80 Then MinLine = 80
Graph1.AutoInc = 0
Graph1.NumPoints = XFinish
For J = XStart To XFinish
Graph1.ThisPoint = J
Graph1.XPosData = J
Graph1.GraphData = ScanLine(ThresScan, J) / (YFinish - YStart)
Write #6, ScanLine(ThresScan, J) / (YFinish - YStart + 1)
Next J
Graph1.GraphType = 6
```

```
Graph1.GraphStyle = 2
Graph1.DrawMode = 2

Next ThresScan

Close #6
Form7.Command1.Value = True
End Sub

Private Sub Command2_Click()

Open "c:\cmtutil\vidpres\Files\Line1.csv" For Output As #4
Open "c:\cmtutil\VidPres\Files\Sto1.csv" For Append As #5
Open "c:\cmtutil\VidPres\Files\FreqSto1.csv" For Append As #15
For ThresFreq = 1 To LastThres
Graph2.AutoInc = 0
Graph3.AutoInc = 0
Graph2.NumPoints = XFinish
Graph3.NumPoints = XFinish
Vectr = -1
For J = XStart To XFinish
Graph2.ThisPoint = J
Graph3.ThisPoint = J
Graph2.XPosData = J
Graph3.XPosData = J

Graph3.GraphData = DYUpLine(ThresFreq, J)
If J = YFinish Then Graph3.GraphData = 0
Graph2.GraphData = DYDownLine(ThresFreq, J)
Write #4, DYUpLine(ThresFreq, J), DYDownLine(ThresFreq, J)
Next J
For X = 1 To XFinish

Write #15, BLineFreq(X, 4), BLineFreq(X, 5), BLineFreq(X, 6), BLineFreq(X, 7), BLineFreq(X,
8), BLineFreq(X, 9), BLineFreq(X, 10), BLineFreq(X, 11), BLineFreq(X, 12), BLineFreq(X, 13),
BLineFreq(X, 14), BLineFreq(X, 15), BLineFreq(X, 16), BLineFreq(X, 17), BLineFreq(X, 18),
BLineFreq(X, 19), BLineFreq(X, 20), BLineFreq(X, 21), BLineFreq(X, 22), BLineFreq(X, 23),
BLineFreq(X, 24), BLineFreq(X, 25), BLineFreq(X, 26), BLineFreq(X, 27), BLineFreq(X, 28),
BLineFreq(X, 29), BLineFreq(X, 30), BLineFreq(X, 31), BLineFreq(X, 32), BLineFreq(X, 33),
BLineFreq(X, 34), BLineFreq(X, 35)

Next X

Graph2.GraphType = 6
Graph3.GraphType = 6
Graph2.GraphStyle = 2
Graph3.GraphStyle = 2
Graph2.DrawStyle = 0
Graph3.DrawStyle = 0
Graph2.ThickLines = 1
Graph3.ThickLines = 1
Graph2.PatternedLines = 0
Graph3.PatternedLines = 0
Graph2.PatternData = 1
Graph3.PatternData = 1
Graph2.DrawMode = 2
Graph3.DrawMode = 2
List1.Clear

For X = 100 To 400
```

```
Rem List1.AddItem X & "     " & BLineFreq(ThresFreq, X) & "    " & BTot(ThresFreq, X) &
"Problem" & Stoch(ThresFreq, X) & "    " & TotalStoch(ThresFreq) / TotStoch(ThresFreq) & "
" & Diff(ThresFreq)

Next X
Write #5, Diff0(ThresFreq), Diff1(ThresFreq), Diff2(ThresFreq), Diff3(ThresFreq),
Diff4(ThresFreq), Diff5(ThresFreq), Diff6(ThresFreq), Diff7(ThresFreq), Diff8(ThresFreq),
Diff9(ThresFreq)
Next ThresFreq
Close #4
Close #5
Close #15
Form9.Command1.Value = True

End Sub
```

MDI form code that is used to connect the individual child forms given above.

```
Private Sub MDIForm_Load()

End Sub

Private Sub mnuFile_Click()

End Sub

Private Sub mnuFileNew_Click()
Form2.Show
Form3.Show
Form4.Show
Form5.Show
Form6.Show
Form7.Show
Form8.Show
Form9.Show
End Sub

Private Sub mnuRun_Click()
Load Form2
Load Form3
Load Form4
Load Form5
Load Form6
Load Form7
Load Form8
Load Form9
End Sub


Private Sub mnuFile_Click()

End Sub

Private Sub mnuFileNew_Click()
Form2.Show
Form3.Show
Form4.Show
Form5.Show
Form6.Show
Form7.Show
Form8.Show
Form9.Show
End Sub
```

```
Private Sub mnuRun_Click()

Load Form2
Load Form3
Load Form4
Load Form5
Load Form6
Load Form7
Load Form8
Load Form9

End Sub
```

Appendix. 2. An example of a neural network program that was used in the thesis.

This neural network program takes inputs from the pre-processing program in appendix 1 and uses a backpropagation neural network to process the extracted features. In this example, the neural network processes a data file containing the features from images of the letters G and r. The network classifies the images according to their printing process. The code is written in MATLAB.

The data file is opened

```
fid = fopen('c:\Gr.csv','r');
A = fscanf(fid,'%g');
status = fclose(fid);
```

The code is formatted into a 52 x m matrix, where m is the length of the datafile

```
B = reshape(A,52,length(A)/52)';
```

The inputs are specified from the 52 x m matrix.

```
Par1 = 1
Par2 = 8
Par3 = 17
Par4 = 10
Par5 = 11
```

The training and validation datasets are selected.

The training set.

```
C=B(1:15,1:52);
D=B(16:30,1:52);

E=B(31:45,1:52);
F=B(46:60,1:52);

Trlaser1=C(:,Par1);
Trlaser2=C(:,Par2);
Trlaser3=C(:,Par3);
Trlaser4=C(:,Par4);
Trlaser5=C(:,Par5);
```

```
Trink1=D(:,Par1);
Trink2=D(:,Par2);
Trink3=D(:,Par3);
Trink4=D(:,Par4);
Trink5=D(:,Par5);
```

The validation set for the letter r.

```
Vrlaser1=E(:,Par1);
Vrlaser2=E(:,Par2);
Vrlaser3=E(:,Par3);
Vrlaser4=E(:,Par4);
Vrlaser5=E(:,Par5);

Vrink1=F(:,Par1);
Vrink2=F(:,Par2);
Vrink3=F(:,Par3);
Vrink4=F(:,Par4);
Vrink5=F(:,Par5);

J1 = [Trlaser1,Trlaser2,Trlaser3,Trlaser4,Trlaser5];
J2 = [Trink1,Trink2,Trink3,Trink4,Trink5];

NTrain = [J1;J2]';
K1 = [Vrlaser1,Vrlaser2,Vrlaser3,Vrlaser4,Vrlaser5];
K2 = [Vrink1,Vrink2,Vrink3,Vrink4,Vrink5];
NVal1 = [K1;K2]';
```

The validation set for the letter G.

```
E=B(61:75,1:52);
F=B(76:90,1:52);

VGlaser1=E(:,Par1);
VGlaser2=E(:,Par2);
VGlaser3=E(:,Par3);
VGlaser4=E(:,Par4);
VGlaser5=E(:,Par5);

VGink1=F(:,Par1);
VGink2=F(:,Par2);
VGink3=F(:,Par3);
VGink4=F(:,Par4);
VGink5=F(:,Par5);

L1 = [VGlaser1,VGlaser2,VGlaser3,VGlaser4,VGlaser5];
L2 = [VGink1,VGink2,VGink3,VGink4,VGink5];
NVal2 = [L1;L2]';
```

These are the the target values for the training set.

```
T=[ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ,...
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
```

These are the training parameters.

```
tp = [1 200 .1 .005]
```

This states the network structure and the transfer function specifications.

```
[W1,b1,W2,b2] = initff(NTrain,4,'tansig',T,'logsig')
```

This specifies that a backpropagation algorithm employing the Levenberg-Marquart algorithm is used.

```
[W1,b1,W2,b2,tr] = trainlm(W1,b1,'tansig',W2,b2,'logsig', NTrain,T,tp)
```

These are the training results.

```
ResTrain = simuff(NTrain,W1,b1,'tansig',W2,b2,'logsig')
```

These are the validation results for the letter r.

```
ResVal1 = simuff(NVal1,W1,b1,'tansig',W2,b2,'logsig')
```

These are the validation results for the letter.

```
ResVal2 = simuff(NVal2,W1,b1,'tansig',W2,b2,'logsig')
```