The Nottingham Trent University

Department of Manufacturing Engineering

Use of Machine Vision in the Automation of Printed Circuit Manufacture

John Keat

This thesis is submitted to The Nottingham Trent University as partial fulfilment for the degree of

Master of Philosophy.

August 1993

Contents

Acknowledgements

The list of people who have helped me through this work and it's writing up is a long one, and those who have put up with my weird behaviour patterns induced by the ups and downs is even longer. To all these people, thank you.

Dawn, my wife, for being unbelievably patient.

My mum and dad, firstly for having me and then for patching up all the subsequent disasters.

Martin and Heather, for support, friendship and for keeping going when things looked dire, thanks for all your efforts.

Dom, for putting up with my mad half hours. Always remember, only risk one eye.

Bala, for inspiration when things looked doomed, and for the next three years!

Alan Hague, for all the worries this project caused him, thank you.

Professor George Thompson for funding the last year of this work.

Phil Breedon and all the technician staff from the Department of Manufacturing Engineering, especially Jez and all the crew in the work shop.

Jim Corlett and all the librarian staff for the sterling work they do.

Dr. Max Robinson for his expertise, advice and for reading through this thesis.

To all my brothers and sisters in Christ who have supported and prayed for me from the very start of this work. You will be rewarded!

Abstract

Use of conducting inks and screen printing has now made it possible to produce multilayer printed circuit boards (PCB's) of increased complexity and reliability. High yield production of such circuitry requires a high degree of accuracy and repeatability in both set up and production, attributes that current screen printing technology is barely able to achieve, even at the top end of the market. One solution to this problem is the integration of a machine vision system with an automated screen printing process.

This thesis describes the implementation of a machine vision system for use in the set-up and registration of a Sveciamatic SM screen printing machine, and its particular application with the production of multilayered PCB's. The screen printing process and its application to PCB manufacturing is briefly described and the reasons for requiring machine vision in this application are discussed. The alternative routes to automation of the process, including the incorporation of machine vision, are described with the advantages and disadvantages of each outlined.

The production of the vision system, from an initial prototype manually assisted registration system, to the final automated set up and registration system is detailed. This includes the selection of hardware and the development of software, along with the relationship of the vision system with the overall control of the machine. The problems that can be encountered when incorporating machine vision within a screen printing process are addressed, with solutions where relevant. Through all the stages of development all aspects of image enhancement and processing that were tried are explained along with possible alternatives, with explanations of why each technique was chosen. Throughout the work current thinking on each subject is mentioned, along with its relevance to the work.

Finally the future of the project and its applications are discussed, with consideration of recent developments in image processing and computing. This includes an outline of both short and long term possibilities for the future development of this work.

Machine vision, as a powerful tool, has been poorly taken up by the manufacturers of screen printing equipment. Where it has been used it is generally no more than an alignment aid. For screen printing to remain a strong contender in the PCB manufacturing industry the manufacturers of printing machinery will have to join the lead taken by those few who have realised the potential improvements that properly implemented machine vision can give.

"What is this life if, full of care,
We have no time to stand and stare?"

Leisure, W. H. Davies.

Chapter 1

Introduction

Chapter 1: Introduction

1.1 The Project

The project was originally started as a collaboration between Nottingham Polytechnic and Keyboard Products Limited (KPL) of Market Harborough. The aim was to develop an automated screen printing machine, for the production of multilayer printed circuit boards. The developed machine should be able to cope with the demands of mass production of such boards, i.e. high production rates with high accuracy and minimal set-up time. The developments of the project were to be implemented on KPL assembly lines in Market Harborough and in Tennessee, USA.

KPL, the original funders of the project, were manufacturers of keyboards, membrane switches and control panels using conducting inks. They were interested in extending the manufacturing technologies to the production of multilayer PCB's using conducting inks, but had found that the production technology currently available was either not suitable or excessively expensive. They approached the Polytechnic explaining their needs, and this resulted in the production of a feasibility study **[1]**, which put forward two aims:-

1) to examine ways of deskilling the existing process, to allow an untrained operator to successfully run the system without detailed knowledge of the process,

2) to recommend possible solutions using turn key equipment that would be capable of fulfilling the aim above for multilayer PCB production.

The study discussed the problems and outlined the possible solutions. A research plan was put forward and accepted. The study recommend the employment of two research assistants (R.A.'s), one to carry out the research and development of the machine vision system (the author) and the other to conduct the research and development of the process control system (Dominic Fulford).

1.2 PCB Manufacture

1.2.1 Single and Double Layer PCB Manufacture- Conventional

A Printed Circuit Board (PCB) is a device that acts as a mounting structure for components within an electrical circuit and provides the connections between these components. The connections have conventionally taken the form of copper foil, creating robust circuits. Almost every commercially available electronic device has some form of PCB within it, therefore the market for PCB's is huge. According to Kuhn **[2]** in an article from Electronic Production the total European market for PCB's in 1990 was worth £2.7 billion, with Britain accounting for 11.5% of that market.

PCB's have traditionally been produced by a subtractive process carried out on boards, usually paper phenolic laminates or glass/epoxy, with a copper foil outer layer, the thickness of which depends on current and voltage requirements of the circuit. The desired circuit is printed (usually by means of a photo-imaging technique) onto the substrate with an etch resist, and then etched, using an etchant such as ferric chloride or chrome/sulphuric acid, to remove the excess copper. Double sided boards are produced by carrying out the above process on both sides of the board and connecting the two sides with plated through holes. The boards need a great deal of preparation and cleaning before production, and importantly this leads to the production of much hazardous waste, especially with the use of etchants and the waste copper that gives heavy metal waste. **Figure 1.1** shows the complex production route required to produce a single circuit, the route may have a wide variety of combinations of the components shown, and each additional process may introduce errors. The need for so many varied processes makes the mass manufacture of PCB's highly capital intensive.

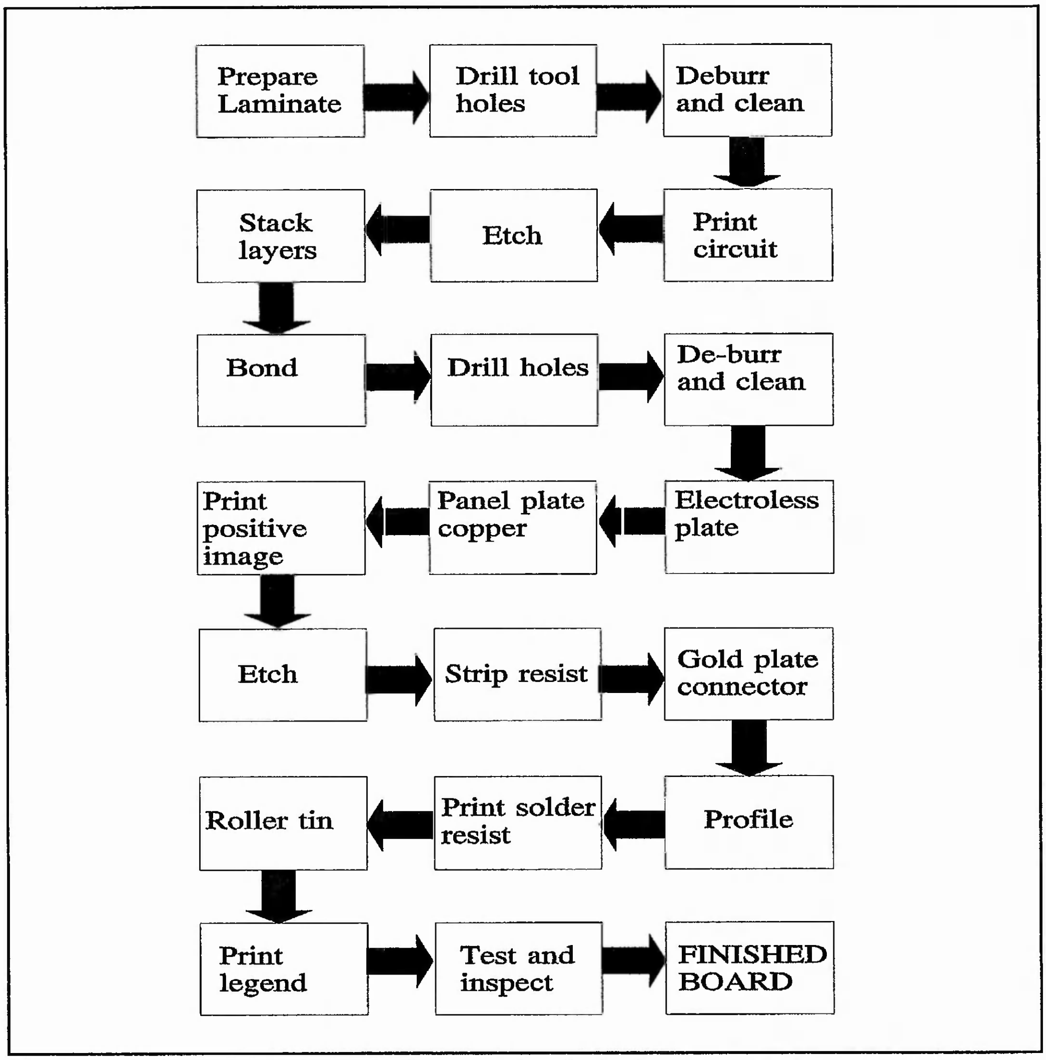


Figure 1.1 Flow chart of a typical conventional single layer rigid board manufacturing process

1.2.2 Multilayer PCB Manufacture- Conventional

There has been a need in recent years to increase circuit packing densities to higher and higher degrees. It may not always be possible to have all the components and connections on one board without going to excessive board sizes, it would therefore seem sensible to have the circuit on a number of smaller boards connected to each other.

This may again lead to an assembly that is too bulky. The alternative is to produce a multilayer board with the individual boards bonded together

Multilayer PCB's have been around for a few years, and until recently have been produced by taking standard single or double sided copper etch boards (as described above) and bonding them together using glass epoxy sheets, with plated through holes providing contact between the layers. It is an over simplification to say that it is just a case of bonding single or double sided boards together. The different layers are not joined at the edge as they would be if they were individual boards put together, the contact points are within the layer, and this requires complex design work. The registration between the layers is paramount if the connection between layers is to be complete. Once registered the different layers are bonded under heat and pressure.

There are various forms of multi-layer board, including double-multi boards (DMB), buried via hole (BVH) and sequential BVH boards, the most common being a configuration of four layers using the inner layers as power and ground planes. These boards are produced with the same thickness (typically 1.6 mm) as standard single or double layer boards. In addition multilayer boards tend to be more robust as the inner signal layers are protected from outside effects. The production route for producing boards of this type may require a number of different production processes in addition to those listed in **Figure 1.1**. These depend on the type of multilayer board produced and may include further cleaning (cleanliness between layers is vital for complete bonding), preparation, and the bonding process. All these additional operations result in high production reject rates (up to 35% **[3]**). Poorly fabricated multilayer boards are also prone to failure at the through hole points where improperly cured epoxy degrades after a time. This fault is not shown up by inspection and this leads to unreliable boards. Conventional multilayer PCB's are therefore very costly to make and not very adaptable if design changes are made.

1.2.3 Multilayer PCB Manufacture- Screen Printing

Recent developments in conducting inks and screen printing technology have made it possible to produce multilayered PCB's of increased complexity **[4]**, but with greatly increased reliability, with the possibility of reduced cost. This method of production of multilayer boards requires fewer production operations giving the potential for increased production rates and lower production rejects. The greater simplicity of the process can be seen by comparing **Figure 1.1** with **Figure 1.2**.

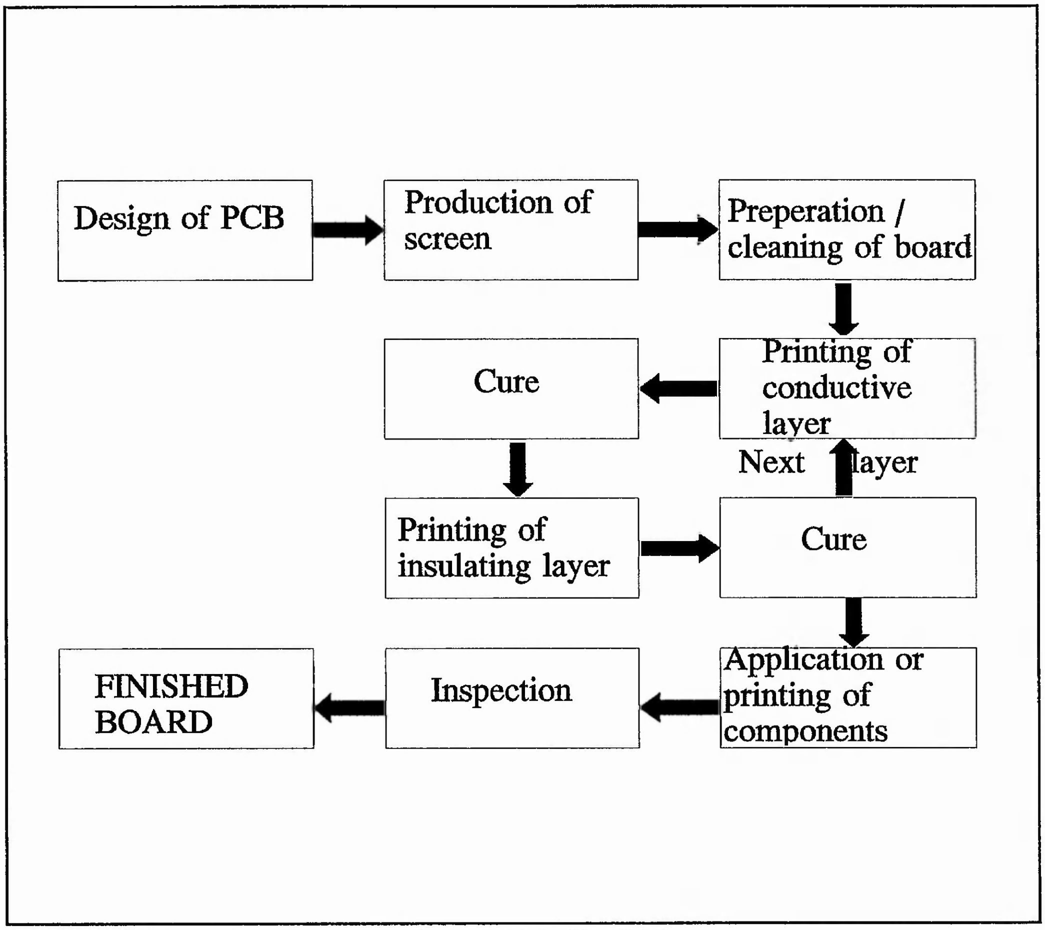

Design of PCB
Production of screen
Preperation / cleaning of board
Printing of conductive layer
Cure
Next layer
Printing of insulating layer
Cure
Application or printing of components
Inspection
FINISHED BOARD

Figure 1.2 Flow chart for production of thick film PCB

The production of multilayered boards using screen printing involves printing tracks with silver loaded inks to produce a thick film. This thick film gives it's name to the circuits produced by the process of printing conducting inks - Thick Film Circuits.

These conducting tracks are insulated from other layers by printed dielectric, it also gives the ability to print resistors and capacitors **[5]**. The number of layers per board that can be produced is currently limited by screen printing technology.

Screen printing of conducting inks is not just confined to the production of PCB's, it has applications in the production of membrane switches, flexible circuits and in rework of boards. PC manufacturers need to keep large stocks of computer motherboards in order to keep up with the large production runs. To keep at the forefront of computer technology the manufacturers regularly need to make changes to board designs. These boards may cost up to $65 to produce so the stock cannot be scrapped and the design changes have to be added, this is done by a process known as hard wiring, the incorporation of changes by soldering in additional wires by hand. This is a complex time consuming manual operation costing up to $65 a board (i.e. doubling the cost of the board). With the use of screen printing it is possible to print the design changes, either directly to the board or on a substrate that may be attached to the board. The estimated cost of this form of rework (according to KPL) was less than $10 per board.

1.3 Screen Printing

Screen printing is one of the oldest printing processes, and is known to date back to Ancient Egypt. The basic principles have changed little since those days. A screen, traditionally made of silk carrying a stencil of the required image, is coated with ink. A flexible squeegee is passed across the screen bringing it into contact with the item to be printed. The ink is forced through the gaps in the screen by the pressure difference between the substrate and the screen. When the squeegee has passed across the screen the image is left on the substrate. **Figure 1.3** shows a schematic of a typical screen printing machine and **Figure 1.4** Shows the process of creating a print. Screen printing is described as an off-contact printing process because the screen only comes into contact with the substrate when forced by the squeegee. The height of the off-contact (snap-off height) is critical to the quality and accuracy of the process and is described in section 1.3.1.

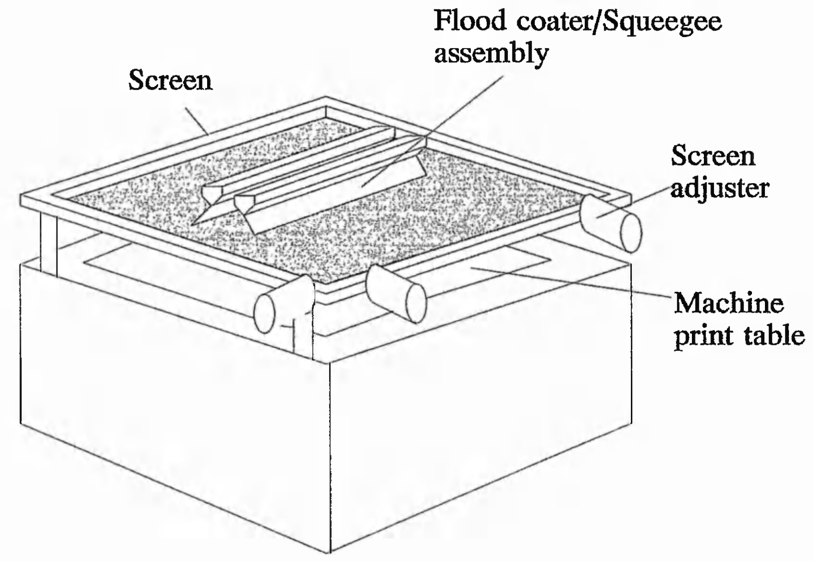


Figure 1.3 Diagram of basic screen printing machine

The main part of the screen is the mesh that is stretched across a strong metal frame. A stencil is applied to the screen in the form of a photo-sensitive emulsion. This emulsion comes in either sheet or liquid form depending on the characteristics desired. The artwork for the desired pattern is applied to the emulsion and then exposed to UV light, the unexposed emulsion is then washed away to leave the finished stencil. The size of the frame used for a particular print is an important factor affecting the print accuracy and quality. The larger the relative size of frame to the image, the less the snap-off height required. This leads to increased print accuracy and quality, with longer screen and stencil life and greater ease of printing. The frame is held in the upper frame of the printing machine, which pivots so that blank substrates may be placed on the machine bed (located by tooling pins or by edge guides).

The frames hold the screen clear of, and parallel to, the surface to be printed. **Figure 1.4** shows the process of producing a print. The ink to be printed is spread across the screen by the movement of a floodcoater (1) that is attached, with the squeegee holder to a moving carriage. The squeegee is passed over the screen (2) pushing the ink into the open holes in the mesh and also bringing the screen momentarily into contact with the substrate. The surface tension caused by the contact between the ink and the substrate forces the ink out of the mesh onto the substrate. After the squeegee has passed the screen peels back (3) from the substrate in a process known as peel off, this prevents smudging of the ink if it works correctly. The discrete particles of ink flow together to yield a uniform coverage of the substrate over the open areas of the screen.

1.3.1 Accuracy of Screen Printing

There are many aspects that affect the accuracy of the printing process. These can be put into two separate categories:-

1) Screen and artwork.
2) Process variables.

Screen and Artwork

For accurate optimum quality the selection of the correct screen is imperative **[6]**. The mesh may be made up of wide variety of materials such as polyester, steel or metal coated plastic threads or even a one piece metallic sheet (such as Hi-mesh). In addition each mesh can made up in different ways to produce a screen, with different thickness' of thread weaves and tensions. The thickness of the thread varies from 0.03 mm for a 165T polyester screen, to 0.12 mm for a 30T polyester screen, where 30T and 165T refer to the threads per cm. The threads per cm may go as high as 200 for use in

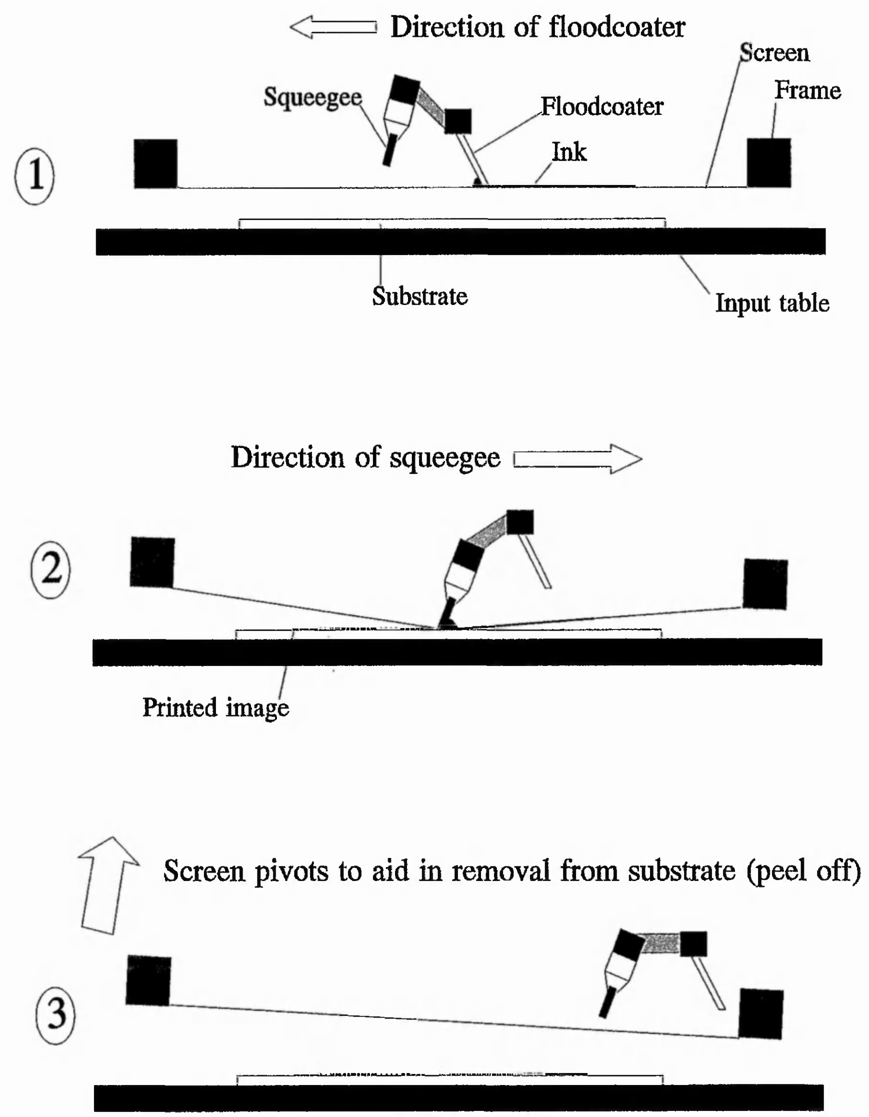


Figure 1.4 Diagram detail screen printing process.

very high accuracy work with low viscosity inks. As a rule higher thread densities are for finer work and/or use with low viscosity inks. The arrangement of the threads, or weave, can also be varied. A typical weave is shown in **Figure 1.5**, this diagram also shows how the mesh is held in the frame and the position of the stencil bearing the image to be printed.

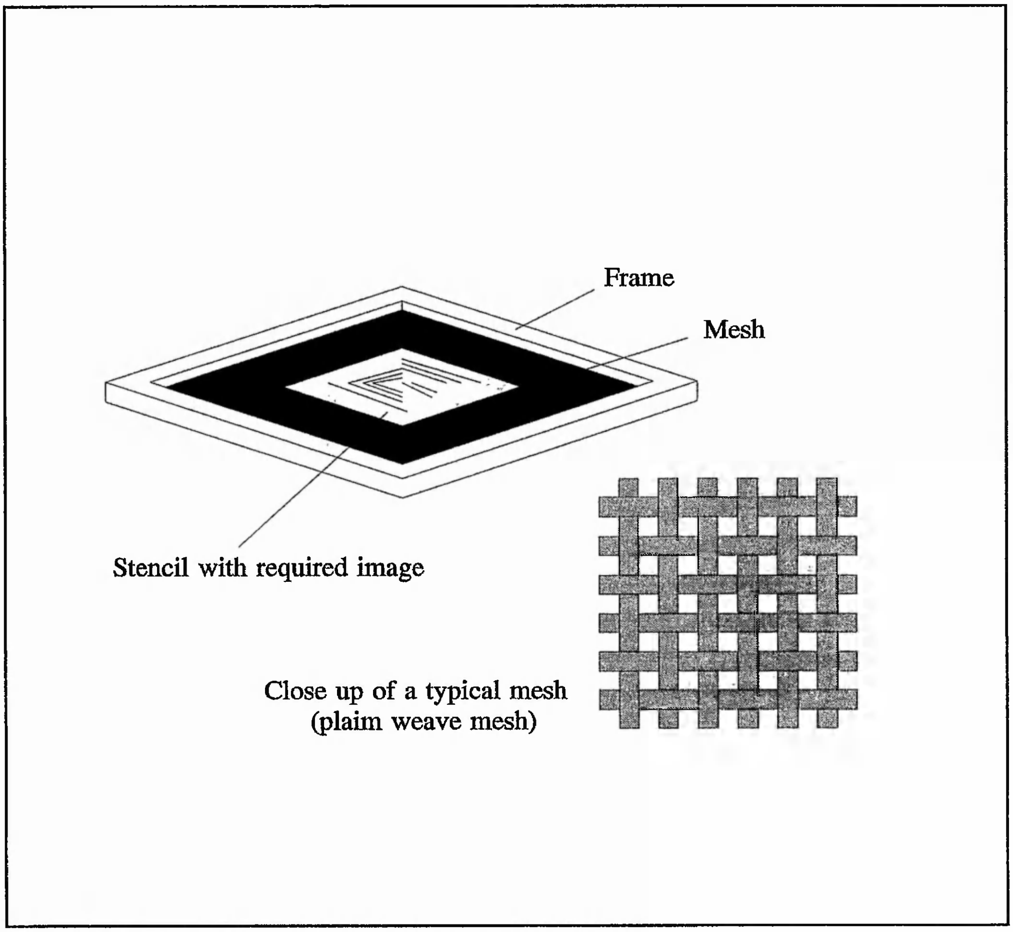


Figure 1.5 Diagram showing components that make up screen

Polyester screens are generally used for low accuracy textile or graphic printing as they tend to distort and stretch, but they are also hard wearing and reusable and tend to have a relatively long process life. For more precise work stainless steel mesh screens are used because they show a lesser degree of distortion for the reason that they can be used at higher tensions, typically 30-40 N/cm as opposed to 12-18 N/cm for polyester meshes. The result is also a longer life expectancy, and because they are less prone to stretching a smaller snap-off height can be used which further increases the accuracy of the screen. Recent developments have been towards screens made up of a mesh more like an electric shaver with hundreds of microscopic holes per square cm. The main advantage of such a material is that screen tension is more even in all directions, therefore giving a more dimensionally stable screen minimising distortion when printing.

The image to be printed is formed on a stencil attached to the screen mesh. There are two different categories of stencil; indirect and direct. Indirect stencils are imaged before application to the mesh while direct stencils are imaged while in direct contact with the mesh fabric. Indirect stencils reproduce the highest resolution and definition achievable by the screen process but tend to be less durable than direct stencils. Stencils come in different thicknesses and the choice of which thickness is used depends on the ink used and resolution required. Increasing the print resolution requires the use of a thinner stencil, and similarly the more viscous the ink the thinner the stencil.

An expert system for selecting the correct screens and stencils for any image and ink was developed as part of the project by Dominic Fulford **[7]**.

Process Variables

The selection of the correct printing parameters has a significant effect not only on the quality of the print, as would be expected, but also on the accuracy. In a later section this thesis will discuss the need for increased technological developments for screen printing to be able to keep up with the demands of the PCB industry. There is also a need for greater research into the actual process of producing a print. Young **[8]** made this point in an article in PC Fabrication in which he also put forward ideas for the increased performance of the process. He concluded that the screen printing industry had to develop its' technology rather than look to the past if it is to compete with other processes.

There are 48 different variables in the screen printing process **[9]** the most relevant being:-

•Snap-off height -

It has already been mentioned that the lower the snap-off height the more accurate the printing, but this gives a greater likelihood of image blurring because the screen remains in contact with the substrate for too long. With lower snap-off heights the amount of screen deformation is less because the squeegee has to move the screen a smaller distance to make contact with the substrate, this is shown in **Figure 1.6**. This has the effect of minimising image deformation.

•Screen alignment (registration) -

Without correct registration screen printing is unsuitable as a means of producing PCB's. Inaccurate registration limits the number of layers that can be printed, the greater the accuracy the registration system the more layers in a circuit can be produced to a greater level of accuracy.

•Integrity of original art work -

If the original artwork is poor the print produced can be no better, a print can only be as good as the original artwork used to produce the screen. During testing at the KPL production line problems were met with the artwork, which was held on an acetate sheet. The acetate expanded and contracted with changes in atmospheric humidity and temperature. This means that artwork for corresponding layers on a multilayer PCB must be produced under the same ambient conditions using the same equipment for optimal printing accuracy.

•Ink -

The ink selected for a print determines the screen and stencil types, therefore ink choice is less flexible. Ink quality must be maintained through out a production run as inks, particularly solvent based types, change viscosity during processing which has a direct effect on the print quality. This may cause blocking of the mesh and can result in incomplete prints. UV cured inks alleviate this problem as they tend to be more stable through a production run.

•Squeegee speed -

Increasing the squeegee speed increases the rate of peel-off, this gives greater print resolution but at too great a speed the ink does not have time to deposit on the substrate leading to incomplete prints.

•Squeegee pressure and hardness -

Softer squeegees tend to give less distortion across the print and are needed when printing on uneven surfaces such as multilayer printing. Harder squeegees are needed for harder smooth surfaces.

1.4 Automation

The aim of the project as stated earlier was to deskill the process, so that it can be successfully run by an untrained operator. The reasons for automating go well beyond this simple statement. Deskilling is not the only effect of automation, additional benefits of automation relevant to screen printing include:

- Registration accuracy,
- Set-up performance and speed,
- Integration within CIM environment,
- Health and safety.

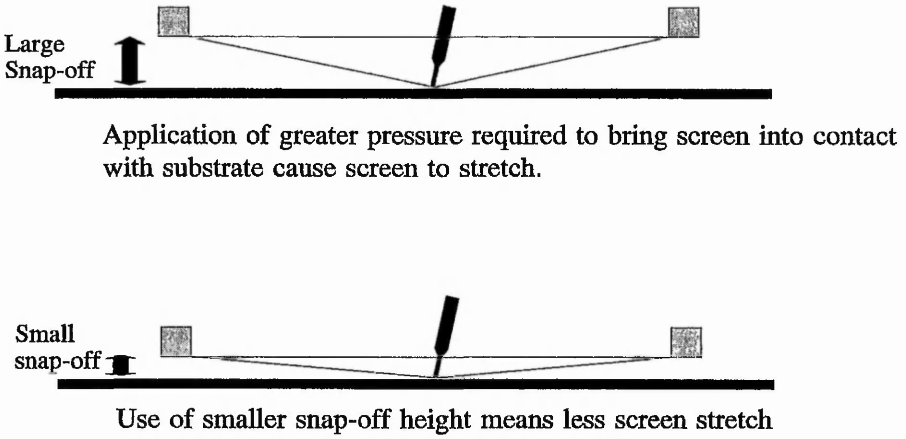

Large
Snap-off
Application of greater pressure required to bring screen into contact with substrate cause screen to stretch.
Small
snap-off
Use of smaller snap-off height means less screen stretch

Figure 1.6 Effects of snap-off height on screen deformation.

1.4.1 Registration Accuracy

The registration accuracy of a standard screen printing machine with manual alignment is approximately +/- 0.1 mm. This implies that when printing a multilayer board of ten layers the accuracy of the last layer may be misplaced, in extreme circumstances, by as much as 1.0 mm. By replacing the operator with an automatic screen alignment system the accuracy can be greatly increased because the subjective nature of alignment relying on the judgement and skill of the operator is replaced by a consistent reliable system.

During operation a screen is usually aligned with the board and fixed in position. No adjustment is made to the screen position until a certain drift in registration is seen. This does not allow for any variance in the position of the first layer on the board relative to the tooling pins. Traditionally boards have been registered to tooling pin positions, this means that the process accuracy relies on the precision of the drilling of the tooling holes. With the use of automated alignment to the circuit, rather than the tooling pins, it becomes feasible to align to every board in a production run. In addition by checking the position of the screen every print, it is possible to remove inaccuracies due to the movement of the input table and the position of the tooling pins. Screen alignment to every board is not possible with a manual system. The ability to register to every board is significant in multilayer PCB manufacture as it reduces the amount of process drift between each layer, allowing more layers to be produced at a much higher tolerance. Registration techniques may be applied to other areas of electronics including solder paste dispensing and component placement.

1.4.2 Set-up Performance and Speed

Screen printing is a highly skilled process, with a great deal of effort required in setting up, it may take a skilled operator 30 minutes to set up one machine for printing, this involves taking a number of test prints to align the screen and to get the printing conditions correct. This set up time is expensive as it involves 30 minute's production down time and the production of waste in the form of the test prints. The substrates used can be very expensive, as can silver inks and so it is obvious that minimizing the set up procedure by automation is virtually essential for economical mass production. Screen changes are regular as no company can devote one machine to a particular screen and each time a screen is replaced the same set up procedure has to be repeated. The shorter the production runs, the more critical it is to shorten set up time. Decreasing set-up time means that screen printing can become economically viable at much lower batch sizes than had been considered possible with conventional manufacturing processes.

1.4.3 Integration Within a CIM/CAM Environment

Once automated the screen printing process would allow a higher degree of in process monitoring of system variables than is possible with standard screen printing. The incorporation of an automated inspection system into the control system would allow adjustment of parameters during production, this closed loop control would greatly improve the consistency in print quality. It follows that by having such control over the production and inspection systems, production can be monitored as part of a factory wide Computer Integrated Manufacturing (CIM) or Computer Aided Manufacturing (CAM) system.

Computer integration of the whole production route opens up many possibilities for the screen printing of PCB's. The design process can be linked into the machine vision inspection process to give location of particular features of the board based on CAD (Computer Aided Design) positional data. The linking of CAD with machine vision has been a topic of much research in robotics. Liew and Tan **[10]** developed what they called a 'Vision System Computer-Aided Modeller' (VCM) to produce a model of an object based on the CAD data that can be interpreted by the robot system. It uses the same representation techniques that the robot vision system uses in the recognition process of locating position and orientation of parts. This work is similar to that done by Sanfeliu and Ananos **[11]** who developed a CAD based modelling system that aided in the recognition and location of partially occluded objects.

The prototype system developed by Mahon **[12]** for the automatic 3-D inspection of solder paste on surface mount PCB's is of particular relevance to the inspection of thick film PCB's. This system is driven by lay out information available from CAD data. These examples are based on 3-D applications but they still have value in screen printing. It would be possible to build up a model of the circuit from the initial CAD design, this could show areas that may be of particular interest to the inspection system such as thin tracks or pads of critical dimensions. From the model parametric data of the design could then be used for location of registration marks, the possibilities are numerous.

Screen printing has only recently taken on machine vision as anything more than an alignment tool, so its part in a CIM environment has not been properly exploited to date.

1.4.4 Health and Safety

The production of PCB's through conventional copper etch resist methods produces many environmental dangers. The cleaning of substrates uses CFC based products, the chemical etching produces heavy metal waste products and the solder contains lead. These factors are coming under growing legislation, particularly in the USA, either prohibiting or restricting their use. Screen printing offers the possibility to greatly improve safety in the production of PCB's. Screen printing, though, is not without its health problems. Over half the inks used in the screen printing industry are solvent based and are therefore a health risk. UV cured inks significantly reduce the environmental dangers of inks **[13]** by removing the need for solvents, but they still contain harmful chemicals.

Automation can cut ink contact to a minimum, thus providing a much safer, and more pleasant, working environment. Automation will not only reduce the need for skilled operators familiar with the process, it will also lessen the number of operators required to run a line as continual observation of the process will be taken over by the automated system.

1.5 Introduction to Machine Vision

Machine vision, or computer vision, has been in existence as a mostly theoretical science since the 1950's, but it is only in recent years that the technology has been available to implement most of the early work within an industrial environment. There is now a vast range of vision systems available to suit most budgets. Evidence of this can be found in surveys of imaging equipment **[14][15]**, these range from basic PC based frame grabbers capable of simple image processing functions, to powerful stand alone systems (based on high performance work stations) able to carry out a vast range of functions at high speed. The range of applications that have taken advantage of machine vision is vast, with possible future applications even greater. Some typical applications include:-

- Inspection of almost any manufactured object for integrity, location or orientation, e.g checking for damaged tiles **[16]**.
- Gauging, e.g engine bore sizes.
- Guidance for robots and AGV's.
- Surveillance.
- As a sensing aid for manufacturing processes, e.g In welding to detect the weld pool edge **[17]**.

Why Machine Vision?

The major advantage a human visual inspection system has over machine vision is that a human is able to view an image with much greater understanding. The human eye often sees things that are not present but are just implied through certain aspects of the image, for instance missing edges, or hidden parts due to shadows. This is also a major disadvantage of human visual inspection. Human inspection relies on the inspectors mental process at that particular time, which means that decisions can be clouded by fatigue and loss of concentration, and different inspectors may have different ideas of what is a 'good' or a 'poor' object. Quality is so often a subjective matter.

With the increased complexity of manufactured items, and the speed and sizes to which they are produced, there is no longer a choice between human inspection and machine vision. A human will have difficulty inspecting all the tracks of a densely packed hybrid circuit, or checking that all surface mount components have been correctly placed, or finding the centre of a registration mark to within +/- 0.01 mm. This task becomes impossible if these boards are being produced at hundred of parts an hour. This technology demands 100%, or near 100%, inspection, only machine vision can cope with these demands with the required reliability and consistency.

Schatz **[18]** in an article in Hybrid Circuit Technology discussed what he saw as the vital importance of the integration of machine vision technology with an automated screen printing system, particularly with the benefits this would give for the integration of the screen printing process into a fully automated SMT (Surface Mount Technology) environment. In the area of alignment, machine vision allows the process to be automated by removing operator input and the reliance on mechanical tooling holes and pins, and therefore giving maximum performance and efficiency.

Figure 1.7 shows the basic components that make up a vision process. The first stage is the acquisition of the image by the vision system camera, this is followed by Image Enhancement which is the process of improving the image, either through hardware such as lights and lenses (as part of the image acquisition), or through computer software. The aim of Image Enhancement also involves minimising the amount of data by eliminating irrelevant information, this may involve such processes as binarizing and filtering, processes that are explained in Chapter 2.

Once the best image has been achieved, the next step is to get the desired information from the image through Image Processing (see Chapter 3). This may involve separating the object of interest (OI) from the rest of the image, a procedure known as segmentation, and then coding the OI to extract the required features (most commonly edge detection). Analysis of the extracted features will give the information required, this could be centre finding or classification of the image. The final stage is that of decision making, taking the information from the image analysis and determining what action is to be taken. This depends on the vision application, for example, it may be a command to move a robot a certain position or to determine whether a component is present or in the correct orientation.

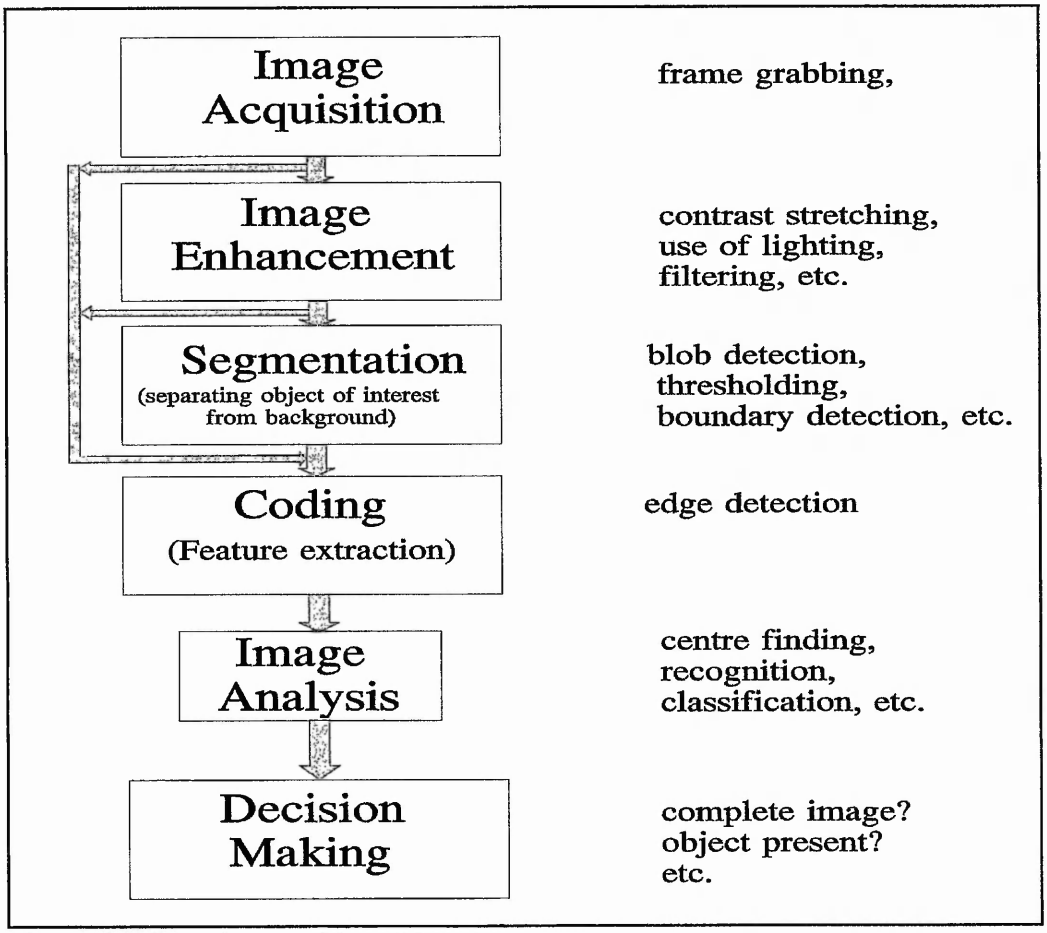


Figure 1.7 Steps in a machine vision process

Chapter 2

The Vision System and Image Enhancing Techniques

Chapter 2: The Vision System and Image Enhancing Techniques

2.1 The Vision System

Chapter 1 introduced the idea of machine vision explaining why it is needed and giving examples of applications. The project aim was to replace the screen printing operator with an automated system. Screen printing requires a significant amount of human visual input, this implies the need for machine vision if the process is to be automated.

The project required a single computer controlling the whole system, and to keep the costs down, an IBM clone 386-33 was chosen. Black and white imaging was considered sufficient and the ability to take input from four cameras with a frame buffer size of 512 x 512. The multiple input was essential to carry out accurate system alignment and the system also required storage (in the form of frame buffers) to carry out image processing on captured images.

2.1.1 Hardware

The board selected was a Matrox MVP-AT, the block diagram of which is shown in **Figure 2.1**. The board itself comprises several sections, these being:

● Memory section - this section consists of 1 Mbyte of video memory. It is accessible by all the MVP sections and the host PC. Its function is to hold the image information. The memory can be divided in three different ways:

- one large 1024 x 1024 x 8-bit frame buffer,
- two medium sized 512 x 512 x 16-bit frame buffers.
- four small 512 x 512 x 8-bit frame buffers.

The different frame buffers are shown in figure 2.2. There are four basic buffers (0, 1, 2 and 3), buffer 4 comprises buffers 0 and 1, buffer 5 comprises buffer 2 and 3.

- Statistical section - this is the hardware section that performs intensity histograms and profiles. A histogram is a probability function describing the grey scale distribution of the image, and is described in more detail later in this chapter (section 2.3.3). A profile is an intensity sum along a predefined axis.

- Graphics section - this is the Hitachi ACRTC. Its function is to supply many drawing primitives which allow the host to perform drawing routines with fewer commands.

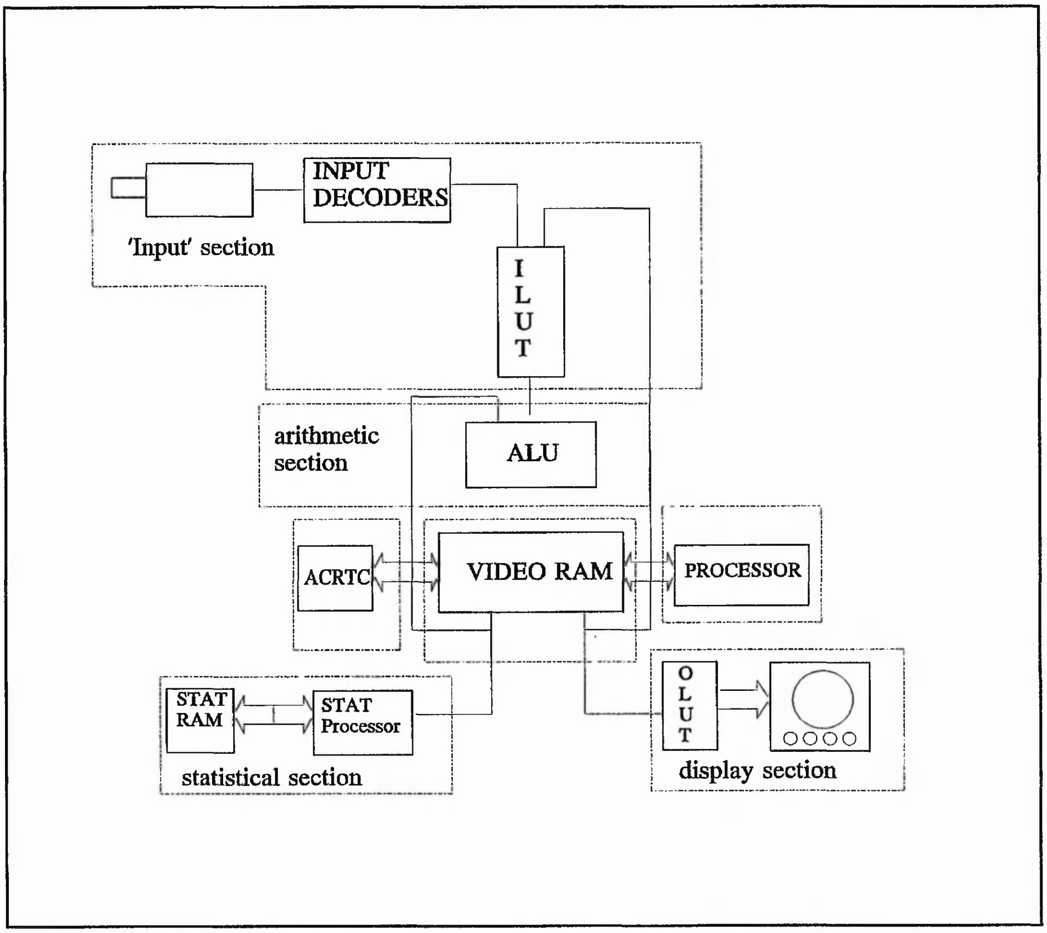


Figure 2.1 MVP-AT block diagram.

- Display section - this section takes information from the memory and converts it to a video signal that is then sent to the display. It has a number of user-programmable features including overlays, windows and keying. This section also gives 24-bit true colour as well as a number of pseudocolour formats, all under software control. It achieves this through a 24-bit Output Look Up Table (OLUT) that can be initialized to a default state by the software or written by the user.

- Arithmetic section - this section performs all arithmetic and logic operations either on one frame buffer, between one frame buffer and a constant, or between two frame buffers. The ALU section operates in real time allowing arithmetical and logical operations with a camera input. Neighbourhood and morphological operations are also possible using this section.

2.1.2 Vision System Software

The MVP-AT comes with its own software library covering a range of basic image processing and enhancement functions. These are all written in 'C' allowing faster development of applications and improved interfacing with automated control systems. The range of functions supplied with the board is shown in **Appendix 3**. The software also includes a shell for development of prototype routines called the Interpreter. Within this shell functions can be tested without writing any 'C' code, this not only acts as a good workbench for developing functions, it also gives a good introduction to the functions supplied with the system.

Besides the supplied software it is also possible to buy 'function libraries' for the MVP-AT and from other suppliers other than Matrox. A survey was carried out by 'Image Processing' **[15]** which shows the large range available, this sort of software tends to be more expensive than Matrox's own but gives a much greater range of functions. A demonstration disk from Foster-Findlay was tested, this was typical of the libraries of functions available. It came with a Microsoft Windows 3 front end that was good as a development tool or as an image processing tutor, but was too slow for industrial applications. The typical cost of this system was more than £2,000. The cost of such software was too much for this project as many of the additional functions available could be written with little difficulty. Such software has its market in industry where instant solutions may be needed to non-complex vision problems, or where the system is to be implemented by operators with little programming skill.

Basic Software Functions

Appendix 3 shows the functions supplied with the MVP-AT, the basic initializing functions are:

outpath - which frame buffer to be shown and whether or not overlay
opmode - Graphic, continuous grab, process
chan - which video chan (1-4) is to be used
video - live video or frame buffer display
sync - internal or external selection
disformat - what operating format is used, pixel aspect ratio, American or European.

The above functions are invoked before any imaging functions are carried out, they are usually incorporated within one 'C' function, see init_mvp() in **Appendix 4**. A typical function would be:-

disformat 0 1 0	Set for interlaced monitor, European TV standard (NTSC).
clear 4 0	Clear frame buffers.
clear 5 0	
opmode 2 4	Processing mode.
outpath 0 -1 0 0	Display Frame Buffer (FB) 2.
inmode 1	Black and white mode.
sync 1 0	External sync.
chan 0	Video channel 0.
video 1 0	Live video source.
snap 0	Snapshot into FB0.
image 0 2 0 80 0	Add 128 (80_{16}) to every pixel in FB0 and put result in FB2.
outpath 2 -1 0 0	Display FB2.

This function adds a value to every pixel in a frame buffer using the function image(). Using this function it is also possible to carry out many other operations on a buffer using the arithmetic and logic capabilities of the ALU, these include addition, subtraction, XOR, OR, NOT and AND.

2.2 Cameras and Lenses

The first, and often neglected, component in a vision system is the part that takes the input, the lens and cameras. Without proper consideration of this input section the initial image given to the processing part of the system will be poor, making the task of the image processing section difficult or impossible. Correct selection will make the processing simpler and far more efficient.

2.2.1 Cameras

The majority of cameras used in machine vision applications fall into two categories; vidicon and solid state.

Vidicon cameras

Vidicon cameras have been around for a few decades and are still popular in television broadcasting. They are basically electron beam sensors with a photosensitive target charged by a scanning electron beam and discharged by incident illumination. The video signal is determined by the amount of energy required to return the surface back to its equilibrium (level) state. They are available with sensitivities to different light wavelengths (UV, X-ray, far-IR). Certain types of vidicon have an advantage over solid state cameras in high resolution applications. These tend to be expensive and like all vidicons they suffer from signal lag and are therefore only suitable in applications with a throughput of less than four images a second. Vidicon cameras are also subject to geometric distortion, burn-in, low filament life, and they are also very fragile and therefore unsuited to many industrial applications. The attributes of vidicon are high resolution, blue spectral sensitivity, relatively inexpensive for lower resolution types, and because they have been around for a number of years they are widely available with a great deal of knowledge in their use.

Solid state cameras

There are many different types (and manufacturers) of solid state cameras, most of which are based on Charge Coupled Device (CCD) technology, these include:

- CID, charge injection devices,
- CPD, charge prime devices,
- MOS, metal-oxide semiconductor,
- CCD, charge coupled devices,
- BCCD, bulk charge coupled devices,

They tend to have poorer spectral response than vidicon cameras in the visible range, with lower resolution and higher costs, but they have many significant advantages. They have a longer life being more rugged, smaller and lighter, making them far more suited to industrial applications. They are not subject to the lag that affects vidicons so they are more useful in real-time applications, some are able to capture up to 12,000 pictures per second and they tend to have better repeatability.

Solid state cameras have an array at the front of the camera that takes in the image. This array is made up of several hundred discrete photosensitive sites, typically arranged in a rectangular form 500 pixels horizontally by 582 pixels vertically. Incoming photons (light) charge the array surface proportional to its intensity and because the arrays are arranged very uniformally they have a better geometric performance than vidicon, making them more useful for measurement or inspection. The uniform arrangement of the array also means there is no need for regular adjustment or calibration.

2.2.2 Lenses

The lens is the first part of the vision system to receive the image and therefore its correct selection is vital to the functioning of the system. There are two questions that have to be considered when selecting a lens for a machine vision application:

- What is to be viewed and what resolution is required?

- In what environment is the lens to be used and are there any problems with positioning, e.g. in robot application is there any chance that the lens may be knocked while in operation?

Basic optics

Lenses are described by their focal length in mm and their aperture, or f-number. These factors determine the magnification and ability to take in light. The focal length determines the magnification of the lens according to the elementary optical equation:

$$\frac{1}{u}+\frac{1}{v}=\frac{1}{f} \qquad \textbf{(2.1)}$$

where u is the distance from the lens to the image,
v is the distance from the lens to the object,
f is the focal length of the lens.

The magnification of the lens is determined by the equation:

$$M=\frac{i}{o} \qquad \textbf{(2.2)}$$

where i = image size,
o = object size.

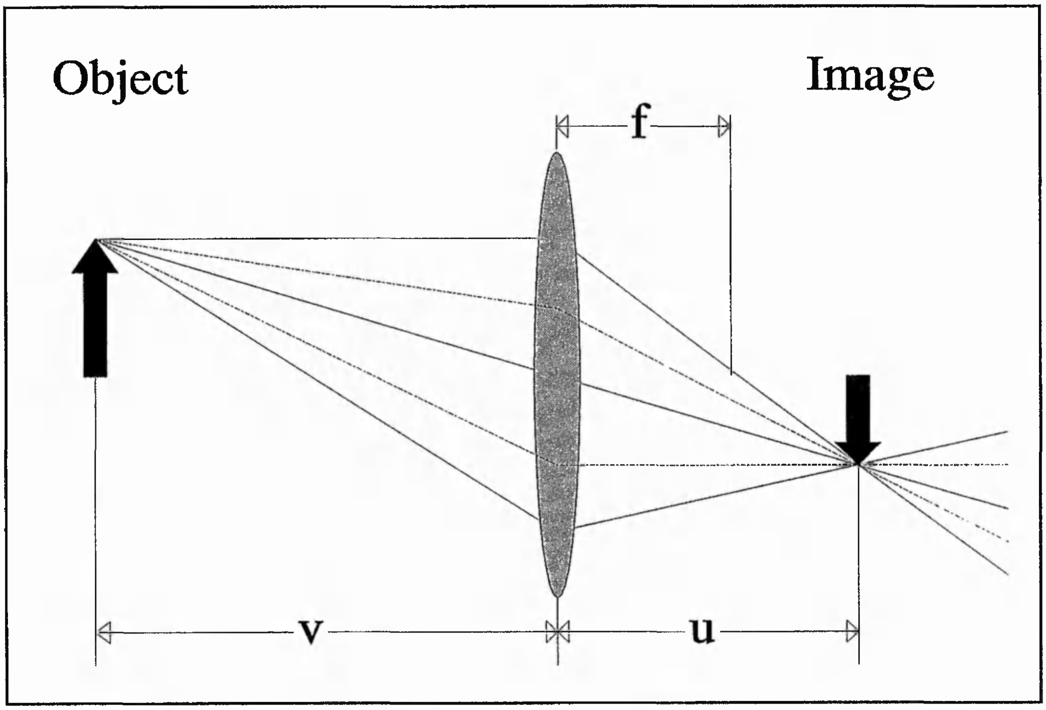


Figure 2.2 Focusing a thin lens.

This can be restated:

$$M=\frac{v}{u} \tag{2.3}$$

$$\therefore f=\frac{uM}{M+1} \tag{2.4}$$

It should be noted that these formulas are designed to be used with ideally thin lenses, but they are useful as a rough guide in lens selection.

The f-number is a measure of the amount of light that is able to pass through the lens and is governed by the aperture setting of the lens, the lower the f-number the greater the amount of light that can pass through the lens (i.e. the wider the aperture). It is the ratio of the focal length of the system to the diameter of the entrance pupil. There is a standard scale for f-numbers: 1.4, 2, 2.8, 4, 5.6, 8, 11, and 16. Each increment of the scale reduces the light intake by a half.

The depth of field is the area in front and behind the image that appears to be in focus, i.e the range of movement of the object within which it remains in focus. It is at its greatest with high f-numbers (i.e. a small aperture), when the camera is focused on a distant object or when the lens has a large focal distance.

$$depth\ of\ field=\frac{depth\ of\ focus}{magnification} \tag{2.5}$$

Lens Operating Environment

An important aspect that has to be considered when choosing a lens is the environment in which it will be used. The lens may have to be vibration resistant if used on an industrial machine, and protection from any chemical attack may be necessary. The location of the lens within the operating environment must also be considered. The lens must be located in a position where it will not be knocked as they are generally intolerant to abuse. A slight knock may leave no mark on the lens casing but the optics may be misaligned leading to image distortion.

2.3 Image Enhancement

Before an image can be processed to find the information required it is necessary to achieve the best possible image quality through a process of image enhancement. The aim of any image enhancement process must be to emphasise the relevant aspects of an image. This may involve filtering out excess detail or noise, removing objects in the image that are of no interest, or just simplifying the image to just black and white (thresholding). It can be done by hardware through effective use of lighting, filters, cameras and lenses, or through software processes like thresholding, contrast stretching, image averaging, segmentation, convolution etc. To understand the process of image enhancement it is necessary to explain some basic principles of machine vision.

2.3.1 Lighting

The vital part played by lighting in any vision system has long been known. It is almost too obvious a fact to state that without lighting a vision system could not perform. It is not just enough to have lighting to illuminate a subject, the lighting system chosen must be the one that most enhances the aspects of the image that are to be investigated.

Important aspects of the lighting that must be considered include the stability of the light source over its life time and the integrity of its power supply, the advantages of having the best lighting system available are negated if attached to a power supply that varies in its output. The spectral output and intensity of the lighting system must also be known, and its relationship with the spectral response of the vision system cameras. Any filtering requirements on either lighting or cameras must be considered, with their resultant effect on the lighting power output. The angle of incidence of the source and the target's ability to absorb light and cause shadows or glare must be taken into account, and the effect of ambient lighting on the complete system.

It is becoming more common for distributors of machine vision systems to have research departments that deal with lighting and that will advise customers on lighting needed for specific applications, some vision systems come with lighting as part of the package. This case is still not the norm, most manufacturers will have no involvement with the vision system once it has left their hands, let alone the use of lighting with the system.

Because of the need to select the correct lighting system for a particular machine vision application much research is going on to produce aids to correct selection. A system has been produced that aids in the choice of the best lighting for a wide range of applications, this is known as "Application Lighting Evaluation System" **[19]**. This system allows the testing of a wide variety of lighting methods, as well as filtering and optical techniques, with very little set-up, enabling the best lighting set-up to be evaluated for a specific object. This is an extremely useful tool for heavy users of machine vision who find they have applications involving a wide range of objects all requiring different lighting techniques. The system allows the testing of a combination of nine different lighting methods, along with a range of colour filters, polarization, and UV and strobe lighting. The effect of this evaluation system is that the best viable lighting solution for a specific application can be found in the fraction of the lead time normally associated with lighting. Colleta and Harding **[20]** realised that lighting was a collection of sciences and with this in mind they put forward guidelines for machine vision lighting design, based on five problems:

1) The object to be viewed and its characteristics,
2) Light levels, and their effect on the subject and the sensor,
3) Light character requirements,
4) Image requirements, resolution, silhouetting, grey-level, etc.,
5) Physical constraints due to the process.

These five problems were given to different machine vision users with different questions relating to each problem. The answers were put in a data base in an attempt to quantify the problems posed in an attempt to help in the production of the guidelines.

These cases outline the importance of considering every aspect of the process in developing the optimum lighting system, with one of the most important factors in any lighting system, the need for stability and repeatability. Machine vision lighting has gone beyond the use of commercial lighting that is designed for lighting up rooms, lighting is being specifically designed for machine vision, and some companies even produce custom light sources. There is also a need to examine the integrity of the power supply to the lighting remove power surges and to make sure the power supplied is constant. Lighting can incorporate feedback loops to ensure a stable supply, to compensate not only for variation in power supply but to compensate for variance due to aging of the lighting, generally light output decreases with age, at varying rates depending on the lamp type.

In choosing a light source the following factors must be considered:

- Knowledge of lamp operation, including how it interacts with the local environment, it must be able to remove problems due to the ambient light, the light may take some time to warm up before its peak output is reached.

- Knowledge of problems that may be encountered in the working environment, if the object is reactive to UV light (e.g UV cured screen printing inks) the light source must not have a significant amount of UV in its output.

- Choice of power supply, the power supply requirements of the lamp must be considered (as mentioned earlier), to ensure constant light output.

- Design of enclosure, the incorporation of reflectors in the enclosure may cause unbalanced light output with bright spots, also light may be absorbed, or the lamps caused to overheat which may affect lamp life and light output.

- Lamp life, and variance in light output with life, many lamps, especially fluorescent, decrease their light output over their life span, this happens at varying rates dependent on the lamp, but working life tends to be shorter the more intense the light source.

- Intensity and wavelength of light source and spectral response of camera. Different wavelengths may act to accentuate different aspects of an object and may increase contrast between the object and its background. Different cameras are sensitive to different wavelengths of light, for example the CCD spectral response curve **[see Chapter 4, Figure 4.6]** shows that they are more sensitive to light in the range 700-850 nm. If the light source emits most of its light in a non-usable form it may have an undesirable effect on the sensor or target. If at all possible the light source should be limited to the desired spectral level, rather than to add filters which may have a detrimental effect on the light levels within the system

● Nature of object (light absorbency, shadows, reflectance etc.), the object must be considered to see whether it is susceptible to shadows or glare (e.g. highly reflective surface such as wet ink). The distance of the object from the lamp must also be considered, if it varies this may have a very detrimental effect because light characteristics may change, even over a very short distance.

The range of lamps available is immense, a description of each type is outside the scope of this thesis, but the most commonly used include:

● Fluorescent, including high frequency and high output types, many different types of fluorescent tube are available 'off the shelf' and as custom designed, high output types incorporate internal reflectors and apertures to increase output and uniformity of light output (up to ten times that of standard tubes). The custom design allows tube of almost any size and shape, including ring lights to fit around lenses and long tubes (100"+). High frequency power supplies(~20 KHZ) are often used to minimise flicker along with feedback loops to minimise diminishing power output due to aging. Biegel **[21]** described the following benefits of high frequency fluorescent lighting: no heat generation, a pure infinitely adjustable light level, colour adjustment for different grey-level applications, longer bulb life, flicker free and higher efficiency.

● Quartz halogen, these are less susceptible to aging and give relatively constant output after a warm up of a few minutes, the most common light source for fibre optics,

● Mercury vapour arc lamps, give a very white light with almost zero red or IR emission, susceptible to voltage changes with a relatively short life span (~500 hours) for high pressure lamps, more for low pressure (~20,000).

● High pressure sodium lamps, small spectral range (550 to 700 nm) giving high output with a long life (~24,000 hours), high temperatures generated.

- Strobe, used in high speed work, particularly with line scan cameras, because the ‘strobing’ nature removes imaging blur, stable light output after initial burn-in period.

- Laser and laser diodes, primarily used for distance measuring applications and as structured light sources.

- Light Emitting Diodes (LED’s), use becoming ever more popular as they are capable of giving a uniform light source with good light output levels when pulsed. They have very small spectral ranges, available in red, green , blues and IR wavelengths. LED’s are very flexible as their small size allows the production of complex arrangements of a few to several thousand individual LED’s.

To go with the different light sources there are also different lighting techniques:

- Back lighting, useful in applications where the edge information is important.

- Front Lighting, including many different angles of lighting, used to bring out different characteristics of an object.

- Structured lighting, used for gauging or surface error detection, most commonly with laser light sources.

The requirements of a lighting system can be summed up in three stages:

1) Optimize image contrast between the object of interest and it’s background,

2) Negate interference and variation due to ambient conditions,

3) Simplify the image for later processing and therefore minimise computing power and time required.

2.3.2 Pixels

A pixel, or picture element, is the smallest element of an image, and each pixel has assigned to it a grey-level, this is a number between 0, representing black, and 225, representing white. It was explained earlier in the description of the MVP-AT vision board that an image is stored in a frame buffer, with a typical size of 512 pixels by 512 pixels. For each of these points in the frame buffer the corresponding grey-level of the pixel is stored. **Figure 2.3** shows a typical section of an image. Each pixel is stored as a character, which is one byte, therefore an image is:

512 x 512 x 1 byte = 262,144 bytes.

This can impose problems for imaging as the 'C' programming language and a PC can only access 64 Kbytes at one time, this means an image has to be analyzed in four 64Kb windows, or by a pixel at a time for neighbourhood operations. The 'C' routine in **Table 2.1** was used to access the information.

```
im_outmode(0);              sets the vision board output mode
im_outpath(0, -1, 0, 0);    sets output path for board
im_opmode(0, 0);            sets operating mode for board
for (j = 0; j < 4; j++) {
        im_cpuwin(j);  selects specific 64Kb window
        nrows = j == 3 ? 96 : 128;
        for (row = 1; row < nrows-1; row++, r++) {
                buildptr(&p,row, 0);
                builds far pointer for 64Kb window
                for ( i=0; i <=512 ; i++,p++){

        }
}

where   nrows is the number of rows in each processing window.
```

Table 2.1 'C' routine to access image data.

100	112	122	158	176	159	147	131	115	105
130	132	135	160	184	176	155	135	129	127
165	156	162	170	187	169	156	161	163	164
124	123	135	160	190	172	160	132	136	128
107	115	140	173	186	170	155	128	108	105
103	112	135	172	185	170	156	130	110	100
98	105	122	146	177	172	151	134	109	101
96	100	112	136	167	160	151	135	108	100
90	100	110	130	159	149	138	130	102	96
88	97	105	120	140	153	135	130	110	97

Figure 2.3 Part of a digital image.

2.3.3 Histogram

A histogram is a statistical representation of an image, giving a relative frequency of all the grey-level values, and as such is probably the most simple, and widely used imaging process. **Figure 2.4a** shows a typical histogram the X-axis gives the relevant grey-level, and the Y-axis the frequency of grey-levels. This histogram is described as being bi-modal, i.e. it has two distinct areas or peaks. It is difficult to separate these two as the histogram has many little 'spikes', these can be smoothed out by a neighbourhood averaging operation. This may mean taking each grey-level point frequency and its two neighbours, calculating an average frequency, and assigning that value to the selected grey-level frequency. This process may be repeated a number of times till the desired level of smoothing is achieved. **Figure 2.4b** shows the results of smoothing a bimodal histogram of **Photograph 2.1b**. The algorithm for producing a histogram is shown in **Table 2.2**.

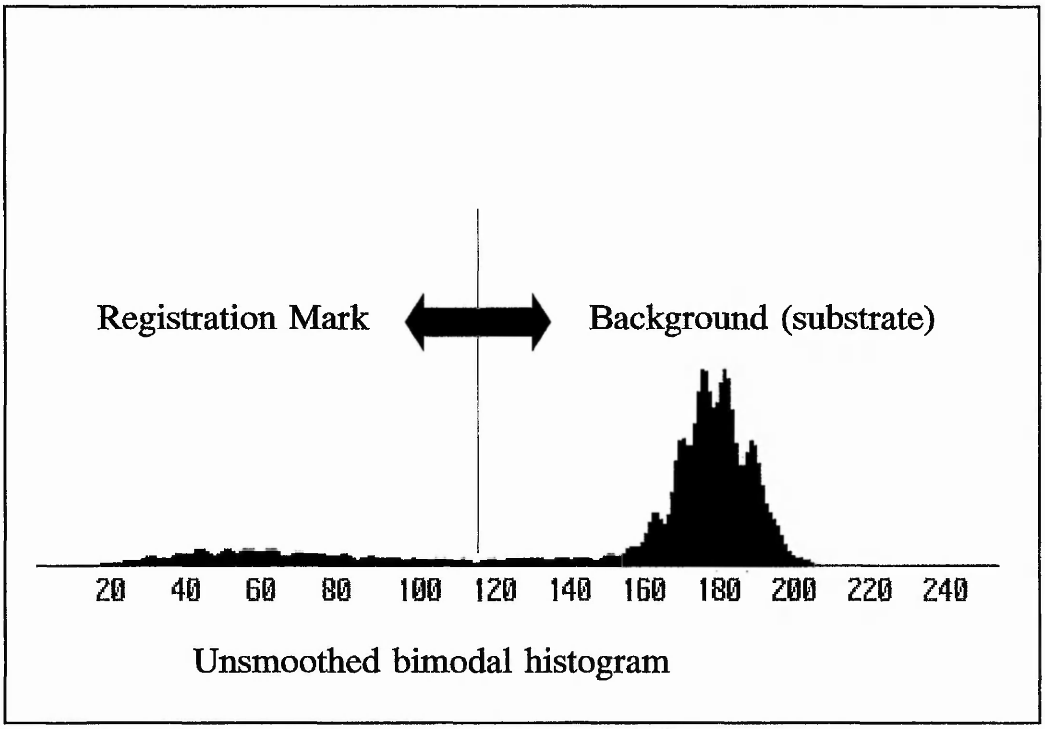


Figure 2.4a Unsmoothed histogram of **Photograph 2.1b**.

```
Initialize histogram buffer histbuf[256];
        for (i=0 to i=255)
                histbuf[i]=0;

        for(each pixel in the image pixel[x,y])
                increment hisbuf[pixel[x,y]] by one;
```

Table 2.2 Algorithm for producing a histogram of an image.

The MVP has a function for calculating histograms and carrying out the smoothing process. It also has simple histogram analysis functions that find the minimum and maximum values of the histogram. This is useful in finding suitable threshold values.

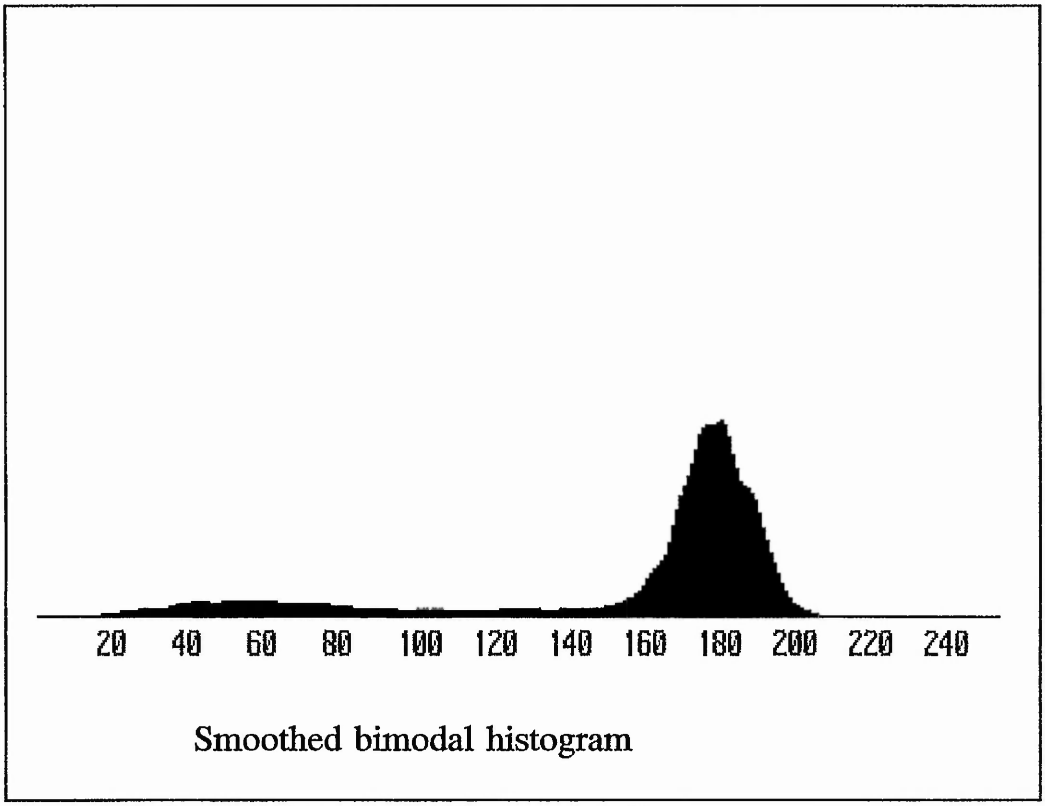
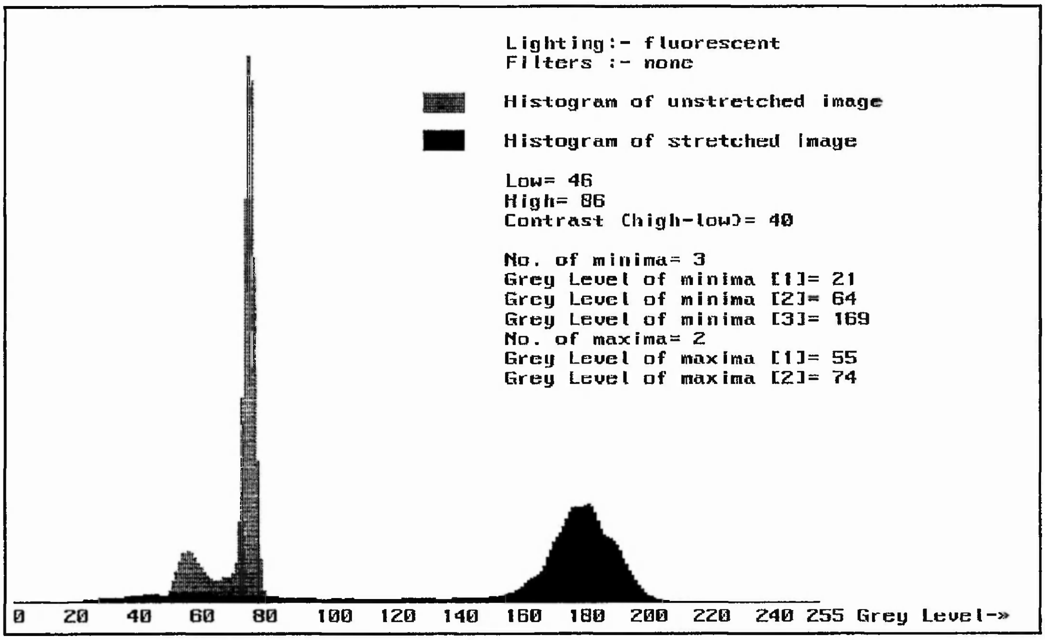


Figure 2.4b Smoothed histogram of **Photograph 2.1b**.

Histogram analysis was used by the vision system to remove unwanted glare from images. When a board is freshly printed the ink is highly reflective and this leads to glare, this can be minimised by using a suitable lighting set-up (see 2.2.1), and polarizing filters, but it is still present. On analysis of a histogram of a freshly printed registration mark, there was no clear peak that corresponded to the glare, but it could be seen that there was a slight rise in the tail of the histogram that corresponded to the glare, this could be converted to black (i.e. part of the registration mark, by thresholding all points above a threshold point on the tail of the histogram. This point was found by calculating the gradient of the histogram at each point from the end of the tail. Once the gradient went over a predefined level (found by experimentation), the point at which it occurred was defined as the glare cut off point and all points above this point were therefore glare and had to be part of the registration mark. This relied on the valid assumption that the grey-level of the glare was always higher than that of the background.

2.3.4 Contrast Stretching

The unstretched histogram shown in **Figure 2.5** covers a range of about 40 grey level points, the rest of the grey levels are not used which means the contrast between the object of interest and the background is not as good as it could be. It is possible to utilise these unused grey levels using a process called contrast stretching. The low and high values of the histogram are calculated, and the low value is subtracted from all the grey level values. This has the effect of shifting the histogram to the left. The histogram is then stretched by multiplying all the grey level values by a scale factor = (255-0)/(low value - 0). The 'C' code for carrying out a contrast stretch is given in **Appendix 4**, with the function stretch();.



```
Lighting:- fluorescent
Filters :- none

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 46
High= 86
Contrast (high-low)= 40

No. of minima= 3
Grey Level of minima [1]= 21
Grey Level of minima [2]= 64
Grey Level of minima [3]= 169
No. of maxima= 2
Grey Level of maxima [1]= 55
Grey Level of maxima [2]= 74
```

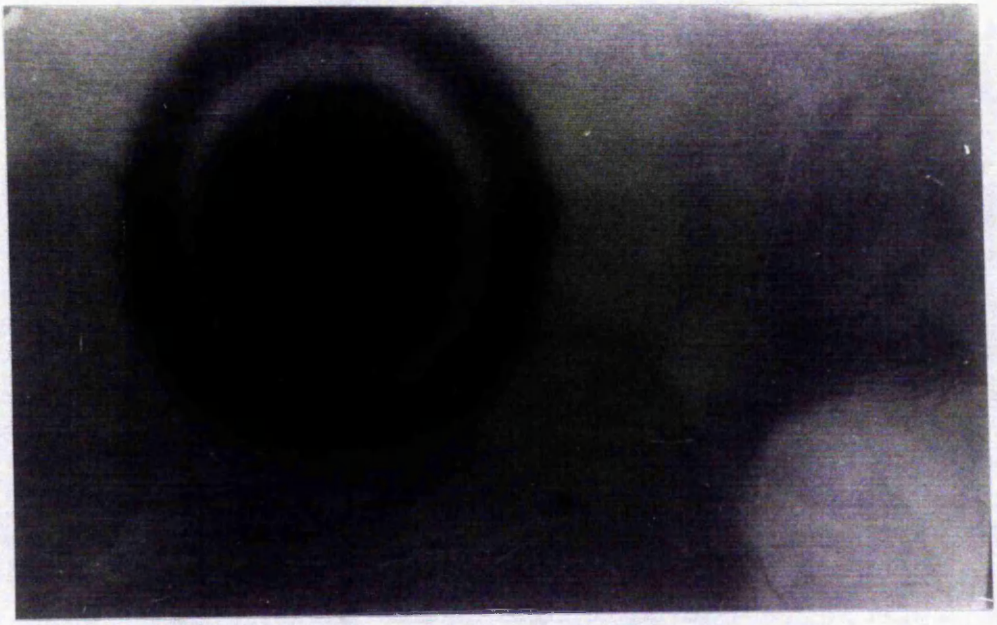
Figure 2.5 Contrast stretching a histogram.

Stretch(); works by creating a LUT buffer, lutbuf[256]. It is filled with all the compensated values for the contrast stretch, using the algorithm shown **Table 2.3.**

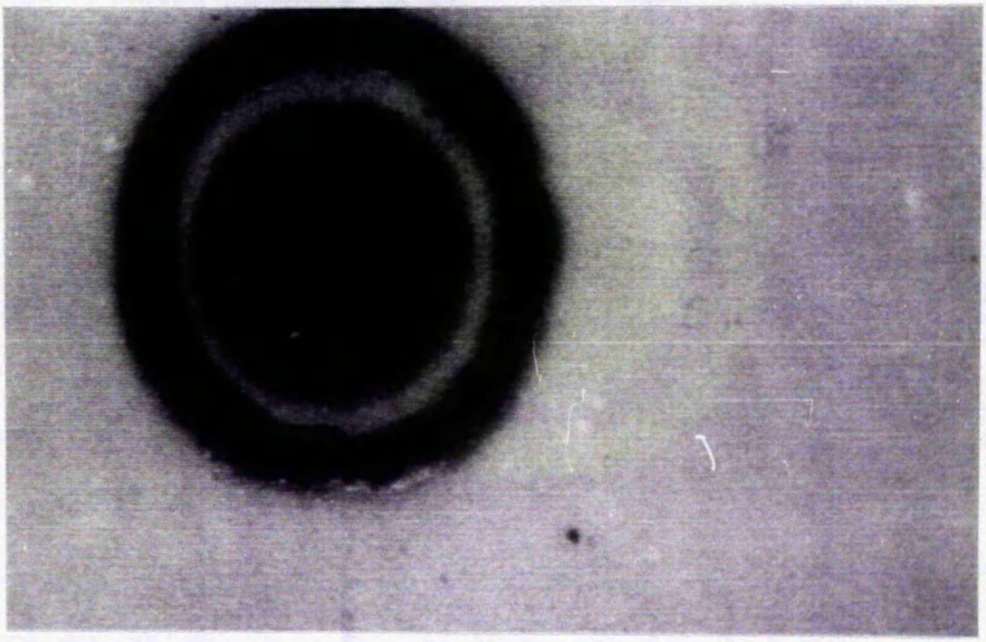
```
scale factor = 255 / (high_value - low_value);
        for(i=0 to i=low_value-1)
                lutbuf[i]=0;
        for(i=low_value to i=high_value)
                lutbuf[i]=(i-low_value)*scale_factor;
        for(i=high_value+1 to i=255)
                lutbuf[i]=255;
        for(each point in image pixel[x,y])
                pixel[x,y]=LUT[pixel[x,y]];]
```

Table 2.3 Algorithm used by stretch(); for contrast stretching.

Photographs 2.1a and **2.1b**, show the effect of applying a contrast stretch to an image of a screen printed registration mark. It clearly emphasises the mark from the background and this makes the whole segmentation process much simpler and constant. With an unstretched image a poor threshold of perhaps five grey scale points out from desired can make an image unclear introducing unwanted noise, or giving an incomplete image. With a stretched image a similar poor selection may make little difference.



Photograph 2.1a Unstretched image



Photograph 2.1b Stretched image.

Look Up Tables (LUT's)

The MVP-AT board is supplied with a number of hardware Look Up Tables, both input (ILUT) and output (OLUT). Each of these LUT's uses pixel intensity values as addresses to position within the table for new pixel grey-levels.

The ILUT takes input data and transforms it before placing it in the frame buffer, the OLUT transforms the image before showing it on the screen of the video monitor. These can be software altered for typical operations like thresholding and contrast stretching. When an image is taken each pixel value is compared with its value in the selected LUT, and assigned that value. This could be used for creating a negative image by writing a LUT table with values 255 to 0, this would have the effect of reversing all the grey-level values. Thresholding an image is done by writing a LUT with values of 0 up to the desired threshold point, say 125, and values of 255 for all points above that value. This would convert all image points with a grey level below the threshold to black, and all points above to white.

The MVP command for selecting a LUT is:

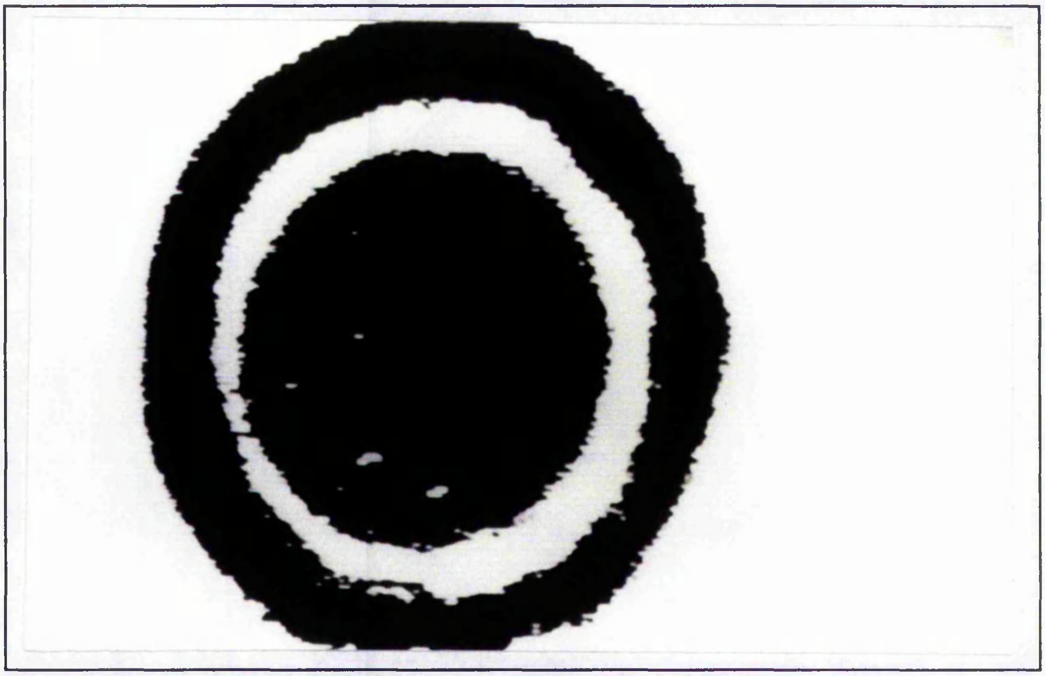
```
slut(LUT,frame_buffer);
```

and for writing to a LUT:

```
wlut(   );
```

2.3.5 Thresholding

It was explained earlier that an image is made up of a certain number of pixels, each with a grey level between 0 and 255. It is possible to simplify the image to just two grey levels, black and white, through a process known as thresholding. This is a process of taking all points with grey level above a defined threshold and converting them to white pixels, and all points at and below the threshold and converting them to black pixels. The results of thresholding **Photograph 2.1b** using the threshold marked in **Figure 2.4a** is shown in **Photograph 2.2**.



Photograph 2.2 Threshold carried out on **Photograph 2.1b.**

The histogram can be clearly seen to have two distinct points and is therefore termed bimodal. These correspond to the registration mark for the point on the left, and the background for the right most point. Clearly by selecting a point at the bottom of the trough between them a threshold point can be chosen. This process is fairly simple for PCB work as there is normally a very clear distinction between the object of interest and it's background, which is made even clearer by contrast stretching. The Algorithm for thresholding an image is shown in **Tables 2.4a** and **2.4b**.

```
For(each pixel within image pixel[x,y]){
        if(pixel[x,y]< <threshold)then pixel[x,y]=BLACK(0);
        else pixel[x,y]=WHITE(255);
}
```

Table 2.4a Algorithm for carrying out a threshold.

A quicker way to do this is to assign values to the ILUT using the algorithm shown in **Table 2.1b**.

```
for(i=0 to i=threshold)
        ILUT[i]=0;
for(i=threshold+1 to i=255)
        ILUT[i]255;     for(each point in image pixel[x,y])
pixel[x,y]=ILUT[pixel[x,y]];]
```

Table 2.4b Algorithm for carrying out a threshold using a LUT.

2.3.6 Filtering and Averaging

The object of image enhancement is to simplify the image before processing and to remove any superfluous information. This can be done before or after thresholding, although it is a less complicated process after thresholding due to the simpler nature of the image.

Noise

Unwanted information may come in a form that is known as noise, this can be defined as corruption of the image of a high spatial frequency, i.e. white specks in a black part of an image, or black specks in a white part of the image. Noise, by its nature, will only be a few pixels in size at most. This can be removed, before thresholding, by a process of image averaging. Multiple images of the scene are taken one after the other and an average image is produced, this is done by averaging each pixel over the range of images. This has the effect of minimising errors due to slight changes in lighting or movement that may occur while imaging, factors that contribute to noise. The MVP has a function for carrying out this averaging:

faverage(n); where n is the number of frames to be averaged.

Another averaging process takes the form of a neighbourhood convolution operation using a local averaging mask as shown in **Figure 2.6**. For each pixel in the image an average value is calculated from itself and the eight pixels around it, this average value is assigned to that pixel location. Noise was defined earlier as being of a high spatial frequency, if the image itself is of a high spatial frequency an averaging process like this will partly degrade the image. It is possible to attach conditions to the averaging process, for example, if the average value calculated is below a defined threshold, do not alter the pixel value. Such conditions were applied as a smoothing operation to smooth the edges of the thresholded image prior to both chain coding and a Novel edge detection process, which will be described in chapter 4. This worked by taking a 5 x 5 neighbourhood average, if the average was below (25*255)/2 (i.e. predominantly black), then the pixel is assigned a value 0 (black), or if it is above that value it is assigned a value of 255 (white).

1/9 (x-1,y-1)	1/9 (x,y-1)	1/9 (x+1,y-1)
1/9 (x-1,y)	1/9 (x,y)	1/9 (x+1,y)
1/9 (x-1,y+1)	1/9 (x,y+1)	1/9 (x+1,y+1)

The average value of the grey levels of all pixels in the mask is calculated and assigned to the centre pixel (x,y).

Figure 2.6 Local averaging mask.

2.4 Summary

This chapter has given some background information detailing how the image is captured using cameras and lenses and enhanced using the vision system software. This is the first part in a machine vision system, how well it is done will have a significant effect on how successfully the complete system operates. Chapter 3 outlines some image processing techniques, these rely on the image enhancement in order to work efficiently.

Chapter 3

Image Processing Techniques

Chapter 3: Image Processing

Chapter 2 explained various aspects of the process of image enhancing, this can be considered as preprocessing, the image has been simplified to make further processing as simple as possible. This chapter outlines the next step in a machine vision system, image processing, not as a complete reference but as an explanation of the techniques used in the development of the automated screen printing system.

3.1 Edge Detection

The next step after image enhancement is to extract the desired features from the image. The feature that defines an image tends to be its edge, most objects can be identified from their profile, once the boundary of an object has been defined it can the be used to describe it's location and shape. In addition to this the edge is more tolerant of, or invariant to, changes in lighting, part texture, and part reflectivity, aspects that are common in nearly all machine vision applications. This makes edge detection the most widely used image processing technique.

By converting an object to just its boundary also has the effect of greatly reducing the amount of memory required to store the image and the computing power required to process the image, an example of this is the registration mark used in the automated registration system. This is a solid circle approximately 250 pixels diameter.

$$\therefore \ \mathit{Area} = \pi\left(\frac{250}{2}\right)^2 \qquad \textbf{(3.1)}$$

$$= 49087 \ \mathit{pixels} \qquad \textbf{(3.2)}$$

$$\mathit{edge\ pixels} = \pi\ 250 = 785\ \mathit{pixels} \qquad \textbf{(3.3)}$$

This clearly shows the advantage of converting an object just to its boundary information, 49087 pixels is 49 Kbytes of information, 785 pixels is just 0.7 Kbytes, i.e.

less than 1% of the storage space and for most image processing applications only 1% of the processing time.

3.1.1 Types of Edge Detection

It is obvious to the human observer what constitutes an edge, and where the edge of an object lies. It is a different matter with machine vision. The simplest definition is that an edge is a discontinuous change in pixel intensity within an area of an image. With this definition in mind, to detect an edge it is necessary to find these discontinuous changes in pixel intensity, and this is the basis of most edge operators.

Because of the importance of edge detection a great deal of work has been done on the subject, resulting in many different processes to carry out the operation. The process of edge detection has traditionally followed one of four different ideas:

1) By measuring the local gradient of pixel grey level against distance, e.g. Sobel, Roberts, Prewitt. These edge detectors tend to be susceptible to noise as this has a high spatial frequency, very similar to the definition of an edge used by such detectors.

2) Using template matching, e.g. Kirsch template, Prewitt template. Templates are defined that would match the description of an edge, and the image is searched to find occurrences of the template, where a match is found an edge is said to exist.

3) Edge fitting methods, an ideal edge can be described by a model of the step discontinuity in intensity at any particular location in an image, using this idea edges can be detected by comparing image neighbourhoods to the model and if there is close correlation an edge can be said to exist at that point. An example of this form of detector is the Heuckel edge fitting technique.

4) Statistical methods look at the homogeneity of a pixel neighbourhood (typically a 7 x 7 pixel area). If the areas is homogeneous then clearly no edge exists, if there is two distinct inhomogeneous areas then an edge may exist. One method that follows this process is used by Yakimovsky **[22]** in which he defines a ratio of the grey level standard deviation of the complete neighbourhood region against the product of the grey level standard deviations of both regions. A threshold value for this ratio is evaluated above and below which an edge is said to exist or not.

It is beyond the bounds of this thesis to give a detailed description of all edge detection processes mentioned above, but a brief explanation of gradient operators and template matching will be given as they are the most common processes used. The merits of individual examples are evaluated in Chapter 4 **(4.4.6)**.

Gradient Operators

Gradient type edge detectors are the most commonly used. If an image function is defined, f(x,y), the rate of change of this function in the x direction can be termed:

$$\frac{\delta f}{\delta x} \qquad \textbf{(3.4)}$$

and in the y direction as:

$$\frac{\delta f}{\delta y} \qquad \textbf{(3.5)}$$

This gives definitions for the rate of change in x and y directions. For the rate of change in a direction Θ measured from the x axis, the equation can be written:

$$\frac{\delta f}{\delta x}\cos\theta + \frac{\delta f}{\delta y}\sin\theta \qquad \textbf{(3.6)}$$

For a maximum value of the rate of change, with a magnitude:

$$\theta = \arctan\left[\left(\frac{\delta f}{\delta y}\right)/\left(\frac{\delta f}{\delta x}\right)\right] \tag{3.7}$$

$$\sqrt{\left(\frac{\delta f}{\delta x}\right)^2 + \left(\frac{\delta f}{\delta y}\right)^2} \tag{3.8}$$

$$\frac{\delta f}{\delta x} = f(x+n,y) - f(x,y) \tag{3.9}$$

$$\frac{\delta f}{\delta y} = f(x,y+n) - f(x,y) \tag{3.10}$$

n is a small integer, most commonly one. This figure is referred to as the "span" of the gradient. The value chosen can vary but as a rule it should be small enough so that the gradient is close to the local changes in the image, but also large enough to overcome small variations of f.

Template Matching

A typical template matching technique is that developed by Kirsch. The Kirsch operator has four different templates that are matched with the image. The operator finds the areas of maximum match and calculates their magnitude and direction. **Figure 3.1** shows the Kirsch templates.

$$\begin{matrix}1&1&1\\1&-2&1\\-1&-1&-1\end{matrix} \qquad \begin{matrix}1&1&1\\-1&-2&1\\-1&-1&1\end{matrix} \qquad \begin{matrix}-1&1&1\\-1&-2&1\\-1&1&1\end{matrix} \qquad \begin{matrix}-1&-1&1\\-1&-2&1\\1&1&1\end{matrix}$$

$$\begin{matrix}-1&-1&-1\\1&-2&1\\1&1&1\end{matrix} \qquad \begin{matrix}1&-1&-1\\1&-2&-1\\1&1&1\end{matrix} \qquad \begin{matrix}1&1&-1\\1&-2&-1\\1&1&-1\end{matrix} \qquad \begin{matrix}1&1&1\\1&-2&-1\\1&-1&-1\end{matrix}$$

Figure 3.1 Kirsch edge templates.

3.1.2 Subpixel Edge Detection

All the edge detectors described so far have a resolution of one pixel and therefore have an accuracy of +/- 0.5 pixels. For most applications this is enough, when an image consists of 800 or more edge pixels this error is averaged out to such a degree that it makes no difference, but for some applications, such a high tolerance gauging, it may be necessary to detect edges to sub-pixel levels. **Figure 3.2** shows the resolution of standard edge detector, compared with the real object.

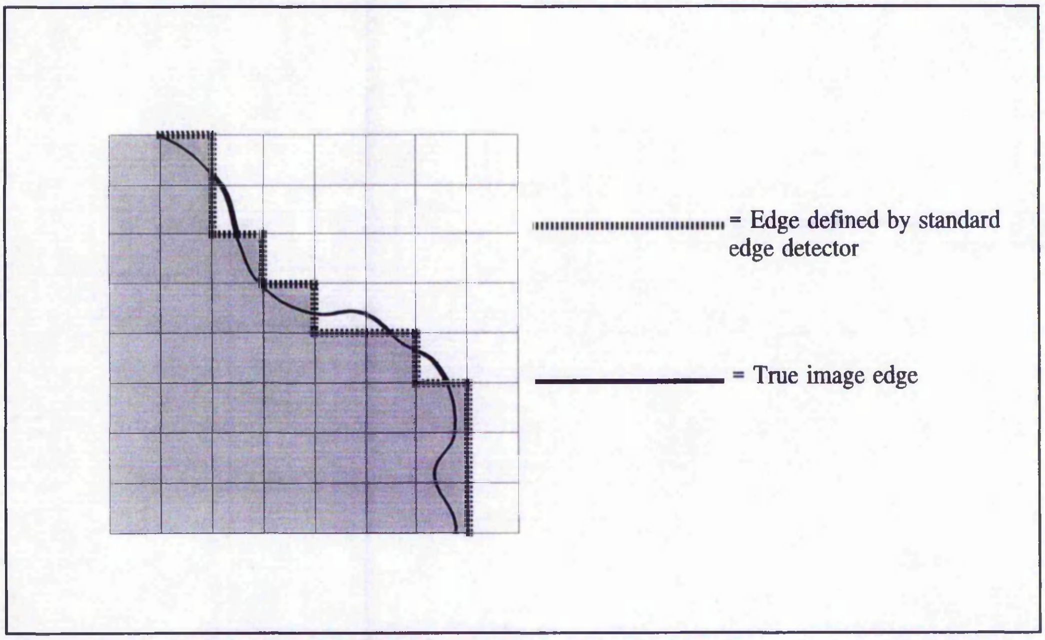


Figure 3.2 Resolution of edge detector

Because subpixel edge detection offers possible improvements in the accuracy of machine vision gauging it has been the subject of much research in recent years. Loomis **[23]** devised a subpixel edge detection algorithm to detect edges within 1/16th of a pixel using standard convolution methods. Crissman **[24]** has looked into the application of four different subpixel edge detection processes looking at their relative performances paying particular interest in a number of aspects of the vision system on the accuracy of the detection. These aspects included the edge model, the relationship between the camera and the vision board, camera sensor discontinuity, optical modelling. He reported subpixel accuracy with one algorithm of +/ 0.025 pixels, but he concluded that for effective use of subpixel edge detection the whole vision system has to be clearly thought out, with particular reference to the factors mentioned above.

3.1.3 Edge Detection on Binary Images

The case of edge detection with a binary image is much simplified as the edge between object and background is a distinct step between black and white. A very simple example of an edge detector for binary images is one that uses a horizontal scan. Starting from the top of the image the state of the pixel is checked to see if it is black or white. If the pixel is black and the previous pixel is white, the pixel is labelled as an edge pixel, the scan then continues until a white pixel is encountered with the previous pixel being black, the previous (black) pixel is then labelled an edge pixel. The scan is continued to the end of the line, and then started at the next line. This is repeated until the whole image has been scanned. This process is fast and simple but has the draw back of being insensitive to horizontal, or near horizontal lines, and it also brings out all noise in the image. The process is refined by incorporating a vertical scan, and the problem of noise is removed by incorporating a counter within the edge detector that measures the distance between edges detected in pixels, if this distance is below a defined threshold, perhaps five pixels, the edge detected is classified as belonging to noise and therefore ignored.

3.2 Edge Tracking

3.2.1 Paperts Turtle

The drawback of many of the edge detectors is that they do not produce a continuous edge, a case which is necessary for producing chain codes (see later section) for image classification. To achieve such an edge a boundary detector is required that follows the outline of the image. A simple process for doing this is known as "Papert's Turtle" **[25]**, the flow chart for which is shown in **Figure 3.3**. It is clear from the example shown in **Figure 3.3** that the boundary produced follows a step pattern as the process only moves

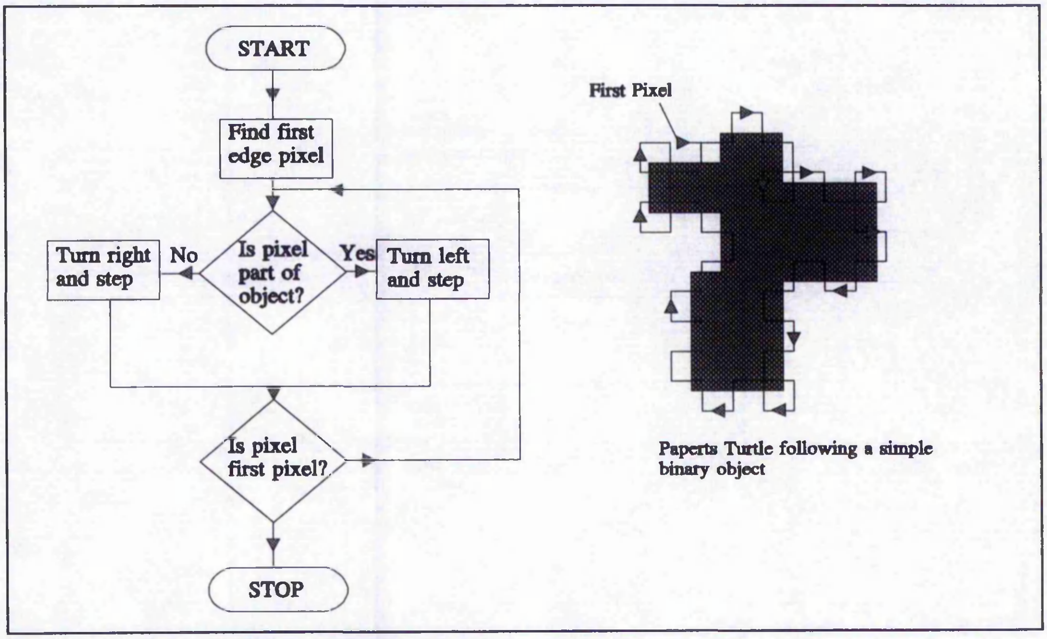


Figure 3.3 Flow chart for "Paperts Turtle"

in four directions (up, down, right, left). This shows the disadvantage of the process as it is designed to work on four-connected block to give a consistent boundary, parts of eight-connected may be missed, hence the exaggerated step boundary. This means that Paperts Turtle is a less than perfect tool for edge tracking of circular objects.

3.2.2 The Novel Edge Tracker

The problems encountered with eight-connected regions mentioned in the description of Paperts Turtle were overcome by a Novel method by the author. This tracker, shown in **Figure 3.6**, followed the outside edge of the object and worked in the following manner:

The first pixel is found and then each pixel in a 3 x 3 neighbourhood of the pixel is examined to see if it is black (i.e part of the object), or white, (i.e. part of the background). An 8 byte character array is created, and the status of each of the eight pixels around the edge pixel, 0=black, 1=white, is entered. The boundary between the object and the background must be the cross over between 0 and 1 (black and white). The array is searched to find the first occurrence of a cross over, finds its position and increments the edge pixel to that point, labelling the original pixel in the frame buffer with a 2 (grey level 125). The process is repeated examining the eight neighbourhood pixels to find the next 1-0 cross over. There is the possibility that there may be two or more possible directions for the tracker to go in, in this case it takes the first option making a note of the point at which it took it (labelled 3 in the array, grey level 60 in the frame buffer) . If the move proves to be wrong (if a dead end is found) it retraces to the point at which it made the choice and takes the next option.

The edge tracked by this method was marked in the frame buffer by altering the grey level of the edge points to 200, and marking other points as referred to above. Originally the detected points were stored in an array, but this proved to be too intensive on system memory. Using the frame buffer as a store the memory on the vision board was utilised instead, freeing much needed system memory.

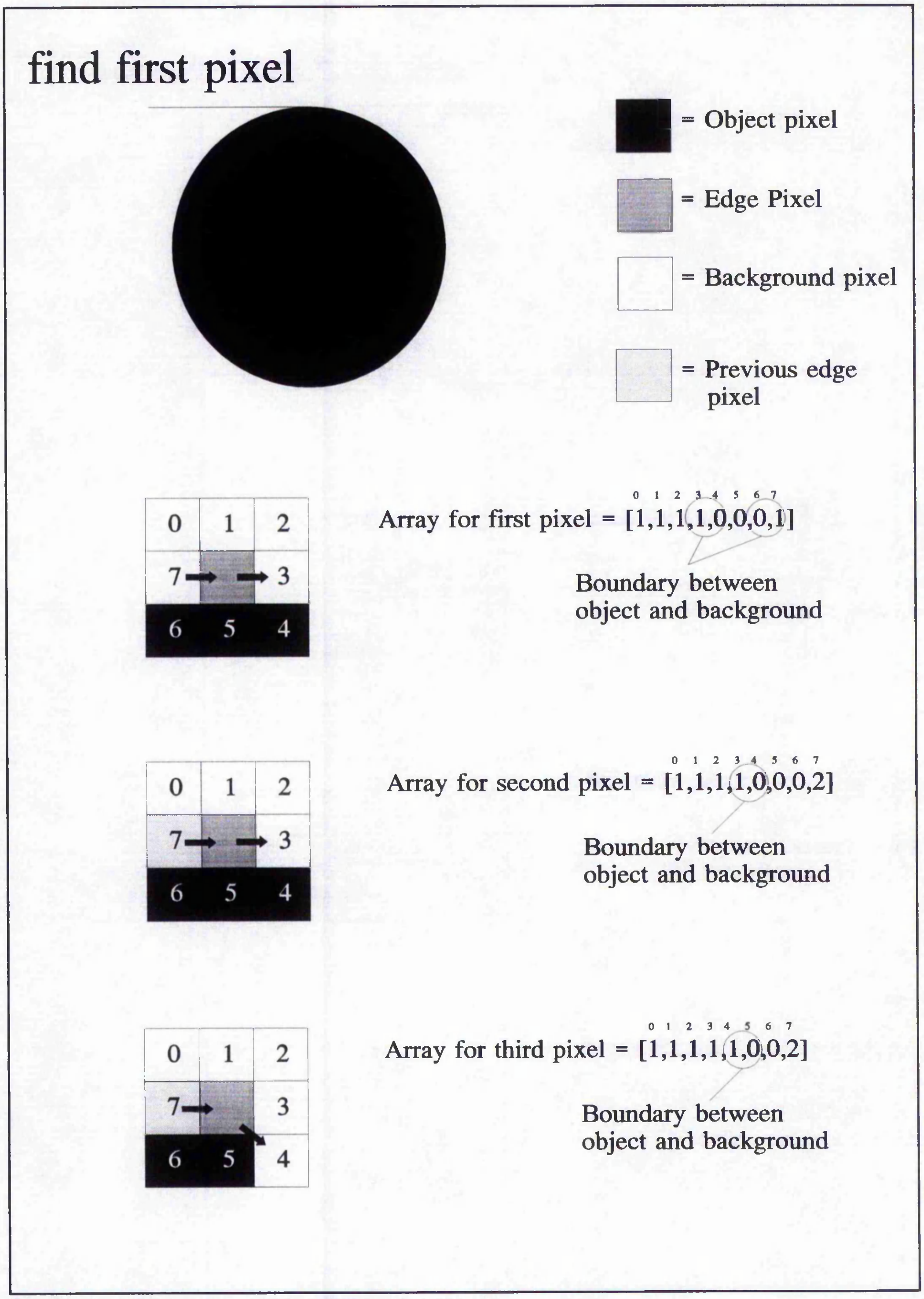

find first pixel
= Object pixel
= Edge Pixel
= Background pixel
= Previous edge pixel
0 1 2
7 3
6 5 4
0 1 2 3 4 5 6 7
Array for first pixel = [1,1,1,1,0,0,0,1]
Boundary between object and background
0 1 2
7 3
6 5 4
0 1 2 3 4 5 6 7
Array for second pixel = [1,1,1,1,0,0,0,2]
Boundary between object and background
0 1 2
7 3
6 5 4
0 1 2 3 4 5 6 7
Array for third pixel = [1,1,1,1,1,0,0,2]
Boundary between object and background

Figure 3.4 Operation of Novel Edge tracker.

3.3 Types of Registration Mark

The registration mark is used to align the screen to the substrate. In the manual screen printing process the screen is aligned by eye with the screen registration mark above the board. A sample print is taken and the displacement of the print due to misalignment and printing errors is shown by the offset of the printed registration mark from the board registration mark. The screen is then moved by the operator to remove this error. This process is repeated until the marks align as closely as possible. In this manual process the operator is checking the alignment of the whole registration mark, in an automated system the vision system finds a particular point on the registration mark and aligns it to the corresponding point on the substrate registration mark. The choice of what mark is used then relies on it having a clearly definable point than can be found automatically.

3.3.1 Arrow Registration Mark

Previous work within the department **[26]** had concentrated on the use of an arrow registration mark **Figure 3.5**. The arrows edge was detected using a horizontal edge detector and the equations of its two constituent lines calculated. From the equations it was possible to calculate the point at which the two lines intersected (i.e. the apex of the arrow). This intersection point was used as the registration point. This form of registration mark had problems because towards the apex of the arrow the edge did not come to a distinct point, but a curve which meant that points towards the centre had to be ignored, as did points towards the ends of the lines. The result of this was that only a few points could be used to define the two lines, and hence their intersection. This meant an accuracy of only +/- 2 pixels. This problem could be overcome by a shape which has a boundary that can be described by one equation with a single reference point.

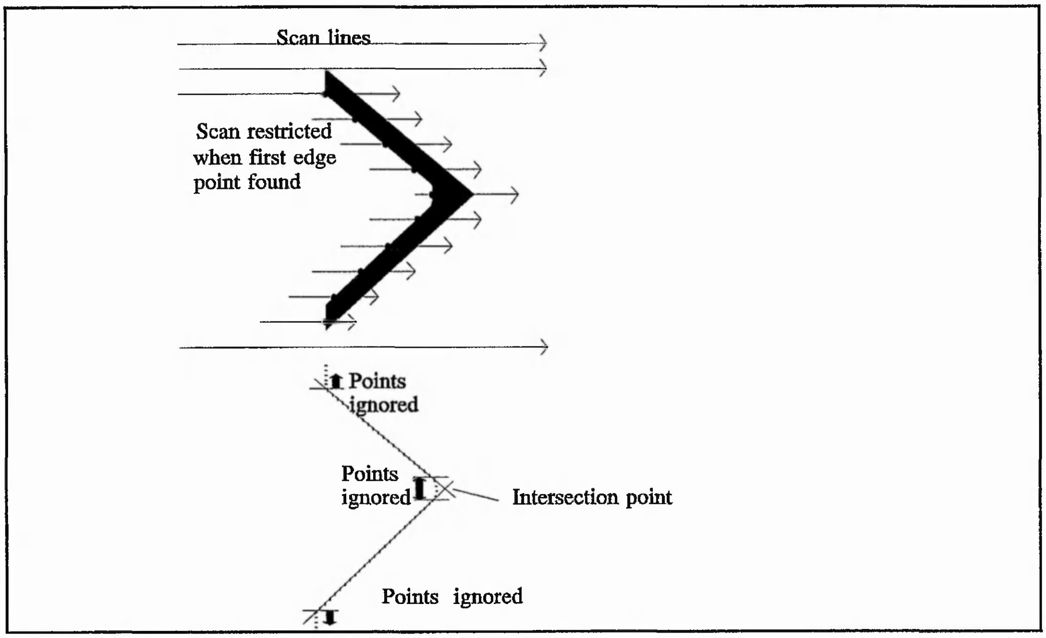


Figure 3.5 Arrow registration mark.

3.3.2 Circle Registration Mark

One shape that has all its edge points aiming towards one common point and may be described by a simple equation is a circle. Most registration marks already used in the printing industry tend to be some form of circle usually incorporating cross-hairs to form a 'bombsight'. Considering these factors work was concentrated on location and recognition of circular registration marks. Three different processes for finding the centre of a circle were used, these were:

1) Least Squares Fit
2) Hough Transform
3) Intersection of chord perpendiculars.

3.4 Circle Centre Finding Methods

3.4.1 Least Squares Fit

If an equation is known to fit a set of data a useful method of fitting the shape defined by that equation to the data is the method of least squares (LS). The LS model says "Find the values of the constants in the chosen equation that minimise the sum of the squared deviations of the observed values from those predicted by the equation." If the registration mark is known to be a circle its equation is known:

$$(x-f)^2 + (y-g)^2 = r^2 \tag{3.11}$$

where (f,g) is the circle centre radius r,

and (x,y) is a known edge point.

Multiplying **(3.11)** out gives:

$$x^2 + y^2 - 2fx - 2gy + f^2 + g^2 - r^2 = 0 \tag{3.12}$$

$$let \quad f^2 + g^2 - r^2 = -c \tag{3.13}$$

Simplifying:-

$$\therefore \; x^2 + y^2 = Z = 2fx + 2gy + c \tag{3.14}$$

$$f(x, y) = q_0 + q_1x + q_2y \tag{3.15}$$

where q_0, q_1 and q_2 are the polynomial coefficients.

The least squares estimates are those of f(x,y) that minimise the function

$$E = \sum_{i=1}^{n} (f(x, y)_i - Z_i)^2 \tag{3.16}$$

$$= \Sigma f_i^2 - \Sigma 2Z_i f_i + \Sigma Z_i^2 \tag{3.17}$$

i.e. the differentials of E with respect to the polynomial coefficients q_0, q_1 and q_2 must equal zero,

$$\frac{\delta E}{\delta q_{0,1,2}} = 0, \tag{3.18}$$

this gives,

$$2\Sigma f \frac{\delta f}{\delta q} - 2\Sigma Z_i \frac{\delta f}{\delta q} = 0 \tag{3.19}$$

$$\therefore \ \Sigma f \frac{\delta f}{\delta q} = \Sigma Z_i \frac{\delta f}{\delta q} \tag{3.20}$$

$$\frac{df}{dq_0} = 1, \ \frac{df}{dq_1} = x, \ \frac{df}{dq_2} = y, \tag{3.21}$$

$$\Sigma f \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \Sigma Z_i \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \tag{3.22}$$

$$\therefore \Sigma \ (q_0 + q_1 x + q_2 \ y) \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} \Sigma Z_i \\ \Sigma Z_i x \\ \Sigma Z_i y \end{bmatrix} \tag{3.23}$$

The above is carried out and solved in the function lsfcent() in **Appendix 4**, this function places the edge pixels in the image using the function matricise() and solves

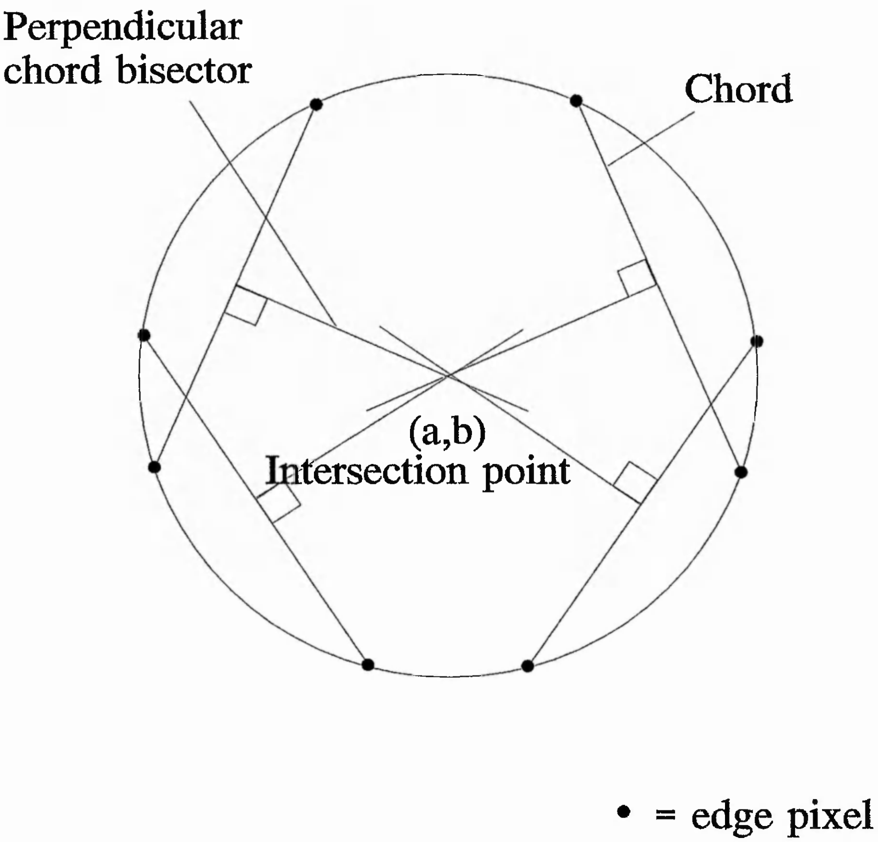
$$\begin{bmatrix} \Sigma 1 & \Sigma x & \Sigma y \\ \Sigma x & \Sigma x^2 & \Sigma xy \\ \Sigma y & \Sigma xy & \Sigma y^2 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} \Sigma Z_i \\ \Sigma Z_i x \\ \Sigma Z_i y \end{bmatrix} \qquad \textbf{(3.24)}$$

(3.24) using the function solve() to give values for q_1 (2f) and q_2 (2g). Knowing these values (f,g) it also possible to calculate the radius of the registration mark.

This method of centre finding was fast and consistent. In stationary centre finding tests it had an accuracy of +/- 0.002 pixels. It also proved robust in the presence of noise and slight imperfections in the registration marks.

3.4.2 Intersection of Chord Bisectors

If a perpendicular is taken from a bisected chord on a circle, the centre of that circle should lie on that perpendicular. If a perpendicular is taken from a second bisected chord on the circle, the point at which the two perpendiculars intersect should be the circle centre. This process is shown in **Figure 3.6.** This is the case for a perfect circle, in the case of the detected edge of a circular registration mark two chords would not define the circle to a very high degree of accuracy, it is therefore necessary to take a number of chords around the edge and to find an average value for x and y. The highest degree of accuracy is found by taking chords at approximately 90° to each other. This is achieved by carrying out an edge detection, and from this determining the number of edge pixels, n. From this value, n, the approximate number of edge pixels in 90°,N, can be calculated (N=n/4). This process is fast and accurate for good images but is unable to cope with incomplete or occluded circles.



Pairs of perpendicular chord bisectors are found using all the edge pixels. For each pair an intersection point is calculated, and an average value for the centre is evaluated from all the pairs.

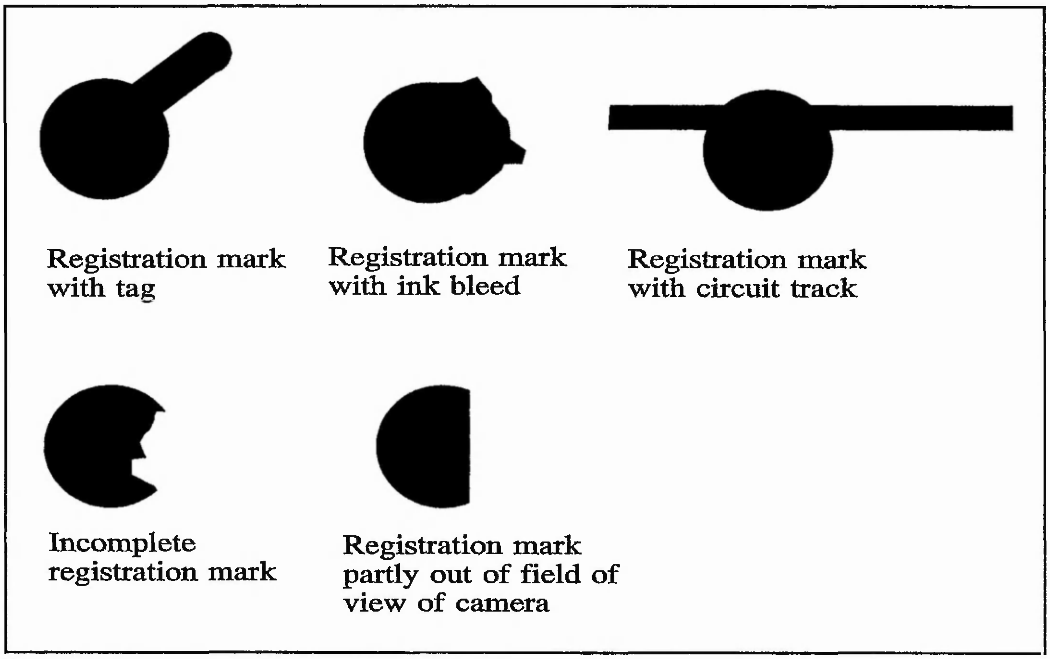
Figure 3.6 Intersection of Chord perpendiculars.

3.4.3 Hough Transform

Some circumstances occur when it is not possible to 'see' the complete registration mark, it may be partly out of the view of the camera, or part of the edge may be distorted due to a poor print, or the registration mark may have an additional tag. Examples of these occluded registration marks are shown in **Figure 3.7**. The two methods of centre finding previously outlined could not cope with these errors without giving some deviation from the true centre, unless given a method to ignore such spurious data. A method of centre finding was required that either ignored the additional data provided by the tag or blur in **Figure 3.7** or creates a situation where that data has no influence on the process of centre finding. Hough proposed a transform **[27]**, the Hough Transform (HT), which generates information about parameterised features such as circles or straight lines from the information of the edge locations. This original work was aimed at machine recognition of complex lines in photographs. The use of the HT for recognizing occluded and distorted objects is well researched. Turney et al. **[28]** used the HT (and template matching) for the recognition of occluded objects for use in industrial automation. The ability of the HT to cope with variance from an expected shape was used by Davies **[29]** to detect blunt corners in food products. Corners are relatively easy to detect with sharply defined metal objects but with food products more variance in corner shapes occurs due to the brittle or soft nature of most foods.

The HT can be used for more than just centre finding, it's main use has been in the process of pattern recognition and location and it is on these subjects that most work has been done. Duda and Hart **[30]** developed the HT for the detection of lines in an image more efficiently than Hough's original. Ballard **[31]** further developed the HT for detection of arbitrary shapes using what he called a Generalised Hough Transform (see later section). The HT's use as a versatile pattern recognition tool will be described in detail in a later section (3.6.2).

Although the HT may be used with any shape a circular registration mark will be used to explain its functioning. The HT is used to generate information about



Registration mark with tag

Registration mark with ink bleed

Registration mark with circuit track

Incomplete registration mark

Registration mark partly out of field of view of camera

Figure 3.7 Occluded or incomplete registration marks.

parameterised features from the information of the edge pixels, in the case of a circle the parameters, or features, that define it are its radius and centre. A circle with centre (a,b) can be defined by the Cartesian equation:

$$(x-a)^2 + (y-b)^2 = R^2 \qquad \textbf{(3.25)}$$

where x and y are the locations of edge pixels,

It may also be descried by the parametric equation:

$$(a+R\cos\theta,\ b+R\sin\theta) \qquad 0 \leq \theta < 2\Pi \qquad \textbf{(3.26)}$$

Once the edge data for an object has been generated by some edge detection process as described earlier, all the x and y edge information is available, although it may not be part of the object, it could be a tag, a blur, or just noise. If the pixel is part of the object it must satisfy the equation:

$$(a,\ b) = (x+R\cos\theta,\ y+R\sin\theta) \qquad 0 \leq \theta < 2\Pi \qquad \textbf{(3.27)}$$

This means that it must lie on a circle radius R with centre at (x,y). The centre of the circle (a,b) is the parameter that is required, therefore knowing R it should be possible to find (a,b). The next step is to define a 'parameter space' within which the values a and b can be found. This space could take in the whole image if nothing about the object is known, but it is usually possible to make some estimate as to where the centre may lie so that the parameter space can be cut to a certain range within which a and b may lie. Within the parameter space an accumulator array is defined, ACCUM[a,b], with all possible values of a and b, initially set to zero. For each pixel generated by the edge operator (x,y) (3.27) is evaluated and if any values are within the range of ACCUM[a,b], the corresponding position within the array is incremented. This process is shown in **Figure 3.8**. Clearly a great deal of data has to be calculated all of which uses floating point operations, which makes the process computationally intensive. **Figure 3.8** shows how much excess information is generated, to make the process less intensive it is necessary to reduce the range of data calculated just to that which lies within the range of the accumulator array. Firstly the boundary gradient is calculated and the centre should lie somewhere in the direction of a perpendicular to this gradient, this gives the possibility of two directions, this can be narrowed down if some idea of the circle centre is known, either through taking moments or least squares. Values are then only calculated if they lie within range of this of this value.

When all edge pixels have been evaluated the accumulator has to be searched to find a maximum value, this corresponds to the circle centre. **Figure 3.9** shows part of the array, with the maximum value. This explanation of the Hough Transform, and it's use for circle centre finding has assumed that the radius of the circle is known. This is rarely the case, usually some idea of its value may be known but not its exact value. The process needs to be able to find the centre without knowledge of the value of R. This may be done by adding R to the parameter space and defining a three dimensional accumulator array ACCUM[a,b,R]. As with restricting the range of values within which a and b lies, it is also necessary to restrict the range for R. The process for generating the accumulator array is carried out as before, but this time using the different values for R. The generated accumulator array is shown in **Figure 3.10** and is searched to find

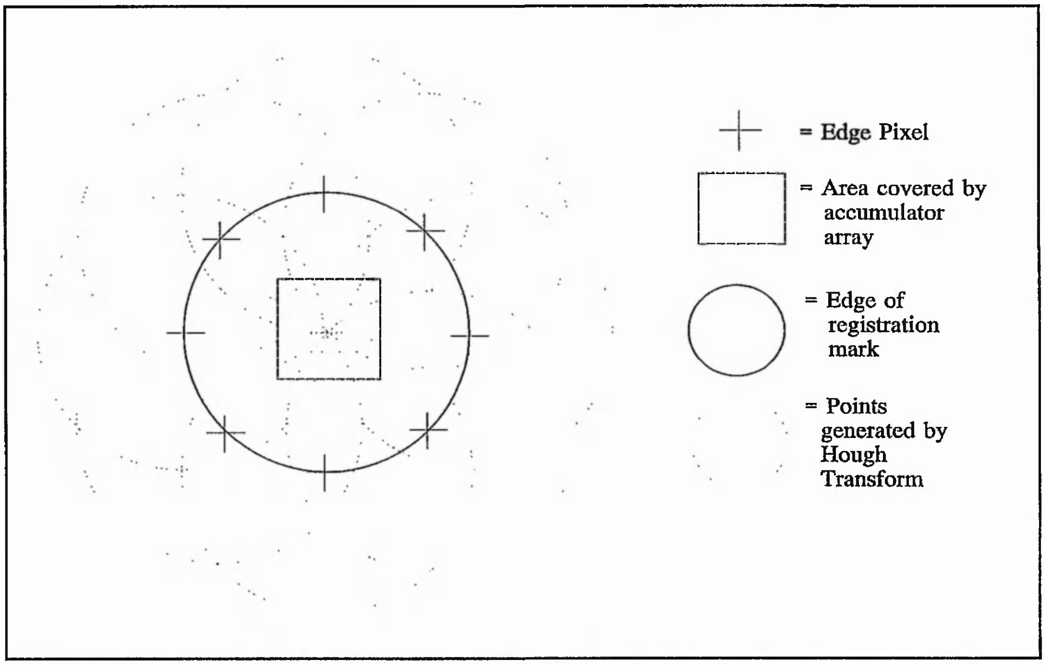
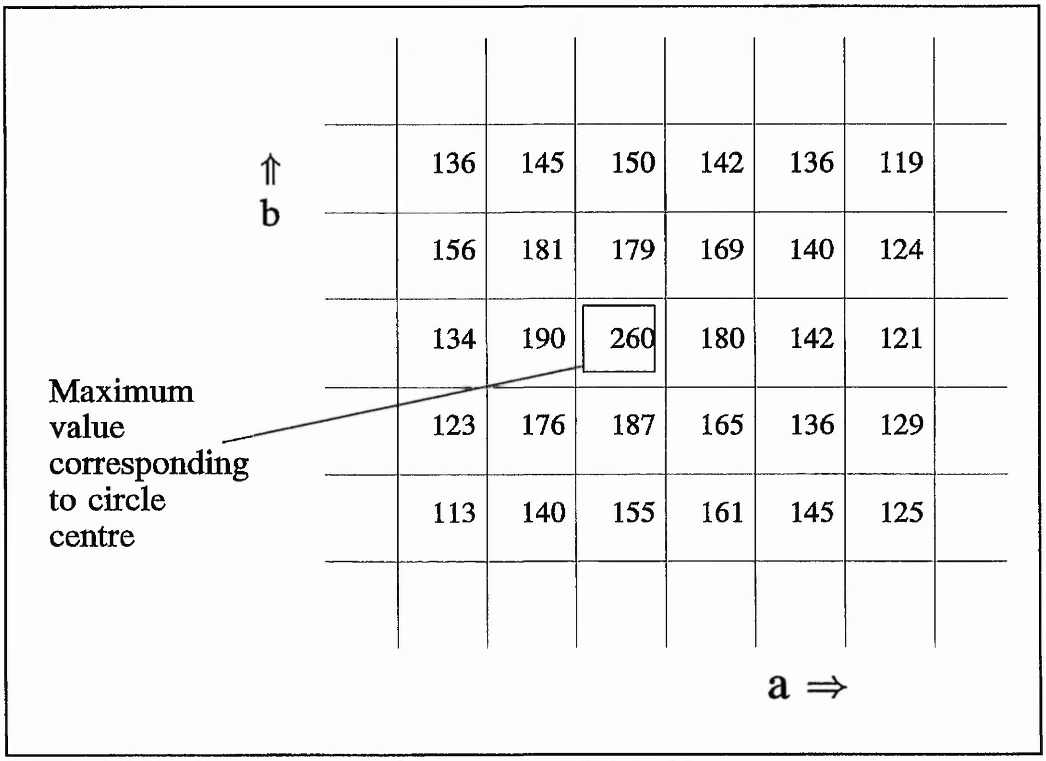

= Edge Pixel
= Area covered by accumulator array
= Edge of registration mark
= Points generated by Hough Transform

Figure 3.8 Hough Transform.


⇑
b
Maximum value corresponding to circle centre
a ⇒

136	145	150	142	136	119
156	181	179	169	140	124
134	190	$\boxed{260}$	180	142	121
123	176	187	165	136	129
113	140	155	161	145	125

Figure 3.9 Section from accumulator array ACCUM[a,b].

the maximum value that corresponds to the circle centre.

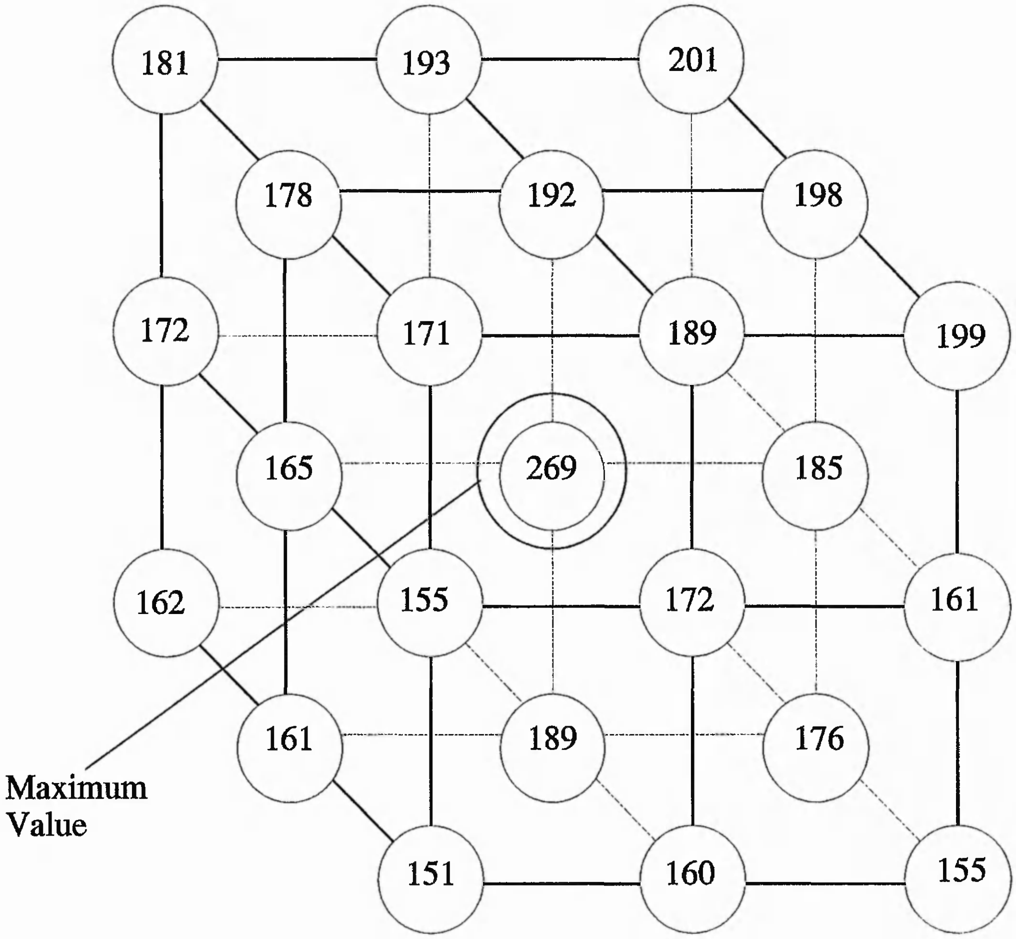


Figure 3.10 Section from three dimensional accumulator array ACCUM[a,b,R].

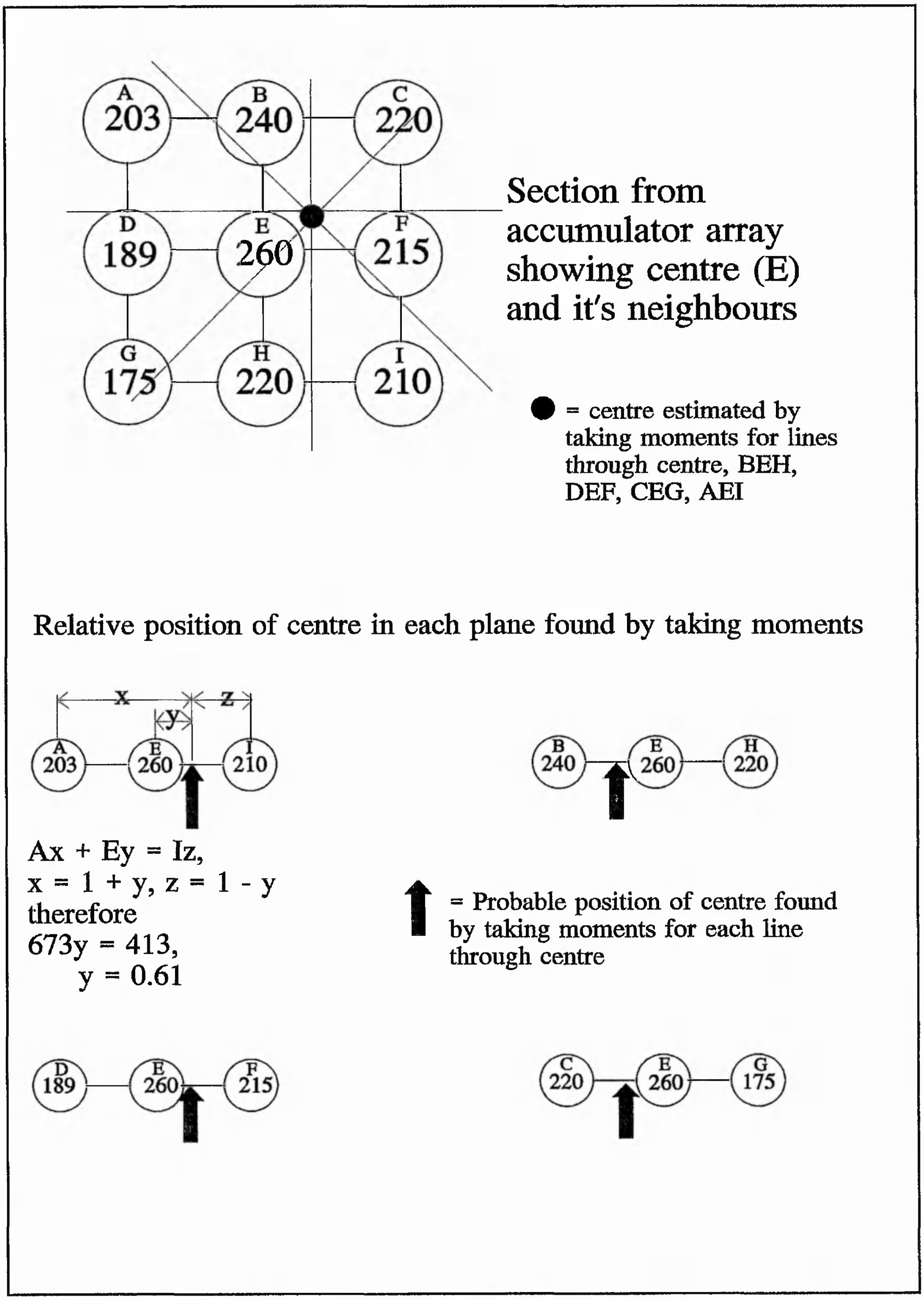

A 203
B 240
C 220
D 189
E 260
F 215
G 175
H 220
I 210
Section from accumulator array showing centre (E) and it's neighbours
● = centre estimated by taking moments for lines through centre, BEH, DEF, CEG, AEI
Relative position of centre in each plane found by taking moments
x
y
z
A 203
E 260
I 210
Ax + Ey = Iz,
x = 1 + y, z = 1 - y
therefore
673y = 413,
y = 0.61
B 240
E 260
H 220
= Probable position of centre found by taking moments for each line through centre
D 189
E 260
F 215
C 220
E 260
G 175

Figure 3.11 Finding centre at sub-pixel level

The HT, although slower and more computationally intensive than previously discussed methods of centre finding, excels when the circle is incomplete or distorted because information generated by the irregularities will lie out of the range of the accumulator array and are thus ignored, only pixels that are part of the circumference will increment the accumulator array within a close proximity of each other (i.e the maximum value). The HT only generates a centre to within a pixel, as this is the level of quantization, but it is possible to examine the centre at a sub-pixel level by either quantizing at a lower level, (i.e. increment the accumulator array at perhaps half or quarter pixel intervals), or to examine the values around the maximum value in the accumulator as shown in **Figure 3.11**.

The algorithm for using the Hough transform to find a circle centre is shown in **Table 3.1**.

```
Initialise accumulator array ACCUM[a,b,r];
        for(each edge pixel (x,y)){
                for(radius range){
                        calculate edge pixels centre(x,y);
                        if(value lies with in range of ACCUM[a,b,r])
                                increment ACCUM[a,b,r];
                }
        }
search ACCUM[a,b,r] for maximum;
```

Table 3.1 Algorithm for circle centre finding using Hough transform.

Generalised Hough Transform

The Hough Transform described earlier for circle centre finding relied on the knowledge of the parametric equation of the shape to perform the transform from image space to the Hough transform space. D.H.Ballard **[31]** devised a process of using the boundaries of an 'arbitrary non-analytic shapes (i.e. shapes that could not be described parametrically) to construct a mapping between image space and Hough Transform space', called the Generalised Hough Transform (GHT). The process used the relationship between the boundary points and a reference point within the shape to produce a look up table for the shape known as an R-Table. The GHT can be used to detect the shape in an image by mapping the image through the R-Table, and it is possible to allow for variations in the shape such as orientation, size and scaling. This has great implications for an inspection or registration system because it is the possible to train the system to recognise any registration mark or object just by calculating an R-Table from a prototype image.

The first stage in the process is to select a reference point within the object of interest (x_{ref},y_{ref}), this is an arbitrary point the only factor necessary for it's selection is that it lies within the object, and when this has been selected an R-Table for the object is produced. The R-table classifies the image in terms of a boundary orientation angle for each boundary point with reference to (x_{ref},y_{ref}) image (Φ), the distance to that point (r), and the relative angle of that point to the reference (α).

Figure 3.12 shows the form of the R-table.

Angle measured from figure boundary to reference point	Set of radii $\{\mathbf{r}^k\}$ where $\mathbf{r} = (r,\alpha)$
Φ_1	$\mathbf{r}_1^{\ 1}, \mathbf{r}_2^{\ 1}, \ldots, \mathbf{rn}_1^{\ 1}$
Φ_2	$\mathbf{r}_1^{\ 2}, \mathbf{r}_2^{\ 2}, \ldots, \mathbf{rn}_1^{\ 2}$
.	.
.	.
Φ_m	$\mathbf{r}_1^{\ m}, \mathbf{r}_2^{\ m}, \ldots, \mathbf{rn}_1^{\ m}$

Figure 3.12 Lay out of the R-table (Ballard, 1981).

The Boundary gradient is calculated by using the method of least squares.

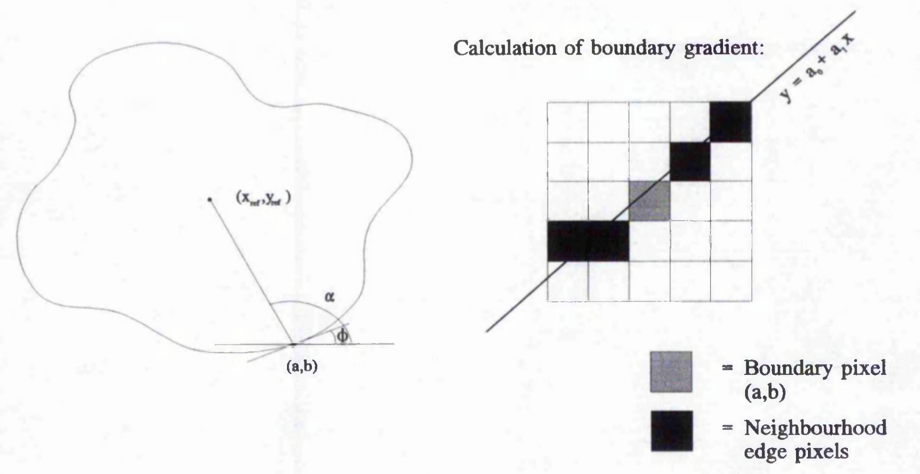

(x_ref, y_ref)
α
ϕ
(a,b)
Calculation of boundary gradient:
y = a0 + a1x
= Boundary pixel (a,b)
= Neighbourhood edge pixels

Figure 3.13 Geometry of shape used in formation of R-Table.

$$y = a_0 + a_1 x \tag{3.28}$$

$$\hat{a}_0 = \bar{y} - \hat{a}_1 \hat{x} \tag{3.29}$$

where $\hat{a}_0$ and $\hat{a}_1$ are least square estimates of a_0 and a_1.

$$\hat{a}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} \qquad \textbf{(3.30)}$$

The gradient of the boundary at the point is given by $\hat{a}_1$, and from this the boundary orientation (Ω) can be calculated.

$$\Omega = \arctan(\hat{a}_1) \qquad \textbf{(3.31)}$$

The boundary orientation angle is used as an address to the R-Table.

The Adaptive Hough Transform

The Hough Transform has been described as being computationally intensive. It's implementation was made more efficient by Illingworth and Kittler **[32]**, they developed what they called an 'Adaptive Hough Transform' (AHT) which uses a small accumulator array to carry out an iterative "coarse to fine" accumulation and search process to find the peaks in the Hough parameter space. The use of a smaller array means far less computation and storage requirements. The AHT works by accumulating the HT in the small array with a "coarse" setting for the parameter space, the array is then searched to find the peak values, once found they are analyzed and the parameter space redefined so that the area of interest can be investigated in greater detail. This process is repeated until the parameters being investigated have been evaluated to a predefined level of accuracy.

3.5 Segmentation

The complete image left by image enhancement is still a complete 256 grey-level image. This represents a large amount of data that gives no quantified description of the individual parts that make up the image, it may be clear to the human eye that a registration mark or circuit tracks are part of the image, but to the vision system there is only 245 Kbytes of grey-level data. The next step following enhancement is the

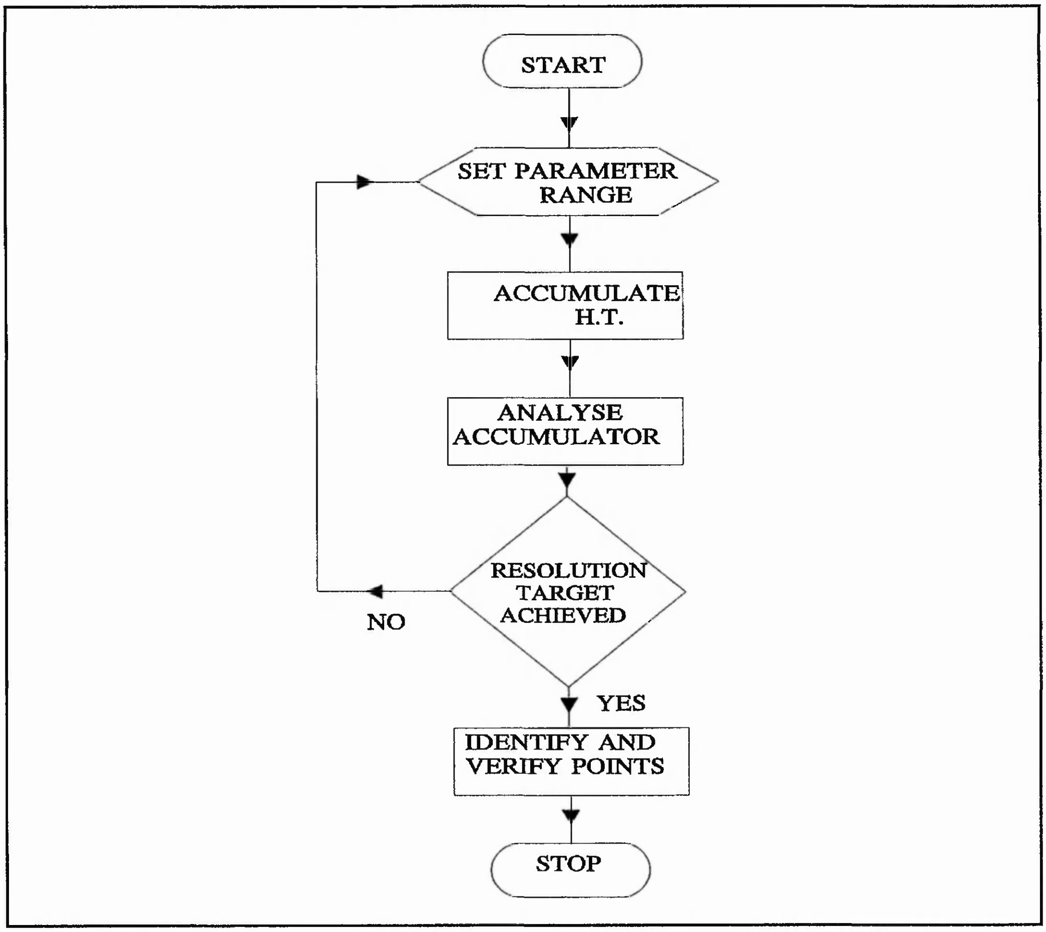


Figure 3.14 Flow chart for AHT algorithm (Illingworth and Kittler 1987)

process of identifying the regions of interest within the image and presenting them in a form understandable to what ever algorithms are to be used to classify and interpret the image. This process is known as segmentation and can take the form of many different processes. The project concentrated on two processes in particular, thresholding and blob detection.

3.5.1 Threshold Selection

Thresholding has already been discussed in Section 2.3.5 as a form of image enhancement in which a 256 grey level image can be transformed to a binary (black and white) image. It can equally be described as being a segmentation process because careful selection of the threshold level enables selection of one particular aspect of the image. The histogram shown in **Figure 2.4a** has two clear peaks and is therefore described as being bimodal. The lower peak is the darker end of the image shown in **Photograph 2.1b**, and corresponds to the registration mark. Selecting a threshold level between the peaks all points below that level will become black and all points above it will become white. The selection of this point is relatively simple using the MVP-AT software function threshold(); on an image histogram. This function carries out an analysis of the histogram to determine the number and location of all maxima and minima associated with the histogram. The format of this function is:

threshold(ntimes, minlist, maxlist, size, histbuf);

where:

- ntimes is the number of smoothing operations carried out on the histogram,
- minlist is the work buffer into which the function places all the minima of the histogram, with the first entry giving the number found,
- maxlist is the work buffer into which the function places all the maxima of the histogram, with the first entry giving the number found,
- size is the limit of the number of extrema to return in minlist and maxlist,
- histbuf is the buffer into which the histogram data is placed.

The thresholding process outlined above is the most basic thresholding technique with just one threshold point which proved to be sufficient for use with bimodal histograms. If the histogram has three or more major peaks and the required peak is a central peak, it is then necessary to carry out a process of multi-thresholding (i.e. the selection of more than one threshold level). An example of this process is the work carried out by Hertz and Schafer [33]. They devised a technique that found the different threshold levels by analysis of the image histogram and then used edge matching to adjust the threshold levels of each of the histogram peaks that corresponded to one part of the image. They demonstrated it's effectiveness in picking out different layers of a multilayer PCB.

The wide range of different thresholding techniques can be seen in the survey produced by Sahoo et al. [34]. This outlined the different approaches to thresholding and discussed the relative performance of nine different techniques.

3.5.2 Blob Detection

Simple thresholding will only separate objects of different grey levels. This is sufficient when the object of interest (OI) has a unique grey level, but when this is not the case then another process is required to differentiate the OI from any other object with a similar grey level. The next step is to separate the individual components of the image in a process known as blob detection.

The blob detection may be based on separating either the individual edges of the different parts or the complete objects. Methods for separating the objects by edge tracking have been discussed elsewhere in this chapter (Novel and Paperts). The other method is to check connectivity between pixels, all pixels that are connected are described as belonging to the same object. This method of blob detection uses a programming process called "Linked Lists" which is described in section 4.4.5.

3.6 Image Analysis and Pattern Recognition

Through the process of segmentation and blob detection the object of interest is separated from the rest of the image or a number of possible objects will have been defined. The next process is the automatic extraction of relevant information the process of image analysis. There are a vast range of Image analysis techniques, such as Hough Transforms, pattern recognition processes like chain coding and template matching, and centre finding processes. The choice of techniques relies on what process is to be carried out, image analysis can be split into three separate categories:

1) Inspection, this is the process of checking if the object is present or complete.
2) Location, finding the relative position and orientation of the object of interest,
3) Identification, determining what type of object is being viewed.

These categories are not necessarily independent of each other, it may be necessary to identify the object before its location is checked, or it's identity might have to be found to choose what inspection process has to be carried out. Each of the categories may also use similar processes, for example, inspection might use a chain coding technique, location may use template matching techniques and the Hough Transform, and identification may use a combination of these. It is therefore necessary to describe a range of techniques and describe their uses within image analysis.

3.6.1 Boundary Representation

The storage of pixel boundary information tends to be memory intensive, boundary representation, usually in the form of chain codes, aims to minimise the information necessary to represent a boundary. Chain coding **[35]** is a method of describing a boundary in the form of 2 or 3-bit numbers representing the boundary direction, together with a start and finish coordinate (x,y) for the boundary. **Figure 3.16** represents the components that make up the chain coding operation in an eight-connected process (i.e. eight directions) but it can also be implemented as a four-connected process. A point is found on the boundary of the object, this is designated the start point, and the edge is tracked using a suitable edge tracking procedure **[see Novel Edge Tracker]**. When moving to the next point along the boundary the direction moved (0-7) is recorded and added to the chain code.

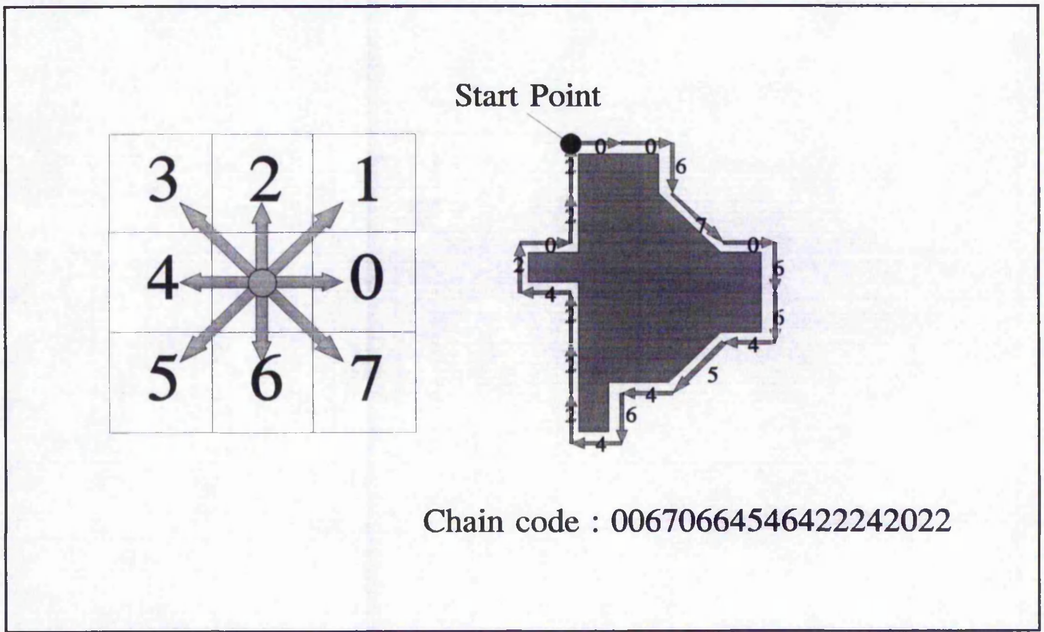


Figure 3.16 Chain Code

Chain coding may also be used to calculate the area enclosed by a boundary if the starting pixel is known, the algorithm for this, using a four-connected example, is shown in Table 3.2.

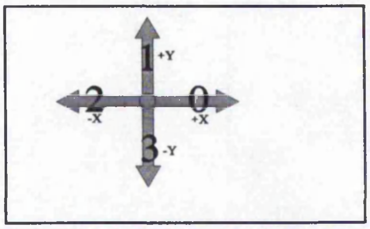


Figure 3.17 Four-connected chain codes

```
starting pixel = (x,y);
initial area = 0;
y_position= y;

for(incrementing through each element in the code){
        if       (code==0)    area = area - y_position;
        else if  (code==1)    y_position = y_position + 1;
        else if  (code==2)    area = area + y_position;
        else if  (code==3)    y_position= y_position - 1;
}

Chain Area = area;
```

Table 3.2 Algorithm for chain coding.

Chain coding, in the case of an eight-connected example, is described as being a non-uniformly sampled function because the lengths of each element in the code vary, 1 for horizontal and vertical links, and $\sqrt{2}$ for diagonal links.

3.6.2 Pattern Recognition - An 'Intelligent' Centre Finding System

It could not be guaranteed that boards from external suppliers would come with registration marks that were of the same design, so it was necessary for the registration system to be able to recognise the registration mark used and decided what centre finding process should be used. This was using the process of pattern recognition as a decision making tool, as opposed to an inspection process in which it is used to decide whether an object is present or complete. The process can be further improved by making the pattern recognition system 'intelligent', this would mean that it could be taught new registration marks and decide what point on that mark should act as the registration point. The major advantage of such a system is its inherent flexibility, the problem with the first system described is that all registration marks to be used by the system must have their own centre finding algorithms, each individually designed and programmed by a highly skilled programmer / vision expert, this can be very expensive if a new registration mark is to be used. The system developed uses a Generalised Hough Transform to give this 'intelligent' registration system. The teaching process involves placing an example of the registration mark in front of one of the vision system cameras and taking a snap shot of the image, the image is enhanced (as described earlier) and an edge detection is carried out. An arbitrary point is selected within the object and an R-Table is drawn up for the object **(Figure 3.12)**. The R-Table is a description of the registration mark in the form of a look-up table, it can now be used to find the object in subsequent images, and in the process of finding the object it will calculate the position of the reference point which is used as the registration point. The drawback of this system is that it tends to be slower than other centre finding procedures (e.g. Least Squares) due to the computational intensity of the Hough Transform, but this problem could be overcome to a certain degree with improved computer hardware (e.g 486 PC or Sun Sparc Station) or by the use of parallel processing **[36][37] (see chapter 5)**. It has also proven to be less accurate than least squares (+/- 1 pixel or +/- 0.01 mm), this could be improved with the use of an Adaptive Hough Transform, which can be used to search smaller areas at a higher resolution than the Generalised Hough Transform.

Chapter 4

System Overview and Development

Chapter 4 : System Overview and Development

4.1 Reasons for the Work

There are two approaches to implementing an automated screen printing process. The first, and perhaps quickest route, is to take a standard automated screen printing machine complete with machine vision, such a system will cost anything from £150,000. Svecia produce such a machine which they describe as being fully automatic, it incorporates machine vision on a limited scale only as an aid to alignment for the operator, there is no automatic recognition and position finding of the registration marks. DEK and Panasonic produce machines with a greater use of machine vision for centre finding of print registration marks but at a greater cost. The drawback of this route to automation is the cost and the inflexibility of the systems, manufacturers can make alterations to machine specifications when they are ordered but additional changes may be impossible to make, an example of this may be the introduction of a control system linked to inspection which would require major machine alterations.

The second route to automation is that taken by this project. A standard semi-automatic machine can be retro-fitted with a control system with machine vision. This allows development to the required specification of the end user allowing for the possibility of future developments, and if they are involved in the development at all stages then additions to the system can be added with greater ease. The major drawback of this approach is that it takes longer to produce a working system with high costs of the development team, but this can be easily out weighed by the cheaper costs of the required hardware and hopefully the increased performance and flexibility of the finished system.

4.2 The Sveciamatic SM

The Sveciamatic SM **(Photograph 4.1)** flat bed screen printer is described as being three quarter automatic in the advertising brochure and technical data that can be found in **Appendix 1** . It is designed for printing on sheet materials such as paper, cardboard, tin plates, and plastic sheets, for use in the graphical and electronics industries. The particular model used in this project can print a maximum surface area of 550 mm x 750 mm. The printing cycle is as follows:

1) The substrate to be printed is manually loaded onto the input table and is held in place by retractable tooling pins and a vacuum bed.

2) The input table moves in, the tooling pins retract when the table is moved in for a print, at which point the substrate is held in place by the table vacuum bed.

3) The printing head is lowered to approximately 4 mm (depending on screen, ink and substrate types) from the substrate (snap-off height).

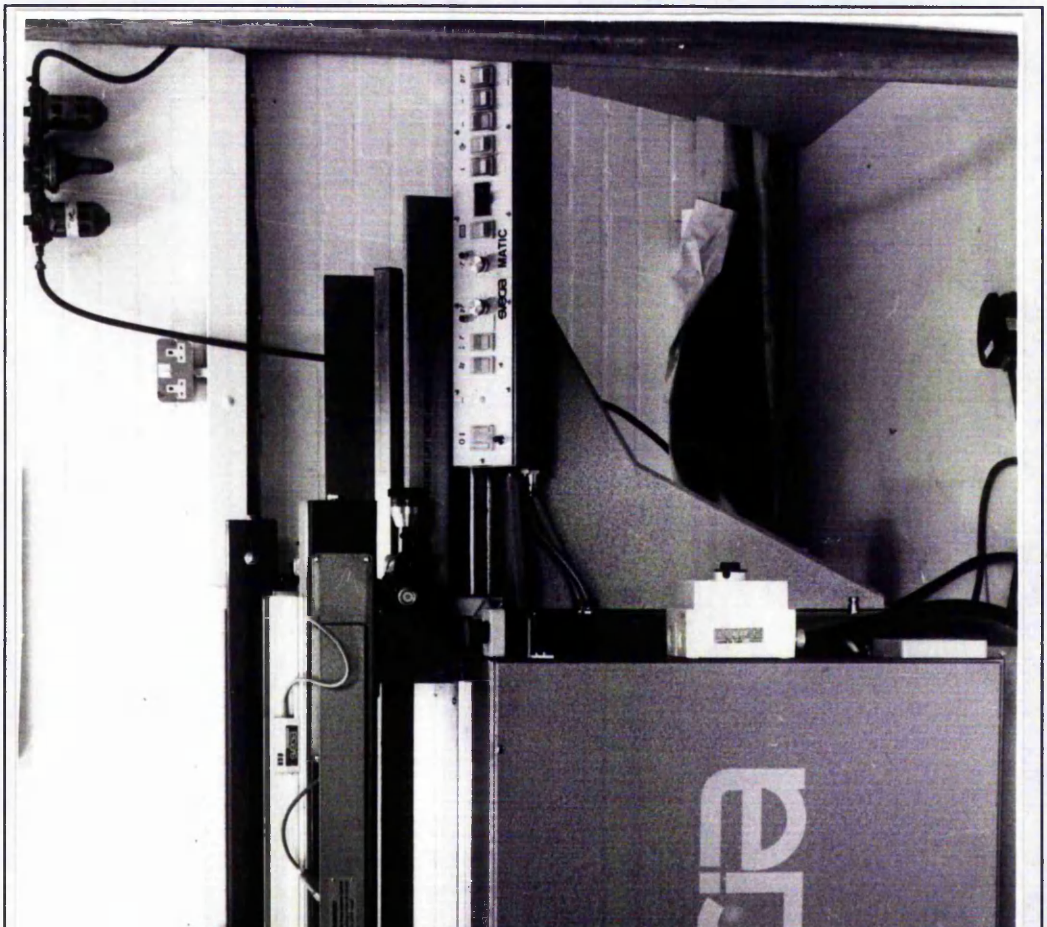
4) The squeegee passes across the screen, bringing it into contact with the substrate. This process passes the ink from screen to substrate by means of the pressure differential created by the movement of the squeegee.

5) The printing head retracts and at the same time the floodcoater redistributes the ink across the screen.

6) The input table moves out and simultaneously grippers remove the substrate to a feeder underneath the print head which feeds to a conveyor if attached.

With the standard machine it is possible to adjust the following machine printing parameters :-

- snap-off height - The height of the screen from the substrate, by means of a wheel mounted at the front of the machine.
- screen position - The position of the screen in the printhead, by means of three micrometer screw adjusters, giving horizontal movement in X, Y and Θ.



Photograph 4.1 Sveciamatic SM

- squeegee/floodcoater speed - The speed at which the squeegee or floodcoater move across the screen, by means of two dials.
- squeegee/floodcoater angle and pressures (height) - The angle and height of the squeegee or floodcoater relative to the screen. The height of the floodcoater determines the amount of ink deposited on the screen before printing. The squeegee pressure (height) affects the amount of ink deposited, and the angle affects the quality of the print, the optimum angle for most printing applications is about 30° to the vertical.
- squeegee/floodcoater stroke - The length of stroke of the squeegee/floodcoater, fixed by position of two limit switches.

The machine can be run in automatic mode, repeating through the above cycle, or semi-automatically jogging through each step of the cycle, the cycle is controlled by a series of cams within the machine.

4.3 The Manual Vision System

4.3.1 Four Camera

The first set-up and registration system developed used machine vision with four cameras, but with the standard manual screen alignment. The cameras were arranged in pairs, one pair over the machine input table, the other pair over the screen, this set-up is shown in **Figure 4.1**. This initial system was developed for two reasons; Firstly to test ideas to be used in the automated system in a full production environment and secondly because the original sponsoring company required a more accurate registration system on their production line.

Two cameras are needed to find a board or screen position, as a single camera would only find an X-Y position and would not be able to measure Θ, the angle of the board to the system. Using two cameras means that the relative positions of two registration marks on a board or screen may be found, making the alignment of screen to board more accurate. The initial system had two pairs of cameras. One pair over the screen to find its position, the other pair over the input table to find the board position.

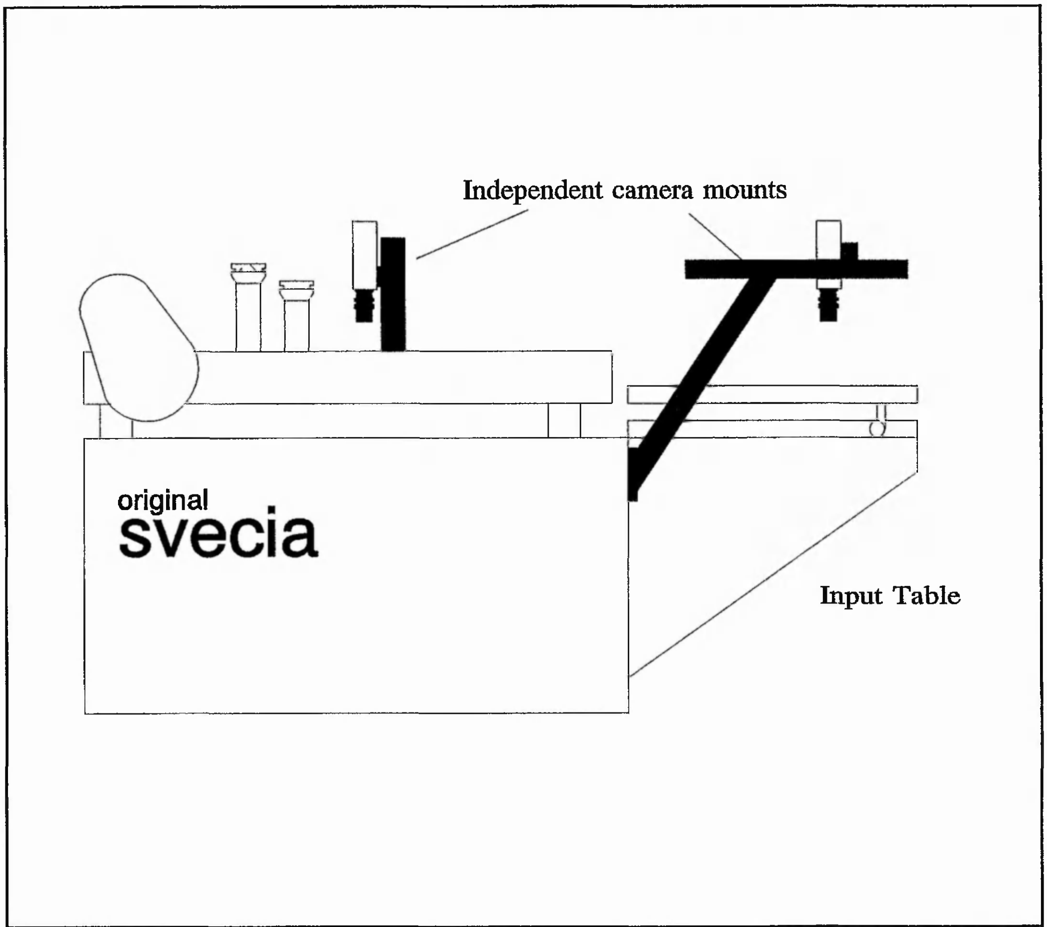


Figure 4.1 Prototype four camera vision aided manual registration system.

The cameras had to be accurately set-up. This involved ensuring that all four cameras were set at equal heights above the screen and input table and that they were all perpendicular to the work surface. This was to ensure that the displacements measured using each camera were to the same scale. The system was menu driven with an option to align the cameras, once this was selected cross-hairs appeared in the centre of the system monitor. Each camera on the input table was aligned so that the cross-hairs corresponded with the board registration marks, once this was achieved the table was brought in and, with the screen removed, the alignment operation was repeated for the screen cameras so that they aligned with the board. The screen was then put in the printing frame, and adjusted so that the screen registration marks aligned with the monitor cross-hairs. At this point the screen registration marks should have been directly above the board registration marks.

The next stage was to select the Print Registration option on the system menu. This function was used to align the screen to subsequent boards. The board to be printed was placed on the input table and a snapshot of each board registration mark was taken, and stored in the vision system memory. The table was then brought in and the screen alignment option was entered on the system menu. This function overlaid a transparent image of the input table over screen for each camera, this allowed the screen to be aligned with the corresponding registration mark on the board for each camera. At this stage the machine was ready to print.

The initial camera set-up procedures only had to be carried out once per print run, but could be repeated if prints showed poor registration, if the screen was changed, or if the cameras were knocked. The screen to board alignment process was carried out for each board if boards were dimensionally inconsistent, but when starting from blank or consistent boards these steps were only carried out once for each screen.

This system decreased the original screen alignment time to just three minutes from about thirty minutes, and for subsequent screen alignments after set-up, just one and a half minutes, the achieved accuracy of alignment was better as well but the possibility of a greater increase in registration accuracy was shown up. The lenses used were Cosmicar 12.5-75 mm zoom C-mount lenses with additional +3 close-up filters to give a horizontal field of view of 12 mm, the equivalent to 0.025 mm per pixel. These allowed closer examination of the resultant print and it showed a degree of shift from perfect registration of the order of 0.1 mm, even though the screen and board had been aligned correctly, in addition to this the error was consistent when the set-up procedure was repeated several times. This drift is shown in **Figure 4.4**. It is well documented that errors occur in printing **[8]** due to screen stretch or screen displacement, but it could not be conclusively said that this was the full reason for the error. For maximum system accuracy it was necessary to investigate any possible sources of error in all aspects of the process. For this purpose the system was divided into two specific parts that could be sources of error:

- Vision system and camera set-up,
- Printing process and machine set-up,

In order to measure errors due to the printing process and the machine it was first necessary to minimise errors due to the vision system. The setting up of four cameras was difficult and was a possible source of error, not only did each pair of cameras have to be at the same height, they also needed the same lens focus setting and over the same relative position on the board, if the four camera system was replaced by two cameras this error could be minimised. It has already been mentioned that two cameras are required to accurately find the position of the screen or board, therefore if two cameras were to be used they would have to be able to see both the screen and the input table. The overlaying of the stored image of the board registration mark on to the live image of the screen was another source of error, to do this the MVP keying function was used. The keying function test the LSB (least significant bit) of the blue OLUT, if it is 0 the camera input is shown and if it is 1 the video output is shown. This has the effect of showing a partly transparent image of the stored frame over the live image of the screen. Because the image is only partly transparent it obscures the screen image which makes alignment only accurate to approximately +/- 4 pixels (0.1 mm). The resolution of the system was 0.025 mm pixel, by increasing this the error due to alignment would be decreased.

To remove the errors due to the printing process it was necessary to improve the vision system as outlined above, this had to be done before any evaluation of printing errors could be achieved. This was done in the production of a two camera system.

4.3.2 Two Camera system

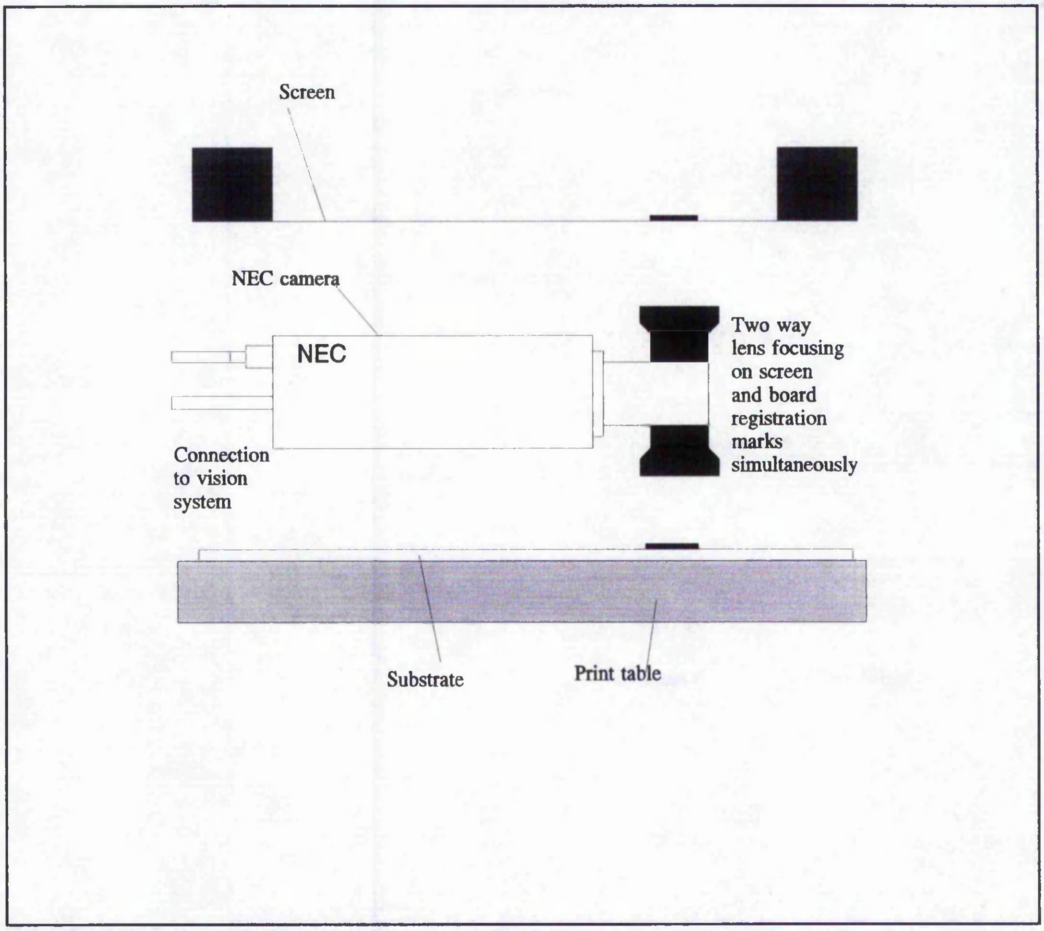
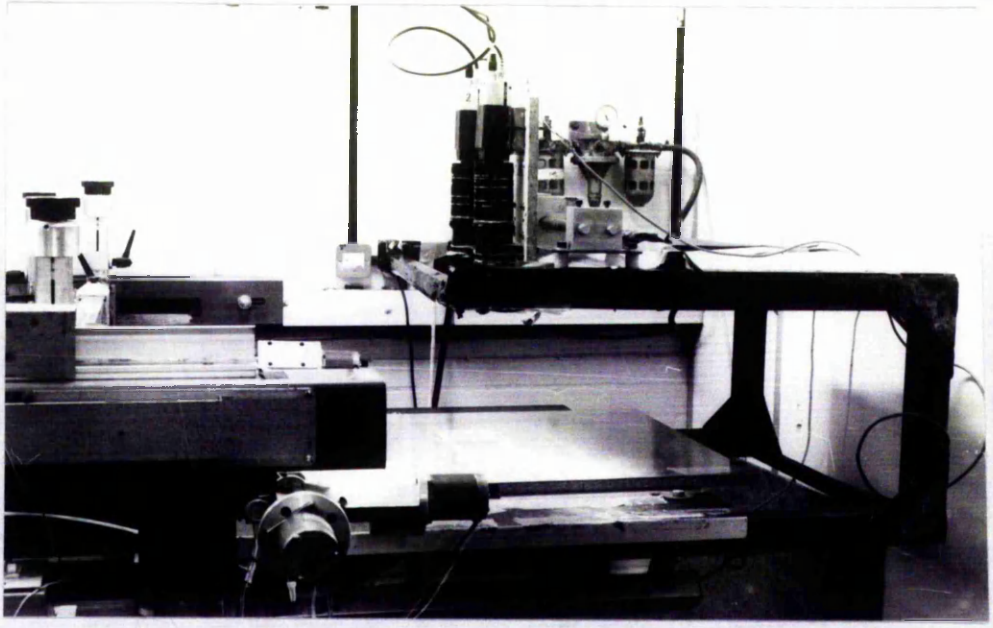

Screen
NEC camera
NEC
Two way lens focusing on screen and board registration marks simultaneously
Connection to vision system
Substrate
Print table

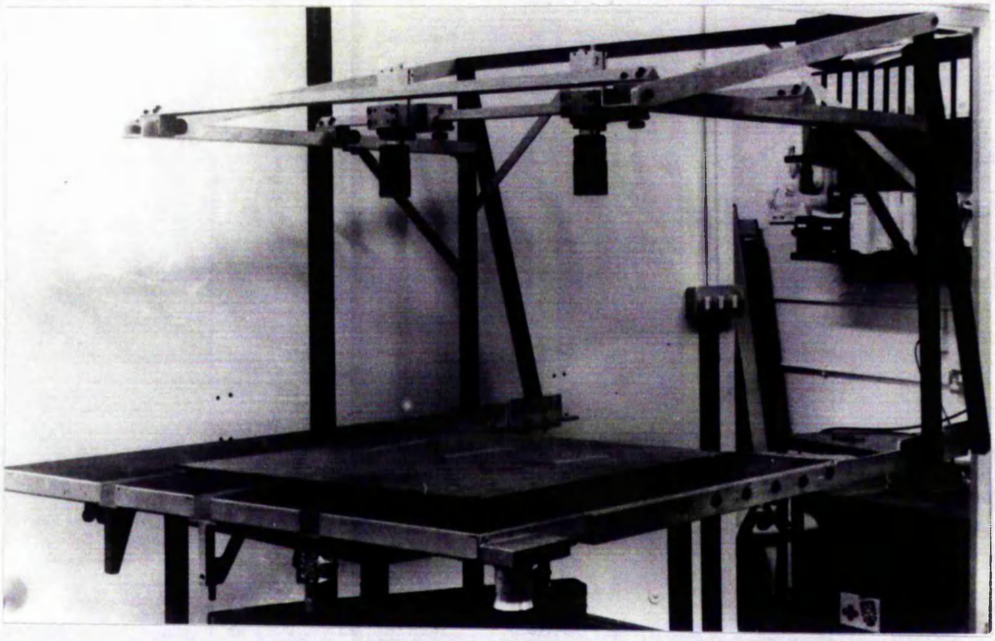
Figure 4.2 Dek two directional lens and camera system.

Dek manufacture a screen printing machine with machine vision, and they solve the two camera problem by having a standard NEC camera with a special two directional lens as shown in **Figure 4.2** illuminated by a ring of LED's. This system had one camera that found both positions by moving between the screen and input table. The printing head for the Dek is very different to the Svecia, and such a system would not be usable with the Svecia. An additional disadvantage of the Dek system is that the lenses were expensive units specially designed for Dek and gave no flexibility in lens magnification and did not give the resolution required.

The only feasible alternative was to mount the cameras to the input table so that they could view the substrate when the table was out, and the screen when the table was in. It can be seen from **Photograph 4.1** that Sveciamatic has very little space to do this, and that the table is made of a very light aluminium honeycomb to which it is difficult to attach anything. The problem is further heightened by the fact that the table moves in and out of the machine at high speed and the camera mount would be subject to high inertia effects. Two camera mounts were produced and are shown in **Photographs 4.2** and **4.3**. The first camera mount **Photograph 4.2** was made from 2" mild steel box section attached to the table by four 8 mm bolts. This was very rigid and showed no significant vibrational effects when the table moved, but had three significant disadvantages; Firstly the cam movement for the input table could only just cope with the weight, this was partially overcome by slowing the table movement down. Secondly the bolt heads came through the table surface, and it was therefore necessary to produce a pallet to raise the substrate above the bolts. Lastly, camera position adjustment was complicated and not very precise.



Photograph 4.2 The first two camera mount design.



Photograph 4.3 Second two camera mount design.

A second camera mount (**Photograph 4.3**) was designed and made from aluminium mounted to the table by approximately 50 Nutserts. This was at least a quarter the weight of the original mount, did not protrude through the table surface, and gave easier camera position adjustment. In static tests the frame proved rigid but when used it tended to vibrate for a few seconds after coming to rest, two additional struts partly removed this problem, but the frame still was not as rigid as the original.

The four camera system had relied on ambient light but this had proved insufficient for examination of the screen. For the two camera system a twin microscope light source was used, this gave a powerful focused beam of light. It should be noted at this stage that very little image processing was used by the system, and lighting was not as crucial as it would later prove to be, but still posed some problems. The registration mark on the substrate presented few problems in lighting other than glare from wet inks and the possibility of shadowing on thick prints. These problems are shown in **Figure 4.3**, the glare is overcome by making the angle of incidence of the light approximately 45 degrees to the horizontal, much shallower than this and shadowing may occur. The

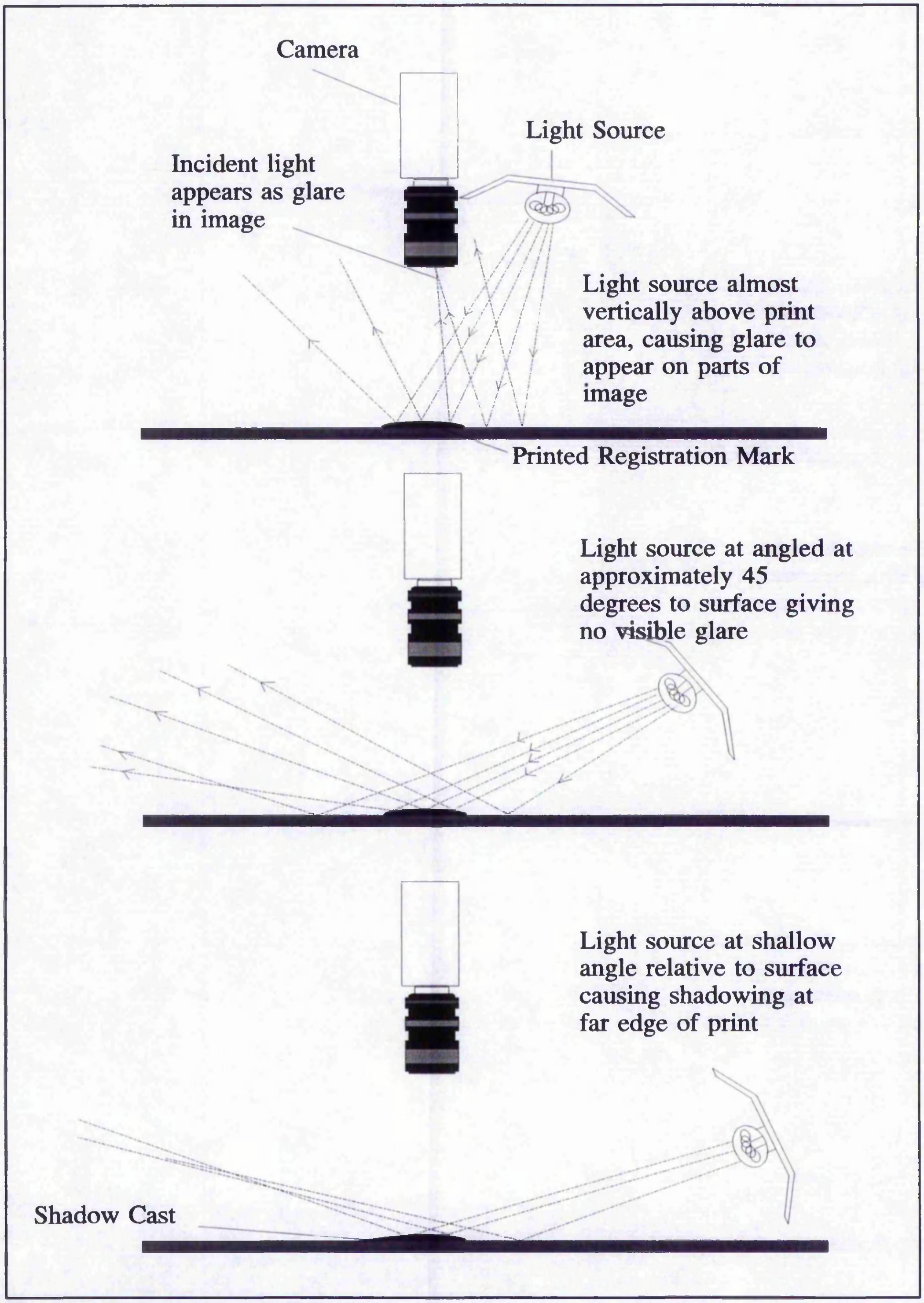

Camera
Light Source
Incident light appears as glare in image
Light source almost vertically above print area, causing glare to appear on parts of image
Printed Registration Mark
Light source at angled at approximately 45 degrees to surface giving no visible glare
Light source at shallow angle relative to surface causing shadowing at far edge of print
Shadow Cast

Figure 4.3 The affect of different light source angles

lighting decision process is described in section 4.4.2. The lighting of the screen registration mark was more complicated. The major problem is that the screen is covered in ink as part of the printing process, so the screen had to be viewed after print and before flood, which meant stopping the machine in its cycle. This was done, and it was possible to view the registration mark clearly enough for alignment, it must be pointed out again that this was only an early prototype development and no software image enhancement techniques were incorporated into the system and so all images viewed had only to be clear to the human eye. This system worked initially using all printing inks on a standard white polyester screen but problems were encountered when a new screen material was used with the system. For higher printing accuracy a new screen material, Himesh, was used. This is a one piece metallic mesh very similar to an electric shaver foil which due to its one piece nature is dimensionally very stable and therefore less susceptible to screen stretch. It was also darker than polyester which meant that there was very little contrast between the mesh and the stencil, the registration mark could be seen faintly without ink on the screen, but as soon as ink was applied very little could be seen using the standard lighting set-up. The initial solution was for the operator to wipe the screen before it was viewed, but this was a far from satisfactory solution bearing in mind that the idea of the project was to minimise operator input. Clearly another solution had to be found. Brighter lighting was tried but made no difference. The next step was to place a reflector under the screen to redirect light through the registration mark, this worked partially but the placement of the reflector and its angle to the lighting was too critical to control and required a complex arrangement to bring the reflector under the screen and remove it before printing. The partial success of the reflector had shown that light under the screen would enhance the registration mark contrast. True back lighting of the screen was not possible as there was only a gap of a few millimetres between the screen and the table but it was possible to shine a focused light beam from a microscope light source under the screen at an acute angle, with the light source at the side of the machine. The effect was different from that expected, instead of the light illuminating the registration mark it cast a shadow which had the effect of showing the mark as a dark area which could be clearly seen without any operator input.

The original Cosmicar C-mount lenses were replaced by Tamron 70-210 mm photographic lenses, fitted to the Pulnix cameras by C to K adaptors. These lenses, along with close up extension tubes and a +1 close-up lens gave the required resolution (0.01 mm per pixel) at a distance of 260 mm from the table surface (260 mm is the closest the cameras can be to the table and still avoid the moving squeegee).

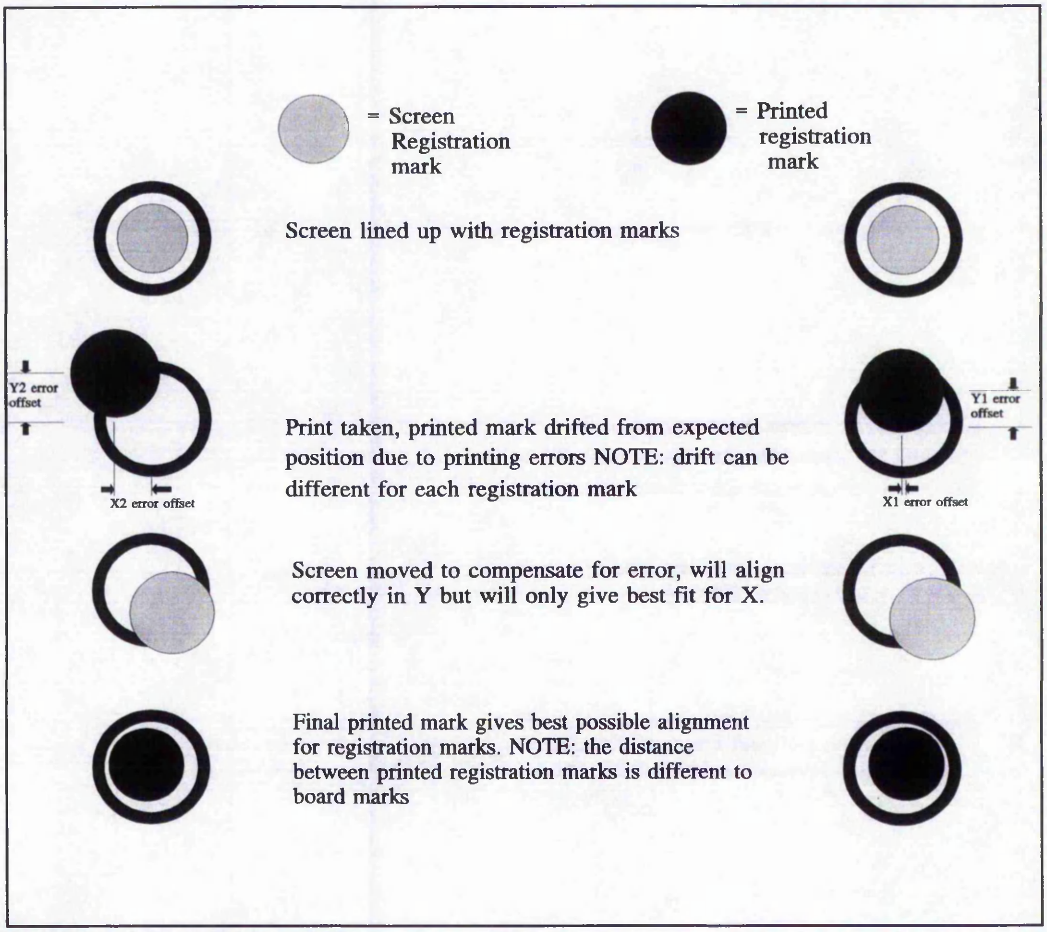

= Screen Registration mark
= Printed registration mark
Screen lined up with registration marks
Y2 error offset
X2 error offset
Print taken, printed mark drifted from expected position due to printing errors NOTE: drift can be different for each registration mark
Y1 error offset
X1 error offset
Screen moved to compensate for error, will align correctly in Y but will only give best fit for X.
Final printed mark gives best possible alignment for registration marks. NOTE: the distance between printed registration marks is different to board marks

Figure 4.4 Registration drift.

An additional feature built into the system was the ability to compensate for screen printing errors. These errors occur when the squeegee passes over the screen causing it to stretch with the resultant effect that the printed image is displaced in the direction of the squeegee movement by varying amounts dependent on the screen/ink combination and the snap-off height. This is shown in **Figure 4.4**. In tests it was shown that these errors were consistent for a certain set of printing conditions. Attempts were made **[38][39]** to calculate these errors, but this proved inaccurate and so the errors were measured by lining the screen up to the registration mark, taking a sample print, and moving a graphical fiducial in the vision system to line up with the printed registration mark. Once lined up the system recorded the position of the registration mark and calculated the error offset displacement. This process was repeated for both cameras and registration mark pairs. In the original four camera system the screen was aligned to the board by overlaying a transparent image of the board registration mark onto a live picture of the registration mark. This process proved inaccurate as it was difficult to distinguish between the two images. For the two camera system the transparent image of the board was replaced by a graphical representation of the registration mark, this is shown in **Photograph 4.4**. This proved easier to align with the screen and also more precise as there was clear edge definition. When aligning the screen to a board the compensation error was added to the monitor fiducials according to the formulas:-

$$Offset\ X = \frac{x1\ disp + x2\ disp}{2} \qquad \textbf{(4.1)}$$

$$Offset\ Y = \frac{disp\ y1 + disp\ y2}{2} \qquad \textbf{(4.2)}$$

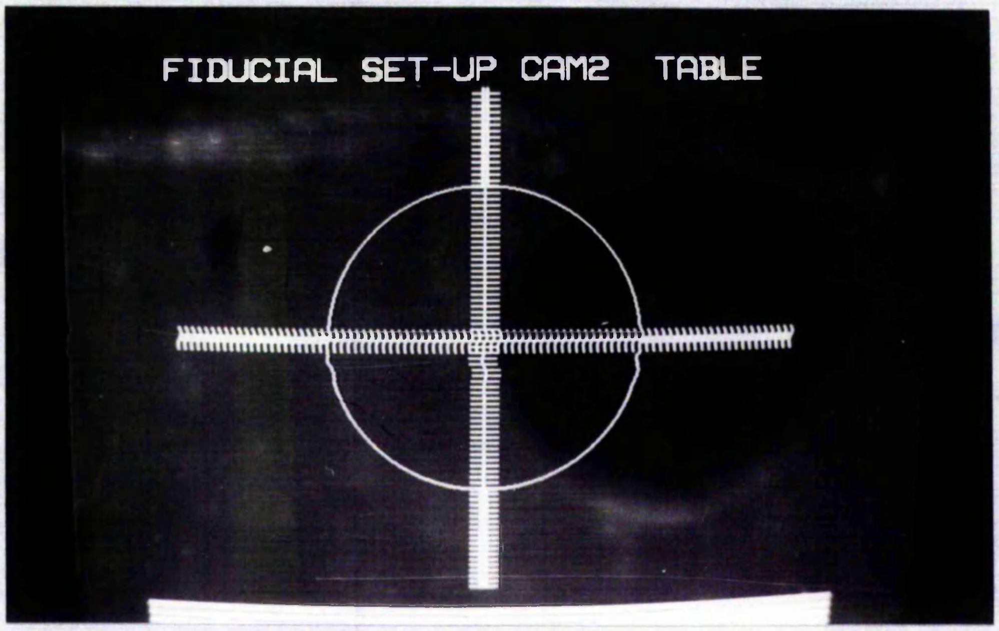
$$x1 = sx1 - (2 * OffsetX) \qquad \textbf{(4.3)}$$

$$y1 = sy1 - (2 * OffsetY) \qquad \textbf{(4.4)}$$

$$x2 = sx2 - (2 * Offset\ X) \qquad \textbf{(4.5)}$$

$$y2 = sy2 - (\ 2 * Offset\ Y) \qquad \textbf{(4.6)}$$

These formulas compensated for the fact that the printed registration marks were further apart than the screen registration marks (due to screen stretch), and moved the monitor fiducials to allow for this and the printing error. This made the lining up procedure more accurate.



Photograph 4.4 Use of graphical overlay to aid manual alignment.

This system was used successfully by the original sponsoring company in a full production environment, but still relied on manual movement of the screen.

4.4 Development of the Automated Alignment System

The two camera system was proven in production but still required a skilled operator to align the screen manually, the vision system only acted as an aid to the operator although it did increase the accuracy by making compensations for errors that could not be seen by the naked eye. It took approximately one and a half minutes to align the screen to the board, and this was in the hands of a skilled operator. To do a full production run aligning the screen to every board was very tiring and only resulted in a production rate of 40 boards an hour, this negated the advantage of having the increased accuracy. The next step in the project was to remove the manual input.

The manual inputs into the operation were:-

1) Aligning of vision system fiducials with board and screen in set up,

2) Aligning vision system fiducial with displaced printed image to measure printing error,

3) Movement of screen to align with either board or vision system.

4) Control of the system.

The automation of these four processes can be divided into two main areas:

- Control of the screen movement and printing parameters, this work was undertaken by Dominic Fulford **[38][39].**

- Recognition and location of the board and screen registration marks, this was undertaken by the author.

In Chapter 1 a brief introduction was given to machine vision and a flow chart was given showing the basic elements that make up a typical machine vision process. This flow chart is redrawn in **Figure 4.5** and shows the operation of the centre finding procedure. Along side the flow charts some of the different techniques to carry out each of these procedures are listed, each of these was described in Chapters 2 and 3. The choice of which technique was chosen to carry out each procedure is discussed in the following sections.

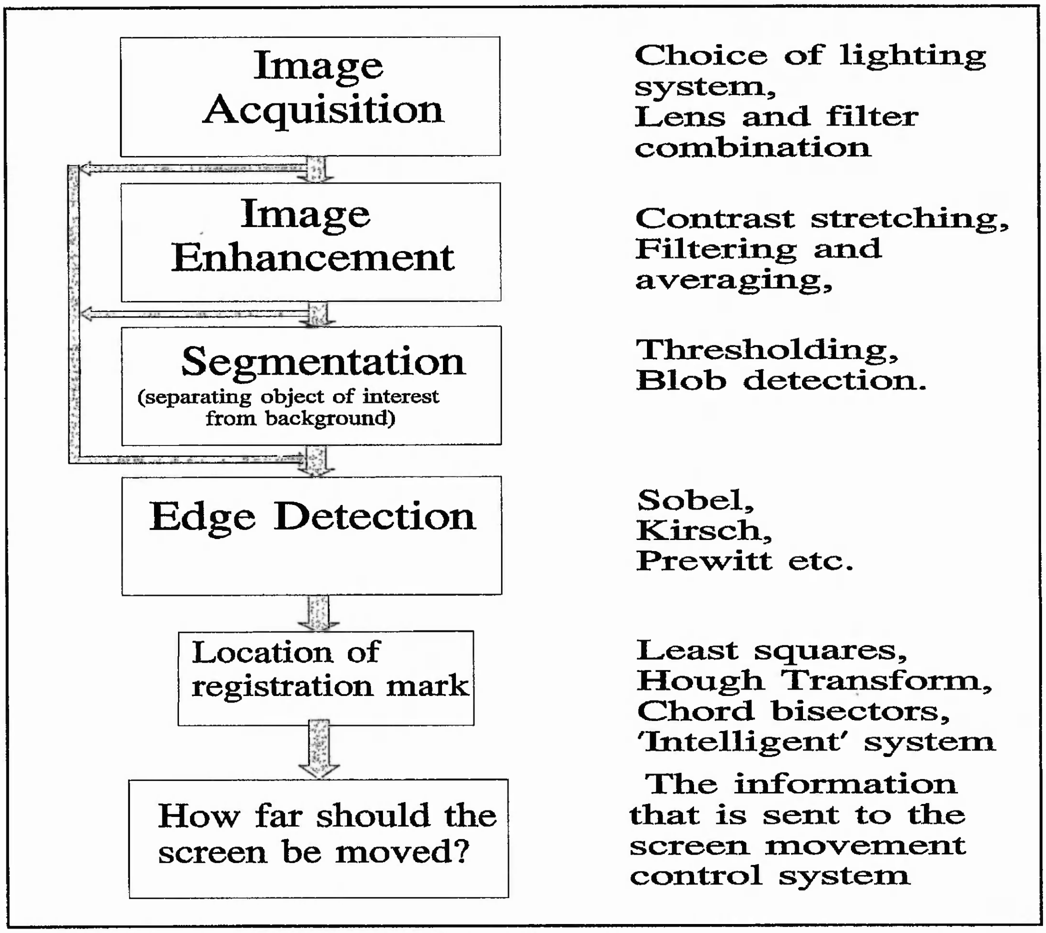


Image Acquisition
Choice of lighting system, Lens and filter combination
Image Enhancement
Contrast stretching, Filtering and averaging,
Segmentation
(separating object of interest from background)
Thresholding, Blob detection.
Edge Detection
Sobel, Kirsch, Prewitt etc.
Location of registration mark
Least squares, Hough Transform, Chord bisectors, 'Intelligent' system
How far should the screen be moved?
The information that is sent to the screen movement control system

Figure 4.5 Steps in finding registration mark position.

4.4.1 Camera Selection

When choosing a camera its performance characteristics and the environment within which will operate must be considered. Vandommelen **[40]** has put forward characteristics he believes should be considered when making the choice and he has also carried out an evaluation of various make and types of camera. His eight selection criteria were:

- Resolution: based on the number of photosites on the sensor and the type of image transfer function on the detector chip.
- Sensitivity: what changes in light level can be detected and over what range.

- Speed/Integration Time: the number of frames per second can the camera cope with.
- Signal to Noise Ratio: how much power in a signal to the amount of noise present in the absence of the signal measured in decibels (dB).
- Spectral Response: in what region of the electromagnetic spectrum is the camera most sensitive.
- Modulation Transfer Function: the ability to distinguish between adjacent high contrast features.
- Environment: vibration, chemicals, moisture etc.
- Cost.

The cameras selected for the project were Pulnix TM-565, the data sheets for which can be found in **Appendix 2**. These are relatively inexpensive at £400 compared with what is available on the market **[41]**, and they are very robust. The spectral response for the camera is shown in **Figure 4.6**, this is a fairly standard response for a CCD device being most sensitive to light in the blue range (0.55 μm) and infra-red (0.9 μm). The camera has an infra-red cut filter which is necessary when using the camera with incandescent lighting.

4.4.2 Choice of Lighting System

Lighting is a vital part of the image enhancing process and as such can not be overlooked. The advantage of having the most powerful image processing software routines and hardware is negated if the image is substandard due to poor lighting. Lighting is not used just to illuminate an image, it is in itself a powerful image enhancement tool. Using different light sources and types can show up aspects of an image that could not otherwise be seen such as subtle changes in grey-level. Using the correct lighting can increase the contrast of an image and minimise the effect in changes due to ambient lighting. Poor use of lighting can lead to glare, shadows and inaccurate data, this is clearly shown in the original development of the automated registration

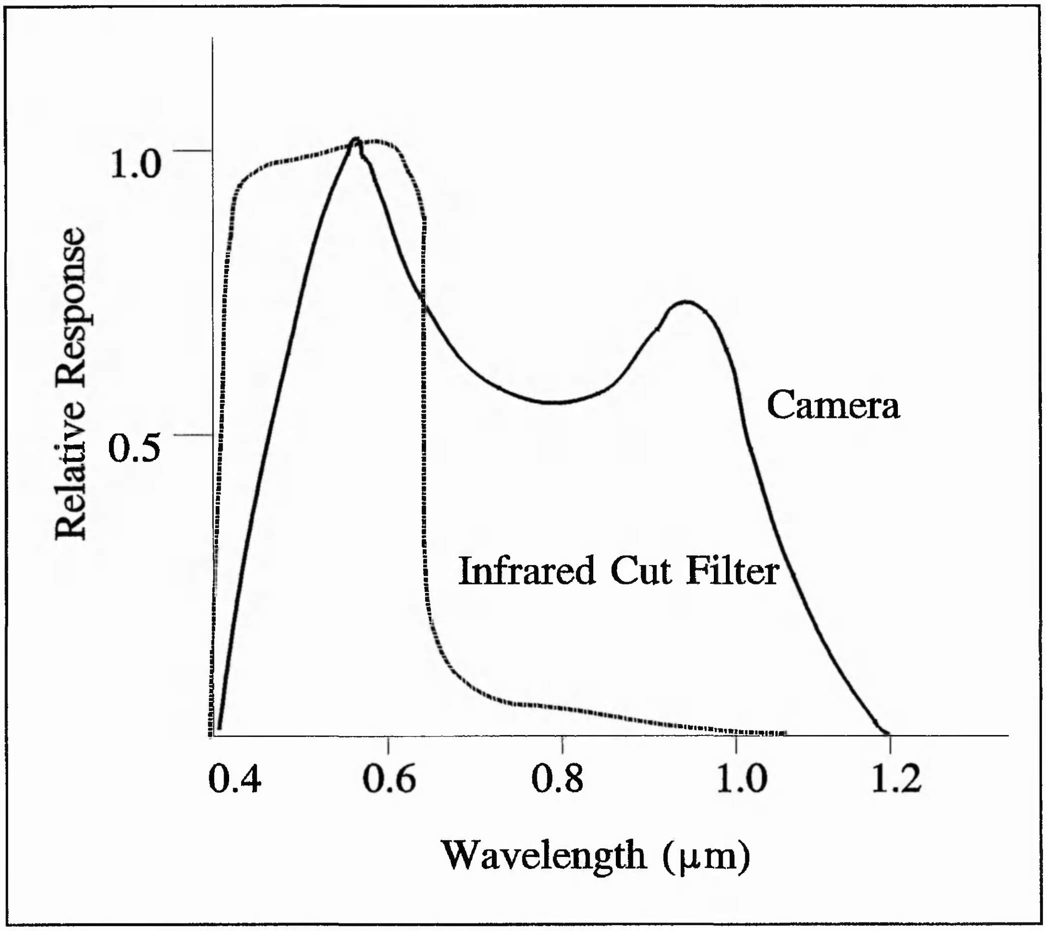


Figure 4.6 Pulnix (CCD) Spectral Response

system. The system had been shown to work to an accuracy of +/- 0.2 pixels (approximately +/- 0.004 mm) but with some readings jumping by about +/- 2 to 3 pixels. The system was tested by moving the screen back and forth between four set points, and the centres of a registration mark on the screen measured. This was left running for a couple of hours and the centre positions were recorded. These values showed that there was a steady drift of four or five pixels (approximately +/- 0.05 mm) over the two hour period, but with occasional unexplained jumps of two or three pixels in some readings. The test was repeated, but this time the position of the screen was monitored using a dial test indicator mounted to the machine to measure the relative movement of the screen, and the position was also checked by live video with the original positions marked on the monitor. The retest showed that the screen movement was very consistent at better than +/- 0.01 mm, but there was still the drift of four or five pixels over the period of the test, with the two or three pixel jumps in individual readings. This meant that the error must have occurred in some aspect of the vision system. The lighting at this stage in the development was supplied by an angle poise lamp with a standard tungsten filament bulb mounted so that the it pointed directly at the screen from above. The system used a Hough Transform to find the centre (described in Chapter 3) which had been proven to be accurate to +/- 0.2 pixels in both static tests and in tests moving a registration mark back and forth using a microscope table, but both of these tests were carried out over a period of just a few minutes. It seemed a mystery as to where this drift was coming from until the lamp was accidentally knocked out of position, this resulted in the centres being found to be more than five pixels out from before hand, even though the registration mark had not been moved and the lighting was still pointing at the mark, all be it at a different angle. Moving the light again showed similar changes in position finding. This was conclusive proof that the lighting was the source of the error. The reason for the individual jumps in position was found to be due to the laboratory door being opened and closed, and the drift was put down to changes in ambient lighting through the day. The lighting system was replaced by two fluorescent tubes which had the effect of drowning out ambient light changes. The tests were repeated and all drift in position finding was removed. This highlighted the need to seriously investigate the system lighting requirements.

The process has a number of lighting requirements. Ink is highly reflective when wet which has a tendency to cause glare, so the lighting must be set-up in a way as to minimise this effect. The process appears to be very much two dimensional which would imply that shadows would not be a problem, but this is not the case, a print has a thickness of approximately 40μm which is enough to cause shadowing. Again the lighting must be set up to minimise this effect. The lighting must be able to show up registration marks on the screen when covered in ink and to give maximum contrast. In order to find the optimum lighting set up a wide range of light sources, set ups and filters were used.

Lighting and Filter Trials

To evaluate the best lighting set-up possible from the limited lighting and filters available to the project a series of lighting trials were designed to show up the best all round lighting arrangement for the system. The following factors were investigated:

- different types of light source,
- different arrangements of light position,
- different types of lens filter.

Images were taken using different combinations of the above factors. These images, along with their histograms, were assessed using the following criteria:

- Contrast, range of grey-level values associated with the peak representing the registration mark on the histogram. (Evaluated on a scale of 1 (no mark visible) to 5 (high contrast))
- Number of histogram minima and maxima, there should only be two maxima representing the background and the registration mark.
- Amount of glare (subjective), when testing lighting angle (Evaluated on a scale of 1 (minimal/no glare) to 5 (high glare making complete mark indistinguishable from background)).
- Amount of shadow (subjective) cast (Evaluated on a scale of 1 (no shadowing) to 5 (high level of shadowing)).

The same object of interest was used for all tests, namely a freshly printed registration mark using dielectric ink on a light grey FR4 substrate. Snapshots were taken and then analyzed to give histograms for each set up.

Lighting Systems Used

The different light sources were set up as listed below, all ambient light was removed to prevent any interference. For each set-up various f-stop values were used across the range of the lens with the best results taken as being representative of that set-up. For each lighting arrangement five sets of readings were recorded, four with lens filters (red, green, orange and yellow) and one set with out any filters.The same lens was used throughout the tests.

Using halogen, fluorescent and tungsten lamps:

1) Two lights at 10° to horizontal,

2) Two lights at 45° to horizontal,

3) Two lights at 90° to horizontal,

Using fluorescent lamps:

3) One light at 10° to horizontal,

4) One light at 45° to horizontal,

5) One light at 90° to horizontal,

Using tungsten lamps:

6) Four lights at 10° to horizontal,

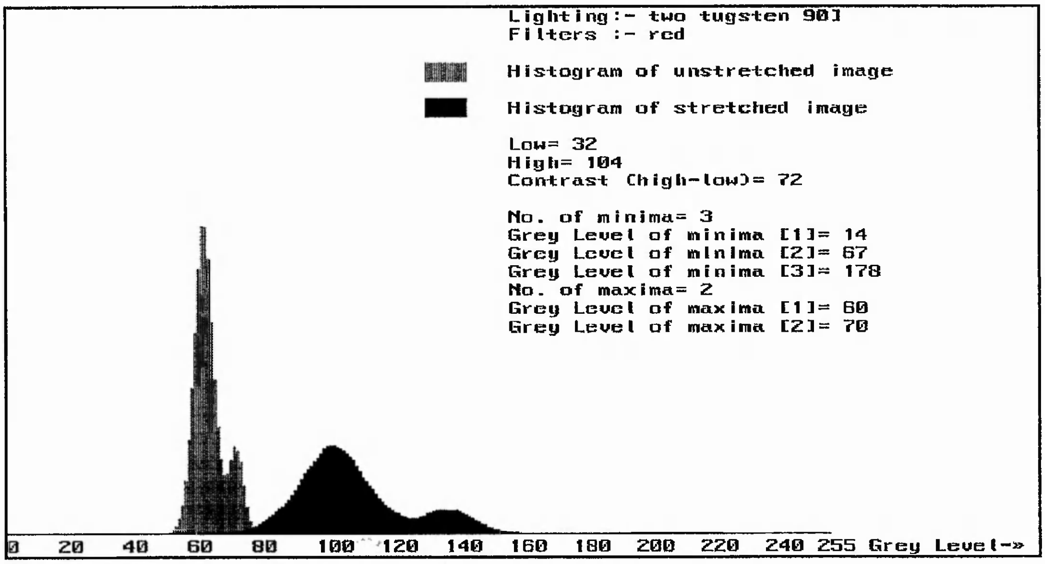
7) Four lights at 45° to horizontal,

8) Four lights at 90° to horizontal,

9) Ring light.

Results:

The above systems were evaluated to give maximum contrast. The results are shown in **Appendix 5**. These show the histograms that resulted from the optimum contrast levels for each of lighting systems tested and the assessed levels of the four factors are given with each histogram. An example of the output from the trials is shown in **Figure 4.7**. Two factors were rated subjectively based on the appearance of the image. These were glare and contrast and were marked on a scale between 0 and 5. In addition to these values a comment was given relating to the appearance.



```
Lighting:- two tugsten 90]
Filters :- red

▨    Histogram of unstretched image

■    Histogram of stretched image

Low= 32
High= 104
Contrast (high-low)= 72

No. of minima= 3
Grey Level of minima [1]= 14
Grey Level of minima [2]= 67
Grey Level of minima [3]= 178
No. of maxima= 2
Grey Level of maxima [1]= 60
Grey Level of maxima [2]= 70
```

Figure 4.7 Example of output from lighting and filter trials.

Discussion:

Examination of the results allows the following statements to be made about each of the different aspects of the lighting and filter arrangements:

Filters:

- Through out the tests green lens filters gave the highest contrast level. This is because the substrate was slightly grey (wave length close to green) which meant the green filter makes the substrate appear lighter (higher grey level) with respect to the red registration mark.

- The red filters had a similar effect on the red registration mark, they became lighter than the substrate giving a 'negative' effect. The contrasts between the mark and the background were generally low, but in most cases the resultant histograms were still bi-modal. This effect could prove useful when printing inks that are the same grey-level but different colour as the substrate, as segmentation through thresholding will be possible.

- The orange filters gave a similar effect as the red filters, but not as distinct. This is because orange's wavelength is close to red. This can be seen by examining the resultant histograms from orange and red filters, there is little difference between the two. The exception was in the use of fluorescent lighting where red filters gave good results and orange gave very poor results.

- The yellow filters resulted in a slight drop in contrast from that gained using no filters. This is due to the filter slightly restricting light input to the lens. Yellow's wavelength has no similarity to any of the elements in the system.

Camera Angles:

- 90° angle lighting gave maximum glare with all light types and generally lower contrast.
- 45° angle lighting gave minimum glare for all light types and generally the best contrast.
- 10° angle lighting gave no shadowing as might have been expected, glare was higher than 45° for some lighting because the uneven surface of the print reflecting light up at the camera.

Light types:

- The ring light (fluorescent) gave the highest contrast with no glare.
- The fluorescent strip lighting gave second best results with very little difference between using one light and two.
- Tungsten lighting gave off a lot of heat and tended to be too bright, they gave more glare due to their undiffused nature (they were too hot to use diffusers).

- The halogen lighting proved too bright even with low power and they resulted in too great a heat output for use with inks. It was not possible to get any contrast between the mark and background using any combination of filters and angles.

Conclusions:

The best lighting/filter arrangement for the particular arrangement of substrate and registration mark proved to be a ring light with a green filter, with the second best all round results from fluorescent tubes mounted at 45° with a green lens filter, with two tubes giving marginally better results than one. The budget restrictions of the project meant it was not possible to purchase another ring light, therefore one fluorescent tube mounted at 45° was selected.

With the limited equipment tested the following general statements can be made to aid in light and filter selection:

- Select lens filter colour to closely match background colour to give maximum contrast, careful choice of colour may also allow objects of same grey-level values but different colours to be differentiated.

- 45° Angled lighting can reduce glare and ring lighting may result in no glare for some applications.

- Fluorescent lighting results in less heat output.

- The combination of lighting/filters must be assessed for every application tested, e.g. the best selection for the combination of a red mark on a green substrate may be the worst for another colour combination.

4.4.3 Lens Selection

Correct lens selection is vital to the performance of the complete vision system. Section 2.2.2 described the basic optical formulas but these only give a rough guide, there is still a degree of trial and error in correct selection. This problem was addressed by White and Leblanc **[42]** who developed an expert system to aid in the design of optics for specific use in machine vision gauging

The original specification of the project was the to align to +/- 0.1 mm, which meant that the vision system would have to have a resolution of at least 0.05 mm per pixel.

Therefore field of view

= 512 (number of pixels across buffer) x 0.05 mm

= 25 mm

The Sveciamatic screen printing machine used in the project has a squeegee that passes over the screen. This means that the camera and lenses had to be mounted clear of it's path which necessitates them having to be mounted at least 260 mm above the screen. This height can impose problems with lens selection. Cosmicar, the industrial lens section of Pentax, are the market leaders in CCTV lenses, their most powerful non-motorised C-mount CCTV lens was a 12.5 mm - 75 mm zoom lens. This, in addition to a +3 close-up lens gave the required magnification at 260 mm, but this was the limit to its magnification. The automated system needed a resolution in excess of the original specification at +/- 0.01 mm, this could not be achieved, at the specified height, by using the original Cosmicar lenses, even with a combination of close-up lenses and extension tubes. It was therefore necessary to use a standard 35 mm photographic 200 mm macro lens attached to the camera by a C-mount to Pentax K-mount adaptor, this with a set of extension tubes and a +3 close up lens, gave a resolution of 0.01 mm per pixel, a field of view of approximately 5 mm. The use of photographic lenses has a number of advantages over standard CCTV lenses:

- A wider range and a better quality of lens available.
- A wider range of filters available.
- They are specifically made for 35 mm photographic film, which is much larger than a standard CCD array, this means that the image produced is much better with a great deal less distortion near the edge of the field of view.

The area of the array in the Pulnix camera is 8.8 mm x 6.6 mm, the width of the object is 2 mm, and the distance of the object from the lens is 260 mm.

$$M = \frac{8.8}{5} = 1.76 \tag{4.7}$$

$$\therefore f = \frac{260\ x\ 1.76}{2.76} = 165.80 \tag{4.8}$$

Using this value of 165.8 mm the nearest lens that was available was a 210 mm F4/5.6 Tamron lens. The specification of the lens is shown in **Appendix 2**. It was not possible to focus the lens at the distance of 260 mm it was designed for long distance use. This meant the use of extension tubes to bring the object into focus. The formula for choice of extension tubes is:

$$Lens\ extension = \frac{focal\ length}{object\ magnification} \tag{4.9}$$

$$\therefore\ lens\ entension = \frac{210}{1.76} = 119.32\ mm \tag{4.10}$$

Use of such a long extension tube results in large loss in light to the camera array. To overcome this a 65 mm long extension tube was used in conjunction with a +1 close up lens and this gave the correct field of view at 260 mm.

4.4.4 Image Enhancement Process

The image enhancement process took place in two steps. The first stage was the acquisition of the image using the faverage(); command supplied with the MVP-AT software. This function takes a number of snapshots of the subject (in this case the registration mark) determined by the operator. This image is further averaged using the average(); function that uses the local averaging mask shown in Chapter 2 **Figure 2.6**. These two averaging processes minimise image noise. The next stage in the process was contrast stretching to maximise image contrast, this process is described in detail in section 2.3.4.

4.4.5 Segmentation Process

Two segmentation process, thresholding and blob detection, were outlined in chapter 3. The blob detection process developed by the author was used to separate areas of the same grey-level in a binary image, therefore it could only be used on thresholded images. In this aspect blob detection was more of a secondary segmentation process. The blob detection used the computing technique of "linked lists" to differentiate between individual objects within the image.

A linked list is a sequence of data elements (structures), in which each element points to its successor, in 'C' the element may take the form:

```
typedef struct {
        int start; /* the start of the object */
        int end;   /* the end of the object */
        int label; /* the object descriptor */
        int next;  /* the address of it's successor */
}LINK;
```

It is the pointer to the next element that keeps the list together. A linked list is known as a sequential access data structure, this means that to access an element in the list it is necessary to start at the beginning of the list and sequentially search through from element to element following the pointers. The major advantage of such a data structure, as opposed to an array, is that you do not need to know how much data is to be stored before you start, as memory can be allocated dynamically (C function malloc();).

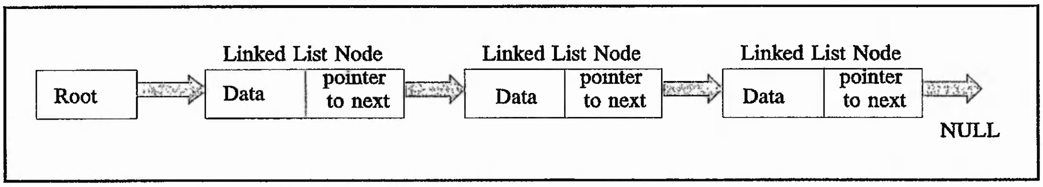


Figure 4.8 Linked list data structure.

The process starts by creating a singularly linked circular buffer, and three integer variables are assigned:

startlist - the index for the start of the list,

now - the index for the last blob pointing to the position for the next,

firstcurr - index for the first blob in the current scan.

The process starts at the top left hand corner of the image, this is the start of the scan. The scan moves to the right incrementing through the image pixels, when a black edge pixel is found a blob is added to the list, the data stored for the blob is the first pixel and last pixel and a label for that particular blob, along with a pointer to the next blob in the scan. The scan continues to the end of the line. At the end of the line all the blobs found within the scan are checked to see if they are connected to blobs found in the previous scan. All the blobs in the previous scan are deleted by setting the index to the start of the list to equal the index of the first blob in the current scan, this is because blobs that exist are represented in the current scan, and those that have shown no connection have ceased to exist. A new label is added to every blob, and the blob is also labelled in the frame buffer by giving it a grey-level equal to the label value. At the end of each scan the labels are changed to allow for new blobs being inserted into the list by use of a hash table.

Once the image has been separated into the individual parts it is then necessary to identify which is the desired object. It is possible to incorporate a pixel counter in to the link lists process, it will then be possible to separate the largest object in the image from the others. It will normally be known what objects should be in the image and what size the registration mark is relative to the other objects. It is likely that the registration mark will be the largest object in the image, therefore it will be easy to separate this from the rest of the image. The object might be able to be identified by its relative pixel density, a ring will have a greater range of x and y pixel values than a solid circle of the same area, and hence it's pixel density will be lower.

The system was able to select which part of the image was to be located by blob detection using two methods:

1) Manual selection using a curser on the vision system monitor,

2) By defining what size the object of interest was compared with any other object that could appear within the field of view of the system. This was a simple case if it could be guaranteed that the object was to be the largest present, but was not feasible when the size of other objects varied.

In practice the blob detection process was too slow for use in production as it took at least four seconds an image depending on the complexity of the image.

It is quite feasible to specify at the circuit design stage that the registration mark is the only object within the field of view of the vision system. If this is the case then thresholding would be a sufficient segmentation processes in itself. The bonus of having just the registration mark within the field of view is that any histogram produced will be distinctly bi-modal making threshold level selection fairly straight forward. Use of the MVP-AT software for threshold selection has already been discussed, it allows the selection of the point between the two peaks of the bi-modal histogram. There are two different situations for thresholding of registration marks in the screen printing process. These are:

1) With board registration marks the marks will generally be darker than the substrate.

2) With screen registration marks the marks will generally be lighter than the screen stencil.

With case (1) all points below the threshold level will be the board registration mark, but with case (2) all points above the level will be the screen registration mark. The Automated system automatically selected the threshold point based on these ideas, but it gave the option for the operator to fine tune this selection to give improved segmentation.

4.4.6 Evaluation of Edge Detecting Techniques

Edge detection was discussed in chapter 3 and the four main categories outlined. The number of different edge detection algorithms (EDA's) is vast, each type was developed for a specific application and as such have they may not always be suitable for all applications. Much work has been done to access the advantages and disadvantages of the most common EDA's for general use **[43][44][45]**. Al-Kindi et al. **[44]** created a ranking for fifteen different EDA's including their own Novel edge detector. The EDA's were accessed on their ability to cope with the following situations:

- noise,
- edges in dark areas,
- diagonal edges, in addition to vertical and horizontal edges,
- shadows,
- corners,
- edges in close proximity,
- edges created by thin and continuous strips.

The EDA's were also ranked in order of their processing time to carry out an edge detection on the same object. Their top five EDA's using the above criteria were:

1) Prewitt D,
2) Robinson,
3) New,
4) Roberts 2,
5) Sobel.

And based on processing speed:

1) Roberts 2,
2) Roberts 1,
3) Fast Novel,
4) Mero and Vassey,
5) Prewitt D.

Their new detector has similarities to the Novel edge tracker put forward in chapter 3 using a similar 3 x 3 mask, but was developed for use on 256 grey level images, where as the Novel edge tracker was developed as a tool for chain coding binary images but also being capable of edge detecting binary images.

Englander [45] carried out a review and evaluation of EDA's for industrial inspection applications putting forward a simple means of categorization them. His assessment was based more on their ability to cope with ambient and system changes using the following criteria:

- Detection sensitivity, ideally the EDA should pick out the same edge as the human eye, giving smooth unbroken edges and straight edges where they exist.
- Consistency of performance when objects in images are rotated (continuity and isotropy), this is particularly relevant in the screen registration process when the screen goes through a rotational movement.
- Accuracy in locating edges (localization), the edge detected need to be as near the true edge as possible, shadowing and blurred edges can cause some EDA's to fail.
- Reliability given image noise and variations in part reflectivity (robustness), most EDA's are highly susceptible to noise and incorporate smoothing routines which tend to slow down the processing speed and decrease the localization.
- Processing speed and hardware considerations, EDA's tend to be computer intensive as a result most have 'fast' versions which use simplified methods with a resultant drop in performance.

Englander differed in his findings to Al-Kindi et al. by recommending the Sobel Magnitude Edge Detection Method, although he agreed that Roberts was the fastest (up to twice as fast as the Sobel on equivalent hardware) but giving poor continuity, isotropy and localization, a high noise level and lack of robustness for use in most industrial applications.

The above evaluation work was carried out on full versions of the EDA's, simplified versions are more often supplied with standard vision system software. These are faster than the fully fledged versions but give poorer results. Edge detectors were used for two processes in the development of the automated set-up system. Firstly to detect edges for centre finding and secondly for chain coding and image segmentation. These two processes require different characteristics of the EDA's. For centre finding good localization and a distinct edge was essential with a minimum of noise. For chain coding and segmentation a smooth complete edge was required, an incomplete edge meant that chain coding could not be done, and rough edges creates excess data. With these factors in mind the following EDA's were tested using the functions supplied with the MVP-AT software: Horizontal, Horizontal and vertical, Sobel, Laplacian, Novel, Prewitt, and Kirsch. The same image of a registration mark was used, and the following was noted:-

- Horizontal - very fast giving a clear edge except on horizontal lines, limited to binary images. This function was developed by the author.

- Horizontal/Vertical - as horizontal but able to cope with vertical and horizontal edges, slightly slower.

- Sobel, the best for use on full grey-level images, but with some noise. Very good edge on binary image but slower than horizontal and horizontal/vertical. The operator used was an approximation to the Sobel edge detector. Only the horizontal and vertical edges are computed and from that information the other edges are approximated.

- Laplacian, Laplacian 1 and 2 tested both gave poor results with both binary and full grey-level images, developed a lot of noise.

- Novel, fast but only on binary images limited, mostly to image segmentation and chain coding. Needs smoothing operation on image to work effectively.

- Prewitt, similar results to Sobel, but not as good on full-grey level images. The function was an approximation to the Prewitt edge operator, only the horizontal and vertical edges are computed and from that information the other edges are approximated.

- Kirsch, very little difference in output to Prewitt. The function used the Kirsch compass gradient technique as described in Chapter 3.

The choice of edge detector was between the Horizontal/Vertical edge detector and the Sobel. Although the Sobel gave the best results for full grey-level images the Horizontal/Vertical was much faster in operation with thresholded images and it was therefore selected for use in the final system.

4.4.7 Choice of Centre Finding Process

Chapter 3 described a number of different centre finding processes to be used on circular registration marks. These were all tried within the production process to access their performance and to try and choose a centre finding procedure that could be used consistently. The following were tried:

- Hough Transform,
- Least Squares,
- Intersection of chords,
- Generalised Hough Transform and the 'Intelligent' Centre Finding System

The tested procedure was to move the registration mark back and forth through a fixed distance one hundred times finding the centres each time. The figures were recorded and the approximate accuracy of the process determined.

Least Squares (Section 3.4.1)

Least squares proved to be by far the most consistent centre finding process, able to find centres at an accuracy of +/- 0.2 pixels at a speed of less than a second a centre. Least squares is less accurate a centre finding method with obscured or incomplete registration marks than the Hough Transform but is able to cope with small amounts of noise and slight print imperfections, generally the increased speed was a greater advantage than the slight loss in accuracy for the occasional poor quality registration mark. The accuracy and speed of this process resulted in least squares being used in the final system to locate the circle centres.

Intersection of Chord Bisectors (section 3.4.2)

This process was as fast as least squares but was far too inconsistent to be used for finding accurate centres, it gave an accuracy of +/- 0.5 pixels with a perfect registration mark with no noise, but as soon as noise was present and the registration mark was slightly imperfect the accuracy went down to about +/- 5 pixels. This was out of tolerance for the process, but was useful as a rough estimate to define the Hough parameter space.

Hough Transform (section 3.4.3)

The Hough Transform was described in section 3.4.2, its implementation was computer intensive as many points had to be calculated. It uses the boundary pixels of the registration mark, for a typical mark this could be as many as 1000 pixels. If all these pixels were used for the Hough Transform and a three dimensional parameter space size 40 pixels (X) x 40 pixels (Y) x 20 pixels (radius) was defined, then the number of calculations to carry out a complete Transform at a resolution of $\pi/200$ radians was:-

$$((2 * \pi) / (\pi/200) * 40 * 40 * 20 = 12.8 \text{ million calculations}$$

Each of these calculations involves two floating point operations which means that the process tends to be slow. In addition to the Hough Transform the approximate centre has to be calculated in order to define the parameter space and minimise processing, and this space needs to be searched after the Transform in order to find the centre at subpixel level. The number of calculations was reduced by restricting the points calculated to an arc of $\pi/25$ radians within the approximate area of the parameter space, this reduced the number of calculations to 2 percent of the original (256,000 calculations). This process took approximately six seconds to find the centre of one registration mark at an accuracy of +/- 0.3 pixels. Although this accuracy was within the required system tolerance the speed was too slow for a production system. The transform resolution was tried at lower levels to determine the resolutions effects on the speed and accuracy of the process. At a decreased resolution of $\pi/100$ radians the system was two seconds faster (it did not halve the time as would have been expected), with an accuracy of +/- 0.5 pixels (only slightly worse than before), but it did show occasional unexplainable readings, out by up to +/- two pixels.

The Hough Transform was also shown to be more susceptible to changes in lighting levels than other centre finding process (see section 4.4.1) a variance of five pixels was observed in tests in which the lighting levels were deliberately altered. No reason could be discovered for this error as other centre finding processes, although susceptible to light changes, did not show such a high variances. The only possible factor was that search resolution was too low as the errors were less at high search resolutions.

The major advantage of this process was in the centre finding of obscured or incomplete registration marks and its ability to cope with image noise. It was capable of finding the centres of registration marks partly out of view of the camera without loss in accuracy, this was also the case for incomplete marks. This was tested by finding the centre of a complete registration mark and then obscuring part of the mark with a piece of paper. This had no effect on the position of the centre recalculated.

Generalised (GHT) and the 'Intelligent' Centre Finding System (section 3.6.2)

The use of the generalised Hough transform as an 'intelligent' centre finding system was described in section 3.6.2. The system, developed by the author, was able to work on any form of registration mark to an accuracy within the specification for the system, but its processing speed was far too slow for use in a production environment. Thazhuthaveetil and Shah **[37]** worked on the Parallel implementation of the Hough transform and reported a decrease in Hough processing time of 1/120th of the original sequential implementation. Such an improvement in processing time would make this system viable in a production environment.

4.4.8 System Registration Marks

The registration marks used by the system needed to cope with registering between many layers, not just one layer to another. This is not such a problem with manual printing as relatively large registration marks are used allowing the use of incomplete registration marks as described in section **3.4**, but because the vision system requires a field of view of 6 mm the registration mark must fall completely within this area with enough clear space around it to allow screen movement for alignment. This means that the chosen registration mark cannot exceed 3 mm x 3 mm ruling out the use of incomplete marks, because each part of the incomplete registration mark represents just one layer, if ten layers are to be printed the registration mark would have to be divided in to ten individual parts each of which would be too small to print accurately. The alternative to the incomplete registration mark system is align marks on top of each other, this is alright for two or three layers but any more and the marks start to blur. To overcome this a series of registration fiducials are printed on the first layer, and subsequent layers are printed to each of the pairs of registration marks. Such an arrangement has the disadvantage of having to realign the system cameras each time the registration mark pairs are changed. This problem was overcome to certain extent by printing sets of rings on the first layer and aligning circular dots within the rings on subsequent layers. By taking the outer edge of the ring to locate the centre of the registration mark and printing smaller dots, the dots should almost always lie clearly within the ring allowing many layers to be printed without distorting the outside edge of the ring. This system was used in the production system produced and resulted in no problems of distorted registration marks after four layers. If more layers are to be produced the system could be further improved by printing a new ring with a larger outer diameter after a number of layers, and subsequently registering to the outside edge of the new ring, it should then be feasible to print at least ten layers off one registration mark.

4.5 System Overview - The Automated Screen Print Set-up and Registration System

The first sections in this chapter described the initial development of a vision-aided screen printing system. This was a vital step in the process of producing an automated alignment system as it allowed the introduction and evaluation of ideas into a production environment. The manual inputs and operating weaknesses in the alignment system described in section 4.3.2 were removed in the development on the automated system which was described in section 4.4. The flow chart for the final automated system is shown in **Figure 4.9**, and is divided into five processes:

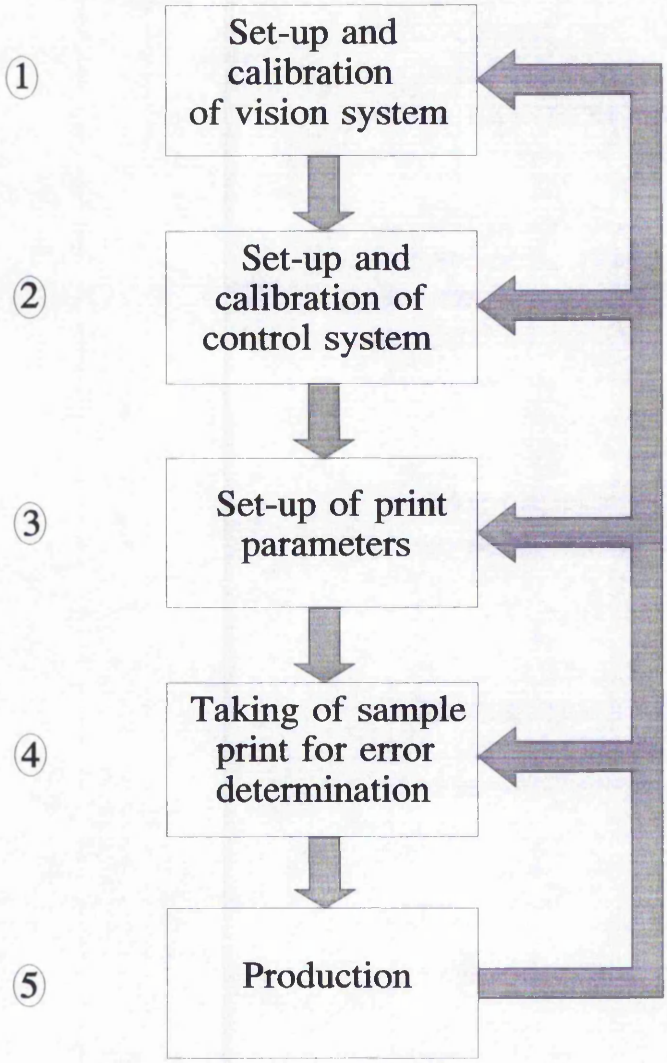
1) Vision set-up and calibration,
2) Control system set-up and calibration,
3) Printing parameter set-up,
4) Determination of print error offset,
5) Production.

The first four steps (set-up) need to be carried out before step 5 (production).

4.5.1 Vision set-up and calibration

The screen movement was automated **[38]** by replacing the three standard screen movement micrometers with Compumotor stepper motors. This gave screen movement in X, Y and Θ, as shown in **Figure 4.10**. The stepper motor movement was controlled by a PC which fed individual distances, in the form of motor steps, speeds and accelerations to the separate motors via the Compumotor control unit. Each revolution of the stepper motors was made up of 12,800 steps, this meant the following screen movement per revolution using a screen movement pitch of 1.5 mm.

12,800 steps = 1 revolution of motor = 1.5 mm screen movement
therefore 1 step = 1.5/12,800 mm = 0.12 x 10^{-3} mm.


1
Set-up and calibration of vision system
2
Set-up and calibration of control system
3
Set-up of print parameters
4
Taking of sample print for error determination
5
Production

Steps 1 to 4 must be carried out before production but may be repeated independantly after production has started.

Figure 4.9 The automated system operation flow chart.

The board to be aligned to was placed on the input table and held in place by

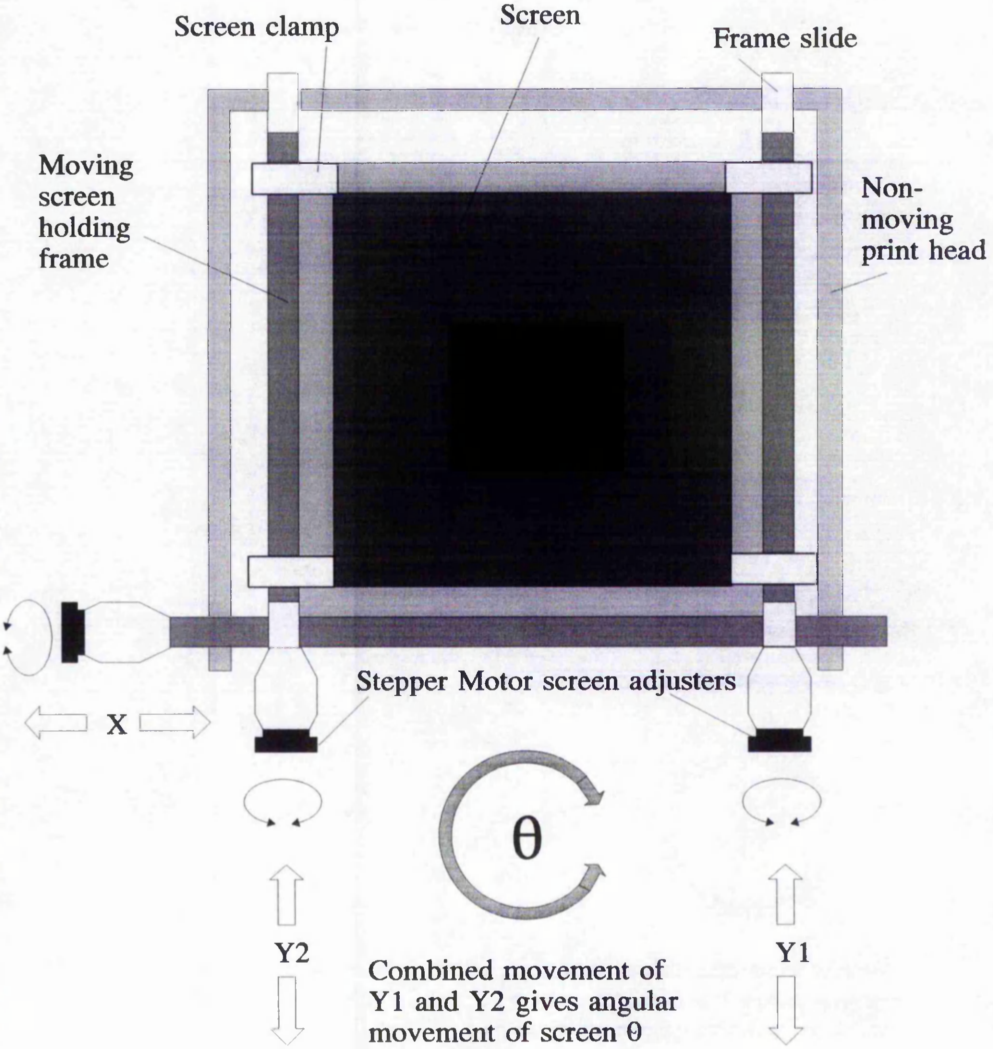

Screen clamp
Screen
Frame slide
Moving screen holding frame
Non-moving print head
X
Stepper Motor screen adjusters
θ
Y2
Y1
Combined movement of Y1 and Y2 gives angular movement of screen θ

Figure 4.10 X, Y and Θ movement of printing frame.

location pins. The cameras were adjusted so that the board fiducials were in the centre of view of the camera. The table was moved into print position. The screen to be used was placed in the printing frame and its position adjusted so that its fiducials were in the centre of view of the camera. At this stage the vision system was calibrated by finding the position of the registration marks, moving the screen a known number of steps (7,000) in X and Y directions, measuring the new centres, and converting the distance moved to a steps per pixel value for each camera. The calibration movement was the same in the X and Y axis, it would therefore be expected that the steps per pixel (following allowance for pixel aspect ratio), should be the same in X and Y for each camera. This was not always the case due to misalignment of the cameras which is shown in **Figure 4.11**. This misalignment (Θ) can be calculated during the camera calibration using the formulae:

$$\alpha = \frac{\Pi}{4}$$

where α is the angle of movement during calibration,

$$\beta = \arctan\left(\frac{Y^I}{X^I}\right)$$

where β is the apparent angle of movement seen by camera,

Y^I is screen movement in the y direction,

X^I is screen movement in the x direction.

$$\theta = \alpha - \beta$$

where θ is the camera misalignment.

$$\therefore \ \theta = \frac{\Pi}{4} - \arctan\left(\frac{X^I}{Y^I}\right)$$

Figure 4.11 shows the camera calibration and the incorporation of the compensation for camera angle. The system allows individual movement of the cameras, although they should always be in the same vertical plane there may be slight deviation, this can be measured during the calibration process and then allowed for in subsequent measurements.

The following 'C' code was used to calculate the camera misalignment and step per pixel values:

```
angle1=(PI/4.0)-atan((ct3.y-ct1.y)/(ct3.x-ct1.x));
angle2=(PI/4.0)-atan((ct4.y-ct2.y)/(ct4.x-ct2.x));
xpix1=7000.0/((ct3.x-ct1.x)*cos(angle1));
ypix1=7000.0/((ct3.y-ct1.y)*cos(angle1));
xpix2=7000.0/((ct4.x-ct2.x)*cos(angle2));
ypix2=7000.0/((ct4.y-ct2.y)*cos(angle2));
```

where angle1 and angle2 are the camera angle misalignment errors of cameras 1 and 2 respectively,

ct1.x and ct2.x are the original x-coordinate positions of the registration marks,

ct1.y and ct2.y are the original y-coordinate positions of the registration marks,

ct3.x and ct4.x are the x-coordinate positions of the registration marks after calibration movement of 7000 steps in X and Y.

ct3.y and ct4.y are the y-coordinate positions of the registration marks after calibration movement of 7000 steps in X and Y.

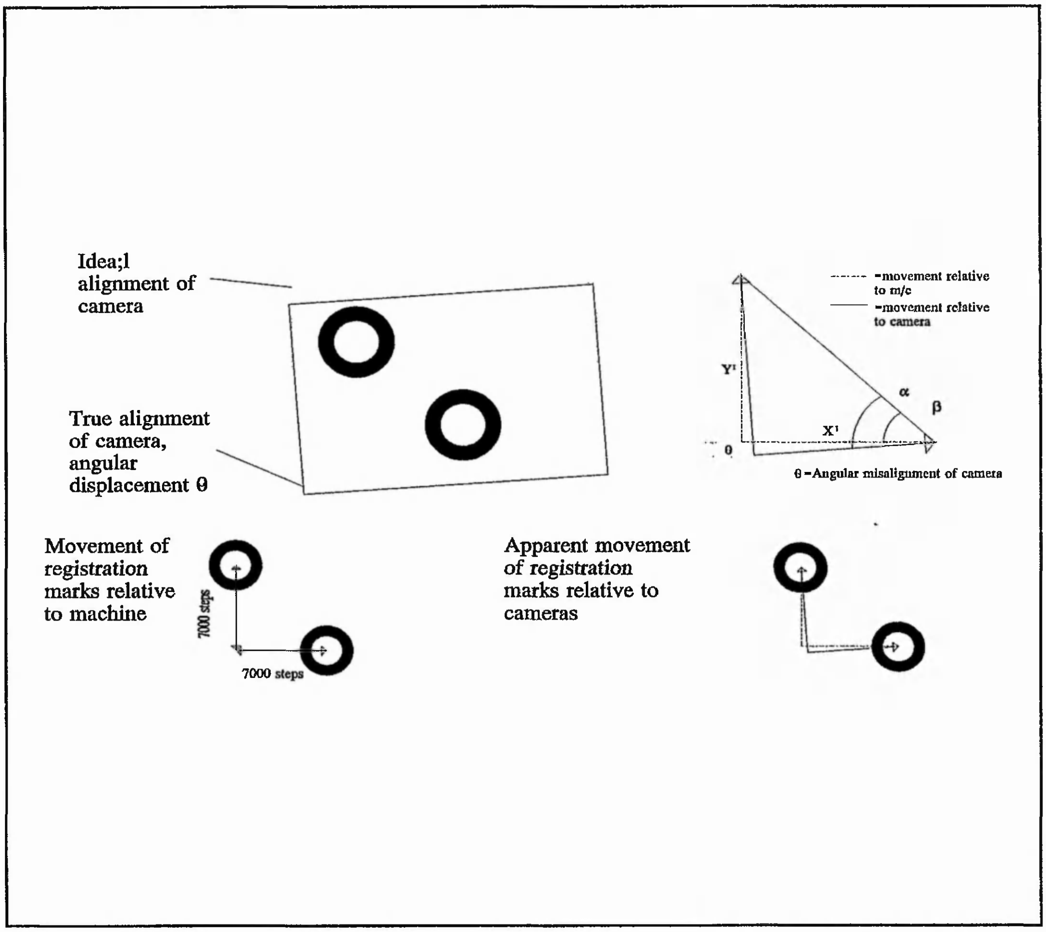

Idea;l alignment of camera
True alignment of camera, angular displacement θ
-movement relative to m/c
-movement relative to camera
Y'
α
β
X'
θ
θ -Angular misaligument of camera
Movement of registration marks relative to machine
7000 steps
7000 steps
Apparent movement of registration marks relative to cameras

Figure 4.8 Calibration of cameras

4.5.2 Control system set-up and calibration

Two control system components have to be set-up and calibrated before the system can be operated, these are:

- Position of screen relative to its pivot points,
- Start and finish positions of squeegee relative to screen,

The positions of the screen registration marks relative to the pivot points of the screen were calculated by rotating the screen a known amount from a set point and determining the resultant offset from the original position. The positions of the pivot points were assumed to be fixed within the frame of the machine, and that the screen moved relative to the points. The position of these points have to be known in order to determine the required distance for each motor to move the screen through a desired translation and rotation. By using a movement of the screen to determine these distances instead of manual measurement, the process is kept automatic and more accurate. This also allowed quick remeasurement in the event of a screen change.

The position of the start and finish of the squeegee strokes were determined by limit switches on the original Svecia control system, in the automated system these positions were defined by the position of the stencil on the screen and were entered into the control system in the set-up process.

4.5.3 Printing parameter set-up

The relevant printing parameters (as described in Chapter 1) were determined with the aid of an expert system **[38]**. These were set manually and as such were only a guide to the operator.

4.5.4 Determination of print error offset

When the board had been aligned a sample print was taken. The printed registration mark would be offset from the board registration due to the inherent screen printing process errors, and this offset was measured and stored for each camera. Work was done to predict these errors, but only part of the error could be accurately predicted. The total error comprises of displacement due to the snap-off height and the friction between the screen and the squeegee. The snap-off displacement could be quantified, but the friction error varied with the type, quantity and thickness of ink as well as screen and squeegee types. In addition ink dynamics varied in process, so it was not possible to quantify its effects on errors with any accuracy. Taking a sample print and measuring the offset proved the most reliable and consistent method of determining error.

The errors were stored for each camera, and the error added to the screen for each subsequent alignment. This was done using the following 'C' code, which includes compensation for camera angle:-

```
Y_OFFSET1 = ((ct1.y - ct3.y)+error.Y1)*ypix1;
Y_OFFSET2 = ((ct2.y - ct4.y)+error.Y2)*ypix2;
X_OFFSET1 = ((ct1.x - ct3.x)+error.X1)*xpix1;
X_OFFSET2 = ((ct2.x - ct4.x)+error.X2)*xpix2;
X1=X_OFFSET1+(tan(angle1)*Y_OFFSET1);
X2=X_OFFSET2+(tan(angle2)*Y_OFFSET2);
Y1=Y_OFFSET1+(tan(angle1)*X_OFFSET1);
Y2=Y_OFFSET2+(tan(angle2)*X_OFFSET2);
```

where Y_OFFSET1, Y_OFFSET2, X_OFFSET1 and X_OFFSET2 are the required registration offset including error compensation,

and Y1, Y2, X1 and X2 are the required screen movements including camera angle compensation in a form to be understood by the screen movement control system.

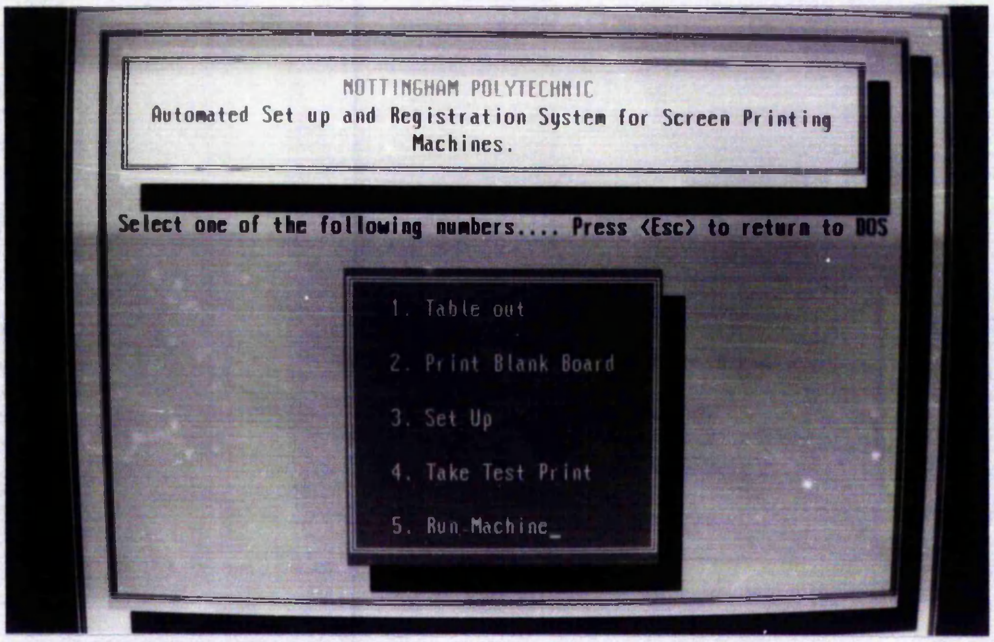
A complete description of how these equations were used by the control system can be found in reference **[38]**.

4.5.5 Production

The substrate was placed on the input table, the system automatically found the positions of the registration marks and the input table was brought in. The position of the screen registration marks were found, and the screen was then moved to align with the board including the error offset. The print was then carried out and the input table brought out. There is an option in the system to print without aligning to each board, this speeds the process up and is used when the accuracy of aligning to every board is not required.

4.6 Testing the System and the Production of a Sample Board

The system was menu driven from the menu shown in **Photograph 4.5**.



NOTTINGHAM POLYTECHNIC
Automated Set up and Registration System for Screen Printing Machines.

Select one of the following numbers.... Press <Esc> to return to DOS

Photograph 4.5 System 'front end' menu.

The accuracy of the automated system was tested using a ring and dot registration system as shown in **Figure 4.12**. The first layer contained two rings internal diameter 1.7 mm 500 mm apart. The second contained two dots with a diameter of 1.5 mm, also 500 mm apart. The first layer was printed and cured, and then the second layer was printed on the first layer using the Automated registration system. A production run of twenty boards was carried out and all twenty were within the specification outlined in the project proposal (+/- 0.02 mm).

The system was used to produce a four layer board for another research project within the department. The board had two conducting and two dielectric layers with surface mount devices attached using conducting adhesive. It is shown in **Photograph**

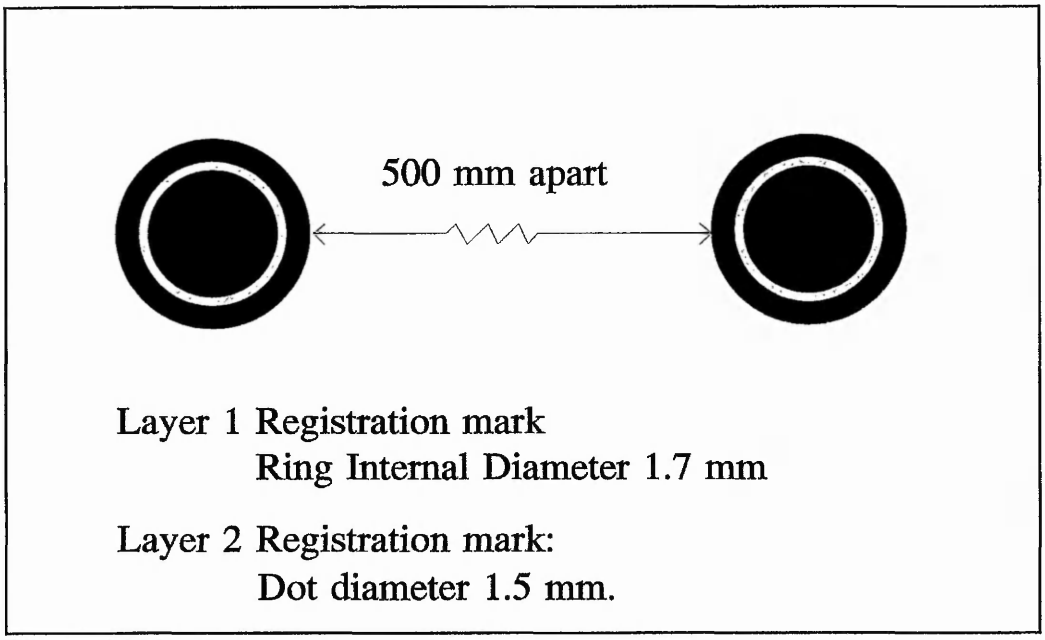
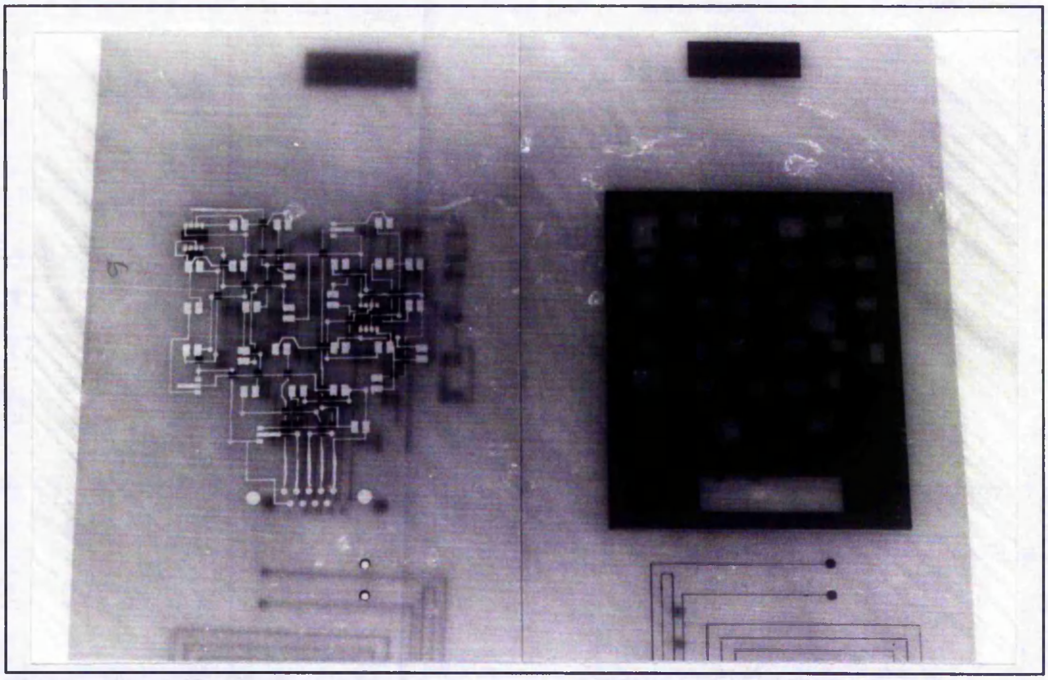

500 mm apart
Layer 1 Registration mark
Ring Internal Diameter 1.7 mm
Layer 2 Registration mark:
Dot diameter 1.5 mm.

Figure 4.12 Registration marks used in testing.

4.6 without the components attached and with the final dielectric layer shown separately. The production of this board showed how simple the production of a board could be. All aspects of the process were carried out within the department except for the production of the screen.



Photograph 4.6 Sample board.

Chapter 5

Conclusions

Chapter 5: Conclusions

5.1 Overall System

The original aims of the project as outlined in Chapter 1 were:

1) To examine ways of deskilling the process, to allow an untrained operator to successfully run the process without detailed knowledge of the process.

2) To recommend possible solutions using turn key equipment.

A screen printing system has been produced that allows fully automated registration to an accuracy greater than the +/- 0.001 inches (0.0254mm) recommended in the original project aims. **Photograph 2.1b** shows the registration of a 1.5 mm dot registration mark within a 1.7 mm internal diameter ring. This was achieved on all boards in testing and shows that the accuracy aims of the project were exceeded. The vision system developed has an accuracy of greater than +/- 0.2 pixels (+/- 0.005 mm with the optical system used) so there is the possibility of even greater printing accuracy in the future.

Determining how significantly the initial set-up of the process has been deskilled depends on the definition of deskilling. The process allows set-up of the system from a clean ink free screen to printing in less than five minutes compared to at least thirty minutes with the original manual screen printing system. Once in production very little operator time is required. In these respects the process has been significantly deskilled.

The system as it stands still requires a limited knowledge of the screen printing process to carry out the initial set-up, but this is aided by the automatic setting of most of the printing parameters and includes the use of an expert system to determine most of the printing parameters **[39]**. It cannot be claimed that the process has been fully deskilled but with minimal training most operators could master it's use as the system is nearly all menu driven. Screen printing has been, and still is, considered an 'art' that takes years to master. The system developed takes away a significant amount of this 'art' and replaces it with technology.

The system has been tested in a limited production environment to produce the four layer printed circuit board described in Chapter 4, but further testing is not possible without industrial involvement and backing. The testing that was done showed that it would be possible to go from the design to a finished board in less than a day given screen production facilities.

Appendix 6 shows the specification of the system and its performance capabilities. It should be noted that the total cost of the system in hardware terms is a little under £50,000 (not including the two year development costs), a factor of at least five less than an 'off the shelf' automated screen printing machine but with increased performance. All the equipment used is readily available or easily reproduced.

The need for printing to such high accuracy (+/- 0.01 mm) is vital for the future of the PCB industry, particularly with development of SMT. There is no doubt that higher degrees of accuracy will be demanded in the future, and this can only be achieved with a greater investment in the research and development of screen printing technology. Screen printing is still considered too much as an art rather than a technology or science, this idea must be put down, screen printing is a vital part of the future of the electronics industry.

5.2 Vision System Software

The vision system allowed for registration in production using circular registration marks to a high degree of accuracy (+/- 0.005 mm) at high speed (less than 1 second per registration mark). Work was done, using a Generalised Hough Transform, to allow registration using almost any type of shape as a registration mark. This was successful and the system could be ‘taught’ to identify any registration mark by just placing it in front of the vision system. This worked but was too slow to be used in production as it took almost a minute a registration mark. This process could be speeded up with further improvements in the software, particularly with the incorporation of an Adaptive Hough Transform which would make the whole process far less computationally intensive. The improvement of this process would make the automated system more flexibile as it would allow registration to any part of the circuit rather than using registration marks.

5.3 Vision System Hardware

The system used an 386-33 PC which two years ago was nearly the top of the range of PC’s, now it is very much a middle to lower range machine with processing speeds at least four times greater available from more up to date 486 machines, these would still be able to use the software that has been developed using the Matrox vision system. Moving the system onto a dedicated VMS based work station would give a great improvement in processing speed, but would need new software to be developed as well as a new vision system, this would be a very expensive option.

Economic restrictions throughout the project have meant that many aspects of the system could be improved with a little investment. The camera mounting system was made by KPL and would not be practical in a production environment because of its weight and poor camera adjustment. Improvements in the camera movement would allow quicker more accurate camera alignment, increasing the speed of set up and allowing less skilled operators to work the system.

Relatively inexpensive lenses were used, these tend to vary in optical quality from lens to lens and the mechanics of the focusing tend to be loose. Higher quality lenses tend to have more consistent optical quality, with better overall build quality. This will result in more accurate lenses for measurement. This could also be achieved using higher resolution cameras.

The lighting system selected (see 4.4.1) was the best that was possible from the limited equipment available, this could be greatly improved with the use of lighting that has been specifically designed for machine vision applications. The lights themselves came from desk top lamps and as such their wavelength and power could not be adjusted. Further work could be carried out to specify the ideal lighting for the system.

Significant improvements in the speed of the vision system could only come from efforts to implement 'state of the art' computing technology such as the use of transputer technology. This is discussed in Chapter 6 (Future Work), this chapter also discusses the way ahead for the project.

5.4 Summary

The original aims of the project were achieved and, to a certain degree, exceeded. The original manual set-up and registration system for the Svecia screen printing machine has been replaced with an automated machine vision based system. The automation has had the result of giving a significant improvement in printing accuracy for the Sveciamatic-SM. From an initial printing accuracy of approximately +/- 0.025 mm a ten fold improvement was obtained giving a repeatable accuracy of +/- 0.0025 mm. This improvement was due to the increased resolution given by interaction of the machine vision and the automated machine control.

In addition to the automated screen printing system a range of image enhancing and processing software has been produced that has had applications in other research work, particularly in the field of 3D surface inspection.

The work outlined in this thesis has been successfully tested in a limited production environment. For this work is to be taken any further industrial backing is essential, not just for the funding, but for the facilities to test the work in a full industrial environment.

Chapter 6

Future Work

Chapter 6 : Future Work

Chapter 5 concluded the project by outlining the present state of the work giving some idea of aspects of the work that given further funding and time could be improved. This chapter will look briefly at how current changes in computing technology could be applied to the machine vision part of the system that has been developed. Finally, possible future developments of the complete automated screen printing system are discussed.

6.1 Vision System

The main 'bottle neck' in the system is because the control system has to wait for the vision processing to be completed before it can carry on, parallel processing using a shared memory parallel computer (transputer) would allow vision analysis to be carried out at the same time as the control system is operating. An example of what can be gained by implementing parallel processing is the registering of the screen to a board. The system locates the board registration marks one at a time and the control system has to wait until the positions have been calculated before the table can be brought in. Using parallel processing it would be possible to grab the images of the two registration marks, and while the table is being moved, the image processing could be carried out in parallel on both images in order to calculate their relative positions. The table takes approximately six seconds to move and the locating of the two registration marks takes approximately three seconds. This means that the image processing could easily be done within the time of the table input, resulting in a three second improvement in cycle time or about a twelve percent.

Parallel processing, with the use of a transputer, would also allow faster implementation of the Hough Transform (HT) and other vision algorithms **[36][37]**, use of this technology would mean a re-think of many the ideas put forward in this thesis, it would not be just a case of putting a transputer in the PC. Thazhuthaveetil and Shah

[37] assessed the use of various parallel HT algorithms on MIMD (Multiple Instruction Multiple Data) shared memory parallel computer, probably the most available parallel processing device, evaluating their effectiveness against the number of processors in the parallel machine. One method of assessment was to define a speed up ratio for the parallel algorithm:

$$speedup = \frac{\underline{Time\ \ taken\ \ by\ \ sequential\ \ algorithm}}{Time\ \ taken\ \ by\ \ p^{\prime}llel\ \ implementation}$$

A speed up figure of over 120 was recorded for one implementation of the HT that assigned one processor to each of the 256 rows in the image. Their final conclusion was that current rapid advancements in parallel processing technology will mean that the use of the HT as a means of shape detection is likely to increase.

Suter et al. **[36]** looked into the use of transputers on a variety of vision algorithm's, particularly edge detection and surface fitting and assessed their ease of implementation. Their conclusion was that a parallel C programming environment under Unix offered the best environment, but that new ways of thinking were required from the implementation of sequential algorithms in order to take full advantage of the benefits of parallel processing.

A subject under much discussion in computing are Neural Network Systems (NNS). NNS are an attempt to simulate human information processing on a computer. Buffa **[46]** discussed the use of NNS and how they work and came up with the following four advantages that they would offer vision system industry:-

1) Major reduction in engineering costs in applications development.
2) More cost effective systems to maintain.
3) Ability to solve problems no other technology can solve.
4) More efficient future hardware architectures.

NNS, although an idea for over thirty years, are still in their technological infancy and their take up in industry is still small. This will not be the case for long as it is the subject of much research (including work within The Nottingham Trent University's Engineering and Computing Faculty), it is likely that NNS will have a large impact on all aspects of computing, including machine vision, particularly in applications where patterns can be seen or implied.

6.2 Further Developments of the Complete the System

Further developments of the control system have been discussed by Fulford **[38]**. The next stage in the development of the complete automated screen printing system would have to be automated inspection. The inks used in the printing of the conductive layers are expensive (Johnson Mathey P1856 silver conducting ink cost approximately £250 for a 500g pot), and a single flaw can mean a board is worthless. This result of this is that inspection of each layer, as it is produced, is necessary so that if any faults are found they can either be corrected, or the board scrapped before any more layers are added. An additional benefit of inspection of each layer is that faults due to screen printing process shifts can be found and corrections to printing parameters made. This fact shows the benefit of inspection as a process control tool. Traditional circuit inspection has relied on electrical conductivity tests to check that circuits are closed. This is not enough in itself as, although a circuit may be closed, there could be point where there are weaknesses, particularly due to tracks that are thinner than they should be, or where pads between different layers only just make contact. It is therefore necessary to incorporate visual analysis into the inspection process.

Machine vision lends itself ideally to the inspection of PCB's because of its fast processing nature, it would be almost impossible to carry out reliable 100% inspection using human vision on anything but the smallest circuit. This has long been realised by the electronics manufacturing industry and it has therefore been the subject of much research. A survey carried out by Chin **[47]** titled "Automated Visual Inspection: 1981 to 1987", listed 69 papers dealing with inspection of printed circuit patterns, 72 papers on inspection of integrated circuits and hybrids, and 54 other papers dealing with various aspects of automated visual inspection in the electronics industry.

Inspection work stations for analysis of standard PCB's are becoming more and more widespread in the PCB industry. Optrotech, Itek and Orbot all manufacture commercially available systems, with many large PCB manufacturing companies (including IBM, DEC and Xerox) developing their own systems. There are three major points that have to be taken into account when looking at the use of automated visual inspection of PCB's:

- The size of the board (main frame boards can be as big as 500 mm x 600 mm), this may mean the use of many cameras in order to inspect a complete board at production rates.

- The conducting track width and the space between tracks, with more and more circuit integration these are becoming smaller, the resolution of the system needs to be such that these can be clearly identifiable from faults.

- The speed of the operation, to be effective the system needs to be able to keep up with production, especially when 100% inspection is needed.

The inspection of multilayer thick-film circuits presents many more problems in addition to those found in inspection of standard PCB's. The addition of each new layer partially obscures the view of previous layers. The problem is not so much inspecting those previous layers, but of differentiating the newest layer from the 'noise' of the rest of the circuit and being able to identify faults. Cooper et al **[48]** worked on classifying different flaws such as shorts and opens in conductive patterns and variation on these (e.g overly thin and thick tracks) that may lead to deterioration in the circuit with time. In the work they classified twelve different flaws that are present in multilayer circuits but gave no solution to their detection with 'noise' of the additional layers. Sanz and Jain **[49]** also looked at classification of errors in both standard PCB's and thick-film / hybrid circuits. They also discussed some of the problems associated with automatic inspection and gauging of thick-film circuits in addition to those mentioned above. These included the wider variance in component widths (0.002 to 0.05 inches) and sensing problems due to the nature of the inks.

A prototype system was developed by Lougheed and Svetkoff **[50]** that addressed the problems mentioned, and above proved the feasibility of automated visual inspection of multilayer thick film circuits. They ruled out the use of standard statistical and template-matching pattern recognition for a number of reasons. Instead the system followed a two step process by firstly segmenting the different areas of interest using knowledge based algorithms with a combination of grey-level and edge information and then processing these areas to detect specific faults. A lot of emphasis on the lighting, sensor and optical system and how they reacted with different inks in order to give as a great a contrast between layers as possible. They concluded that such automated inspection was feasible and that it will produce a significant improvement in product quality.

The different aspects mentioned above have outlined the difficulties that automated visual inspection presents, but with it comes much greater control of process quality. It would be flippant to say that automated inspection of the printed boards would have to be the next step in the development of the system because the cost of developing such a system would be great (the system put forward by Kikuchi et al. **[51]** used 10 CCD cameras and 10 image processors). Analysis of printer output would allow feedback to the machine control system, this would close the control loop to give far greater control over the quality and accuracy of the printing. Walker et al. **[52]** have developed an automated 100% in-line inspection system for unpopulated hybrid circuitry that can feed back process control information to the operator so that changes to parameters can be made. This type of system would be a useful first stage, the next step would be to replace the operator possibly using an expert system to determine the cause of the error, nearly all previous work has concentrated on classifying faults, not determining their cause and improvements in the control system as outlined by Fulford **[38]**. The use of an expert system in machine vision recognition has been looked at by Solinsky **[53]** for use in situations where little knowledge of the object of interest. Development of such a system would be a significant step forward in screen printing technology but could only be developed with major industrial involvement, not just for funding, but for facilities such as a production line staff with in depth knowledge of the screen printing process and its errors.

Bibliography

Bibliography

(1) "Printed Circuit Assembly", Fred W. Kear, Marcel Decker Inc. 1987.

(2) "Pattern Recognition Practice", edited by E. S. Gelsema and L. N. Kanal, North Holland, Proceedings of International Workshop, Amsterdam May 21-23 1980.

(3) "Machine Vision - Automated Visual Inspection and Robot Vision", David Vernon, Prentice Hall 1991.

(4) "Computer Vision - A First Course", R. D. Boyle and R. C. Thomas, Blackwell Scientific Productions Limited, 1988.

(5) "Readings in Computer Vision - Issues, problems, Principles, and Paradigms", 1980.

(6) "Pattern Recognition - Introduction and Foundations", edited by Jack Sklansky, Benchmark Papers in Electrical Engineering and Computer Science 1973, Dowden Hutchinson and Ross Inc.

(7) "Pattern Recognition Techniques", J. R. Ullmann, Butterworths 1973.

(8) "Computer Vision", D. H. Ballard and C. M. Brown, Prentice Hall 1982.

(9) "Proceedings of Second International Conference on Image Processing and its Applications, IEE June 1986.

(10) "Computer Image Processing and Recognition", Ernest L. Hall, Computer Science and Applied Maths, Academic Press, 1979.

(11) "Applying Machine Vision", Nello Zeuch, J. Wiley and Sons, 1988.

(12) "Digital Picture Processing - Vols. 1 and 2", Azriel Rosenfeld and Avinash C. Kak, Academic Press, 1982.

(13) "Machine Vision", Nello Zeuch and Richard K. Miller, Van Nostrand Reinhold, 1987.

(14) "C Programming Language", Kernighan and Ritchie

(15) "Microsoft C Run Time Library",

(16) "Screen Printing Electronic Circuits", Albert Kosloff, The Signs of the Times Publishing Company, Cincinnati, Ohio, U.S.A. 1980.

(17) "Digital Image Processing", W. K. Pratt, Wiley, New York, 1978.

(18) "Fitting Equations to Data, Computer Analysis of Multifactor Data", Second Edition, Cuthbert Daniel and Fred S. Wood, John Wiley and Sons Inc., 1980.

(19) R. M. Hodgson and S. J. McNiell, "The design of image processing systems for real-time inspection applications", *Proceedings of the 2nd International Conference on Image Processing and Its Applications*, June 24-26 1986, Imperial College.

(20) A. C. M. Gieles, W. D. van Amstel and W. J. Venema. "Automatic optical inspection for surface mounting technology", *Proceedings of 5th CIM Europe Conference*, Brussels, May 17-19 1989, pp.306-319.

(21) I. F. Pau, "Integrated testing and algorithms for visual inspection of integrated circuits", *IEEE Transactions on Pattren Analysis and Machine Intelligence*, Vol. PAMI-5, No.6, November 1983.

(22) J. R. Mandeville, "Novel Method for analysis of printed circuit images", *IBM Journal of Research and Development*, Vol. 29, No. 1, January 1985. pp.73-86.

(23) A. P. Sprague, M. J. Donahue and S. I. Rokhlin, "A Method for automatic inspection of printed circuit boards", *CVGIP: Image Understanding*, Vol 54, No. 3, November pp.401-415, 1991.

(24) R. D. Taylor, "Automatic optical inspection of thin film circuits", *ISHM '87 Proceedings*, pp.700-706.

(26) J. A. Calkins, "A New Approach to Visual Inspection", *Circuits Assembly*, December 1990, pp. 45-48.

(27) G. A. W. West, "A System for the Automatic Visual Inspection of Bare-Printed Circuit Boards", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-14, No. 5, September/October 1984.

(28) K-S. Fu, "Pattern Recognition for Automatic Visual Inspection", *Computer*, December 1982, pp.34-40.

(29) W. K. Pratt, "Quantative Design and Evaluation of Enhancement / Threshold Edge Detectors",*Proceedings of I.E.E.E.*, Vol. 67, No. 5, May 1979.

(30) A. Rosenfeld, "Survey: Image Analysis and Computer Vision: 1991", *CVGIP: Image Understanding*, Vol.55, N0.3. May 1992.

(31) G. X. Ritter, J. N. Wilson, and J. L. Davidson, "Image Algebra: An Overview", *Computer Vision, Graphics, And Image Processing*, 49, pp.297-331, 1990.

(32) C-Y, Wang and D. Lee, "Integration of manufacturing Systems: a vision and image processing perspective", IEEE 1990.

(33) J. Morris, "An integrated approach to vision and control", *Sensor Review.* (1991) pp.165-169.

(34) J. F. Claridge, "Automatic Glass Examination", *Sensor Review*, Vol. 11, No.1, 1991, pp. 17-21.

(35) A. R. Hidde and A. Prusak, "The use of artificial intelligence for printed circuit board manufacturing", *Computers in Industry* 16 1991, pp.1-11.

(36) S. A. Partridge, "The Role of the stencil in high definition screen printing", *Circuit World*, Vol. 13, No.2, 1987 pp.4-13.

(37) P. Plummer, "The benefits of colour- Colour Vision", *Image Processing*, september/October 1990.

(38) R. J. Beattie, "Threshold Selection In The Presence of Noise", *Proceedings of the 2nd International Conference on Image Processing and Its Applications*, June 24-26 1986, Imperial College.

(39) A. N. Jain and D. B. Krig, "A robust Hough technique for machine vision", *Proceedings of Vision '86*, Machine Vision Association opf Society of Manufacturing Engineers, Detroit, Michigan, June 3-6 1986.

(40) J. W. Davison, "Cost Efective Screen Printing", *Circuit World*, Vol. 3, Part 1, Pages 53-5, 1976.

References

References

(1) M. Howarth and A. Giles, Project Feasability Study, 1989.

(2) A. Kuhn, "Windows into Europe", Electronic Production, January 1991.

(3) "Printed Circuit Assembly", Fred W. Kear, Marcel Decker Inc. 1987.

(4) B. H. Carlisle, "Screen Printing Promises Smaller Cheaper PCB's", *Machine Design*, December 1988.

(5) E. N. Heesom, "Adding Polymer Circuitry to PCB's", *Electronic Production*, pp 19-27, April 1987.

(6) R. M. Nersesian, "Screen Printing is the answer", *Printed Circuit World Expo '80 Proceedings* pp.69-72.

(7) M. Howarth, D. Fulford, J. Keat and M. Robinson, "Screen print automation for printed circuit manufacture", Paper submitted to Eigth National Conference on Manufacturing Research, Birmingham Polytechnic, 8-10 September 1992.

(8) M. Young, "Screen Process Printing Techniques", *PC Fab*, pp 77-92, February 1987.

(9) R. W. Atkinson, "An Intelligent Approach to Screen Printing", *International Journal of Hybrid Microelectronics*, Vol.4 No. 2 October 1981.

(10) B. K. Liew and C. L. Tan, "CAD-based robotic vision", *International Journal of Computer Applications in Technology*, Vol. 4, No. 2, pp. 88-92. 1991.

(11) A. Sanfeliu and M. Ananos, " A CAD based vision system for identifying industrial work pieces", *Information Control Problems in Manufacturing Technology*, Madrid, Spain 1989, pp.599-604.

(12) J. Mahon, "Automatic 3-D inspection of solder paste on surface mount printed circuit boards", *Journal of Materials processing Technology*, 26 (1991), pp.245-256.

(13) D. A. Bolon, G. M. Lucas and S. H. Schroeter, "Radiation Curable Conductive Ink", *IEEE Trans. Electr. Insul.*, Vol EI-13 No 2, April 1978.

(14) K. Parslow, "Fundamental Functions- Board Survey", *Image Processing*, July/August 1990.

(15) D. Braggins, "Hard facts on software- Software Survey", *Image Processing*, September/October 1991.

(16) M. Coulthard, "Tile Inspection- The Right Approach", *Sensor Review*, Vol. 11, No.2, 1991, pp.15-18.

(17) D. Brzakovic and D. T. Khani,"Weld Pool Edge Detection for Automatic Control of Welding", *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 3, June 1991.

(18) P. Schatz and G. Freeman, "Automatic alignment aids silk screening", *Hybrid Technology*, pp. 19-24, February 1988.

(19) R. C. Daum, and S. Flom, "Advances and applications in machine vision lighting", *22nd International Symposium on Industrial Robots*, 1991 International Robots and Vision Automation Conference, Detroit, Michigan, USA, October 21-24, 1991.

(20) M. P. Colleta and K. G. Harding, "Lighting Science- Tools to guide machine vision", *Proceedings of Vision '89*, Machine Vision Association of Society of Manufacturing Engineers, Chicago, Illinois, April 24-27 1989.

(21) G. Biegel, "High frequency fluorescent lighting option for image processing", *22nd International Symposium on Industrial Robots*, 1991 International Robots and Vision Automation Conference, Detroit, Michigan, USA, October 21-24, 1991.

(22) T. Yakimovsky, "Boundary and object detection in real world images", *JACM*, Vol. 23, No. 4, pp. 599,618.

(23)J. S. Loomis, "Edge-finding algorithm with subpixel resolution", *Proceedings of Vision '89*, Machine Vision Association of Society of Manufacturing Engineers, Chicago, Illinois, April 24-27 1989.

(24) M. A. Crissman, "Subpixel edge detection", *22nd International Symposium on Industrial Robots*, 1991 International Robots and Vision Automation Conference, Detroit, Michigan, USA, October 21-24, 1991.

(25) S. Papert, "Uses of technology to enhance education", Techniacl Report 298, AI Lab, MIT, 1973.

(26) P. Horne, "The use of a vision system in the alignment of silk screens", 1990, student thesis Nottingham Polytechnic.

(27) P. V. C. Hough, "Methods and means for recognizing complex patterns", *US patent No. 3,069,654*, December 18, 1962.

(28) J. L. Turney, T. N. Mudge and R. A. Volz, "Recognizing partially occluded objects", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No.4, July 1985.

(29) E. R. Davies, "Corner Detection Using the Generalised Hough Transform", *Proceedings of the 2nd International Conference on Image Processing and Its Applications*, June 24-26 1986, Imperial College.

(30) R. O. Duda and P. E. Hart, "Use of the Hough Transform to detect lines and curves in pictures", *Graphics and image Processing*, Communications of The Association for Computing Machinary, Vol. 15, No. 1, January 1972.

(31) D. H. Ballard, "Genaralizing the Hough Transform to detect arbitary shapes", *Pattern Recognition*, Vol. 13, No. 2, pp.111-122, 1981.

(32) J. Illingworth and J. Kittler, "The Adaptive Hough Transform", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 5, September 1987.

(33) L. Hertz and R. W. Schafer, "Multilevel Thresholding Using Edge Matching", *Computer Vision, Graphics, and Image Processing* **44**, pp. 279-295 1988.

(34) P. K. Sahoo, S. Soltani, A. K. C. Wong and Y. C. Chen, "SURVEY- A survey of thresholding techniques", *Computer Vision, Graphics, And Image Processing*, 41, pp. 233-360, 1988.

(35) H. Freeman, "Computer Processing of Line-Drawing Images", *Computing Surveys*, Vol. 6, No. 1, March 1974.

(36) D. Suter, X. Deng, H. A. Cohen and T. S. Dillon, "Development and implementation of parallel vision algorithms", *Proceedings of Vision '89*, Machine Vision Association of Society of Manufacturing Engineers, Chicago, Illinois, April 24-27 1989.

(37) M. J. Thazhuthaveetil and A. V. Shah, "Parallel Hough Transform algorithm performance", *Image and Vision Processing*, Vol. 9, No. 2 April 1991.

(38) D. Fulford, "Automatic Control of High Resolution Screen Printing", Nottingham Polytechnic MPhil thesis, September 1992.

(39) D. Fulford, M. Howarth, and M. Robinson, "Automating the screen print process for high accuracy, high volume, printed circuit production", *Advances in Manufacturing Technology VI*, 1991, pp. 266-70.

(40) C. H. Vandommelen, "Fundamentals of camera selection for machine vision", *22nd International Symposium on Industrial Robots*, 1991 International Robots and Vision Automation Conference, Detroit, Michigan, USA, October 21-24, 1991.

(41) E. Martin, "Close up on cameras", *Image Processing*, Winter 1990.

(42) T. P. White and S. M. Leblanc, "An expert system for the design of optics for precision gauging with machine vision", *22nd International Symposium on Industrial Robots*, 1991 International Robots and Vision Automation Conference, Detroit, Michigan, USA, October 21-24, 1991.

(43) W. K. Pratt, Digital Image Processing, 1978, Wiley, New York, Chapters 2, 3, and 5.

(44) G. A. Al-Kindi, R. M. Baul and K. F. Gill, "An example of automatic two-dimensional component inspection using computer vision", *Proceedings of the Institute of Mechanical Engineers* Vol. 205, Part B: Journal of Engineering Manufacture, 1991.

(45) A. C. Englander, "Edge detection techniques for industrial machine vision", *Proceedings of Vision '86*, Machine Vision Association opf Society of Manufacturing Engineers, Detroit, Michigan, June 3-6 1986.

(46) M. G. Buffa, "Neural-Network Systems Will Make Many Machine Vision Applications Economically Justifiable", *Proceedings of Vision '89*, Machine Vision Association of Society of Manufacturing Engineers, Chicago, Illinois, April 24-27 1989.

(47) R. T. Chin, "Survey - Automated Visual Inspection: 1981 to 1987", *Computer Vision, Graphics, And Image Processing*, 41, pp. 346-381, 1988.

(48) P. W. Cooper, H, J, Curnan, and C. Knarr, "Machine vision inspection criteria for hybrid microelectronics multilayer substrates", *Proceedings of Vision '86*, Machine Vision Association opf Society of Manufacturing Engineers, Detroit, Michigan, June 3-6 1986.

(49) J. L. C. Sanz and A, K, Jain, "Machine-vision techniques for inspection of printed wiring boards and thick film circuits", *Journal of the Optical Society of America*, Vol. 3, No. 9, September 1986, pp.1465-1482.

(50) R. M. Lougheed and D. J. Svetkoff, "A system for automatic high speed visual inspection of multilayer thick film circuits", *International Journal for Hybrid Micro Electronics*, Vol. 4, Part 2 pp.401-409, 1983.

(51) H. Kikuchi, Y. Utsumi and H. Taga, "Automated Optical Inspection for High-Density Printed Wiring Boards", Technical Paper No. B10/3, 5th Printed Circuit World Convention, UK, 1990.

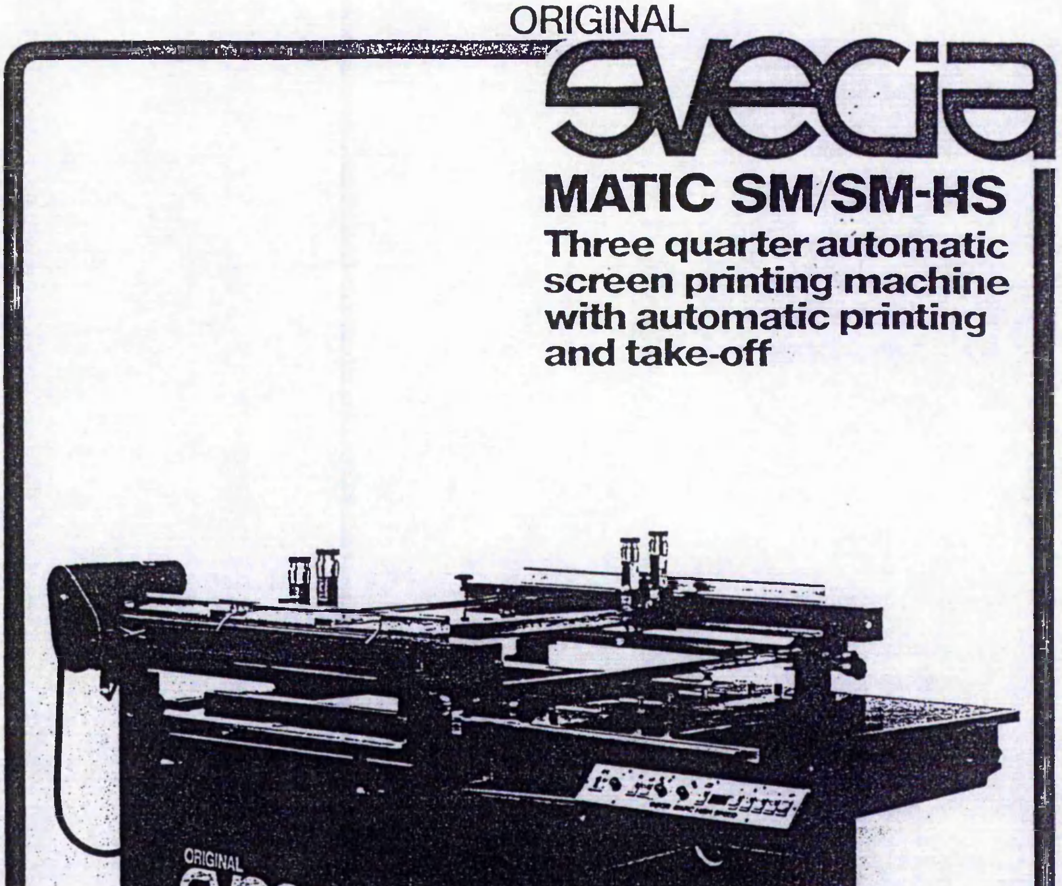
(52) R. A. Walker, G. E. Frame, and D. W. Neases, "Automated in-line machine vision verification of hybrid circuitry", *Proceedings of Vision '86*, Machine Vision Association of the Society of Manufacturing Engineers, Detroit, Michigan, June 3-6 1986.

(53) J. C. Solinsky, "The use of expert systems in machine vision recognition", *Proceedings of Vision '86*, Machine Vision Association opf Society of Manufacturing Engineers, Detroit, Michigan, June 3-6 1986.

Appendix 1

Sveciamatic SM

Description and Technical Data



ORIGINAL

svecia

MATIC SM/SM-HS

Three quarter automatic screen printing machine with automatic printing and take-off

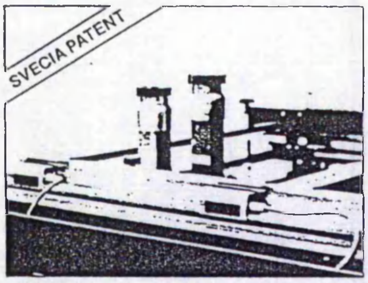
The flat-bed p
with the decis

Quick set-up.

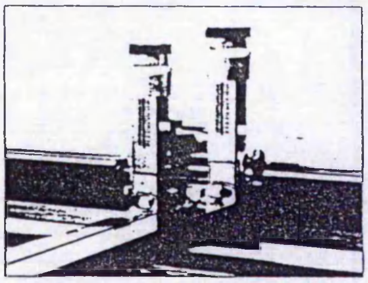
An important factor for maximising output is to achieve the shortest possible set-up and changeover time. Simple handling and practical adjustments operated without the use of any special keys or tools contribute to quick, easy operation and thus superior production capacity.

Built-in precision.

The Original Svecia Matic is precision built on a welded, very rigid frame that gives the press extreme stability and torsional strength—the basis for the accuracy, reliability and durability of the machine. The vibration-free drive absolutely ensures the highest printing quality. As all carriage and frame movements are mounted on precision ground bearings running on accurately machined steel tracks, the machine keeps its accuracy almost indefinitely.



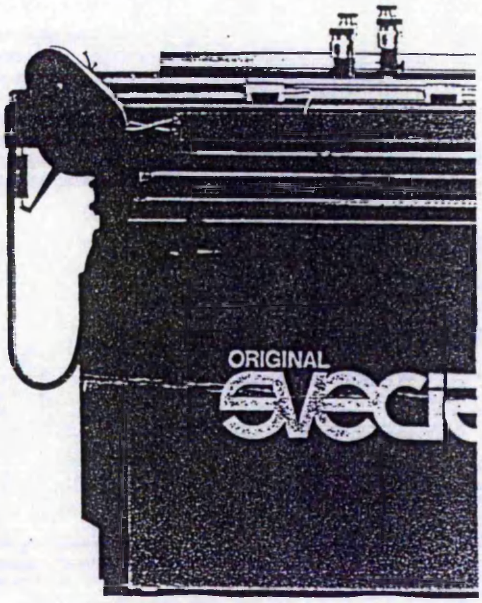
Sliding proximity switches for instantaneous adjustment of squeegee and stroke length without tools gives a higher output when printing small areas on large material sizes.

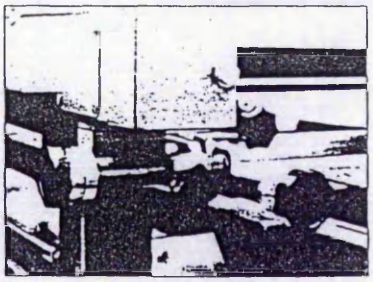


Squeegee and flood coater pressures and angles are steplessly adjustable by graduated micrometer dials, resulting in exceptional printing accuracy.

The superior "Svecia Principle"
the parallel moving printing frar

Original Svecia Matic works on the "Svecia Principle"
which means the printing frame remains horizontal wh
the printing table has a reciprocating movement—unl
the old "Hinge Principle" with the swinging frame. As a

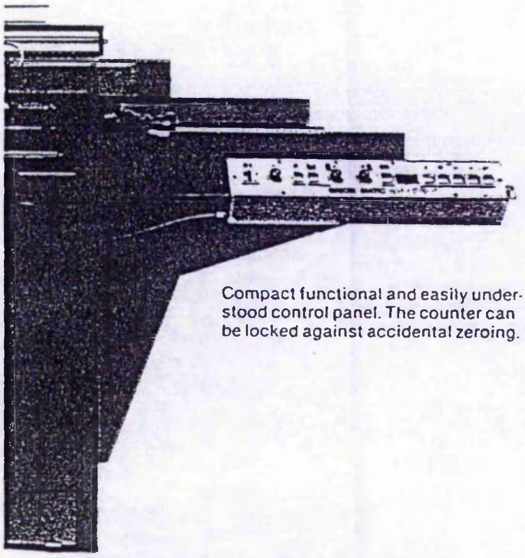




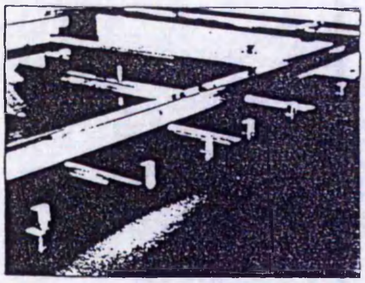
High printing speeds are necessary but an even more impor
time-saving factor is set-up time. The original Svecia Matic i
designed to minimise the set-up time, having a practical sys
for adjusting the frame in X and Y directions by easily acces
sible and lockable micrometer controls.

inting machine ve advantages

sult of the "Svecia Principle" you can print on a wide variety of materials from extremely thin stocks up to a thickness of 6 mm. This unique "Svecia Principle" also minimises the risk of drying-in and dust problems caused by turbulence from a swinging frame. Another advantage of the frame remaining in a horizontal position is that far superior ink control is available.



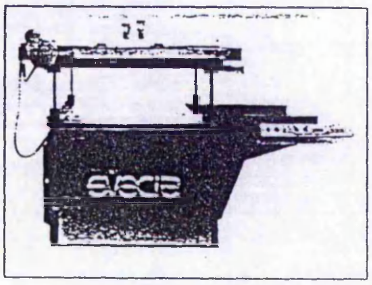
Compact functional and easily understood control panel. The counter can be locked against accidental zeroing.



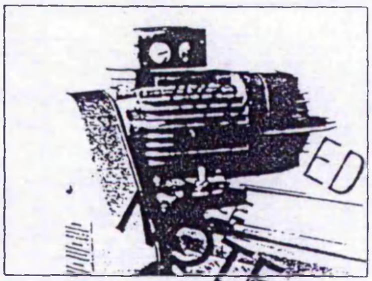
Self-compensating lay stops which rise and fall with the table movement are provided with hardened steel contact faces. A simple mechanism allows stops to be selected according to sheet sizes and for side-laying to either right or left hand. An electromagnetic gripper cancelling system permits the return of a sheet for inspection. Automatic continuous cycling (with a variable dwell of 0—10 seconds in the feed position) can be interrupted by the footswitch if required.

Outstanding technology gives more economic production.

The reliability and well proven technical design results in secure production. Svecia's long experience in designing and building screen printing machines, continuous dialogue with printers and the utilisation of the most up-to-date technology confirms the leading position of Svecia and guarantees you the best results. Technical features such as variable turbo vacuum and adjustable air cushion for feeding and delivery of difficult materials enable reliable and accurate printing. All types of material can be accommodated from very thin and soft materials to heavy and rigid stock. Excellent flexibility with high output can thus be achieved.

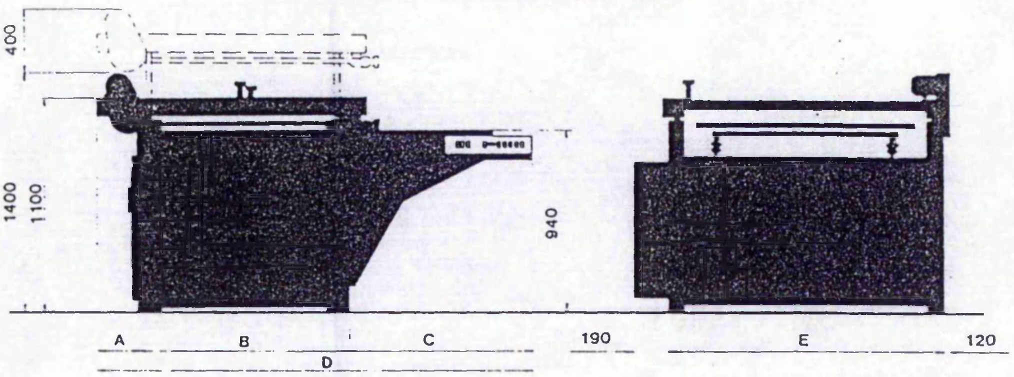


The printing head is accurately guided at each corner by a pillar which provides a high degree of accuracy. The whole frame assembly can be raised to a height of 400 mm to allow inspection and cleaning.



Squeegee speed is variable in three steps with a release arm to enable manual operation for set-up and cleaning.

Technical data



Size	A	B	C	D	E
400 x 600	210	1060	835	2105	1500
550 x 750	210	1060	835	2105	1500
650 x 900	210	1160	935	2305	1650
765 x 1070	210	1260	1035	2505	1800
880 x 1250	210	1360	1135	2705	2000
1000 x 1400	210	1640	1235	3085	2450
1200 x 1600	210	1800	1375	3385	2600

Model SM/SM-HS							
Max. print area = max. sheet size	400 x 600	550 x 750	650 x 900	765 x 1070	880 x 1250	1000 x 1400	1200 x 1600
Max. frame size, outer dimensions	1000 x 1090	1000 x 1090	1100 x 1190	1200 x 1390	1350 x 1590	1650 x 1850	1700 x 2000
Machine net weight	470	470	550	620	680	870	1000
Max. capacity at max. stroke length* SM SM-HS	1300 1500	1300 1500	1200 1400	1100 1300	1000 1200	900 1100	800 1000
Max. capacity at min. stroke length* SM SM-HS	1500 1700	1500 1700	1400 1600	1200 1400	1100 1300	1000 1200	900 1100
Squeegee speed m/sec. SM SM-HS	SM: 0.5—0.8—1.0 m/sec. SM-HS: 0.1—1.5 m/sec. (Other speeds available on request)						
Max. sheet thickness	5. (Thicker material can be printed in special executions)						
Max. sheet weight	3.5 (Heavier material can be printed in special executions)						
Angle adjustment of squeegee and flood coater	0—30						
Adjustment in height of squeegee and flood coater	50						
Fine register adjustment in both planes	± 16						
Snap off	0—25						
Current rating	3 kW, 10 A at 380 V and 13 A at 220 V. (Available for other electrical supplies)						

*Feeding time may need to be added

Please note: SVECIA policy is one of continuous improvment and accordingly the manufacturers reserve the right to change specifications without prior notice.
Please, do not hesitate to contact us for further information on our extensive programme of machinery, dryers and auxiliary equipment for the screen printing industry.

svecia

SVECIA SILKSCREEN MASKINER AB
S-145 81 Norsborg/Stockholm · Sweden
Tel 753-690 00 · Tlx 11297
Fax 753-756 58 · Tgm Sveciascreen

SVECIA HANDELSGES. M. B. H.
Bunuesstrasse 8 · A-5071 Wals/Salzburg · Austria
Tel 662-85 10 21 · Tlx 632474

Svecia GMBH Siebdruckmaschinen
Mühllach 11 · 8505 Röthenbach/Peg.
West Germany · Tel 0911-570 82/570 83 · Tlx 623882

SVECIA FRANCE SARL
73, Avenue de la République · 91650 Crosne
France · Tel 6-948 27 17 · Tlx 691523

SVECIA EAST LTD
7-D, Kingsford Industrial Bldg
26-32, Kwai Hei Street · Kwai Chung, N.T.
Hong Kong · Tel 0-26 16 01 · Tlx 38729
Fax 0-25 11 39 · Tgm Sveciaeast

SVECIA USA, INC.
220 Distribution Street · San Marcos, Ca. 92069
USA · Tel 619-471-1880 · Fax 619-471-2372

SVECIA ESPANA S.A.
c/Aluminio 5 · Poligono Industrial Finanzauto
Arganda del Rey · Madrid · Spain · Tel 1-871 71 20

Oy SVECIA AB
Lauttasaarentie 54 · SF-00200 Helsingfors 20 · Finland
Tel 009 358-0-692 60 33 · Tlx 57-125318 · Tgm Seri-Fantasy

Your representative:

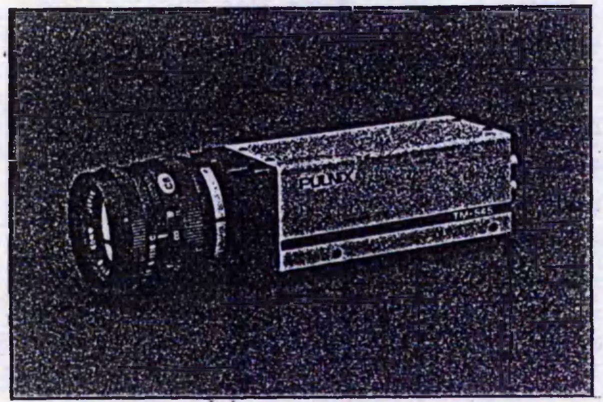
Svecia service and sales facilities are available throughout the world.

Appendix 2

Camera and Lens Specifications

PULNiX

TM-545W (EIA)
TM-565W (CCIR)
2/3" CCD CAMERA



FEATURES

- 510 pixel, 2/3" Interline Imager
- Low light sensitivity (2 lux)
- External H_DV_D sync locking
- Small size, low power consumption
- Low cost, all-purpose camera
- Rugged package
- C-mount with backfocus adjustment
- 3-Year warranty

GENERAL DESCRIPTION

The TM-545W and TM-565W Series cameras offer the resolution and image format of the popular TM-540 Series in a less expensive package for general purpose imaging. Featuring the same sharp 510(H) X 492(V) interline transfer CCD imager as the TM-540, the TM-545W and TM-565W are excellent for general surveillance, inspection, and machine vision applications.

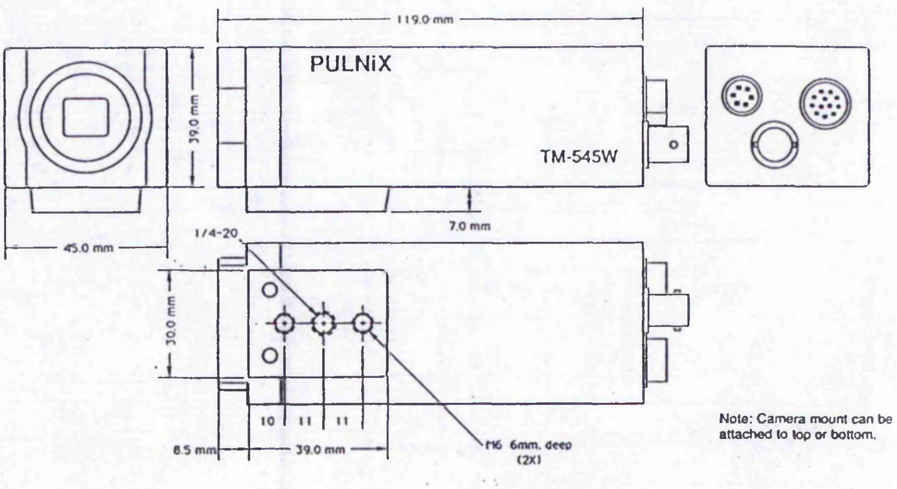
The TM-545W and TM-565W accept external H_D/V_D sync. These cameras are perfect replacement devices for failing 2/3" tube cameras. The 510x492, 2/3" format is also ideal for use in low cost vision systems.

Options available: AGC disable, blemish free imager, internal IR cut filter, gamma = 1.0.

SPECIFICATIONS

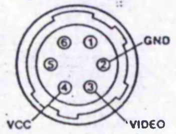
	TM-545W	TM-565W
Imager	2/3" Interline transfer CCD	
Pixel	510 (H) x 492 (V)	500 (H) x 582 (V)
Scanning	525 lines, 60 Hz, interlace (EIA)	625 lines, 50 Hz, interlace (CCIR)
Sync.	Internal/External auto switching	
External Sync.	EXT H_DV_D	EXT H_DV_D
Video Output	1.0 Vp-p, Sync negative, 75 ohm	
Minimum Illumination	2.0 Lux at F = 1.4	
TV Line Resolution	370 H, 350 V	370 H, 420 V
AGC, Gamma	AGC ON, Gamma = 0.45	
Power Requirement	12 VDC 300 mA	
Lens Mount	C-mount (Back focus adjustable)	
Auto-Iris	12 V and video output	
Operating Temperature	–10°C to +50°C (+14°F to +122°F)	
Size (inches)	1.50" (H) x 1.75" (W) x 4.68" (L)	
Weight	8.5 oz. (250 grams)	
Shock and Vibration	7 G (11 to 200 Hz), 60 G shock	

PHYSICAL DIMENSIONS mm



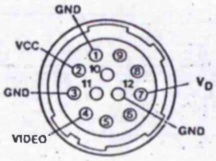
Note: Camera mount can be attached to top or bottom.

PIN CONFIGURATION



EXTERNAL VIEW 6-PIN

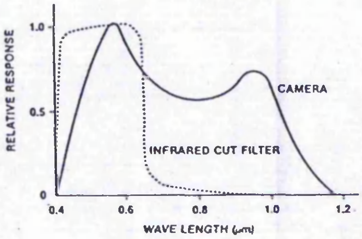
1. NC
2. GND
3. VIDEO LEVEL
4. + 12 VDC OUT
5. NC
6. NC



EXTERNAL VIEW 12-PIN

1. GND
2. + 12 VDC IN
3. GND
4. VIDEO
5. NC
6. NC
7. VD
8. NC
9. H_D
10. NC
11. NC
12. GND

SPECTRAL RESPONSE



CABLES AND ACCESSORIES

PC12P—	12 pin (power and sync connector)
PC6P —	connector for auto iris
KC-10 —	10 foot cable with connector
KC-X —	custom length power/sync cable

PULNiX

In the U.K.
PULNiX Europe Ltd,
Kapa Complex, Wade Road,
Basingstoke, Hants, RG24 0PL.
Tel: 0256 475555 Tlx: 859532
Fax: 0256 466268

PULNiX Europe Ltd,
35 Leslie Hough Way,
Salford University Business Park,
Manchester, M6 6AJ.
Tel: 061 745 7996 Fax: 061 745 8654

In Germany
PULNiX Europe Ltd,
P.O. Box 1202, D-8755 Alzenau,
West Germany.
Tel: 06023 4666 Fax: 06023 4667

Appendix 3

MVP-AT Commands

Name	Mode	Description
chan	all	selects the input channel
disformat	P	customizes the display format
ega	P	enables the ega
formatbuf	all	formats the frame buffers for output in American Standard
gain	all	sets the gain function of the input A/D converters
init	all	initializes the board to a default state
inmode	all	selects the input data path
key	all	selects input keying function
mask	all	enables the user to selectively write to any combination of frame buffer bit planes
offset	all	adjusts the lower reference voltage on the A/D converter
opmode	all	selects the operating mode
outmode	all	selects outpath operating mode
outpath	P&C	selects the display source
outzoom	P	zooms the background screen
pan	P&C	pans the displayed image relative to its current position on the screen
procwin	P&C	selects window
reststate	all	restores a previously saved state of the board
rpan	P&C	relative pan
rscroll	P&C	relative scroll
scroll	P	scrolls to an absolute position on the screen
selboard	all	selects an MVP-AT board
softinit	all	resets the board to the default state
sync	all	selects the sync source
video	all	selects display source; enables/disables display
viewwin	all	sets the viewing window size and location
wblank	all	waits for vertical blanking

Table A.1: Control and Display Commands

Name	Mode	Description
cgrab	C	controls continuous grabbing mode
cstart	C	selects first field to start continuous grabbing
grabzoom	C	sets the zoom factor while in continuous grabbing mode
inpan	C	absolute pan of input grabbing screen
inrpan	C	relative pan of input grabbing screen
inscroll	C	absolute scroll of input grabbing screen
inrscroll	C	relative scroll of input grabbing screen
selzoom	C	selects type of zoom allowed when in continuous grabbing mode

Table A.2: Continuous Grabbing Commands

Name	M	Description
absolute	P	calculates the absolute values of an image
clear	P	clears screen to the specified index value
clip	P	clips contents of FB4 or FB5 to 255
compcol	P	compresses an RGB and overlay image into a 16-bit frame buffer
disgrab	P	disables frame grabbing
engrab	P	enables frame grabbing
faverage	P	averages the image from the camera input
fimage	P	performs an image operation between a stored
	P	image and camera input
image	P	performs function between image and constant
inmap	P	maps pixels according to a selected map
interimage	P	performs a function between two images
pixblt	P	copies between rectangles
mix	P	combines two frame buffers on the basis of a third
snapshot	P	takes a snapshot

Table A.3: Point-to-Point Commands

Name	Mode	Description
fhisto	P&C	calculates faster histogram
histo	P	calculates intensity histogram of source FB
histsample	all	reduces entries in a histogram (or profile)
maximum	P	calculates the maximum pixel value of a FB
minimum	P	calculates the minimum pixel value of a FB
modhist	all	modifies a histogram through mapping
obhist	P	calculates the intensity histogram of an object
obsum	P	calculates the total intensity of an object
profile	P	takes an intensity profile
sumvector	P	calculates the total intensity of a vector
threshold	all	finds the minima and maxima of an image's histogram to be used for thresholding

Table A.4: Statistical Commands

Name	Mode	Description
cbl	all	converts 3 byte arrays to a long word array
cbw	all	converts 2 byte arrays to an integer array
clb	all	converts array of long words to 3 byte arrays
cwb	all	converts array of integers to 2 byte arrays
invert	all	inverts a character array
olutlay	all	generates an overlay LUT
rampscale	all	generates a ramp pixel mapping function
rlut	all	reads values from a LUT
sawscale	all	generates a sawtooth pixel mapping function
scaling	all	creates a scaled pixel mapping function
slut	all	selects a LUT palette
threshscale	all	generates threshold pixel mapping function
wlut	all	writes entries to a LUT

Table A.5: Lookup Table Commands

Name	M	Description
average	P	performs noise reduction
bincomp	P	compares 2 binary images in different FBs
connectivity	P	makes a connectivity map
convolve	P	convolves an image
dilate	P	dilates a user defined object
erode	P	erodes a user-defined object
horedge	P	performs horizontal edge detection
kirsch	P	performs edge detection using the Kirsch compass gradient operators
lp1	P	performs Laplacian edge detection - type 1
lp2	P	performs Laplacian edge detection - type 2
prewitt	P	performs edge detection using the Prewitt compass gradient operators
rvector	P	reads an object
sharp1	P	performs image sharpening - type 1
sharp2	P	performs an image sharpening - type 2
sobel	P	performs an approximation of Sobel edge detection
thin	P	thins a line
vertedge	P	performs vertical edge detection

Table A.6: Neighbourhood Commands

Name	M	Description
areaptn	G&C	Sets the ACRTC area pattern
circle	G&C	draws a circle
dot	G&C	draws a dot
drawhist	G&C	draws a histogram/profile in the frame buffer
drawmode	G&C	sets the drawing mode
ellipse	G&C	draws an ellipse
frect	G&C	draws a filled rect from current pen position
gclear	G&C	clears the graphics window
graphwin	G&C	sets the graphics window
gtext	G&C	outputs text string to screen
histmode	G&C	selects numeric base for x-axis labelling of drawhist
line	G&C	draws a line
lineptn	G&C	sets a new line pattern
move	G&C	moves the current pen position
paint	G&C	fills a closed area
rdot	G&C	does a relative move and then draws a dot
rect	G&C	draws a rectangle
rfrect	G&C	draws a filled rectangle relative to current pen position
rline	G&C	draws a line relative to current pen position
rmove	G&C	moves the pen position
rrect	G&C	draws a relative rectangle
selfont	G&C	selects a user defined font to be used with gtext
setbcolor	G&C	sets the background drawing color
setcolor	G&C	sets the current drawing color

Table A.7: Graphics Commands

Name	M	Description
colr	I/O&C	reads a pixel column from a frame buffer
colw	I/O&C	writes a pixel column to a frame buffer
cpuwin	I/O&C	selects the CPU access window
decode	I/O&C	performs run-length decoding of user-defined file
encode	I/O&C	performs run-length encoding of user-defined file
fromdisk	I/O&C	copies an image from a file to a frame buffer
iowin	I/O&C	sets the I/O window
pixr	I/O&C	reads pixel value from a frame buffer
pixw	I/O&C	writes pixel value to a frame buffer
rowr	I/O&C	reads a row from a FB and writes it into a workbuffer
roww	I/O&C	write a row from a workbuffer into a frame buffer
todisk	I/O&C	copies an image from a frame buffer to a disk file

Table A.8: I/O Commands

Appendix 4

Vision System 'C' Functions

The 'C' code listed here is the menu driven development test bench used in the project. Most of the vision enhancement and processing techniques described within the thesis can be found in this listing. The complete test bench allows testing of all the techniques and is therefore a useful teaching aid as an introduction to machine vision.

HEADER FILE MENU5.H

```
#include <stdio.h>
#include <dos.h>
#include <string.h>
#include <conio.h>
#include <bios.h>
#include <ctype.h>
#include <stdlib.h>
#include <process.h>
#include <memory.h>
#include <malloc.h>
#include <graph.h>

#define MVPMEM        0xd000
#define MVPIO         0x300

#define MEG10         0
#define MEG12         1
#define NON_INTERLACED 0
#define INTERLACED     1
#define EUROPEAN       0
#define AMERICAN       1

#define OLUTS         1
#define ILUTS        0
#define DEFAULT        0

#define REDM          0
#define GREENM         1
#define BLUEM         3

#define START         0
#define ONE          1
#define COUNT         256

#define CAMERA         0
#define EXTERN        1
#define FRAME_BUFFER    1

#define DISABLE        0
#define ENABLE         1
#define BUFFER         1

#define IO          0
#define GRAPHICS        1
#define PROCESS        2
#define CGRAB         3

#define OVERLAY         2
#define COLOUR         6

#define BW           1
#define NTSC          3

#define FB0          0
#define FB1          1
```

```
#define FB2         2
#define FB3         3
#define FB4         4
#define FB5         5
#define FB6         6

#define NONE          0
#define NULPTR        -1

#define BLACKM          0
#define WHITEM          255

#define FOREVER       -1

#define TRUE      1
#define NUM       6
#define NUM2      6
#define NUM3      9
#define NUM4      5
#define CLEAR     "\x1B[2J"
#define ERASE     "\x1B[K"
#define NORMAL     "\x1B[0m"
#define REVERSE   "\x1B[7m"
#define HOME      "\x1B[1;1f"
#define BOTTOM     "\x1B[20;1f"
#define U_ARRO    72
#define D_ARRO    80
#define INSERT    82

#define  PALETTE 0
#define  END     255
#define  BLACK   0
#define  WHITE   255

#define  PI         3.141592654
#define  RADIUS     10
#define  XCOORD     20
#define  YCOORD     20

#define  TAB_ENTS   30
#define  INCREMENTS 45

typedef struct {  /*== Structure for linked lists ==*/
        int start;
        int end;
        int label;
        int next;
}LINK;

typedef struct {     /*== STRUCTURE FOR PRECISE CENTRES ==*/
        double x;
        double y;
        int check;
        double r;
} CENTRE;


typedef struct {     /*== STRUCTURE FOR R-TABLE ==*/
        double r[TAB_ENTS];
        double alpha[TAB_ENTS];
        int entry;
} R_TABLE;


extern void display(char *arr[],int size,int pos);
extern int getcode(void);
extern void action(int pos);
extern int action2(int pos);
extern extern void init_mvp(void);
extern void snap(void);
extern void reed(void);
extern void store(void);
extern void live(void);
extern void image_menu(void);

extern int action3(int pos);
extern int action4(int pos);
extern void edge_menu(void);
extern void centre_menu(void);
extern void stretch(void);
extern void threshold(void);
extern void blob(void);
extern void threshhist(int xxxx);
extern CENTRE hough(CENTRE);
extern void horedge(void);


unsigned int table[1500];
unsigned int blobbuff[256];

static char path[] = "d:\\mvp\\image";
static char filespec[]="******.***";
static char name[20];
static int thresh_chk=0;
static int stretch_chk=0;
static unsigned char bufg[256] ,bufb[256] ,bufr[256] ,hbuf[256];
static unsigned long hisbuf[258];
static unsigned long hisbuf2[258];
static unsigned char maxbuf[512] ,minbuf[256];
static unsigned char maxbuf2[512] ,minbuf2[256];
R_TABLE john[INCREMENTS];
```

PROGRAM MENU5.C

```
#include <menu.h>



char spec[]="*****.***";

void main(void);
void display(char *arr[],int size,int pos);
int getcode(void);
void action(int pos);
int action2(int pos);
int action3(int pos);
int action4(int pos);
void init_mvp(void);
void snap(void);
void reed(void);
void store(void);
void live(void);

void main(void)
{
        static char *items[NUM]=
            {
                "RETRIEVE IMAGE",
                "TAKE SNAPSHOT",
                "STORE",
                "LIVE IMAGE",
                "IMAGE PROCESSING",
                "QUIT", };
        int curpos;
        int code;
        init_mvp();
        printf(CLEAR);
        curpos=0;
        while(TRUE){
            display(items,NUM,curpos);
            code=getcode();
            switch(code){

                case U_ARRO: if(curpos>0)--curpos;break;
                case D_ARRO: if(curpos<NUM-1)++curpos;break;
                case INSERT: action(curpos);break;
            }
        }
}

/* Display menu choice */

void display(char *arr[],int size,int pos)
{
int j;

        printf(HOME);
        for(j=0;j<size;j++){
            if(j==pos)printf(REVERSE);
            printf("%s\n",*(arr+j));
            printf(NORMAL);
        }
        printf(BOTTOM);
}

/* Get menu selection */
```

```
int getcode(void)
{
int key;

      while(getch()!=0);
      return(getch());
}

/* Carry out operation selected from menu */

void action(int pos)
{
      printf(ERASE);
      switch(pos){
          case 0 : reed();system("cls");break;
          case 1 : snap();system("cls");break;
          case 2 : store();system("cls");break;
          case 3 : live();system("cls");break;
          case 4 : image_menu();break;
          case 5 : _setvideomode(_DEFAULTMODE);exit(1);
      }
}

/* Initialise Vision Board */

void init_mvp(void)
{
      im_init(MVPMEM, MVPIO);
      im_disformat(0,1,0);
      im_sync(0,0);
      im_chan(1);
      im_clear(4,0);
      im_clear(5,0);
      im_inmode(BW);
      im_outpath(0,NULPTR,0,0);
      im_video(1,1);

}

/* Retrieve image from disc */

void reed(void)
{
char fname[34];
struct find_t c_file;
int i;
int filec=0;
int file=0;

      system("cls");
      _dos_findfirst("*.img",_A_NORMAL,&c_file);
      filec++;
      printf("\n%s\n",c_file.name);
      while( _dos_findnext(&c_file)==0){
          printf("%s\n",c_file.name);
          filec++;
      }
      printf("file count= %d ",filec);
      printf("WHICH PICTURE DO YOU WISH TO RETRIEVE\n");
      gets(name);
      strcpy (spec,name);
      strcpy (fname, path);
      strcat (fname, "\\");
      strcat (fname, spec);
      strcat (fname, ".img");
```

```
        init_mvp();
        im_opmode(0,0);
        im_fromdisk(fname);
        im_opmode(2,0);
        im_outpath(0,NULPTR,0,0);

}

/* Copy an image to disc */

void store(void)
{
int i;
int c;
char fname[34];

        i=0;
        system("cls");
        printf("WHAT WOULD YOU LIKE TO CALL THE PICTURE\n");
        gets(name);

        strcpy (spec,name);
        strcpy (fname, path);
        strcat (fname, "\\");
        strcat (fname, spec);
        strcat (fname, ".img");
        im_opmode(0,0);
        im_todisk(fname);
        im_opmode(2,0);
        im_outpath(0,NULPTR,0,0);
}

/* Display live image from selected channel */

void live(void)
{
int chan;

        init_mvp();
        printf("\nWhat Channel");
        chan=getch();
        im_opmode(2,0);
        im_chan(chan);
        im_outpath(0,-1,0,0);
        im_video(1,0);

}

/* Take snapshot from selected channel */

void snap(void)
{
int chan;

        init_mvp();
        printf("\nWhat Channel");
        chan=getch();
        im_opmode(2,0);
        im_chan(chan);
        im_outpath(0,-1,0,0);
        im_video(1,0);
        printf("\n Press a key to take snapshot");
        getch();
        im_video(1,1);
        im_opmode(2,0);
```

```
     im_chan(chan);
     im_outpath(0,-1,0,0);
     im_snapshot(chan);

}

/* Menu for image processing technique selection */

void image_menu(void)
{
int checker=0;
static char *items[NUM2]=
     {
         "CONTRAST STRETCH",
         "THRESHOLD",
         "BLOB FIND",
         "EDGE DETECT",
         "RETURN TO MAIN MENU",
         "QUIT", };
int curpos,code;

     printf(CLEAR);
     curpos=0;
     while(TRUE){
         display(items,NUM2,curpos);
         code=getcode();
         switch(code){

              case U_ARRO: if(curpos>0)--curpos;break;
              case D_ARRO: if(curpos<NUM2-1)+ +curpos;break;
              case INSERT: checker=action2(curpos);
                                   if(checker= =1){system("cls");return;}
                                       break;
         }
     }
}

/* Carry selected image processing technique */

int action2(int pos)
{
     printf(ERASE);
     switch(pos){
         case 0 : stretch();break;
         case 1 : threshold();break;
         case 2 : blob();break;
         case 3 : edge_menu();break;
         case 4 : return 1;break;
         case 5 : _setvideomode(_DEFAULTMODE);exit(1);
     }
}
```

PROGRAM IMAGE_MEN.C

```
#include <menu.h>

void stretch(void);
void threshold(void);
void threshhist(int xxxx, int chan);
void stretch2(void);
void draw_stretch(int low, int high);
void draw_hist(int colour, unsigned long *his);
void draw_line(int colour,int value);

struct videoconfig vc;


int select(int x, int y);
void screen1(int x,int y);
void key(void);

/* Contrast stretching carried out on image, displays stretched and unstretched histograms */

void stretch(void)
{
        int  high,low,thresh,i;
        double scale_fact;
        int yorn,smooth=15,incdec;

        _clearscreen(_GCLEARSCREEN);

        stretch2();
        draw_stretch(low,high);
        draw_hist(16, hisbuf);
        getch();
        printf("\nDo you want to change the histogram smoothing?");
        yorn=getch();
        incdec= =0;
        if((yorn= ='y')||(yorn= ='Y')){
            while(incdec!=13){
                printf("\rSmoothing value is presently %2d, +/- to increase/decrease",smooth);
                incdec=getch();
                if(incdec= =43)smooth++;
                else if(incdec= =45)smooth--;
                im_histo(1,hisbuf);
                im_threshold(smooth,minbuf,maxbuf,20,hisbuf);
                _clearscreen(_GCLEARSCREEN);
                draw_hist(16,hisbuf);
            }
        }
        im_average(1,0);
        im_inmap(FB0, 1, 0);
        im_outpath(1,-1,0,0);
        _clearscreen(_GCLEARSCREEN);
}

/* Threshold level is automatically selected. The operator is able to 'fine tune' the threshold level by moving a curser up or down a histogram of the image. As the curser moves up and down the results of the threshold are shown on the system monitor. */

void threshold(void)
{
        int xxxx;
        int choice,inc;
        int i,s;

        thresh=minbuf[4];               /* Set threshold level */
        draw_hist(16,hisbuf);
        draw_line(1,thresh+2);

        while((s=getch())!=13){
            if(s= =43){
                        thresh+ +;
                        draw_line(1,thresh+2);
                        draw_line(16,thresh+1);
            }
            if(s= =45){
                        thresh--;
                        draw_line(16,thresh+1);
                        draw_line(1,thresh+2);
            }
        }
        xxxx=thresh;
        threshhist(xxxx,MASTER);
        im_outpath(0,-1,0,0);

        printf("\nDo you want to adjust threshold level?");
        if((choice= ='n')| |(choice= ='N'))return;

        while(inc!=110){
            inc=getch();
            switch(inc){
                case 43: xxxx+ =8;
                                threshhist(xxxx,MASTER);
                                draw_line(1,xxxx+2);
                                draw_line(16,xxxx-8+1);
                                break;
                case 45: xxxx-=8;
                                threshhist(xxxx,MASTER);
                                draw_line(16,xxxx+8+1);
                                draw_line(1,xxxx+2);
                                break;
                case 42: xxxx+ +;
                                threshhist(xxxx,MASTER);
                                draw_line(1,xxxx+2);
                                draw_line(16,xxxx+3);
                                break;
                case 47: xxxx--;
                                threshhist(xxxx,MASTER);
                                draw_line(16,xxxx+3);
                                draw_line(1,xxxx+2);
                                break;
                case 13:
                                _clearscreen(_GCLEARSCREEN);
                                thresh=xxxx;
                                im_clear(2,0);
                                im_inmap(0, 2, 0);
                                im_outpath(2,-1,0,0);
                                im_clear(4,0);    /* clear buffers 0 and 1*/
                                return;
                                break;
            }
        }
}

/* Carry out threshold on image using level xxxx */

void threshhist(int xxxx, int chan)
 {
        im_threshscale (START, BLACKM, xxxx, WHITEM, END, bufr);
        im_cwb (bufr, bufr, hbuf, COUNT);
        im_wlut (ILUTS, PALETTE, START, COUNT, bufr, bufr, bufr);
        im_slut (ILUTS, PALETTE);
        im_inmap(1, 0, 0);
}

void draw_stretch(int low, int high)
{
        int i;

        printf("\nEnter Lighting Used:- ");
        gets(light);
        printf("\nEnter Filters  Used:- ");
        gets(filter);

        _settextcolor(1);
        _setcolor(4);
        _settextposition((short)8,(short)40);
        printf("Low= %d",low);
        _settextposition((short)9,(short)40);
        printf("High= %d",high);
        _settextposition((short)10,(short)40);
        printf("Contrast (high-low)= %d",high-low);
        _settextposition((short)12,(short)40);
        printf("No. of minima= %d",minbuf2[0]);

        for(i=1;i< =minbuf2[0];i+ +){
             _settextposition((short)12+i,(short)40);
             printf("Grey Level of minima [%d]= %d",i,minbuf2[2*i]);
        }
        _settextposition((short)13+minbuf2[0],(short)40);
        printf("No. of maxima= %d",maxbuf2[0]);
        for(i=1;i< =maxbuf2[0];i+ +){
             _settextposition((short)13+minbuf2[0]+i,(short)40);
             printf("Grey Level of maxima [%d]= %d",i,maxbuf2[2*i]);
        }

        _settextposition((short)4,(short)40);
        _rectangle(_GFILLINTERIOR,260,48,285,63);
        printf("Histogram of unstretched image");
        _settextposition((short)6,(short)40);
        _setcolor(15);
        printf("Histogram of stretched image");
        _settextposition((short)30,(short)0);
        _rectangle(_GFILLINTERIOR,260,79,285,94);
        printf("0");
        _settextposition((short)30,(short)(255/4.0)+1);
        printf("255 Grey Level-»");
        draw_hist(4,hisbuf2);
}

void draw_hist(int colour, unsigned long *his)
{
        int i;

        _setcolor(colour);
        for(i=2;i<258;i+ +){   /* draw histogram*/
             _moveto(2*(i-2),460);
             _lineto(2*(i-2),460-(his[i-2]/100));
             _moveto((2*(i-2))+1,460);
             _lineto((2*(i-2))+1,460-(his[i-2]/100));
             if(i%20= =0){
                   _settextposition((short)30,(short)(i/4.0));
                   printf(" %d",i);
             }
        }
}
```

```
void draw_line(int colour,int value)
{
        int i;

        _setcolor(colour);
        _moveto(2*(value-2),460);
        _lineto(2*(value-2),460-(hisbuf[value]/100));
        _moveto((2*(value-2))+1,460);
        _lineto((2*(value-2))+1,460-(hisbuf[value]/100));
}


/* Carry out blob detection on thresholded image, allows manual selection of a specific blob */

void blob(void)
{
unsigned char far *p;
int i, j , r, row, nrows, mincol, maxcol,x;
int thresh,threshval,john,now,startlist,startcurr;
LINK linklist[200];
int high,low,list,ilist,index,count,newlabel,max,bcount;
unsigned int *q,ii,l,m,*a;

        for(i=0;i<256;i++)blobbuff[i]=0;
        for(i=0;i<1500;i++)table[i]=0;

        r=count=0;
        mincol = 1;
        maxcol = 511;
        system("cls");

        for(i=0;i<200;i++)linklist[i].next=i+1;
        linklist[199].next=0;
        startlist=now=startcurr=0;
        newlabel=0;
        im_outmode(0);
        im_outpath(0,-1,0,0);
        im_opmode(0,0);

        for (j = 0; j < 4; j++) {
            im_cpuwin(j);
            nrows = j == 3 ? 96 : 128;
            for (row = 0; row < nrows;row++,r++ ) {
                startcurr=now;
                buildptr(&p,row, mincol);
                for ( count=0,i=mincol; i <= maxcol; i++,p++){
                    if(*p==BLACK){
                        if(count==0){
                            if(++newlabel==sizeof(table)){
                                printf("\nnewlabel=%d
                                    size=%d",newlabel,sizeof(table));
                                printf("\nlist table too small");
                                return ;
                            }
                            if(linklist[now].next==startlist){
                                printf("\nCircular buffer too small");
                                return ;
                            }
                            linklist[now].label=index=newlabel;
                            table[newlabel]=newlabel;
                            linklist[now].start=low=i;
                            count++;
                        }
                    }
                    else if (*p==WHITE){
```

```
                    if(count==1){
                        linklist[now].end=high=i;
                        count=0;
                        ii=startlist;
                        if(table[linklist[now].label]==0){
                            printf("\rrow=%3d  label=%4d  newlabel=%d",
                                row,table[linklist[now].label],newlabel);
                        getch();
                        }
                        /* checking for connection */
                        while(ii!=startcurr){
                            if(!(linklist[ii].start > high || linklist[ii].end < low)){
                                list=table[index];
                                ilist=table[linklist[ii].label];
                                q=table;
                                if(list>ilist) table[index]=ilist;
                                else while(*++q)if(*q==ilist)*q=list;
                            }
                            ii=linklist[ii].next;
                        }
                        for(x=high;x>=low;x--)im_pixw(x,r,ilist);
                        now=linklist[now].next;
                    }
                }
            }
        startlist=startcurr;
        }
    }
    max=select(250,240);
    im_outmode(0);
    im_outpath(0,-1,0,0);
    im_opmode(0,0);
    r= 0;
    for (j = 0; j < 4; j++) {
        im_cpuwin(j);
        nrows = j == 3 ? 96 : 128;
        for (row = 0; row < nrows; row++, r++) {
            buildptr(&p,row, mincol);
            for ( count=0,i=mincol; i <= maxcol; i++,p++){
                if((*p==max)) {
                    im_pixw(i,r,0);
                }
                else if(*p!=max)im_pixw(i,r,255);
            }
        }
    }
    getch();
    _setvideomode(_DEFAULTMODE);
}

/* Build far pointer for accessing pixel data */

buildptr (ptr,row,col)
unsigned *ptr, row, col;
{
    *((unsigned *)ptr + 1) = MVPMEM;
    *((unsigned *)ptr)     = 512*row + col;
}


/* Performing keying operation for overlaying image on monitor */


void key(void)
{
        im_opmode(2,0);
        im_key(1);
}

/* Routine to select manually select specific blob after blob detection, returns grey level of blob selected */

int select(int x,int y)
{
char c;
int area;

        im_opmode(1,0);
        im_drawmode(1);
        im_setcolor(125);
        screen1(x,y);
        while((c=getche())!=13){
            switch(c) {
                  case ’8’ :y--;
                              printf("\r");
                              screen1(x,y);
                              break;
                  case ’2’ :y++;
                              printf("\r");
                              screen1(x,y);
                              break;
                  case ’4’ :x--;
                              printf("\r");
                              screen1(x,y);
                              break;
                  case ’6’ :x++;
                              printf("\r");
                              screen1(x,y);
                              break;
                  case ’H’ :y-=20;
                              printf("\r");
                              screen1(x,y);
                              break;
                  case ’P’ :y+=20;
                              printf("\r");
                              screen1(x,y);
                              break;
                  case ’K’ :x-=20;
                              printf("\r");
                              screen1(x,y);
                              break;
                  case ’M’ :x+=20;
                              printf("\r");
                              screen1(x,y);
                              break;
            }
        }
        im_opmode(1,0);
        area=im_pixr(x+3,y+3);
        return area;
}

/* Vision curser drawing */

void screen1(int x,int y)
{

        im_opmode(1,0);
        im_drawmode(1);
        im_move(x,y);
        im_line(x+20,y);
        im_move(x,y);
        im_line(x-20,y);
        im_move(x,y);
        im_line(x,y+20);
        im_move(x,y);
        im_line(x,y-20);
        im_outpath(0,-1,0,0);
        key();
}

/* Hough transform circle centre finding */

CENTRE hough(CENTRE john)
{
unsigned char far *p;
int i, j , r, row, nrows, mincol, maxcol;
double x,y,acnt,bcnt;
int amin,bmin,amax,bmax,ex,ey;
int a,b,c,arem,brem,hr;
unsigned int tran[RADIUS][XCOORD][YCOORD];
double sigma,theta;
int maxa,maxb,rmax,radmax,max;
double rad;

        r= 1;
        mincol = 0;
        maxcol = 511;
        _setvideomode(_VRES16COLOR);

        amax=john.x+10;
        bmax=john.y+10;
        amin=john.x-10;
        bmin=john.y-10;
        ex=john.x;
        ey=john.y;
        rad=john.r;
        printf("\nSTARTING HOUGH");
        for(a=0;a!=RADIUS;a++)
            for(b=0;b!=XCOORD;b++)
                for(c=0;c!=YCOORD;c++)
                        tran[a][b][c]=0;

        im_outmode(0);
        im_outpath(2, -1, 0, 0);
        im_opmode(0, 2);
        for (j = 0; j < 4; j++) {
            im_cpuwin(j);
            nrows = j == 3 ? 96 : 128;
            for (row = 1; row < nrows-1; row++, r++) {
                buildptr(&p,row, mincol);
                for ( i=mincol; i <= maxcol; i++,p++){
                    if(*p>50){
                        for(hr=0;hr<RADIUS;hr++){
                            if((ex!=i)||(ey!=r))sigma=atan((ey-i)/(ex-r));
                            else if(ex==i)sigma=(PI/2);
                            else if(ey==r)sigma=0;
                            for(theta=sigma-(PI/50); theta<=sigma+(PI/50);                    theta+=(PI*0.005)){
                                acnt=i + ((hr+rad)*(cos(theta)));
                                bcnt=r + ((hr+rad)*(sin(theta)));
                                a=acnt;
                                b=bcnt;
                                arem=acnt-a;
                                brem=bcnt-b;
                                if(arem>=0.5)a=a+1;
                                if(brem>=0.5)b=b+1;
                                if(arem<0.5)a=a;
                                if(brem<0.5)b=b;
                                if((a>amin) && (a<amax) && (b>bmin) && (b<bmax)){
                                    tran[hr][a-amin][b-bmin]++;
                                }
                            }
                        }
                    }
                }
            }
        }
        for(a=0;a!=RADIUS;a++){
            for(b=0;b!=XCOORD;b++){
                for(c=0;c!=YCOORD;c++){
                    if(tran[a][b][c]>=max){
                        radmax=a/2;
                        rmax=a;
                        maxa=b;
                        maxb=c;
                        max=tran[a][b][c];
                    }
                }
            }
        }
        printf("\nmax=%d",max);
}
```

PROGRAM EDGE_MEN.C

```
#include <menu.h>

int action3(int pos);
int action4(int pos);
void edge_menu(void);
void centre_menu(void);
void horedge(void);

static double p[3];          /* sums z, xz, yz .....  */
static double q[3];          /* evaluated polynomial COEFFICIENTS */
                                        /* NOTE: solution is [*A][q] = [p] */
static double L[3][3];        /* lower/upper triangular matrices */
static double U[3][3];
static double E[6];

static double *A[3][3]={      /* pointers to elements in E */
        &E[0], &E[1], &E[2],
        &E[1], &E[3], &E[4],
        &E[2], &E[4], &E[5],
};

void dodo(double,double);

CENTRE lsfcent(void);
void solve(double *coeff);
void initvars(void);
void matricise(void);

void sobel(void);
void kirsch(void);
void lp1(void);
void lp2(void);
void prewitt(void);
void horizontal(void);
```

/* Menu for edge detection technique election */

```
void edge_menu(void)
{
int checker=0;
static char *items[NUM3]=
       {  "SOBEL",
          "HORIZONTAL",
          "PREWITT",
          "LAPLACIAN 1",
          "LAPLACIAN 2",
          "EDGE DETECTING BUG",
          "CENTRE FIND",
          "RETURN TO MAIN",
          "QUIT", };
int curpos,code;

       printf(CLEAR);
       curpos=0;
       while(TRUE){
           display(items,NUM3,curpos);
           code=getcode();
           switch(code){
                case U_ARRO: if(curpos>0)--curpos;break;
                case D_ARRO: if(curpos<NUM3-1)++curpos;break;
                case INSERT: checker=action3(curpos);
                                      if(checker==1){system("cls");return;}
```

```
                                break;
                }
            }
}

/* Carry out menu selection */

int action3(int pos)
{
        printf(ERASE);
        switch(pos){
            case 0 : sobel();getch();system("cls");break;
            case 1 : horedge();system("cls");break;
            case 2 : prewitt();system("cls");break;
            case 3 : lp1();system("cls");break;
            case 4 : lp2();system("cls");break;
            case 5 : johnedge();system("cls");break;
            case 6 : centre_menu();system("cls");break;
            case 7 : return 1;system("cls");break;
            case 8 : _setvideomode(_DEFAULTMODE);exit(1);
        }
}

/* Menu for choice of centre finding technique */

void centre_menu(void)
{
int checker=0;
static char *items[NUM4]=
        {  "LSF",
            "HOUGH",
            "SEARCH WITH CURRENT R TABLE",
            "RETURN TO MAIN",
            "QUIT", };
int curpos,code;

        printf(CLEAR);
        curpos=0;
        while(TRUE){
            display(items,NUM4,curpos);
            code=getcode();
            switch(code){
                case U_ARRO: if(curpos>0)--curpos;break;
                case D_ARRO: if(curpos<NUM4-1)++curpos;break;
                case INSERT: checker=action4(curpos);
                                        if(checker==1){system("cls");return;}
                                            break;
            }
        }
}

/* Carry out menu selection */

int action4(int pos)
{
        CENTRE john;
        printf(ERASE);
        switch(pos){
            case 0 : lsf=lsfcent();printf("\nX=%f, Y=%f ",john.x,john.y);system("cls");break;
            case 1 : r_table();system("cls");break;
            case 2 : find();system("cls");break;
            case 3 : return 1;system("cls");break;
            case 4 : _setvideomode(_DEFAULTMODE);exit(1);
        }
}
```

```
void sobel(void)
{
        im_sobel(0,2,5);
        im_outpath(2,-1,0,0);
}

void prewitt(void)
{
        im_outpath(2,-1,0,0);
        im_prewitt(0,2,5);
}

void kirsch(void)
{
        im_outpath(2,-1,0,0);
        im_kirsch(0,2,5);
}

void lp1(void)
{
        im_outpath(2,-1,0,0);
        im_lp1(0,2);
}

void lp2(void)
{
        im_outpath(2,-1,0,0);
        im_lp2(0,2);
}

/* Least Squares Fit centre finding */

CENTRE lsfcent(void)
{
CENTRE john;
int i,n;
double xin,Cx, Cy,Rad2,Rad;

        initvars();
        matricise();
        solve(q);
        Cx = q[1]/2.0;
        Cy = q[2]/2.0;
        Rad2= q[0] + Cx*Cx + Cy*Cy;
        Rad=sqrt(Rad2);
        john.x=Cx;
        john.y=Cy;
        john.r=Rad;
        im_opmode(1,2);
        im_setcolor(250);
        im_drawmode(4);
        im_move((int)john.x,(int)john.y);
        im_ellipse((int)(Rad*3.0/4.05),(int)(Rad*1.07));
        getch();
        im_outpath(2,-1,0,0);
        _setvideomode(_DEFAULTMODE);
        printf("\ncentre: x=%f,\ty=%f",john.x,john.y);
        getch();
        return(john);
}

/* Initialise variable for LSF */

void initvars(void)
{
        int i;
        for(i=0; i<6; i++)  E[i] = 0.0;
        for(i=0; i<3; i++)  p[i] = 0.0;
}

/* Enter pixel values in matrices */

void matricise(void)
{
unsigned char far *p;
int i, j , r, row, nrows, mincol, maxcol;
double x,y;

       r= 1;
       mincol = 0;
       maxcol = 511;
       initvars();
       im_outmode(0);
       im_outpath(0, -1, 0, 0);
       im_opmode(0, 0);
       for (j = 0; j < 4; j++) {
           im_cpuwin(j);
           nrows = j == 3 ? 96 : 128;
           for (row = 1; row < nrows-1; row++, r++) {
                buildptr(&p,row, mincol);
                for ( i=mincol; i <= maxcol; i++,p++){
                     if(*p==125){
                                 x =(double)i;
                                 y =(double)r;
                                 dodo(x,y);
                     }

                }
           }
       }
}

void dodo(double x, double y)
{
        double  z, x2, y2;

             x2 = x*x;
             y2 = y*y;
             z  = x2+y2;
             E[0] += 1.0;
             E[1] += x;
             E[2] += y;
             E[3] += x2;
             E[4] += x*y;
             E[5] += y2;

             p[0] += z;
             p[1] += z*x;
             p[2] += z*y;
}


void solve(double *coeff)
{
int i,j,k,size;
double t;

        size=3;

        for(i=0; i<size; i++){
```

```
        L[i][i]=1.0;
        coeff[i]=p[i];
        for(j=0; j<size; j++)     U[i][j]=*A[i][j];
    }
    for(k=0; k<size-1; k++){
        for(i=k+1; i<size; i++){
            t = L[i][k] = U[i][k]/U[k][k];
            for(j=k; j<size; j++)   U[i][j] -= t*U[k][j];
        }
    }
    for(i=1; i<size; i++)
        for(j=0; j<=i-1; j++)     coeff[i] -= L[i][j]*coeff[j];
    for(i=size-1; i>=0; i--){
        for(j=i+1; j<size; j++)   coeff[i] -= U[i][j]*coeff[j];
        coeff[i] /=U[i][i];
    }
}


void horedge(void)
{
    unsigned char far *p;
    int i,ii,jj,j , r, row, nrows, ncols,mincol, maxcol;
    int maxrow,rad,minrow,col,avch;
    int count=0;


    r= 0;
    maxrow = 480;
    mincol = 0;
    maxcol = 511;
    minrow=0;


    _setvideomode(_VRES16COLOR);
    im_outmode(0);
    im_outpath(0, -1, 0, 0);
    im_opmode(0, 0);
    for (j = 0; j < 4; j++) {
        im_cpuwin(j);
        nrows = j == 3 ? 96 : 128;
        for (row = 0; row < nrows; row++, r++) {
            buildptr(&p,row, mincol);
            for ( i=mincol; i <= maxcol; i++,p++){
                if(*p == BLACK){
                    if(count==0){
                        _setpixel(i,r*0.8);
                        for(ii=-1;ii<1;ii++){
                            im_pixw(i+ii,r,125);
                        }
                        count++;
                    }
                }
                else{
                    if(count==1){
                        _setpixel(i,r*0.8);
                        for(ii=-1;ii<1;ii++){
                            im_pixw(i+ii,r,125);
                        }
                        count=0;
                    }
                }
            }
        }
    }
```

```
        getch();
        printf("\nDo You wish to Average (smooth/filter) thresholded image?(y or n)");
        avch=getch();
            if(avch=='y'||avch=='Y'){
                printf("\nPerforming averaging");
                average1();
                system("cls");
            }
        _setvideomode(_DEFAULTMODE);
}

/* Build far pointer for vertical edge detection */

buildptrc (ptr,row,col)
unsigned *ptr, row, col;
{
        *((unsigned *)ptr + 1) = MVPMEM;
        *((unsigned *)ptr)     = 512*col + row;
}

/* Vertical edge detection */

void vertedge(void)
{
unsigned char far *p;
int i, j , r, row, nrows, ncols,mincol, maxcol;
int maxrow,rad,minrow,col;
int count=0;

        r= 0;
        maxrow = 480;
        mincol = 0;
        maxcol = 511;
        minrow=0;

        _setvideomode(_VRES16COLOR);
        im_outmode(0);
        im_outpath(0, -1, 0, 0);
        im_opmode(0, 0);
        i= 0;
        count=0;
        for (j = 0; j < 4; j++) {
            im_cpuwin(j);
            ncols = 128;
            for (col=0 ; i < ncols; col++,i++){
                for (r = 0; r < maxrow;  r++) {
                buildptrc(&p, minrow,col);
                    if(*p == BLACK){
                        if(count==0){
                            _setpixel(i,r*0.8);
                            im_pixw(i,r,125);
                            count++;
                        }
                    }
                    else{
                        if(count==1){
                            _setpixel(i,r*0.8);
                            im_pixw(i,r,125);
                            count=0;
                        }
                    }
                }
            }
        }
        getch();
        _setvideomode(_DEFAULTMODE);
}
```

PROGRAM HOUGH.C

```
#include <menu.h>
#include <math.h>
#include <float.h>


#define RIGHT 0
#define LEFT  1
#define PREVIOUS 2

unsigned char accum[256][256];
char filespec[]="accum.dat";

void fileops(void);
void filexwrite(void);
void filexread(void);
double boundary_angle(int,int);
void r_table(void);
int search(int);
int find2 (int);
void drawaccum(void);
void freeman (int);
POINT start(int);
void show_search(void);
void show_2(void);
void show_3(int, int);

R_TABLE ace[INCREMENTS];

/* Calculate boundary angle for R-Table */

double boundary_angle(int x,int y)
{
int y1,y2,i,j,avcount=0;
int xavi=0,yavi=0;
int xlowlimit,xhighlimit;
int ylowlimit,yhighlimit;
double xav, yav, sigbot,sigtop,fi,a0,a1;

    if(x<7){xlowlimit=0;xhighlimit=6;}
    else if((x>=7)&&(x<505)){xlowlimit=-6;xhighlimit=6;}
    else if(x>=505){xlowlimit=6;xhighlimit=0;}
    if(y<7){ylowlimit=0;yhighlimit=6;}
    else if((y>=7)&&(y<473)){ylowlimit=-6;yhighlimit=6;}
    else if(y>=473){ylowlimit=6;yhighlimit=0;}

    for(i=xlowlimit;i<=xhighlimit;i++){
        for(j=ylowlimit;j<=yhighlimit;j++){
            if(im_pixr(x+i,y+j)==125){
                xavi=xavi+(x+i);
                yavi=yavi+(y+j);
                avcount++;
            }
        }
    }
    if(avcount==0)return 0;
    xav=(double)xavi/(double)avcount;
    yav=(double)yavi/(double)avcount;

    sigtop=sigbot=0.0;
    for(i=-6;i<=6;i++){
        for(j=-6;j<=6;j++){
            if(im_pixr(x+i,y+j)==125){
                sigtop=sigtop+(((double)(x+i)-xav)*((double)(y+j)-yav));
                sigbot=sigbot+pow((double)(x+i)-xav,2.0);
            }
        }
    }
    if((sigbot!=0.0)&&(sigtop!=0.0)){
        a1=sigtop/sigbot;
        a0=yav-(a1*xav);
        fi=atan(a1);
        _setcolor(__BLUE);
        y1=(a1*(double)(x+30))+a0;
      /* _moveto(x+30,y1*0.75);*/
        y2=(a1*(double)(x-30))+a0;
        /*_lineto(x-30,y2*0.75);    */
        _setcolor(__RED);
    }

    else if((sigbot==0.0)&&(sigtop!=0.0))fi=PI/2.0;
    else if((sigbot!=0.0)&&(sigtop==0.0))fi=0.0;

    return fi;
}

/* Produce R_table */

void r_table(void)
{
double rad,radsq,beta,omegaf,ommax,ommin,xbit,ybit;
int omega,entry,entries;
unsigned char far *p;
int count,i,j,r,row,nrows,mincol,maxcol;
int xref,yref;
int rtx,rty,n,m;
double power;

    r= 0;
    entries=0;
    xref=lsf.x-10;
    yref=lsf.y-10;
    mincol = 10;
    maxcol = 500;
    power=2.0;
    ommin=ommax=0.0;

    _setvideomode(_VRES16COLOR);
    _setcolor(__RED);

    for(i=0;i<INCREMENTS;i++){
        ace[i].entry=0;
        for(j=0;j<10;j++){
            ace[i].r[j]=0.0;
            ace[i].alpha[j]=0.0;
        }
    }
    im_outmode(0);
    im_outpath(0, -1, 0, 0);
    im_opmode(0, 0);
    for (j = 0; j < 4; j++) {
        im_cpuwin(j);
        nrows = j == 3 ? 96 : 128;
        for (row = 0; row < nrows; row+=12, r+=12) {
            buildptr(&p,row, mincol);
            for ( i=mincol; i <= maxcol; i++,p++){
                if(*p ==125){
                    _setpixel(i,r*0.75);
                    omegaf=boundary_angle(i,r);
                    if(omegaf<ommin)ommin=omegaf;
                    if(omegaf>ommax)ommax=omegaf;
                    omegaf=((omegaf+PI/2)*(180/PI))/4;/* was omegaf +PI/2*/
                    omega=omegaf;
                    radsq=pow(i-xref,power)+pow(r-yref,power);
                    rad=sqrt(radsq);
                    if    ((i<xref)&&(r<yref))beta=-acos((double)(i-xref)/rad);
                    else if((i>xref)&&(r<yref))beta=-acos((double)(i-xref)/rad);
                    else if((i<xref)&&(r>yref))beta=acos((double)(i-xref)/rad);
                    else if((i>xref)&&(r>yref))beta=acos((double)(i-xref)/rad);
                    entry=ace[omega].entry;
                    ace[omega].r[entry]=rad;
                    ace[omega].alpha[entry]=beta;
                    ace[omega].entry++;
                    if(ace[omega].entry> =TAB_ENTS)printf("\nOUTOF BOUNDS");
                    entries++;
                }
            }
        }
    }
    printf("\rEntries= %d, Ω max= %f, Ω min= %f",entries, ommax, ommin);
    getch();
    _setcolor(__BRIGHT_WHITE);
    im_opmode(1,0);
    im_drawmode(1);
    for(i=0;i<INCREMENTS;i++){
        for(j=0;j<ace[i].entry;j++){
            ybit=ace[i].r[j]*sin(ace[i].alpha[j]);
            xbit=ace[i].r[j]*cos(ace[i].alpha[j]);
            rtx=xref+xbit;
            rty=yref+ybit;

            /*if(xbit<0)rtx=xbit+xref;
            else if(xbit> =0)rtx=xbit-xref;
            if(ybit<0)rty=ybit+yref;
            else if(ybit> =0)rty=ybit-yref;*/

            im_move(rtx,rty);
            for(n=0;n< =3;n++){
                        im_circle(n);
            }
            _ellipse(_GFILLINTERIOR,rtx-2,(rty-2)*0.75,rtx+2,(rty+2)*0.75);
        }
    }
    getch();
    _setvideomode(_DEFAULTMODE);
    for(i=0,r=0;i<5;i++ ){
        for(j=0;j<9;j++ ){
            printf("%3d",ace[r].entry);
            r++;
        }
        printf("\n");
    }
    shape=search(0);
    getch();
}

/* Search for object defined by R-table */

int search(int bit)
{
int xref,yref;
double xbit,ybit;
double xcd,ycd;
int rtx,rty,n,m,radinc;
```

```
double omegaf,xpos,ypos;
int omega,yorn;
char choice;
unsigned char far *p;
int xp,yp,count,i, j , r, row, nrows, mincol, maxcol;
int x_min,y_min;
int x_max,y_max,maxacc;
double ratio;
double x,y;

    x_min=lsf.x-128;
    y_min=lsf.y-128;
    mincol = 10;
    maxcol = 500;
    for(i=0;i<256;i++ )
        for(j=0;j<256;j++ )accum[i][j]=0;

    _setvideomode(_VRES16COLOR);
    _setcolor(__DARK_GRAY);
    im_outmode(0);
    im_outpath(0, -1, 0, 0);
    im_opmode(0, 0);
    r=0;
    for (j = 0; j < 4; j++) {
        im_cpuwin(j);
        nrows = j == 3 ? 96 : 128;
        for (row=0; row < nrows; row++, r++) {
            buildptr(&p,row, mincol);
            for ( i=mincol; i <= maxcol; i++,p++){
                if(*p ==125){
                            _setpixel(i,r*0.75);
                }
            }
        }
    }
    getch();
    _setcolor(__MAGENTA);
    r=0;

    for (j = 0; j < 4; j++) {
        im_cpuwin(j);
        nrows = j == 3 ? 96 : 128;
        for (row=0; row < nrows; row++, r++) {
            buildptr(&p,row, mincol);
            for ( i=mincol; i <= maxcol; i++,p++){
                if(*p ==125){
                    _setpixel(i,r*0.75);
                    omegaf=boundary_angle(i,r);
                    omegaf=((omegaf+PI/2)*(180/PI))/4;
                    omega=omegaf;
                    if(omega<0.0)omega=-omega;
                    if ((ace[omega].entry!=0)&&(omega<45)&&(omega>=0)){
                        for(m=0;m<=ace[omega].entry;m++){
                            for(radinc=-5;radinc<=5;radinc++){
                                xpos=(double)i+((ace[omega].r[m]+radinc)*cos(ace[omega].alpha[m]));
                                ypos=(double)r+((ace[omega].r[m]+radinc)*sin(ace[omega].alpha[m]));
                                xp=xpos;
                                yp=ypos;
                                if((xp>x_min)
                                    &&(xp<x_min+256)
                                        &&(yp>y_min)
                                            &&(yp<y_min+256)){
                                                accum[xp-x_min][yp-y_min]++;
                                    }
                                xpos=i-(ace[omega].r[m]*cos(ace[omega].alpha[m]));
                            ypos=r-(ace[omega].r[m]*sin(ace[omega].alpha[m]));
                            xp=xpos;
                            yp=ypos;
                            if((xp>x_min)
                                &&(xp<x_min+256)
                                    &&(yp>y_min)
                                        &&(yp<y_min+256)){
                                            accum[xp-x_min][yp-y_min]++;
                                            /*  _setcolor(__BRIGHT_WHITE);
                                            _setpixel(xp,yp*0.75);
                                            _setcolor(__RED);*/
                            }
                        }
                    }
                }
            }
        }
    }
}
fileops();
x_max=0;
y_max=0;
maxacc=0;
for(i=0;i<256;i++ ){
    for(j=0;j<256;j++ ){
        if(accum[i][j]>maxacc){
            maxacc=accum[i][j];
            x_max=i;
            y_max=j;
        }
    }
}
xref=x_min+x_max;
yref=y_min+y_max;
printf("\nXref should be %.3f (%d), Yref should be %.3f (%d) number=%d",lsf.x,xref,lsf.y,yref,maxacc);
yorn=0;
_setcolor(__LIGHT_RED);
if(bit==1)yorn=find2(maxacc);
if(yorn==1){
    _setcolor(__WHITE);
    im_opmode(1,0);
    im_drawmode(1);
    for(i=0;i<INCREMENTS;i++){
        for(j=0;j<ace[i].entry;j++){
            ybit=ace[i].r[j]*sin(ace[i].alpha[j]);
            xbit=ace[i].r[j]*cos(ace[i].alpha[j]);
            rtx=xref+xbit;
            rty=yref+ybit;
            im_move(rtx,rty);
            for(n=0;n<=3;n++){
                im_circle(n);
            }
            _ellipse(_GFILLINTERIOR,rtx-2,(rty-2)*0.75,rtx+2,(rty+2)*0.75);
        }
    }
}
getch();
drawaccum();
getch();
_setvideomode(_DEFAULTMODE);
for(i=-12;i<12;i++){
    for(j=-12;j<12;j++){
        printf("%3d",accum[x_max+i][y_max+j]);
    }
    printf("\n");
```

```
        }
        getch();
        return maxacc;
}

/* Decide if object is present */

void find (void)
{
        int check;
        double ratio;

        check=search(1);
        ratio=(double)check/(double)shape;
        printf("\n shape= %d check= %d s/c= %f",shape,check,ratio);
        getch();
        if(check<(shape*0.55)){
             printf("\nobject not found");
             printf("\n Press a key to continue");
             getch();
        }
        else {
             printf("\nobject found, Congratulations!");
             printf("\n Press a key to continue");
             getch();
        }
}

int find2 (int check)
{
double ratio;

        ratio=(double)check/(double)shape;
        if(check<(shape*0.55)){
             return 0;
        }
        else {
             return 1;
        }
}



/* Save/retrieve Hough accumulator array */

void fileops(void)
{
  char c;

  printf("Save/Retrieve : ");
  fflush(stdin);
  c=getchar();
  if (c=='s' || c=='S') {filexwrite();}
  else if (c=='r' || c=='R') {filexread();}
  else return ;
}

/* Save accumulator array */

void filexwrite(void)
{
        FILE *stream;
        char fname[34];
        unsigned int count[1];
        unsigned int numwritten,i,j,m,n,nw;
        char *p =(char *)&accum[0][0];

        strcpy (fname, path);
        strcat (fname, "\\");
        strcat (fname, filespec);

        if((stream = fopen(fname, "w+b")) == NULL){
          printf("Could not open new file %s to write\n", fname);
          getch();
          return;
        }
        numwritten=0;
        for(i=0;i<256;i++)numwritten=numwritten+fwrite(accum[i],256,1, stream);
        printf("\nNumber written  %d ",numwritten);
        printf("\nNumber written  %u ",(255*numwritten)+255);
        getch();
        fclose(stream);
}

/* Retrieve accumulator array */

void filexread(void)
{
        FILE *stream;
        char fname[34];

        strcpy (fname, path);
        strcat (fname, "\\");
        strcat (fname, filespec);

        stream = fopen(fname, "r+b");
        fread(accum, sizeof(accum),1 , stream);
        fclose(stream);
}

/* Draw Hough accumulator array */

void drawaccum(void)
{
double xcd,ycd,vald,x,y;
int i,j;

        _setvideomode(_VRES16COLOR);
        _setcolor(__RED);
        _setlogorg(50,350);
        _moveto(0,0);
        x=cos(PI/6)*256;
        y=sin(PI/6)*256;
        _lineto(256,0);
        _moveto(256,0);
        _lineto(x+256,y);
        _moveto(0,0);
        _lineto(x,y);
        _moveto(x,y);
        _lineto(x+256,y);
        _setcolor(__BLUE);
        _setpixel(0,0);
        getch();
        for(i=0;i<256;i++ ){
             for(j=1;j<256;j++ ){
                          ycd=(-(double)accum[i][j])+(sin(PI/6)*(double)j);
                          xcd=(double)i+(cos(PI/6)*(double)j);
                          _setcolor((accum[i][j]/16))+1;
                          if(accum[i][j]!=0)_setpixel(xcd,ycd);
        }
    }
}

/* Find first point of object for edge tracking */

POINT start(int colour)
{
unsigned char far *p;
int xp,yp,count,i, j , r, row, nrows, mincol, maxcol;
POINT first_point;

    mincol = 0;
    maxcol = 511;

    _setcolor(__GREEN);
    im_outmode(0);
    im_outpath(0, -1, 0, 0);
    im_opmode(0, 0);
    r=0;
    for (j = 0; j < 4; j++) {
        im_cpuwin(j);
        nrows = j == 3 ? 96 : 128;
        for (row=0; row < nrows; row++, r++) {
            buildptr(&p,row, mincol);
            for ( i=mincol; i <= maxcol; i++,p++){
                if(*p ==colour){
                    _setpixel(i,r*0.75);
                }
            }
        }
    }
  r=0;
  for (j = 0; j < 4; j++) {
        im_cpuwin(j);
        nrows = j == 3 ? 96 : 128;
        for (row=0; row < nrows; row++, r++) {
            buildptr(&p,row, mincol);
            for ( i=mincol; i <= maxcol; i++,p++){
                if(*p ==BLACK){
                    first_point.x=i;
                    first_point.y=r;
                    _setcolor(__BRIGHT_WHITE);
                    _setpixel(i,r*0.75);
                    getch();
                    return(first_point);
                }
            }
        }
    }
}

/* Function for carrying out Papert's turtle blob detection */

void turtle(void)
{
    POINT first_point;
    unsigned int x,y,i,j,cd=0;
    unsigned int x0,y0,xtemp,ytemp;
    int turn;
    int lastturn;

    _setvideomode(_VRES16COLOR);
    _setcolor(__BRIGHT_WHITE);
    first_point=start(BLACK);
    x=x0=first_point.x;
    y=y0=first_point.y;
    _setpixel(x,y*0.75);
    x++;
    y++;
    lastturn=LEFT;
    _setcolor(__RED);
    /*for(i=0;i<20000;i++)                    */
    while((y!=first_point.y))/*&&(x!=first_point.x))*/

    {
        if(im_pixr(x,y)==BLACK){   /* Turn left and step */
            turn=LEFT;
        /*  printf("\nLEFT");*/
        }
        else if(im_pixr(x,y)==WHITE){   /* Turn left and step */
            turn=RIGHT;
            /*printf("\nRIGHT");*/
        }
        _setpixel(x,y*0.75);
        xtemp=x;
        ytemp=y;

        if      (turn==RIGHT) {
            y++;
            x--;
            printf("1 R\t");
        }
        else if (turn==LEFT)  {
            y++;
            x++;
            printf("1 L\t");
        }

        x0=xtemp;
        y0=ytemp;
        _setpixel(x,y*0.75);
        printf("x=%d (%d) y=%d (%d)\r",x,first_point.x,y,first_point.y);
    }
    getch();
    _setvideomode(_DEFAULTMODE);
}

/* Novel Edge Tracker */

void johnedge(void)
{
int checker[8],tempck[8],xtemp,ytemp;
int i,j,ii,jj,x,y,x0,y0,np,ck,ooh,npc;
POINT first_point;
int colour,chick,cd=0;
int points=0;
unsigned int direction[3][3];
char code[1000];

    direction[0][0]=3;
    direction[0][1]=4;
    direction[0][2]=5;
    direction[1][0]=2;
    direction[1][1]=8;
    direction[1][2]=6;
    direction[2][0]=1;
    direction[2][1]=0;
    direction[2][2]=7;
    _setvideomode(_VRES16COLOR);
    _setcolor(__YELLOW);


    first_point=start(BLACK);
    _setcolor(__RED);
    x=first_point.x-1;
    y=first_point.y;
    _setpixel(x,y*0.75);
    x0=x;
    y0=y-1;
    /*show_search();*/
    getch();

    do{
        xtemp=x; /* Temporary store for search position */
        ytemp=y;
        ii=x0-x; /* Calculate relative position of previous search position */
        jj=y0-y; /* was x-x0 and y-y0 */
        /* Check status of surrounding pixels */
        show_search();
        for(i=-1,ck=0;i<=1;i++){
            for(j=-1;j<=1;j++){
                if((x+i)<0||(x+i)>511||(y+j)<0||(y+j)>480)colour=255;
                    /* Assigns white value to points choosen out of frame buffer range */
                else colour=im_pixr(x+i,y+j);

                if(colour==WHITE||colour==125||colour==150){
                    tempck[ck]=1;
                    _rectangle(_GFILLINTERIOR,
                            75+(25*i),75+(25*j),100+(25*i),100+(25*j));
                    if((i==ii)&&(j==jj)){
                        tempck[ck]=PREVIOUS;
                        _setcolor(__RED);
                        _rectangle(_GFILLINTERIOR,
                            75+(25*i),75+(25*j),100+(25*i),100+(25*j));
                        _setcolor(__BRIGHT_WHITE);
                    }
                }
                else if(colour==50){
                    tempck[ck]=3;
                    _setcolor(__YELLOW);
                    _rectangle(_GFILLINTERIOR,
                            75+(25*i),75+(25*j),100+(25*i),100+(25*j));
                    _setcolor(__BRIGHT_WHITE);
                }
                else if(colour==25){
                    tempck[ck]=4;
                    _setcolor(__MAGENTA);
                    _rectangle(_GFILLINTERIOR,
75+(25*i),75+(25*j),100+(25*i),100+(25*j));
                    _setcolor(__BRIGHT_WHITE);
                }
                else if(colour==BLACK){
                    tempck[ck]=0;
                    if((i==ii)&&(j==jj)){
                        tempck[ck]=PREVIOUS;
                    }
                }
                if((i==0)&&(j==0)){
                    _setcolor(__BLUE);
                    _rectangle(_GFILLINTERIOR,
75+(25*i),75+(25*j),100+(25*i),100+(25*j));
                    _setcolor(__BRIGHT_WHITE);
                    ck--;
                }
                ck++;
            }
        }
        show_2();

/*      tempck =  0 3 5    checker =    0 1 2                    */
/*                1   6                 7   3              */
/*                2 4 7                 6 5 4               */

        checker[0]=tempck[0];
        checker[1]=tempck[3];
        checker[2]=tempck[5];
        checker[3]=tempck[6];
        checker[4]=tempck[7];
        checker[5]=tempck[4];
        checker[6]=tempck[2];
        checker[7]=tempck[1];
        printf("\r");
        for(i=0;i<8;i++){
            printf("%d",checker[i]);
        }
        i=npc=0;
        /* Count direction options (npc) */
        while(i<8){
            if(checker[i]==1){
                if(i==0){
                    if((checker[7]==0)||(checker[i+1]==0)){npc++;}
                }
                else if((checker[i+1]==0)||(checker[i-1]==0)){npc++;}
            }
            i++;
        }
        /* Choose direction to go (np) */
        i=0;
        while(i<8){
            if(checker[i]==1){
                if(i==0){
                    if((checker[7]==0)||(checker[i+1]==0)){np=0;i=8;}
                }
                else if((checker[i+1]==0)||(checker[i-1]==0)){np=i;i=8;}
            }
            i++;
        }
        _setcolor(__LIGHT_MAGENTA);
        /* Increment edge bug according to np value */
        if    (np==0){
            code[cd]=0;
            x--;y--;i=-1;j=-1;
            if(im_pixr(x,y)==BLACK||im_pixr(x,y)==125)   {                    im_pixw(x,y,50);x++;y++;cd--;
            }
        }
        else if(np==1){
            code[cd]=1;
            y--;i=0;j=-1;
            if(im_pixr(x,y)==BLACK||im_pixr(x,y)==125)   {
                im_pixw(x,y,50);y++;cd--;
            }
        }
        else if(np==2){
            code[cd]=2;
            x++;y--;i=1;j=-1;
            if(im_pixr(x,y)==BLACK||im_pixr(x,y)==125)   {
                im_pixw(x,y,50);x--;y++;cd--;
            }
```

```
    }
    else if(np==3){
        code[cd]=3;
        x++;i=1;j=0;
        if(im_pixr(x,y)==BLACK||im_pixr(x,y)==125)  {
            im_pixw(x,y,50);x--;cd--;
        }
    }
    else if(np==4){
        code[cd]=4;
         x++;y++;i=1;j=1;
        if(im_pixr(x,y)==BLACK||im_pixr(x,y)==125)  {
            im_pixw(x,y,50);x--;y--;cd--;
        }
    }
    else if(np==5){
        code[cd]=5;
        y++;i=0;j=1;
        if(im_pixr(x,y)==BLACK||im_pixr(x,y)==125)  {
            im_pixw(x,y,50);y--;cd--;
        }
    }
    else if(np==6){
        code[cd]=6;
        x--;y++;i=-1;j=1;
        if(im_pixr(x,y)==BLACK||im_pixr(x,y)==125)  {
            im_pixw(x,y,50);x++;y--;cd--;
        }
    }
    else if(np==7){
        code[cd]=7;
        x--;i=-1;j=0;
        if(im_pixr(x,y)==BLACK||im_pixr(x,y)==125)  {
            im_pixw(x,y,50);x++;cd--;
        }
    }
    _rectangle(_GFILLINTERIOR,75+(25*i),75+(25*j),100+(25*i),100+(25*j));
    _setcolor(_BRIGHT_WHITE);

    _setpixel(x,y*0.75);
    if(npc<=1)im_pixw(x,y,125);
    else if(npc>1)im_pixw(x,y,150);
    x0=xtemp;
    y0=ytemp;
    show_3(x,y);
    points++;
    cd++;

} while((x!=first_point.x-1)||(y!=first_point.y));
show_3(x,y);
getch();
printf("\nPress a key to reconstitute object from chain code");
getch();
_setcolor(1);

x=first_point.x-1;
y=first_point.y;
for(i=0;i<cd;i++){
    if(code[i]==0){x--;y--;}
    else if(code[i]==1){y--;}
    else if(code[i]==2){x++;y--;}
    else if(code[i]==3){x++;}
    else if(code[i]==4){x++;y++;}
    else if(code[i]==5){y++;}
    else if(code[i]==6){x--;y++;}
```

```
            else if(code[i]==7){x--;}
            _setpixel(x,y*0.75);
        }
        getch();
        /*printf("\nDo you wish to code? y or n ");
        chick=getch();
        if(chick=='y'||chick=='Y')freeman(points);*/
        _setvideomode(_DEFAULTMODE);
}
```

/ Show graphically Novel edge tracker esearching for edge */*

```
void show_search(void)
{
        _setcolor(__BLACK);
        _rectangle(_GFILLINTERIOR,50,50,125,125);
        _setcolor(__RED);
        _moveto(50,50);
        _lineto(125,50);
        _moveto(125,50);
        _lineto(125,125);
        _moveto(125,125);
        _lineto(50,125);
        _moveto(50,125);
        _lineto(50,50);

        _moveto(75,50);
        _lineto(75,125);
        _moveto(100,50);
        _lineto(100,125);
        _moveto(50,75);
        _lineto(125,75);
        _moveto(50,100);
        _lineto(125,100);
        _setcolor(__BRIGHT_WHITE);

}
```

/ Show graphically Novel edge tracker esearching for edge */*

```
void show_2(void)
{
        _setcolor(__RED);
        _moveto(50,50);
        _lineto(125,50);
        _moveto(125,50);
        _lineto(125,125);
        _moveto(125,125);
        _lineto(50,125);
        _moveto(50,125);
        _lineto(50,50);

        _moveto(75,50);
        _lineto(75,125);
        _moveto(100,50);
        _lineto(100,125);
        _moveto(50,75);
        _lineto(125,75);
        _moveto(50,100);
        _lineto(125,100);
        _setcolor(__BRIGHT_WHITE);

}
```

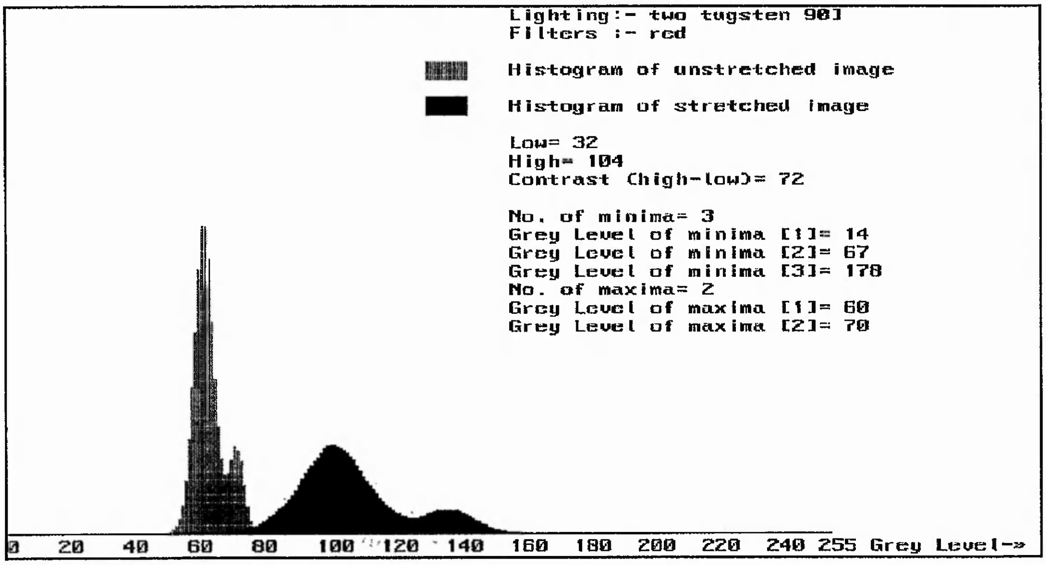
```
/* Smooth object edge */

void average1 (void)
{
    int i,j,x,y;
    int average,colour;

    _setvideomode(_VRES16COLOR);
    _setcolor(__BRIGHT_WHITE);


    for(x=0;x<511;x++){
       for(y=0;y<479;y++){
        if(im_pixr(x,y)==125||im_pixr(x,y)==0){
              for(i=-1,average=0;i<=1;i+=1){
                 for(j=-1;j<=1;j+=1){
                    if((x+i)<0||(x+i)>511||(y+j)<0||(y+j)>480)colour=255;
                       /* If point is out off frame buffer range */
                    else colour=im_pixr(x+i,y+j);
                    average=average+colour;
                 }
              }
              if(average>=1350)im_pixw(x,y,255);/* was 1350 */
              else if(average<1350){im_pixw(x,y,0);/*_setpixel(x,y*0.75);*/}
          }
       }
    printf("\rX=%3d, Y=%3d",x,y);
    }
    getch();
    _setvideomode(_DEFAULTMODE);

}

/* Smooth object edge */

void average2 (void)
{
    int i,j,x,y;
    int average,colour;

    _setvideomode(_VRES16COLOR);
    _setcolor(__BRIGHT_WHITE);

    for(x=0;x<511;x++){
       for(y=0;y<479;y++){
              for(i=-1,average=0;i<=1;i+=1){
                 for(j=-1;j<=1;j+=1){
                    if((x+i)<0||(x+i)>511||(y+j)<0||(y+j)>480)colour=255;
                       /* If point is out off frame buffer range */
                    else colour=im_pixr(x+i,y+j);
                    average=average+colour;
                 }
              }
              if(average>=1150)im_pixw(x,y,255);
              else if(average<1150){im_pixw(x,y,0);_setpixel(x,y*0.75);}
          }
       printf("\rX=%3d, Y=%3d",x,y);
    }
    getch();
    _setvideomode(_DEFAULTMODE);

}

/* Show graphically Novel edge tracker esearching for edge */
```

```
void show_3(int x,int y)
{
int i,j,colour;

      for(i=-7;i<=7;i++){
          for(j=-7;j<=7;j++){
               colour=im_pixr(x+i,y+j);
               if(colour==BLACK){
                    _setcolor(__BLUE);
                    _rectangle(_GFILLINTERIOR,
                              250+(10*(i+7)),250+(10*(j+7)),260+(10*(i+7)),260+(10*(j+7)));
               }
               else if(colour==WHITE){
                    _setcolor(__BRIGHT_WHITE);
                    _rectangle(_GFILLINTERIOR,
                              250+(10*(i+7)),250+(10*(j+7)),260+(10*(i+7)),260+(10*(j+7)));
               }
               else if(colour==50){
                    _setcolor(__YELLOW);
                    _rectangle(_GFILLINTERIOR,
                              250+(10*(i+7)),250+(10*(j+7)),260+(10*(i+7)),260+(10*(j+7)));
               }
               else if(colour==125||colour==150){
                    _setcolor(__GREEN);
                    _rectangle(_GFILLINTERIOR,
                              250+(10*(i+7)),250+(10*(j+7)),260+(10*(i+7)),260+(10*(j+7)));
               }
               else if(colour==25){
                    _setcolor(__MAGENTA);
                    _rectangle(_GFILLINTERIOR,
                              250+(10*(i+7)),250+(10*(j+7)),260+(10*(i+7)),260+(10*(j+7)));
               }
          }
      }
      _setcolor(__RED);
      _rectangle(_GFILLINTERIOR,250+(10*7),250+(10*7),260+(10*7),260+(10*7));
      _setcolor(__BRIGHT_WHITE);

}


/* Produce chain code for edge */

void freeman (int points)
{
      char *code;
      POINT first_point;
      unsigned int x,y,i,j,cd=0;
      unsigned int direction[3][3];

      printf("\nPoints = %d ",points);
      getch();
      code=malloc(points);
      direction[0][0]=3;
      direction[0][1]=4;
      direction[0][2]=5;
      direction[1][0]=2;
      direction[1][1]=8;
      direction[1][2]=6;
      direction[2][0]=1;
      direction[2][1]=0;
      direction[2][2]=7;

      first_point=start(125);
      x=first_point.x;
```

```
        y=first_point.y;
        for(i=-1;i<=1;i++){
            for(j=-1;j<=1;j++){
                if(im_pixr(x+i,y+j)==125){
                    *code=direction[i+1][j+1];
                    code++;
                }
            }
        }
        for(i=0;i<points;i++){
            printf("\n%d",code);
            code++;
        }
}

/* Perform average on function using average mask */

void average3 (void)
{
        int i,j,x,y;
        int average,colour;
        double avcol;

        _setvideomode(_VRES16COLOR);
        _setcolor(__BRIGHT_WHITE);

        for(x=0;x<511;x++){
            for(y=0;y<479;y++){
                        for(i=-1,average=0;i<=1;i+=1){
                            for(j=-1;j<=1;j+=1){
                                if((x+i)<0||(x+i)>511||(y+j)<0||(y+j)>480)colour=255;
                                    /* If point is out off frame buffer range */
                                else colour=im_pixr(x+i,y+j);
                                average=average+colour;
                            }
                        }
                        avcol=(double)average/9.0;
                        im_pixw(x,y,avcol);
            }

            printf("\rX=%3d, Y=%3d",x,y);
        }
        getch();
        _setvideomode(_DEFAULTMODE);

}
```

Appendix 5

Results from Lighting and Filter Trials



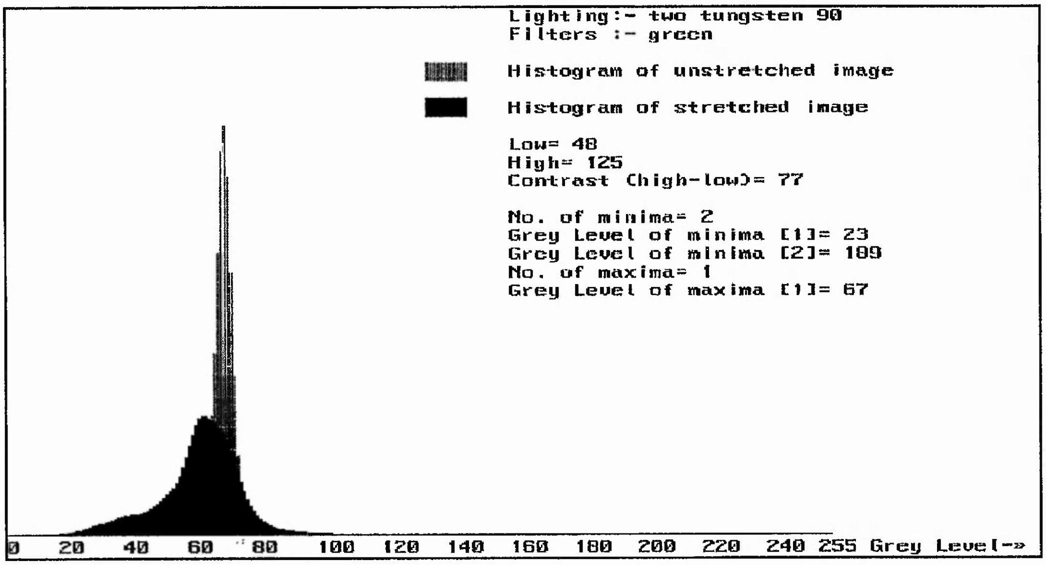
```
Lighting:- two tugsten 90]
Filters :- red

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 32
High= 104
Contrast (high-low)= 72

No. of minima= 3
Grey Level of minima [1]= 14
Grey Level of minima [2]= 67
Grey Level of minima [3]= 178
No. of maxima= 2
Grey Level of maxima [1]= 60
Grey Level of maxima [2]= 70
```

Peaks = 2

Glare = 4

Contrast = 3

Red filter has caused mark to be lighter than substrate.



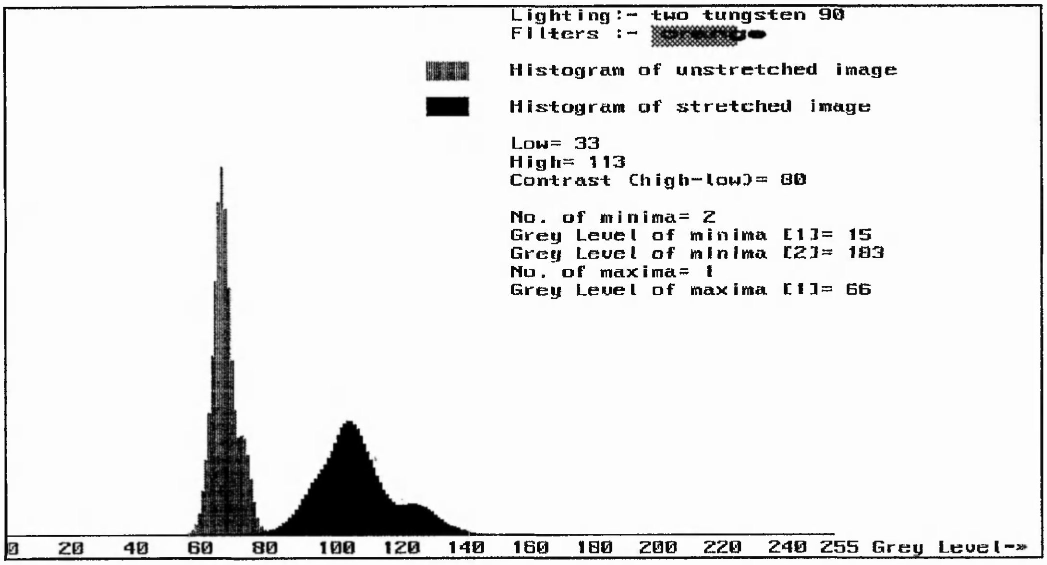
```
Lighting:- two tungsten 90
Filters :- green

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 48
High= 125
Contrast (high-low)= 77

No. of minima= 2
Grey Level of minima [1]= 23
Grey Level of minima [2]= 189
No. of maxima= 1
Grey Level of maxima [1]= 67
```

Peaks = 1

Glare = 4

Contrast = 2

Mark could just be seen, distinction unclear on Histogram.



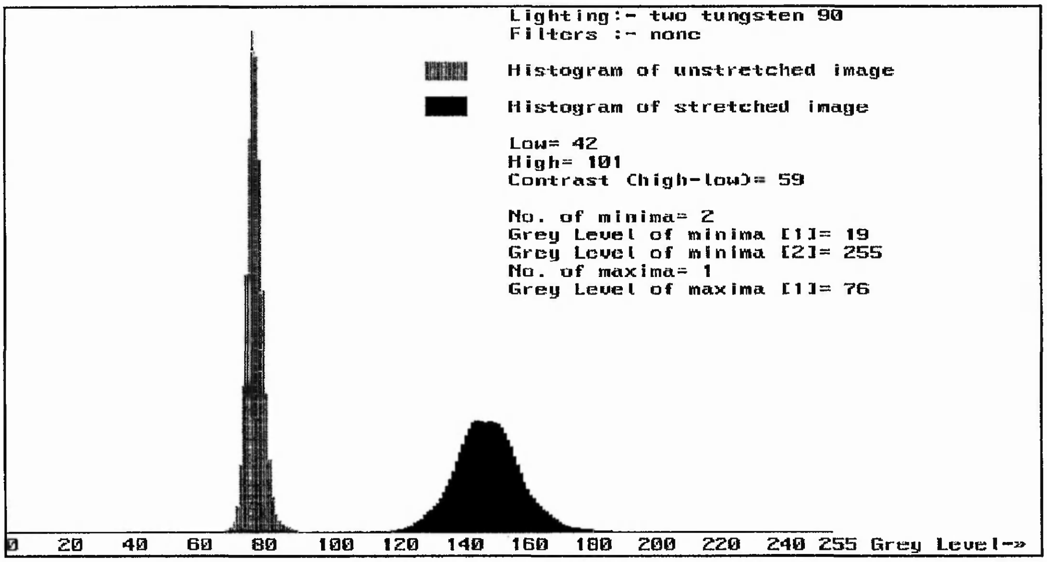
```
Lighting:- two tungsten 90
Filters :- orange

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 33
High= 113
Contrast (high-low)= 80

No. of minima= 2
Grey Level of minima [1]= 15
Grey Level of minima [2]= 183
No. of maxima= 1
Grey Level of maxima [1]= 66
```

Peaks = 2

Glare = 4

Contrast = 3

Filter has caused mark to be lighter than substrate.



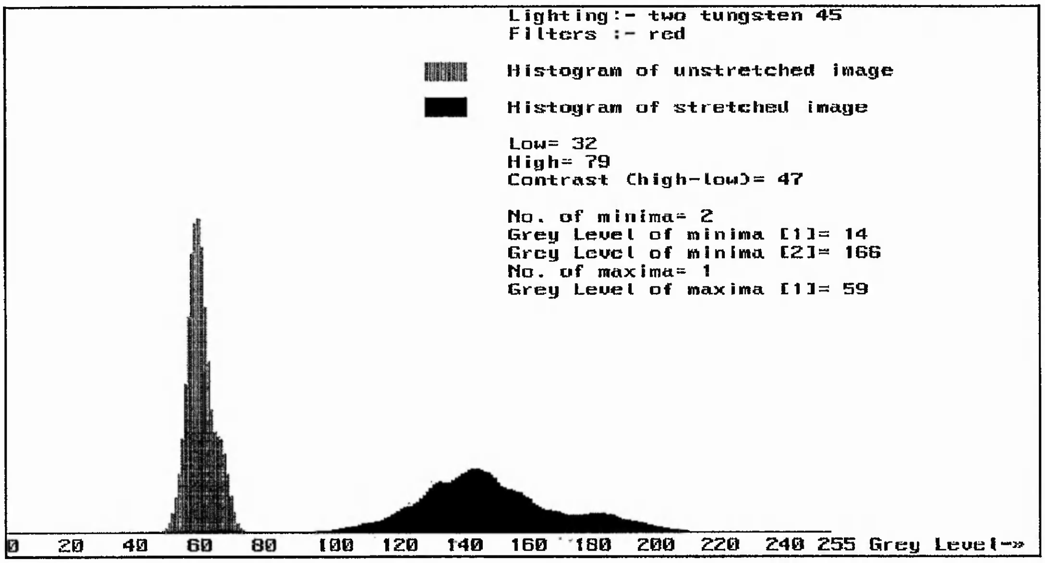
```
Lighting:- two tungsten 90
Filters :- none

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 42
High= 101
Contrast (high-low)= 59

No. of minima= 2
Grey Level of minima [1]= 19
Grey Level of minima [2]= 255
No. of maxima= 1
Grey Level of maxima [1]= 76
```

Peaks = 1

Glare = 4

Contrast = 1

Mark could not be seen.



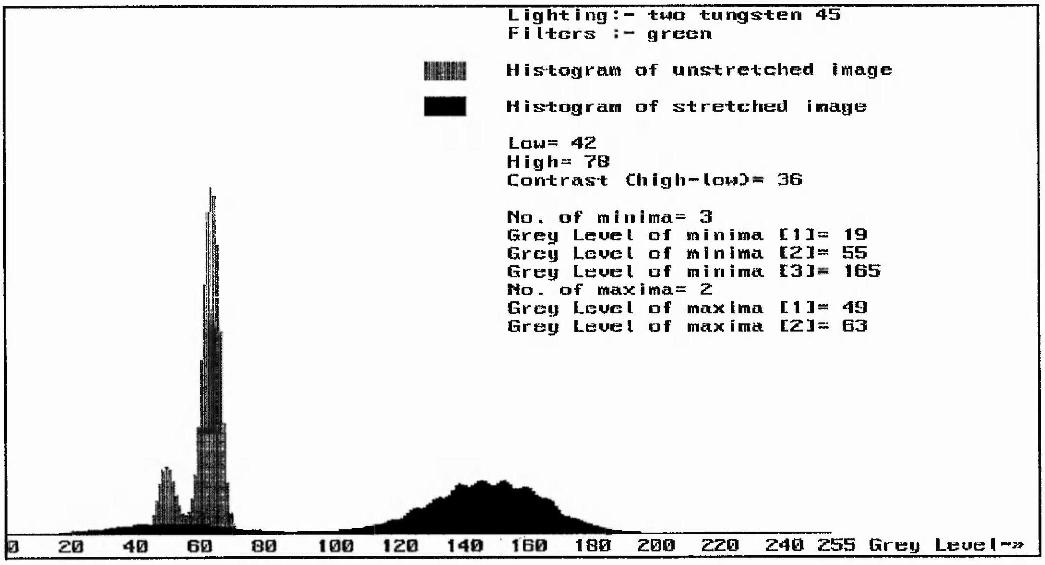
```
Lighting:- two tungsten 45
Filters :- red

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 32
High= 79
Contrast (high-low)= 47

No. of minima= 2
Grey Level of minima [1]= 14
Grey Level of minima [2]= 166
No. of maxima= 1
Grey Level of maxima [1]= 59
```

Peaks = 1

Glare = 3

Contrast = 2

Mark could just be seen, lighter than substrate.



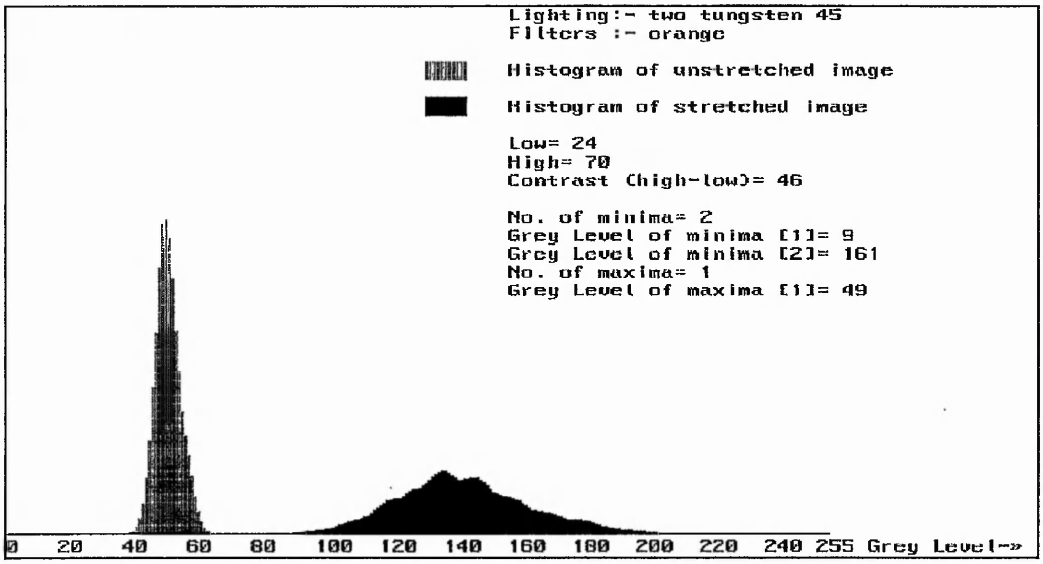
```
Lighting:- two tungsten 45
Filters :- green

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 42
High= 78
Contrast (high-low)= 36

No. of minima= 3
Grey Level of minima [1]= 19
Grey Level of minima [2]= 55
Grey Level of minima [3]= 165
No. of maxima= 2
Grey Level of maxima [1]= 49
Grey Level of maxima [2]= 63
```

Peaks = 2

Glare = 2

Contrast = 5

Very clear.



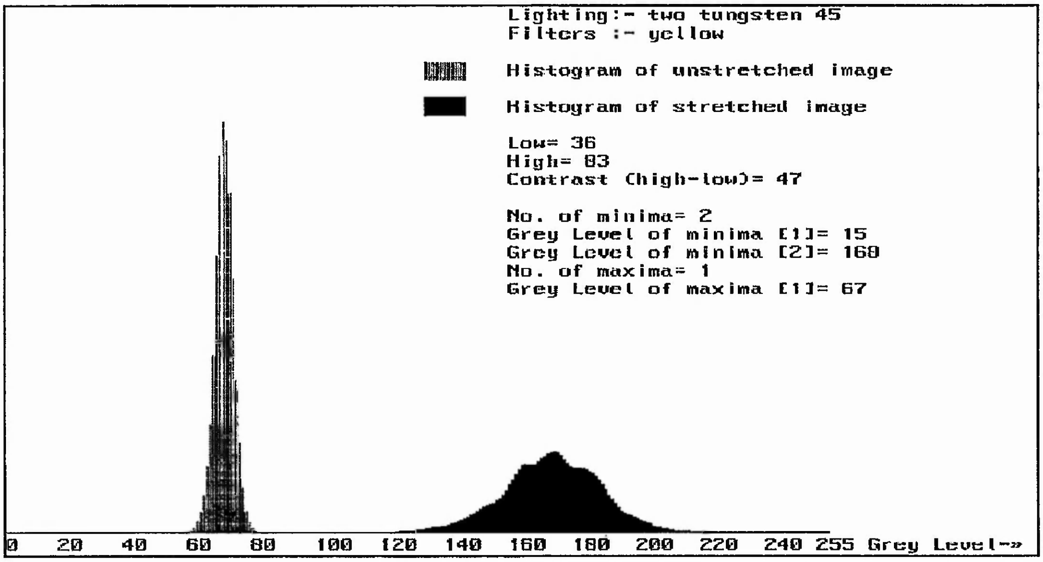
```
Lighting:- two tungsten 45
Filters :- orange

▥ Histogram of unstretched image

■ Histogram of stretched image

Low= 24
High= 70
Contrast (high-low)= 46

No. of minima= 2
Grey Level of minima [1]= 9
Grey Level of minima [2]= 161
No. of maxima= 1
Grey Level of maxima [1]= 49
```

Peaks = 1

Glare = 2

Contrast = 1

No visible mark.



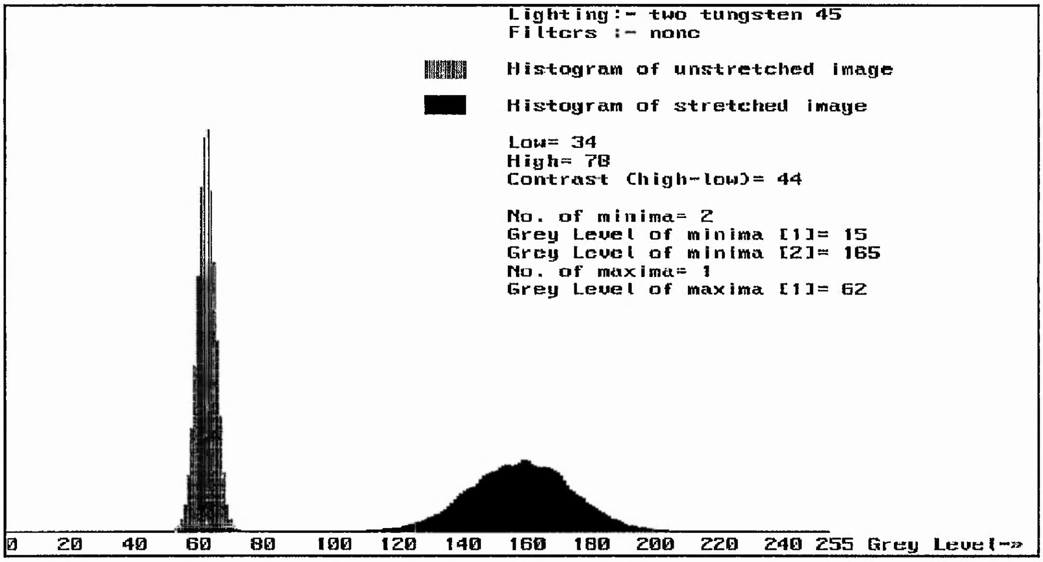
```
Lighting:- two tungsten 45
Filters :- yellow

▥ Histogram of unstretched image

■ Histogram of stretched image

Low= 36
High= 83
Contrast (high-low)= 47

No. of minima= 2
Grey Level of minima [1]= 15
Grey Level of minima [2]= 168
No. of maxima= 1
Grey Level of maxima [1]= 67
```

Peaks = 1

Glare = 2

Contrast = 1

No visible mark.



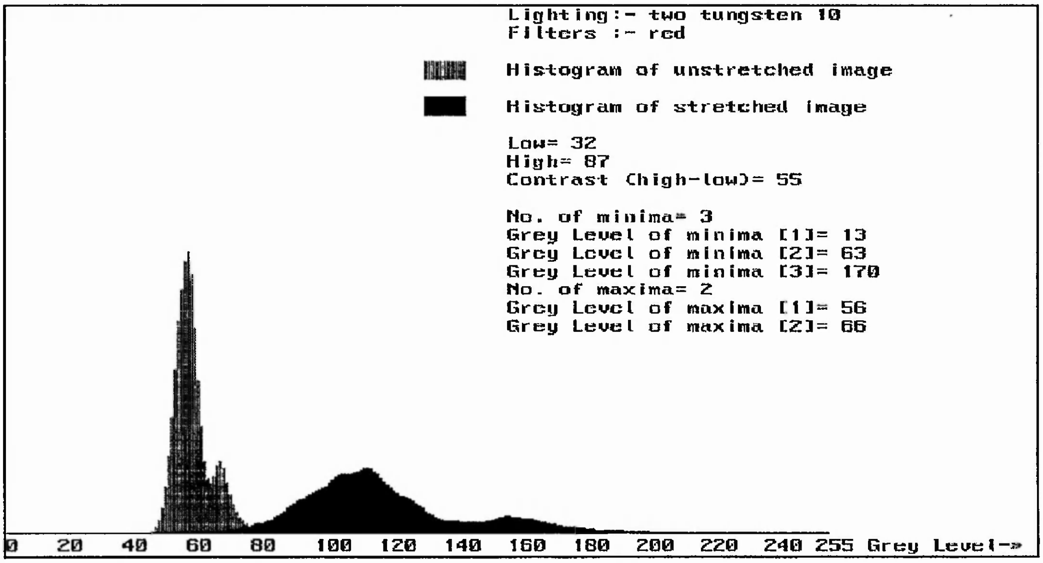
```
Lighting:- two tungsten 45
Filters :- none

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 34
High= 78
Contrast (high-low)= 44

No. of minima= 2
Grey Level of minima [1]= 15
Grey Level of minima [2]= 165
No. of maxima= 1
Grey Level of maxima [1]= 62
```

Peaks = 1

Glare = 3

Contrast = 1

No visible mark.



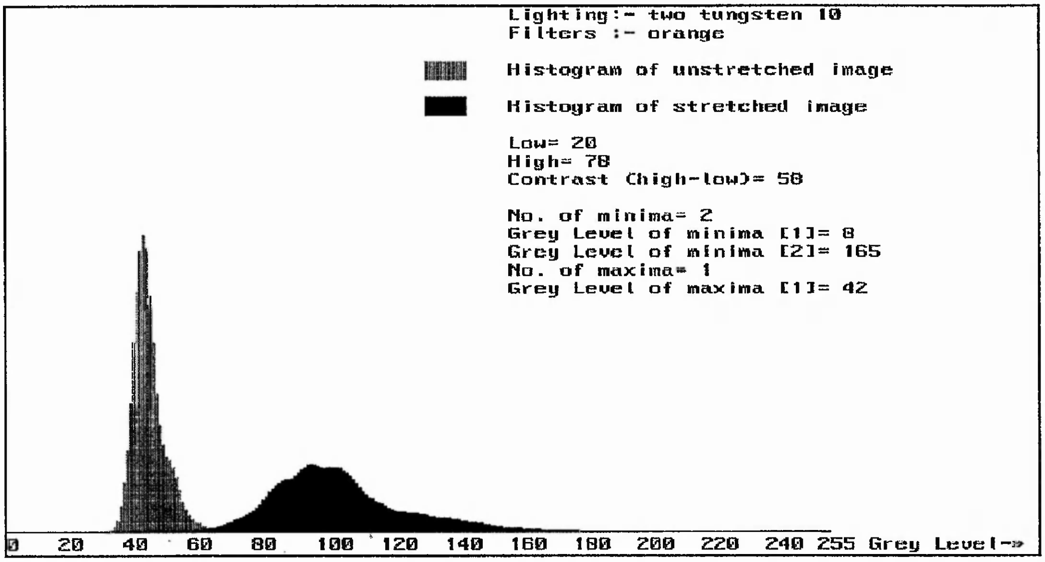
```
Lighting:- two tungsten 10
Filters :- red

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 32
High= 87
Contrast (high-low)= 55

No. of minima= 3
Grey Level of minima [1]= 13
Grey Level of minima [2]= 63
Grey Level of minima [3]= 170
No. of maxima= 2
Grey Level of maxima [1]= 56
Grey Level of maxima [2]= 66
```

Peaks = 2

Glare = 2

Contrast = 3

Mark lighter than substrate.



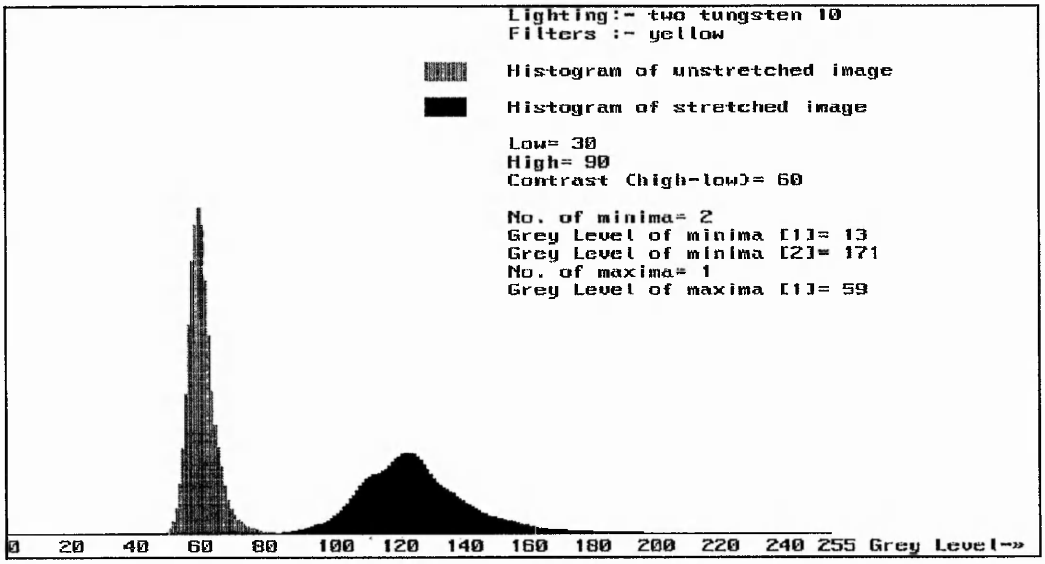
```
Lighting:- two tungsten 10
Filters :- orange

▨  Histogram of unstretched image

■  Histogram of stretched image

Low= 20
High= 78
Contrast (high-low)= 58

No. of minima= 2
Grey Level of minima [1]= 8
Grey Level of minima [2]= 165
No. of maxima= 1
Grey Level of maxima [1]= 42
```

Peaks = 1

Glare = 2

Contrast = 2

Mark lighter than substrate but only just visible to naked eye .



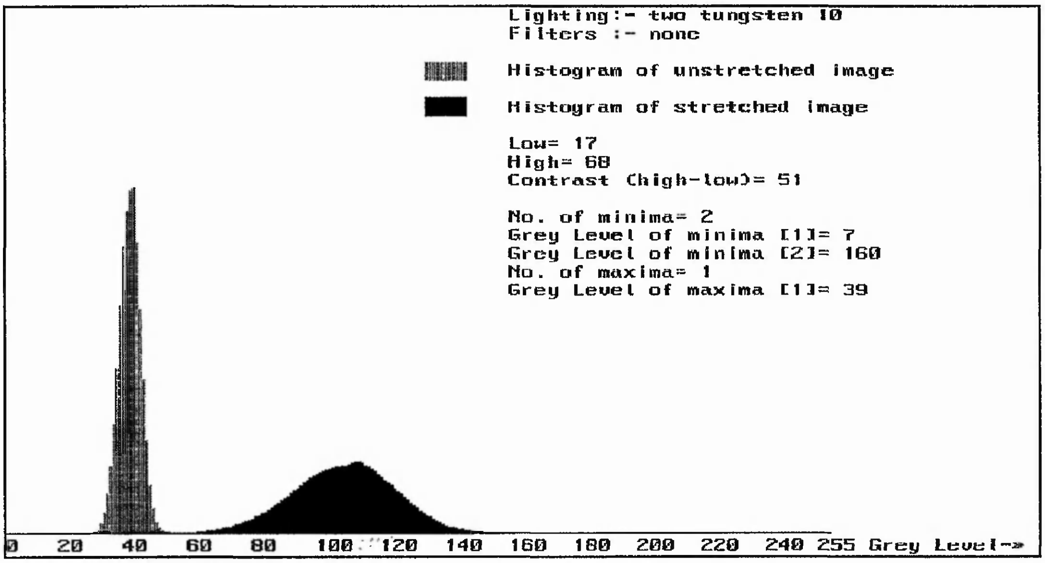
```
Lighting:- two tungsten 10
Filters :- yellow

▨  Histogram of unstretched image

■  Histogram of stretched image

Low= 30
High= 90
Contrast (high-low)= 60

No. of minima= 2
Grey Level of minima [1]= 13
Grey Level of minima [2]= 171
No. of maxima= 1
Grey Level of maxima [1]= 59
```

Peaks = 1

Glare = 2

Contrast = 2

Barely visible to naked eye .



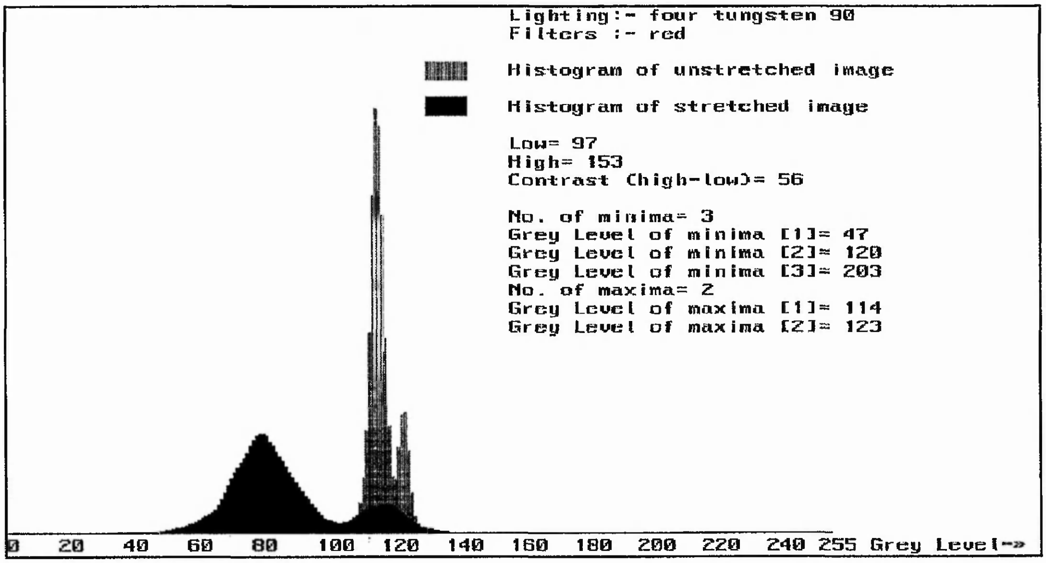
```
Lighting:- two tungsten 10
Filters :- none

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 17
High= 68
Contrast (high-low)= 51

No. of minima= 2
Grey Level of minima [1]= 7
Grey Level of minima [2]= 160
No. of maxima= 1
Grey Level of maxima [1]= 39
```

Peaks = 1

Glare = 2

Contrast = 1

Mark not visible.



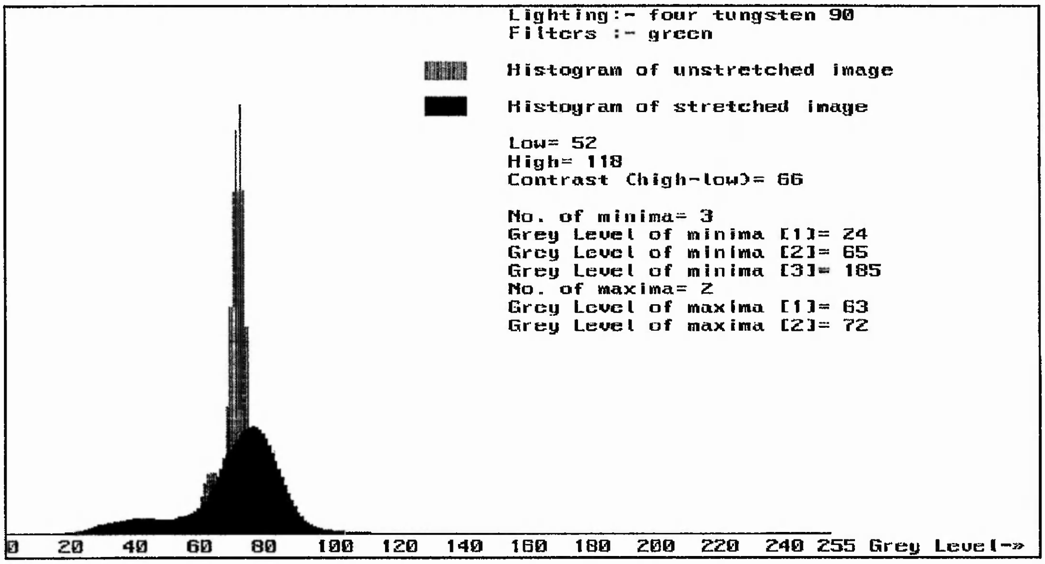
```
Lighting:- four tungsten 90
Filters :- red

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 97
High= 153
Contrast (high-low)= 56

No. of minima= 3
Grey Level of minima [1]= 47
Grey Level of minima [2]= 120
Grey Level of minima [3]= 203
No. of maxima= 2
Grey Level of maxima [1]= 114
Grey Level of maxima [2]= 123
```

Peaks = 2

Glare = 3

Contrast = 3

Mark lighter than substrate.



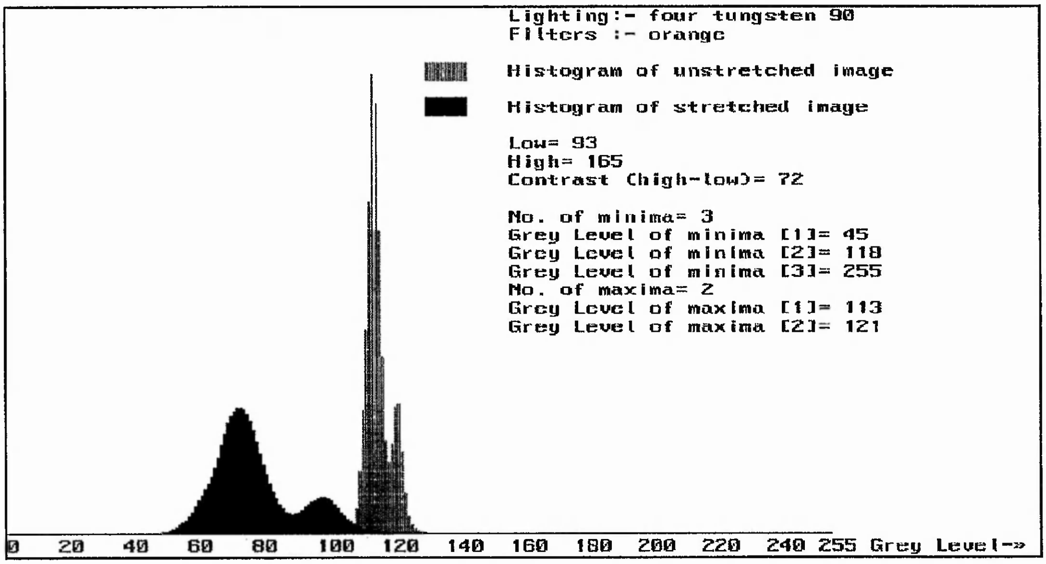
```
Lighting:- four tungsten 90
Filters :- green

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 52
High= 118
Contrast (high-low)= 66

No. of minima= 3
Grey Level of minima [1]= 24
Grey Level of minima [2]= 65
Grey Level of minima [3]= 185
No. of maxima= 2
Grey Level of maxima [1]= 63
Grey Level of maxima [2]= 72
```

Peaks = 2

Glare = 3

Contrast = 3

Mark clear to naked eye but less distinguishable on histogram.



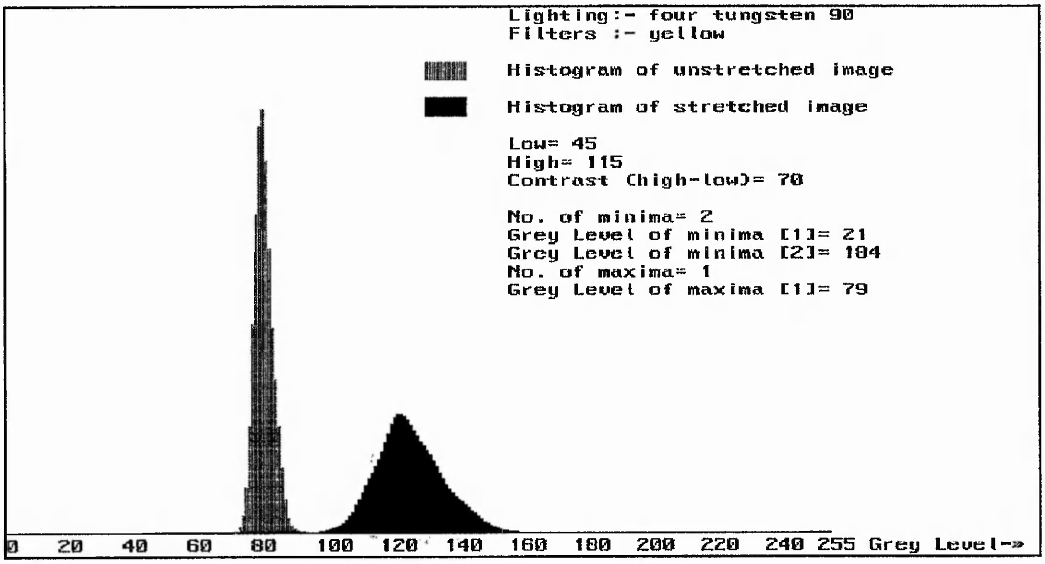
```
Lighting:- four tungsten 90
Filters :- orange

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 93
High= 165
Contrast (high-low)= 72

No. of minima= 3
Grey Level of minima [1]= 45
Grey Level of minima [2]= 118
Grey Level of minima [3]= 255
No. of maxima= 2
Grey Level of maxima [1]= 113
Grey Level of maxima [2]= 121
```

Peaks = 2

Glare = 3

Contrast = 3

Mark lighter than substrate.



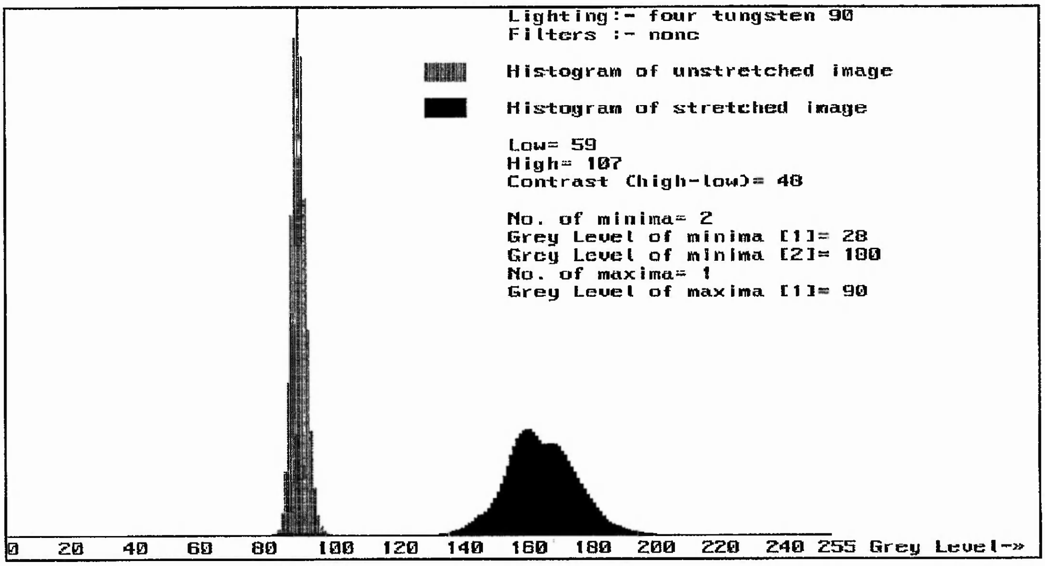
```
Lighting:- four tungsten 90
Filters :- yellow

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 45
High= 115
Contrast (high-low)= 70

No. of minima= 2
Grey Level of minima [1]= 21
Grey Level of minima [2]= 104
No. of maxima= 1
Grey Level of maxima [1]= 79
```

Peaks = 1

Glare = 3

Contrast = 3

Mark invisible.



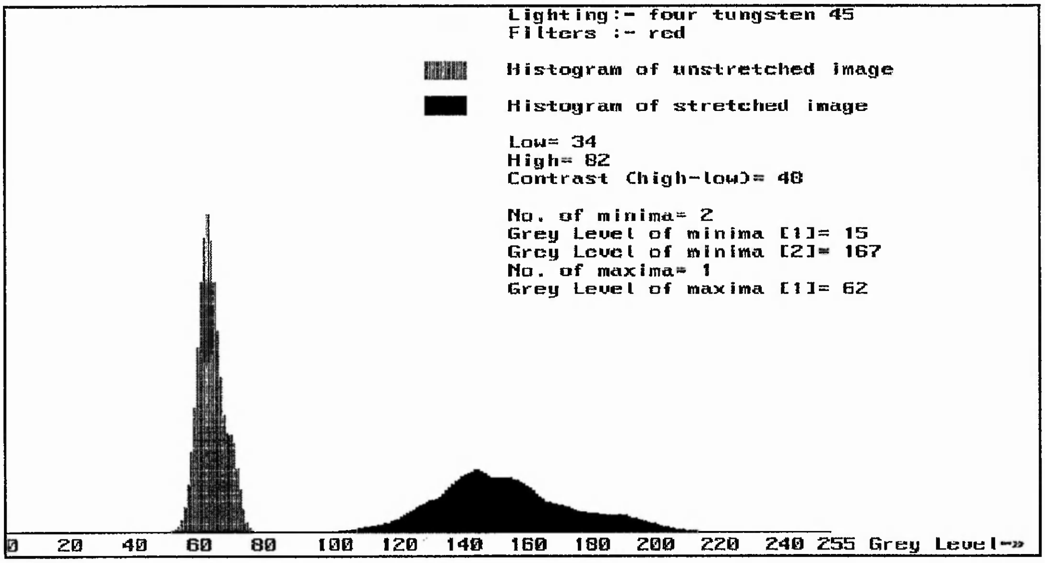
```
Lighting:- four tungsten 90
Filters :- none

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 59
High= 107
Contrast (high-low)= 48

No. of minima= 2
Grey Level of minima [1]= 28
Grey Level of minima [2]= 180
No. of maxima= 1
Grey Level of maxima [1]= 90
```

Peaks = 1

Glare = 3

Contrast = 3

Mark invisible.



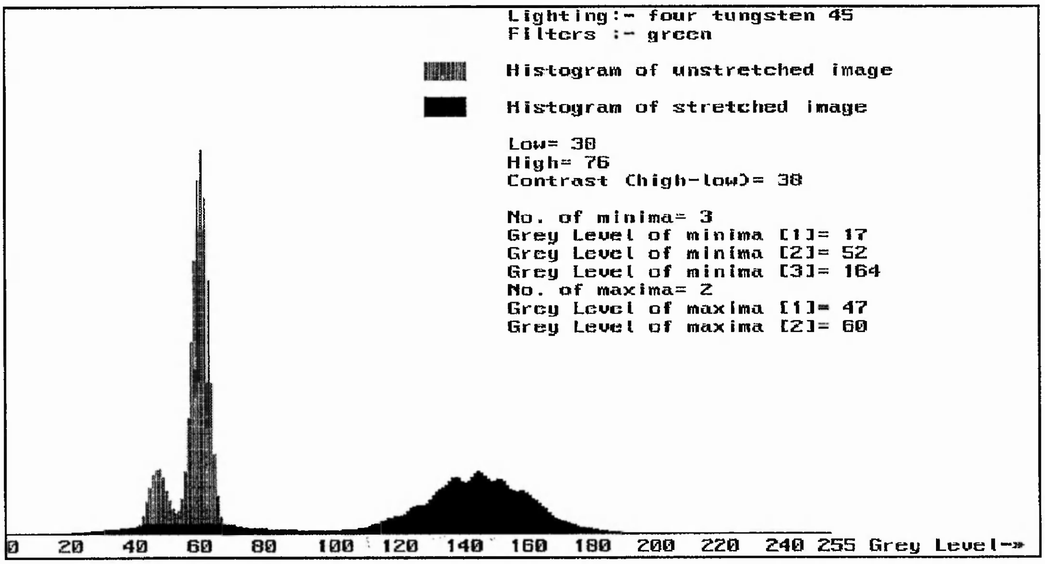
```
Lighting:- four tungsten 45
Filters :- red

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 34
High= 82
Contrast (high-low)= 48

No. of minima= 2
Grey Level of minima [1]= 15
Grey Level of minima [2]= 167
No. of maxima= 1
Grey Level of maxima [1]= 62
```

Peaks = 1

Glare = 3

Contrast = 2

Mark just visible, lighter than substrate.



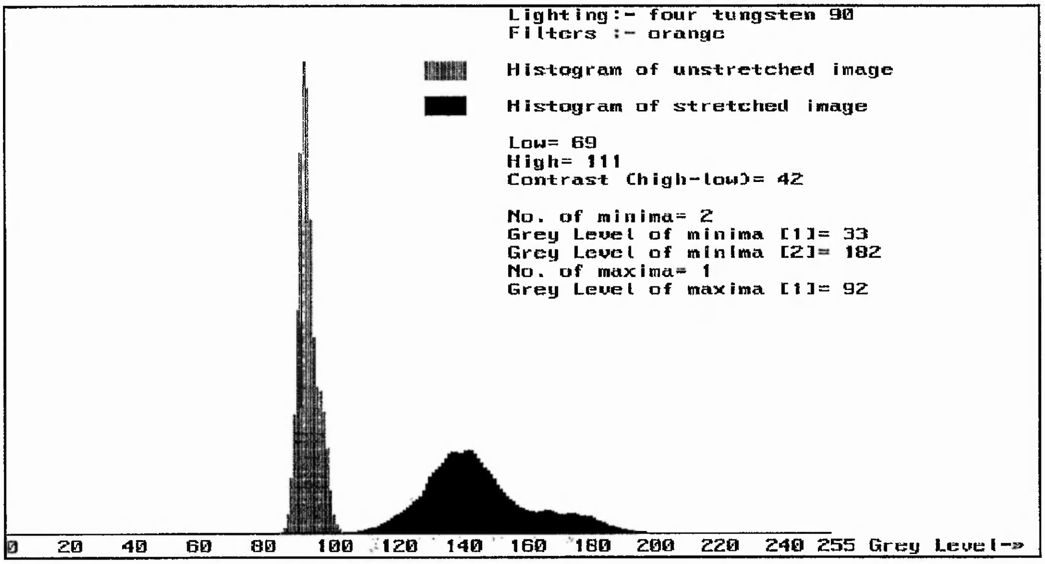
```
Lighting:- four tungsten 45
Filters :- green

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 38
High= 76
Contrast (high-low)= 38

No. of minima= 3
Grey Level of minima [1]= 17
Grey Level of minima [2]= 52
Grey Level of minima [3]= 164
No. of maxima= 2
Grey Level of maxima [1]= 47
Grey Level of maxima [2]= 60
```

Peaks = 2

Glare = 3

Contrast = 4

Mark very clear but a lot of glare on edges.



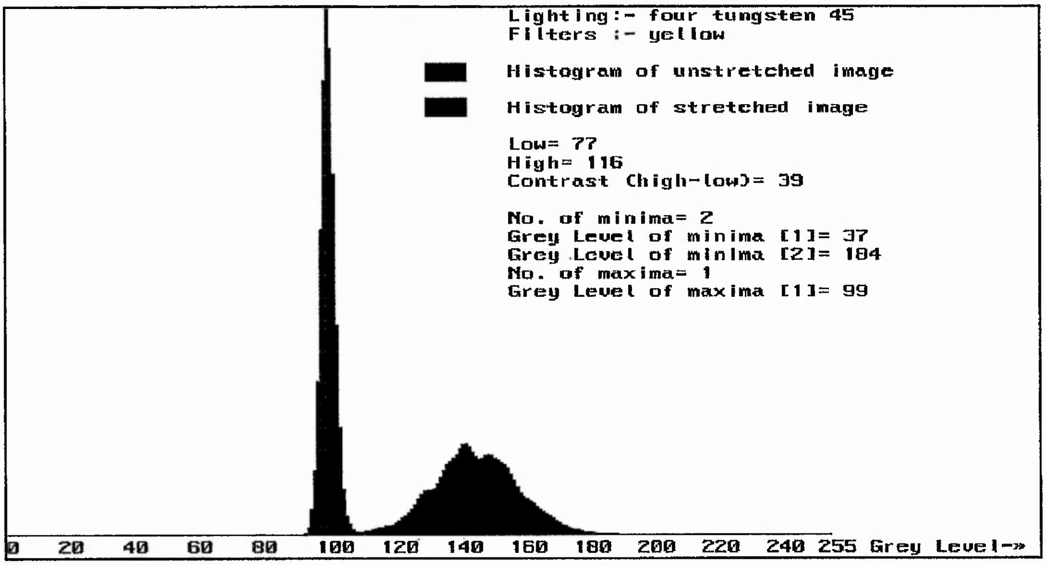
```
Lighting:- four tungsten 90
Filters :- orange

▥  Histogram of unstretched image

■  Histogram of stretched image

Low= 69
High= 111
Contrast (high-low)= 42

No. of minima= 2
Grey Level of minima [1]= 33
Grey Level of minima [2]= 182
No. of maxima= 1
Grey Level of maxima [1]= 92
```

Peaks = 1

Glare = 3

Contrast = 2

Mark just visible, lighter than substrate.



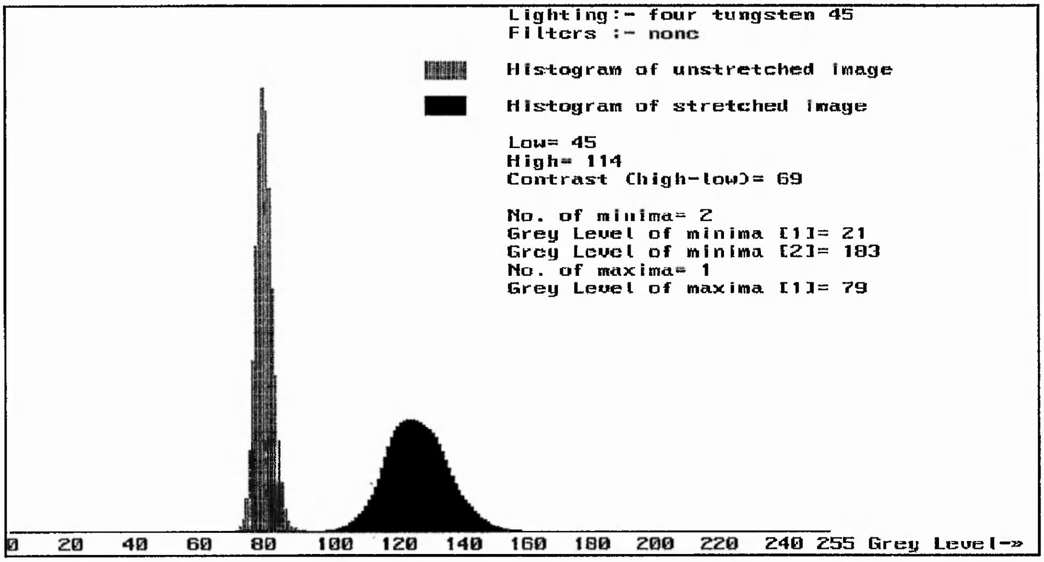
```
Lighting:- four tungsten 45
Filters :- yellow

■  Histogram of unstretched image

■  Histogram of stretched image

Low= 77
High= 116
Contrast (high-low)= 39

No. of minima= 2
Grey Level of minima [1]= 37
Grey Level of minima [2]= 184
No. of maxima= 1
Grey Level of maxima [1]= 99
```

Peaks = 1

Glare = 2

Contrast = 1

Mark not visible.



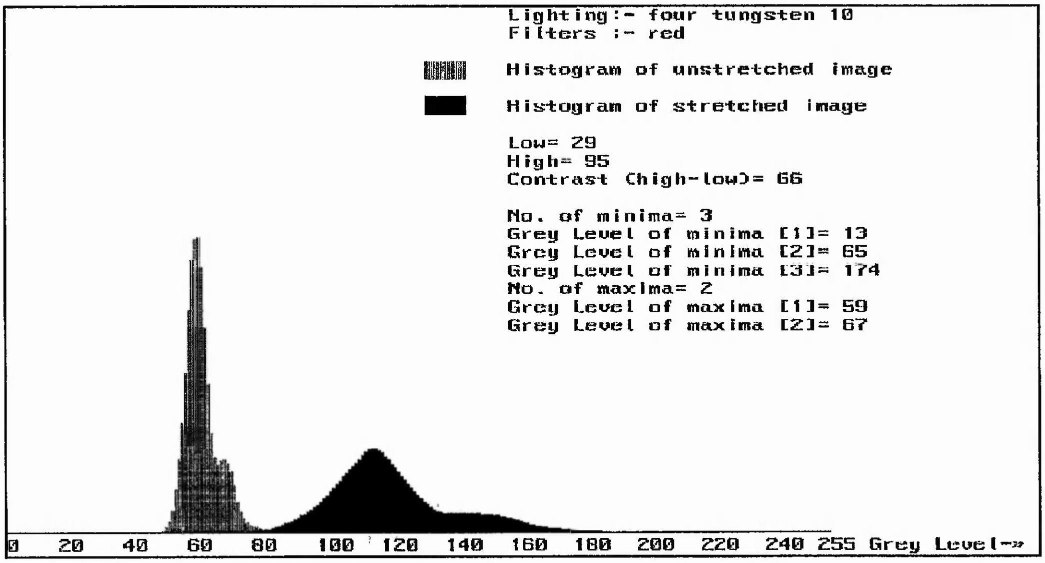
```
Lighting:- four tungsten 45
Filters :- none

▥ Histogram of unstretched image

■ Histogram of stretched image

Low= 45
High= 114
Contrast (high-low)= 69

No. of minima= 2
Grey Level of minima [1]= 21
Grey Level of minima [2]= 183
No. of maxima= 1
Grey Level of maxima [1]= 79
```

Peaks = 1

Glare = 2

Contrast = 1

Mark not visible.



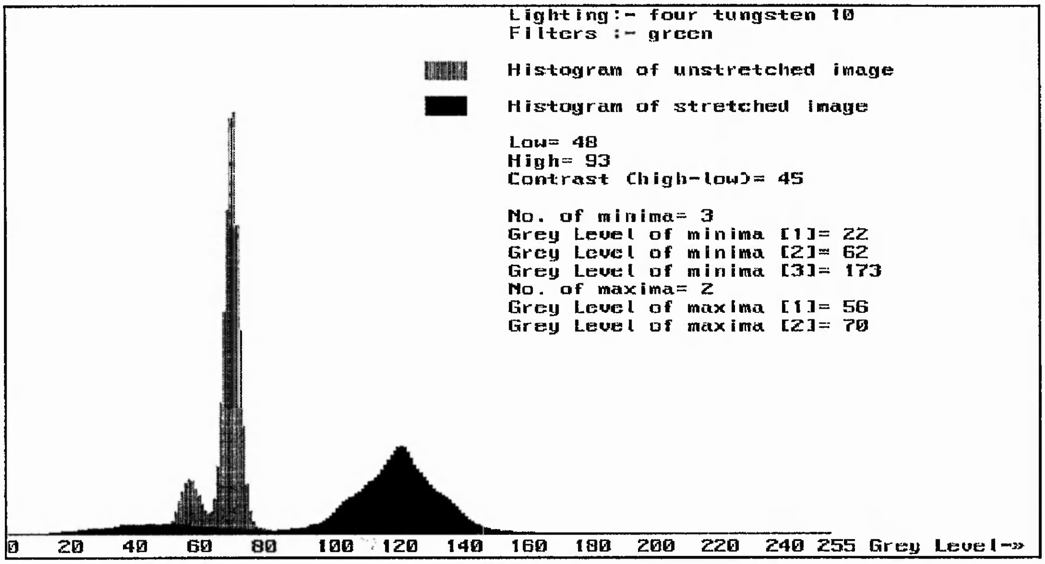
```
Lighting:- four tungsten 10
Filters :- red

▥ Histogram of unstretched image

■ Histogram of stretched image

Low= 29
High= 95
Contrast (high-low)= 66

No. of minima= 3
Grey Level of minima [1]= 13
Grey Level of minima [2]= 65
Grey Level of minima [3]= 174
No. of maxima= 2
Grey Level of maxima [1]= 59
Grey Level of maxima [2]= 67
```

Peaks = 2

Glare = 2

Contrast = 2

Mark just visible, lighter than substrate.



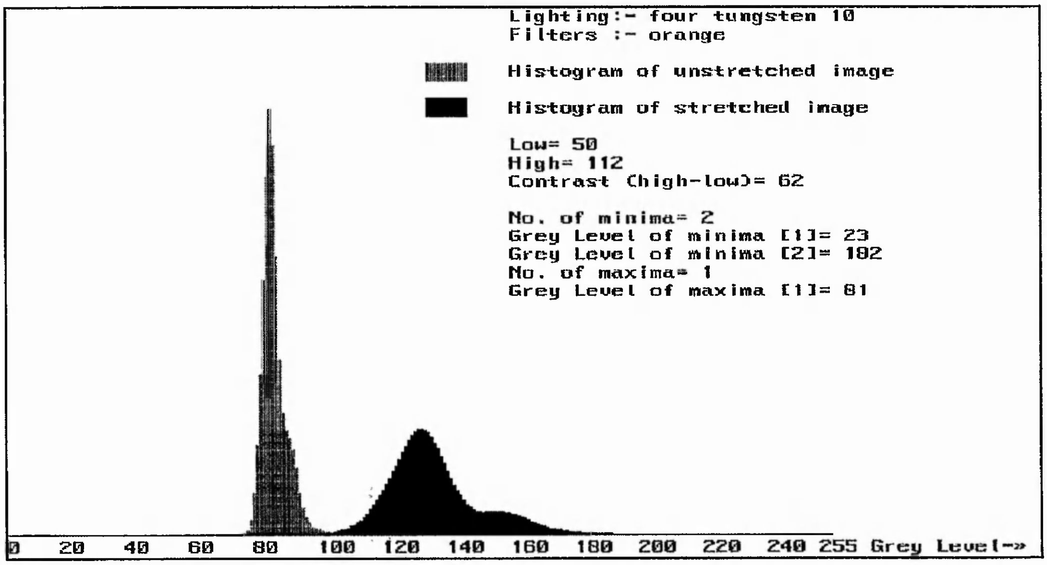
```
Lighting:- four tungsten 10
Filters :- green

▥ Histogram of unstretched image

■ Histogram of stretched image

Low= 48
High= 93
Contrast (high-low)= 45

No. of minima= 3
Grey Level of minima [1]= 22
Grey Level of minima [2]= 62
Grey Level of minima [3]= 173
No. of maxima= 2
Grey Level of maxima [1]= 56
Grey Level of maxima [2]= 70
```

Peaks = 2

Glare = 2

Contrast = 4

Very clear.



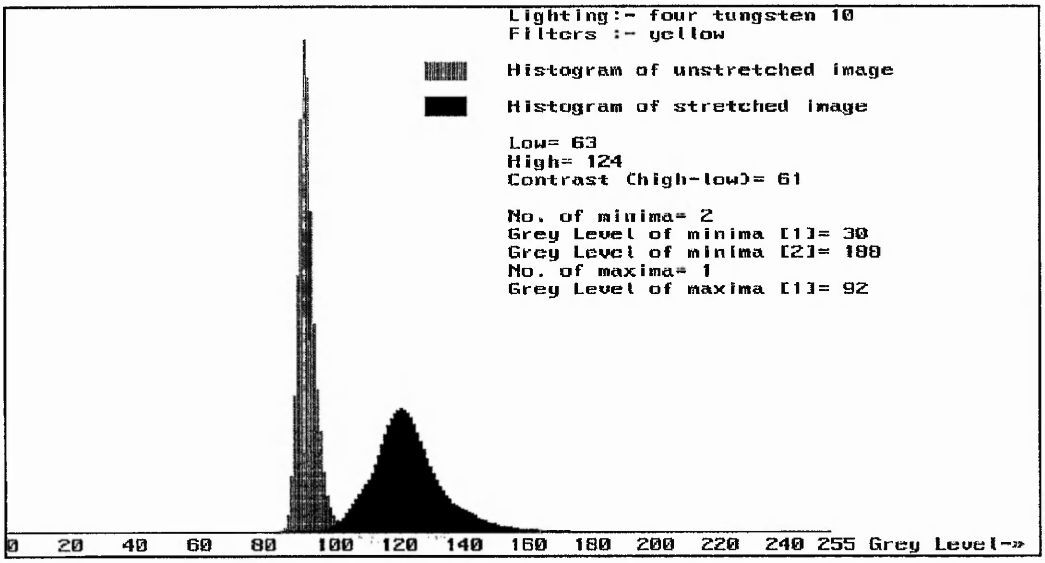
```
Lighting:- four tungsten 10
Filters :- orange

▥ Histogram of unstretched image

■ Histogram of stretched image

Low= 50
High= 112
Contrast (high-low)= 62

No. of minima= 2
Grey Level of minima [1]= 23
Grey Level of minima [2]= 182
No. of maxima= 1
Grey Level of maxima [1]= 81
```

Peaks = 1

Glare = 2

Contrast = 2

Mark just visible, lighter than substrate.



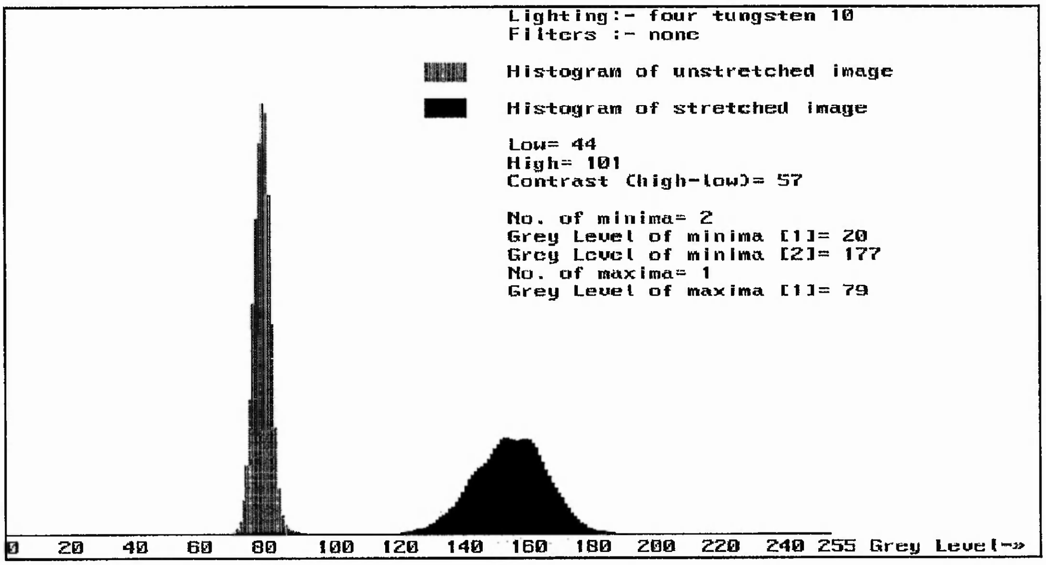
```
Lighting:- four tungsten 10
Filters :- yellow

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 63
High= 124
Contrast (high-low)= 61

No. of minima= 2
Grey Level of minima [1]= 30
Grey Level of minima [2]= 180
No. of maxima= 1
Grey Level of maxima [1]= 92
```

Peaks = 1

Glare = 2

Contrast = 1

Mark not visible.



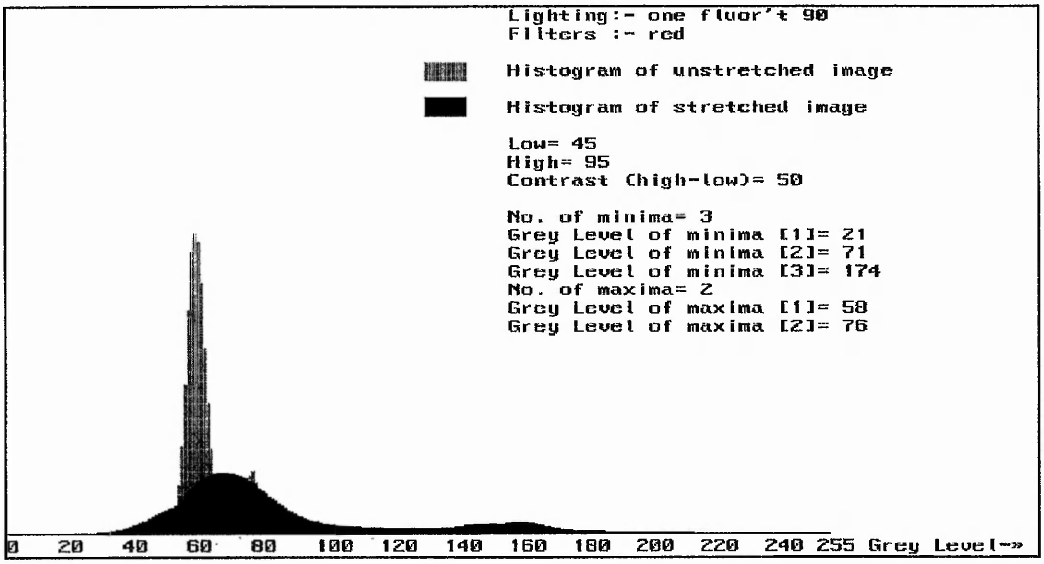
```
Lighting:- four tungsten 10
Filters :- none

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 44
High= 101
Contrast (high-low)= 57

No. of minima= 2
Grey Level of minima [1]= 20
Grey Level of minima [2]= 177
No. of maxima= 1
Grey Level of maxima [1]= 79
```

Peaks = 1

Glare = 2

Contrast = 1

Mark not visible.



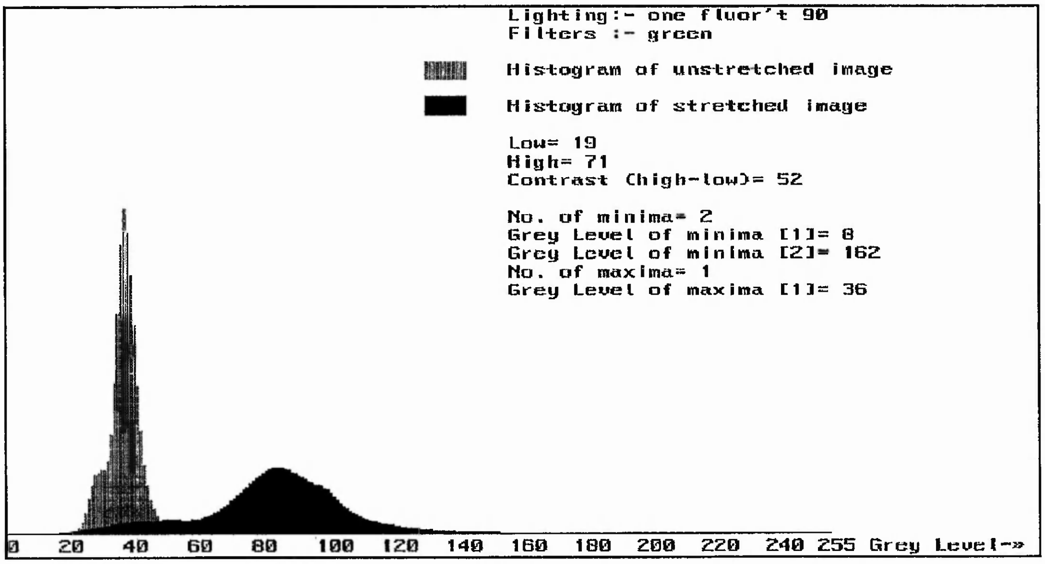
```
Lighting:- one fluor't 90
Filters :- red

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 45
High= 95
Contrast (high-low)= 50

No. of minima= 3
Grey Level of minima [1]= 21
Grey Level of minima [2]= 71
Grey Level of minima [3]= 174
No. of maxima= 2
Grey Level of maxima [1]= 58
Grey Level of maxima [2]= 76
```

Peaks = 2

Glare = 3

Contrast = 4

Mark very clear, lighter than substrate.



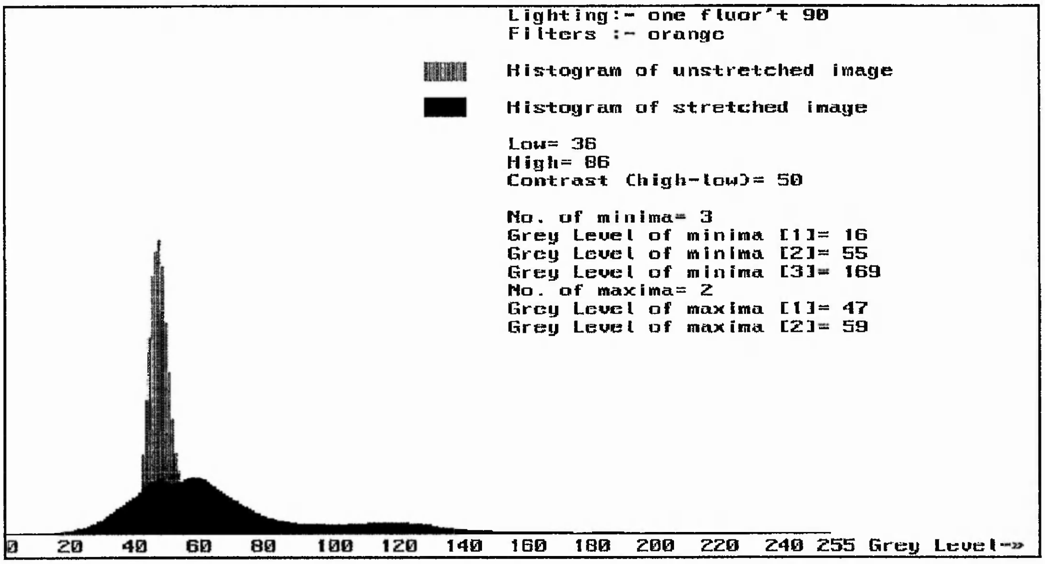
```
Lighting:- one fluor't 90
Filters :- green

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 19
High= 71
Contrast (high-low)= 52

No. of minima= 2
Grey Level of minima [1]= 8
Grey Level of minima [2]= 162
No. of maxima= 1
Grey Level of maxima [1]= 36
```

Peaks = 1

Glare = 4

Contrast = 2

Mark clear to naked eye, unclear on histogram.



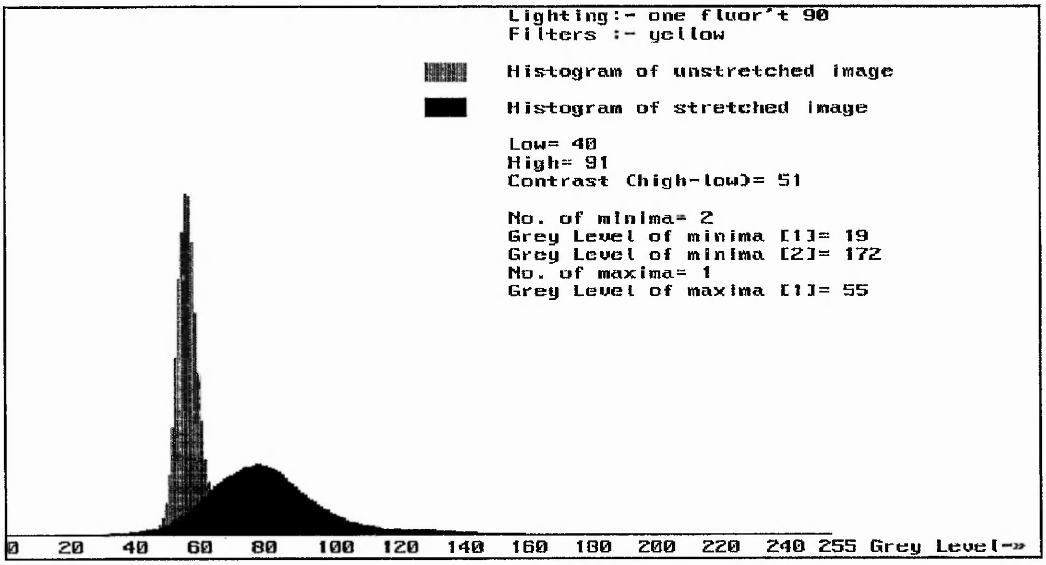
```
Lighting:- one fluor't 90
Filters :- orange

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 36
High= 86
Contrast (high-low)= 50

No. of minima= 3
Grey Level of minima [1]= 16
Grey Level of minima [2]= 55
Grey Level of minima [3]= 169
No. of maxima= 2
Grey Level of maxima [1]= 47
Grey Level of maxima [2]= 59
```

Peaks = 2

Glare = 3

Contrast = 3

Mark clear, lighter than substrate.



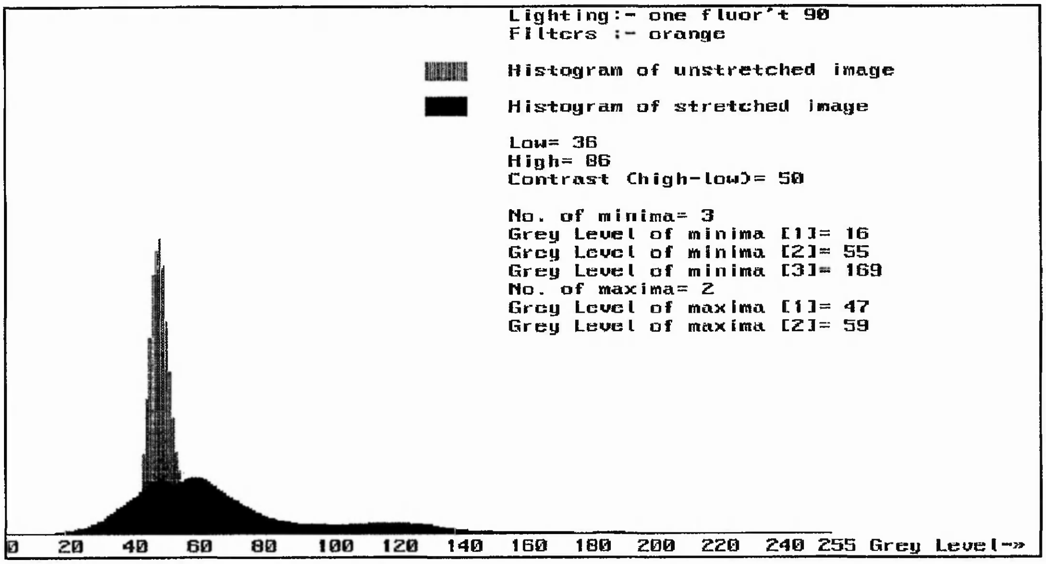
```
Lighting:- one fluor't 90
Filters :- yellow

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 40
High= 91
Contrast (high-low)= 51

No. of minima= 2
Grey Level of minima [1]= 19
Grey Level of minima [2]= 172
No. of maxima= 1
Grey Level of maxima [1]= 55
```

Peaks = 1

Glare = 4

Contrast = 1

Mark not visible, high glare.



```
Lighting:- one fluor't 90
Filters :- orange

▥ Histogram of unstretched image

■ Histogram of stretched image

Low= 36
High= 86
Contrast (high-low)= 50

No. of minima= 3
Grey Level of minima [1]= 16
Grey Level of minima [2]= 55
Grey Level of minima [3]= 169
No. of maxima= 2
Grey Level of maxima [1]= 47
Grey Level of maxima [2]= 59
```

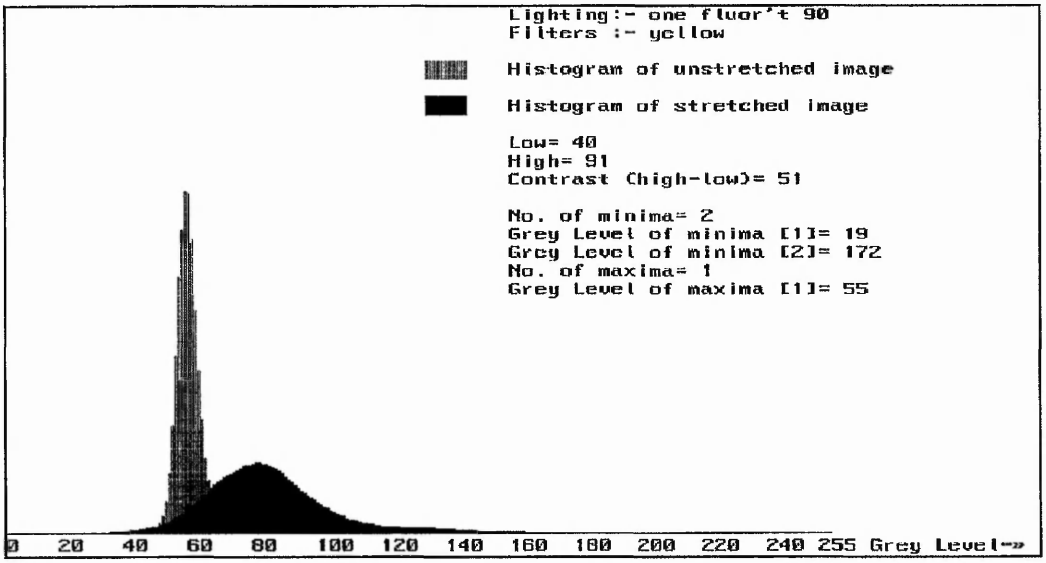
0 20 40 60 80 100 120 140 160 180 200 220 240 255 Grey Level-»

Peaks = 2

Glare = 3

Contrast = 3

Mark clear, lighter than substrate.



```
Lighting:- one fluor't 90
Filters :- yellow

▥ Histogram of unstretched image

■ Histogram of stretched image

Low= 40
High= 91
Contrast (high-low)= 51

No. of minima= 2
Grey Level of minima [1]= 19
Grey Level of minima [2]= 172
No. of maxima= 1
Grey Level of maxima [1]= 55
```

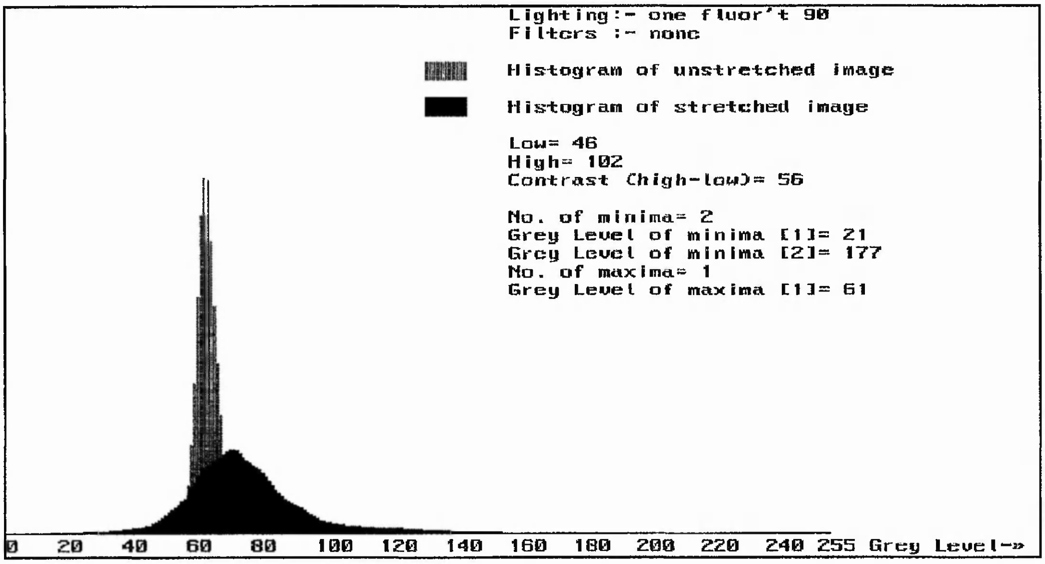
0 20 40 60 80 100 120 140 160 180 200 220 240 255 Grey Level-»

Peaks = 1

Glare = 4

Contrast = 1

Mark not visible, high glare.



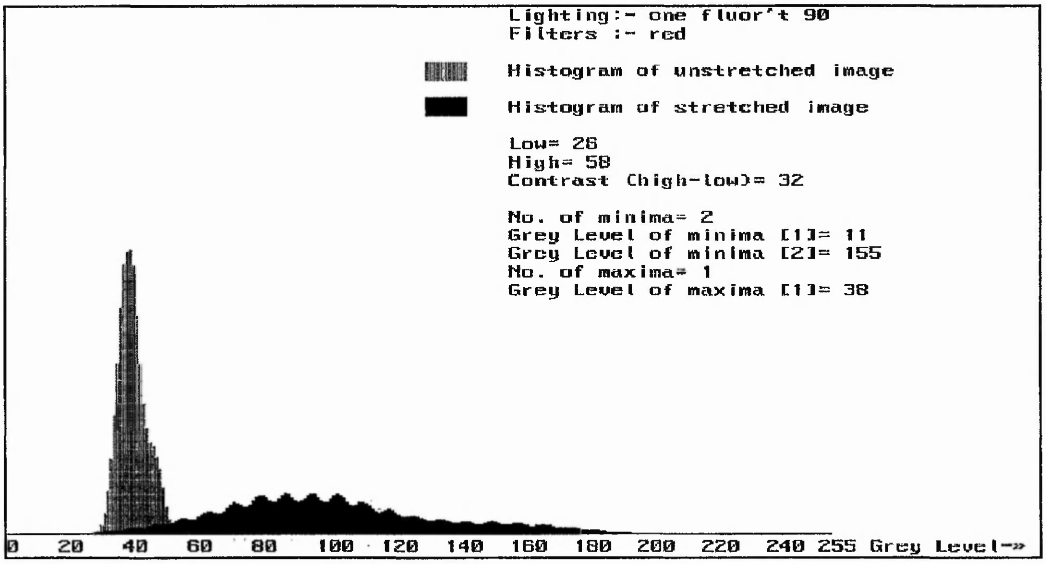
```
Lighting:- one fluor't 90
Filters :- none

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 46
High= 102
Contrast (high-low)= 56

No. of minima= 2
Grey Level of minima [1]= 21
Grey Level of minima [2]= 177
No. of maxima= 1
Grey Level of maxima [1]= 61
```

Peaks = 1

Glare = 4

Contrast = 1

Mark not visible, high glare.



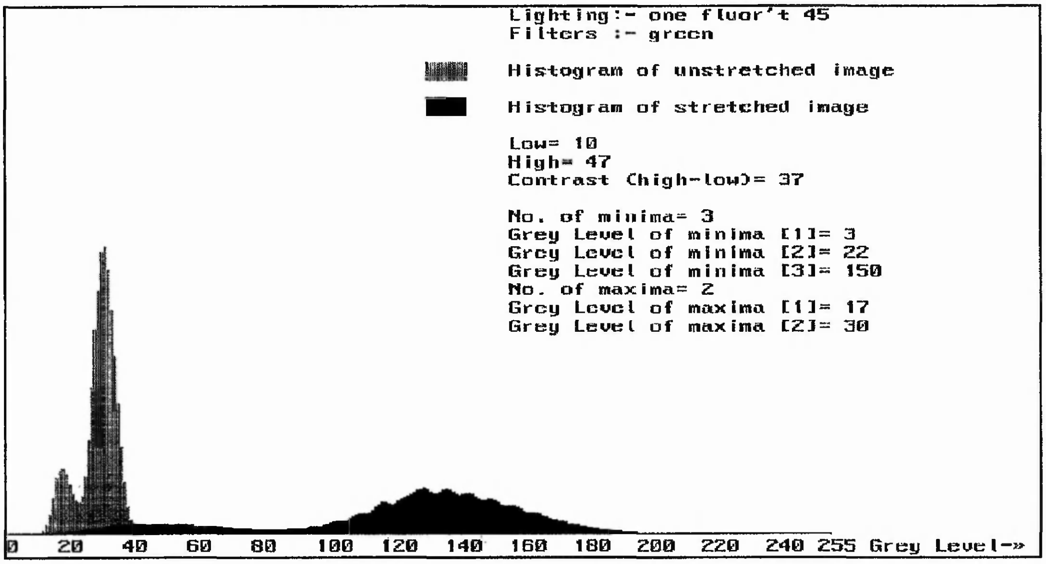
```
Lighting:- one fluor't 90
Filters :- red

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 26
High= 58
Contrast (high-low)= 32

No. of minima= 2
Grey Level of minima [1]= 11
Grey Level of minima [2]= 155
No. of maxima= 1
Grey Level of maxima [1]= 38
```

Peaks = 1

Glare = 3

Contrast = 2

Mark clear to eye lighter than substrate, unclear on histogram.



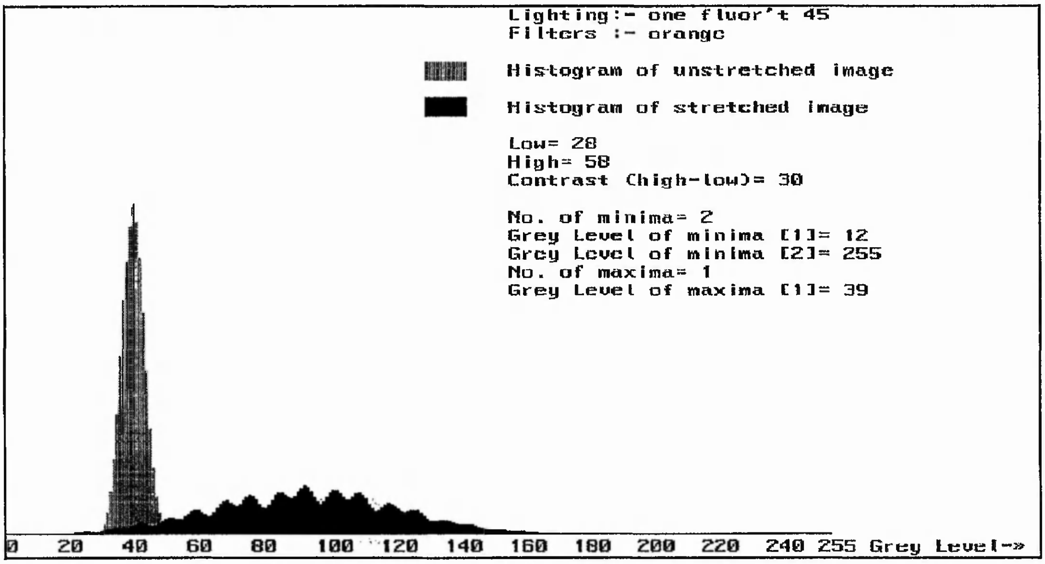
```
Lighting:- one fluor't 45
Filters :- green

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 10
High= 47
Contrast (high-low)= 37

No. of minima= 3
Grey Level of minima [1]= 3
Grey Level of minima [2]= 22
Grey Level of minima [3]= 150
No. of maxima= 2
Grey Level of maxima [1]= 17
Grey Level of maxima [2]= 30
```

Peaks = 2

Glare = 2

Contrast = 4

Mark very clear.



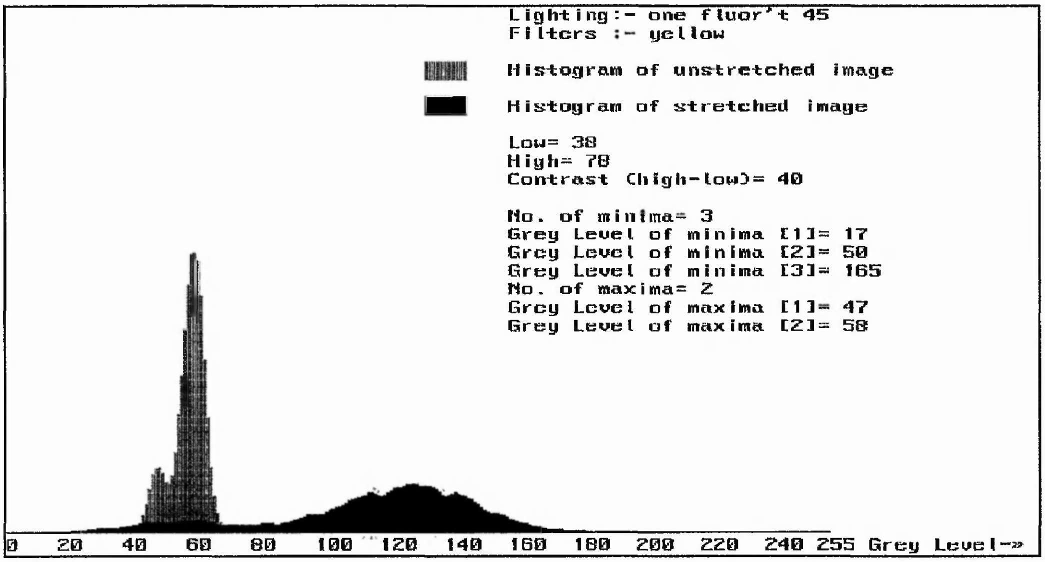
```
Lighting:- one fluor't 45
Filters :- orange

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 28
High= 58
Contrast (high-low)= 30

No. of minima= 2
Grey Level of minima [1]= 12
Grey Level of minima [2]= 255
No. of maxima= 1
Grey Level of maxima [1]= 39
```

Peaks = 1

Glare = 2

Contrast = 2

Mark barely visible, lighter than substrate.

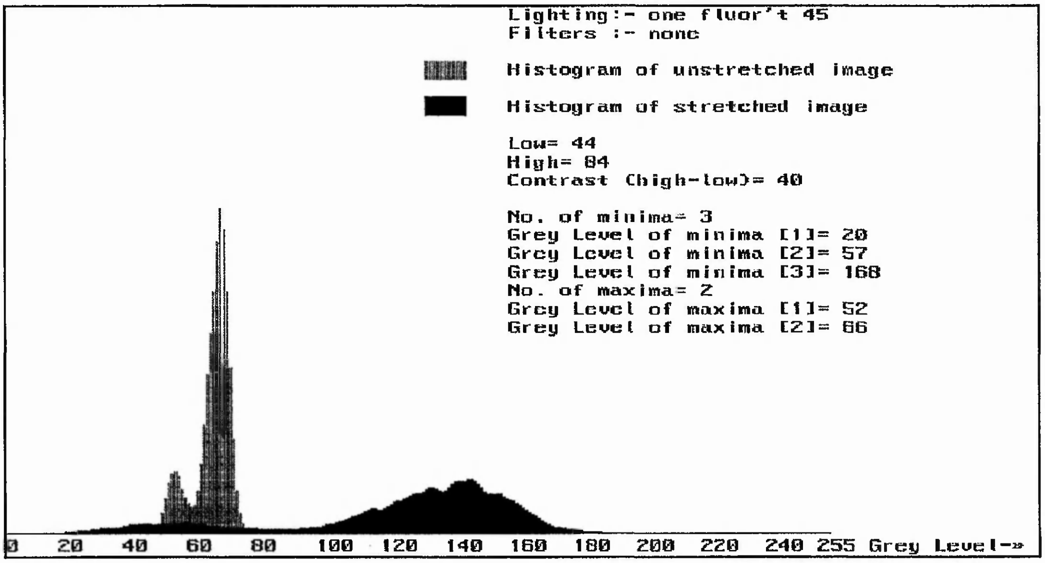


```
Lighting:- one fluor't 45
Filters :- yellow

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 38
High= 78
Contrast (high-low)= 40

No. of minima= 3
Grey Level of minima [1]= 17
Grey Level of minima [2]= 50
Grey Level of minima [3]= 165
No. of maxima= 2
Grey Level of maxima [1]= 47
Grey Level of maxima [2]= 58
```

Peaks = 2

Glare = 2

Contrast = 3

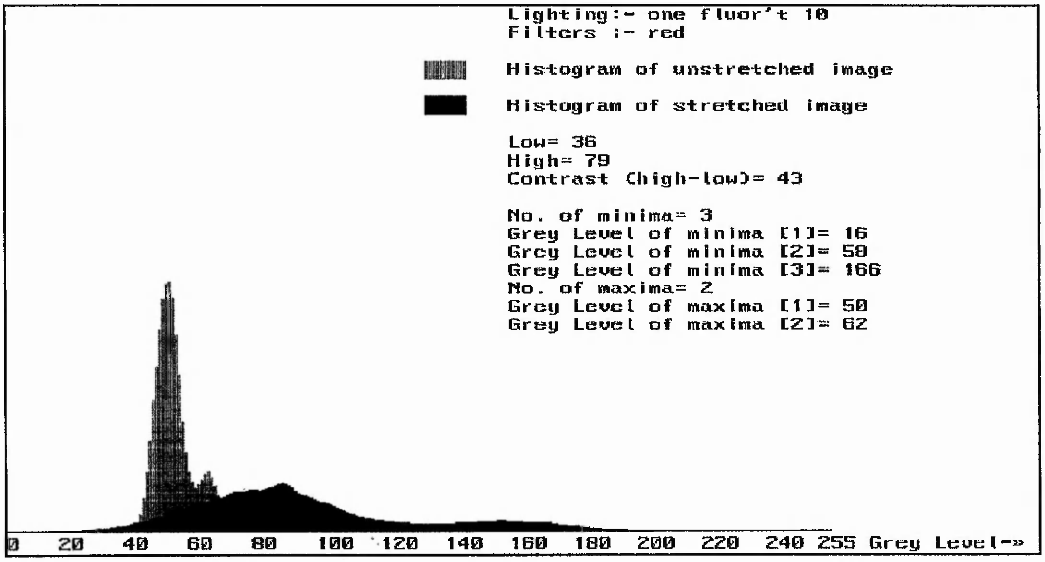


```
Lighting:- one fluor't 45
Filters :- none

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 44
High= 84
Contrast (high-low)= 40

No. of minima= 3
Grey Level of minima [1]= 20
Grey Level of minima [2]= 57
Grey Level of minima [3]= 168
No. of maxima= 2
Grey Level of maxima [1]= 52
Grey Level of maxima [2]= 66
```

Peaks = 2

Glare = 2

Contrast = 3



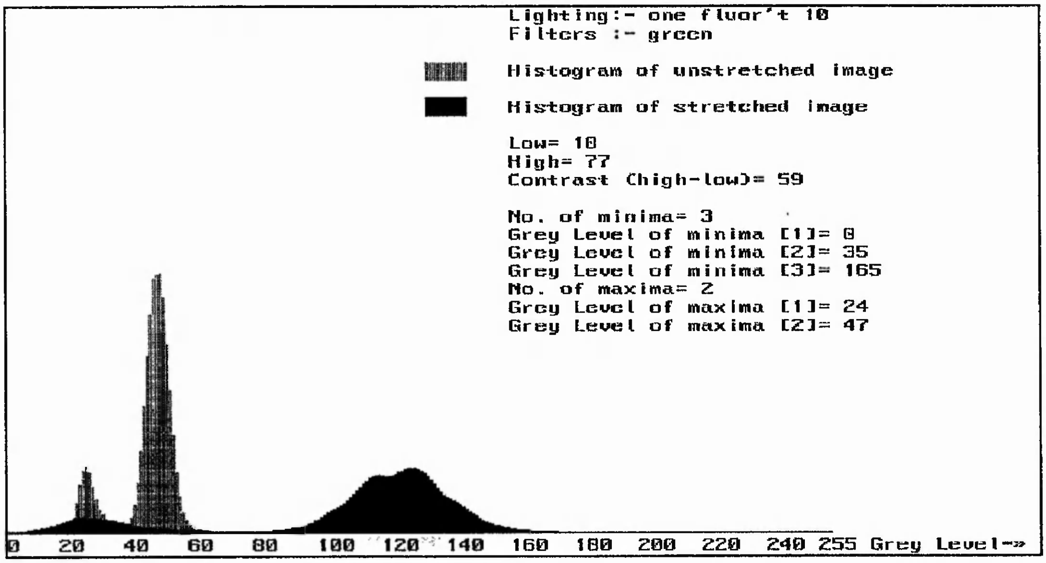
```
Lighting:- one fluor't 10
Filters :- red

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 36
High= 79
Contrast (high-low)= 43

No. of minima= 3
Grey Level of minima [1]= 16
Grey Level of minima [2]= 58
Grey Level of minima [3]= 166
No. of maxima= 2
Grey Level of maxima [1]= 50
Grey Level of maxima [2]= 62
```

Peaks = 2

Glare = 2

Contrast = 4

Very clear, lighter than substrate.



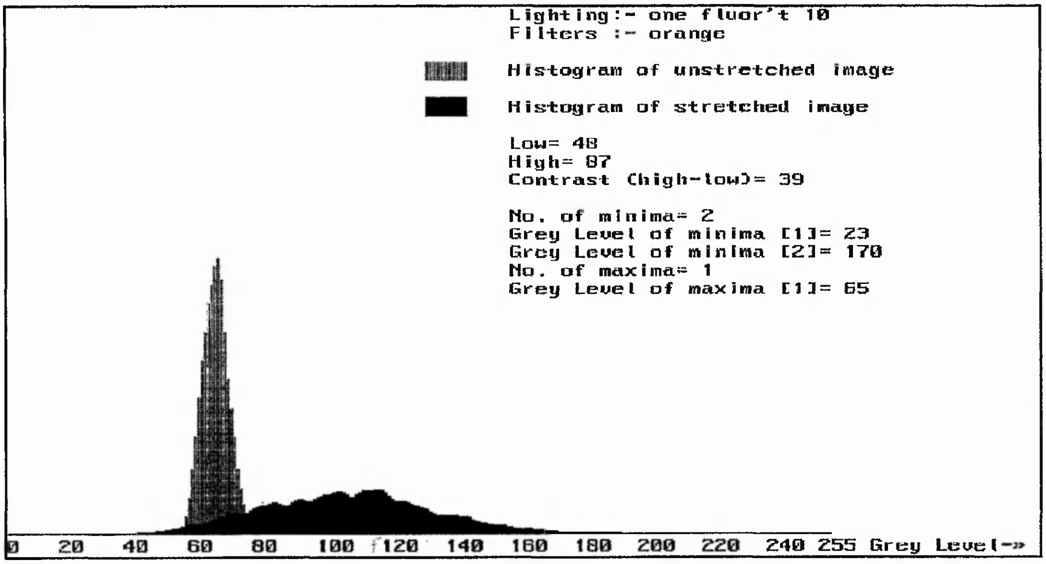
```
Lighting:- one fluor't 10
Filters :- green

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 18
High= 77
Contrast (high-low)= 59

No. of minima= 3
Grey Level of minima [1]= 8
Grey Level of minima [2]= 35
Grey Level of minima [3]= 165
No. of maxima= 2
Grey Level of maxima [1]= 24
Grey Level of maxima [2]= 47
```

Peaks = 2

Glare = 2

Contrast = 4

Very clear, with highly distinct peaks.



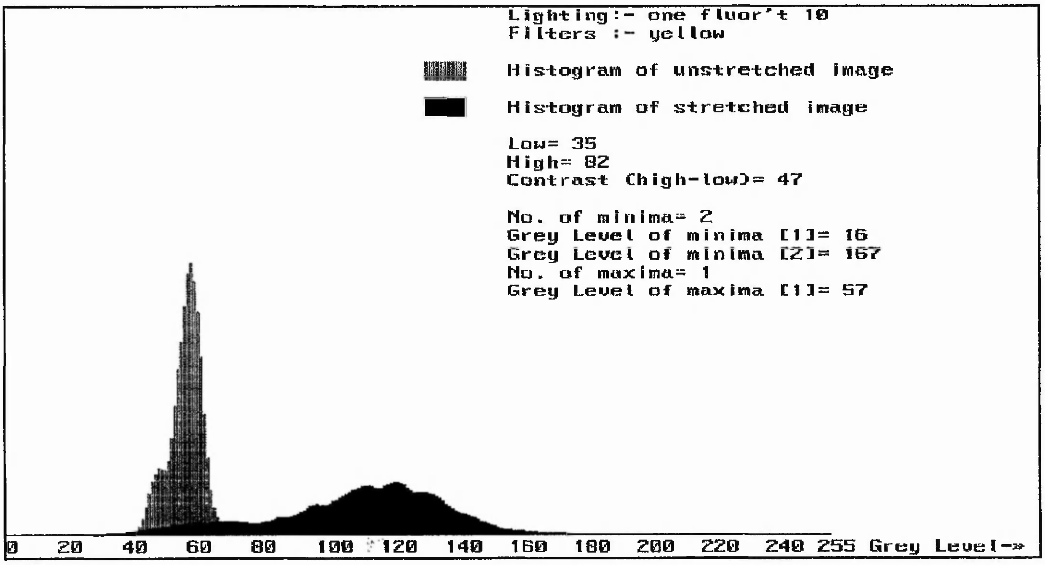
```
Lighting:- one fluor't 10
Filters :- orange

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 48
High= 87
Contrast (high-low)= 39

No. of minima= 2
Grey Level of minima [1]= 23
Grey Level of minima [2]= 170
No. of maxima= 1
Grey Level of maxima [1]= 65
```

Peaks = 1

Glare = 2

Contrast = 2

Mark barely visible to naked eye, lighter than substrate.



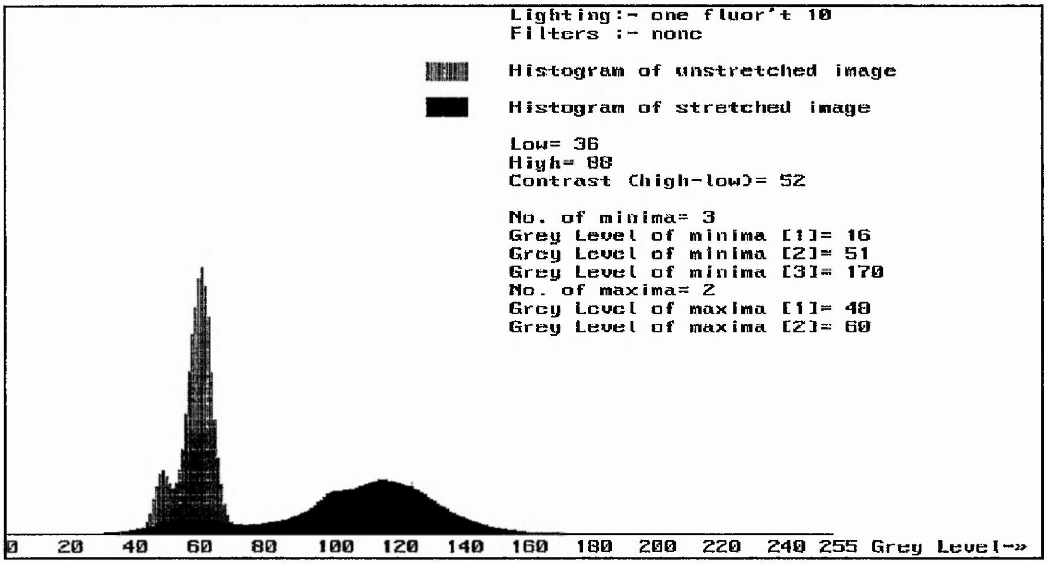
```
Lighting:- one fluor't 10
Filters :- yellow

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 35
High= 82
Contrast (high-low)= 47

No. of minima= 2
Grey Level of minima [1]= 16
Grey Level of minima [2]= 167
No. of maxima= 1
Grey Level of maxima [1]= 57
```

Peaks = 1

Glare = 2

Contrast = 2

Mark just visible to naked eye, unclear on histogram.

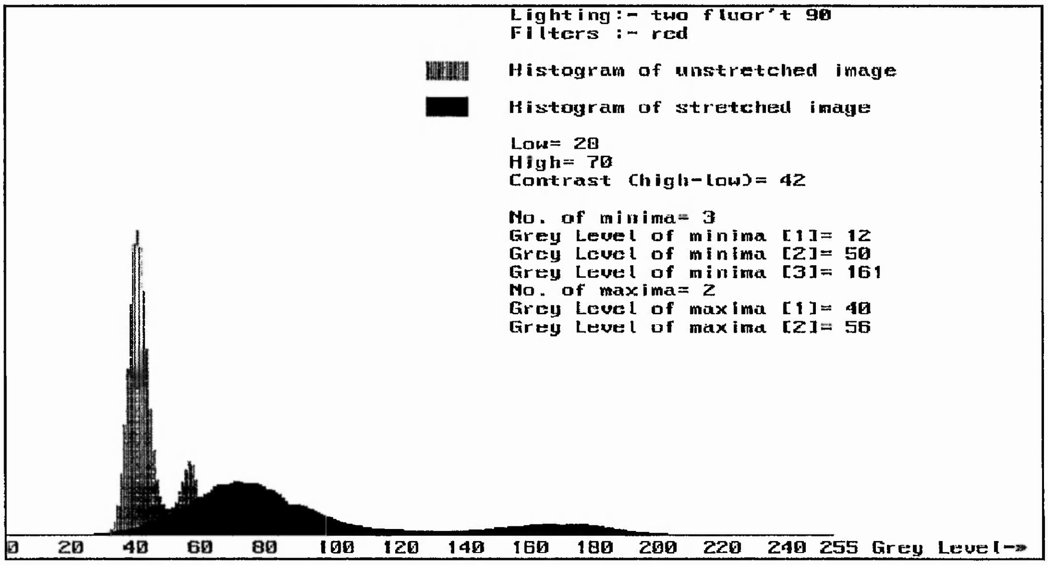


```
Lighting:- one fluor't 10
Filters :- none

▨  Histogram of unstretched image

■  Histogram of stretched image

Low= 36
High= 88
Contrast (high-low)= 52

No. of minima= 3
Grey Level of minima [1]= 16
Grey Level of minima [2]= 51
Grey Level of minima [3]= 170
No. of maxima= 2
Grey Level of maxima [1]= 40
Grey Level of maxima [2]= 60
```

Peaks = 2

Glare = 2

Contrast = 3



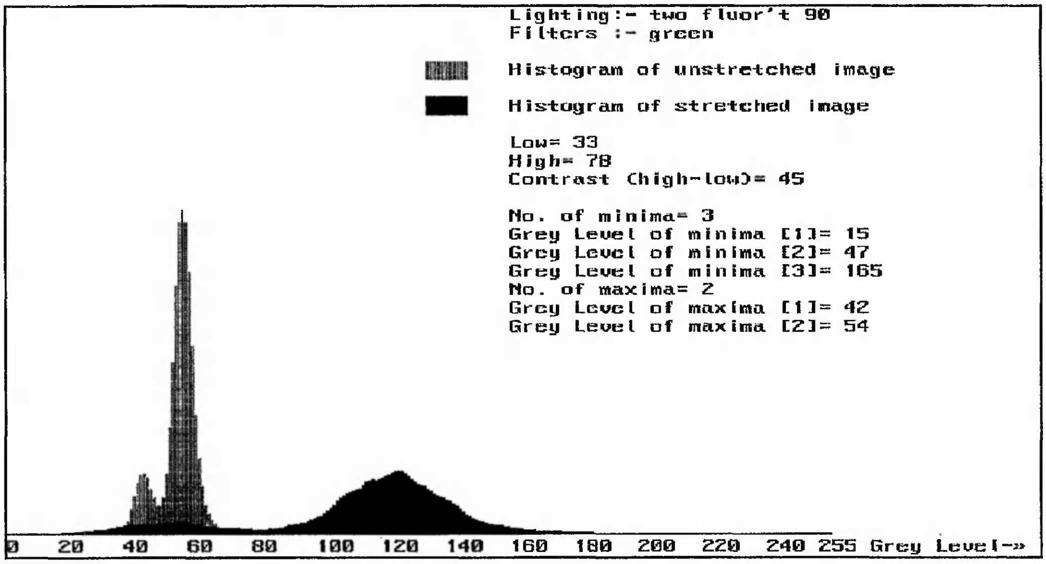
```
Lighting:- two fluor't 90
Filters :- red

▨  Histogram of unstretched image

■  Histogram of stretched image

Low= 28
High= 70
Contrast (high-low)= 42

No. of minima= 3
Grey Level of minima [1]= 12
Grey Level of minima [2]= 50
Grey Level of minima [3]= 161
No. of maxima= 2
Grey Level of maxima [1]= 40
Grey Level of maxima [2]= 56
```

Peaks = 2

Glare = 3

Contrast = 4

Mark very clear, lighter than substrate.



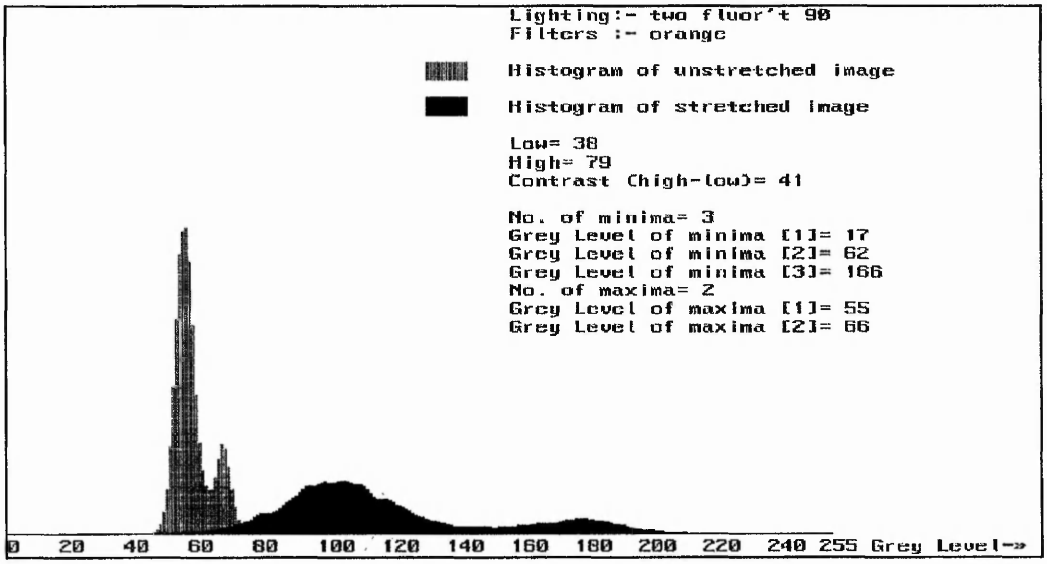
```
Lighting:- two fluor't 90
Filters :- green

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 33
High= 78
Contrast (high-low)= 45

No. of minima= 3
Grey Level of minima [1]= 15
Grey Level of minima [2]= 47
Grey Level of minima [3]= 165
No. of maxima= 2
Grey Level of maxima [1]= 42
Grey Level of maxima [2]= 54
```

Peaks = 2

Glare = 3

Contrast = 4

Mark very clear.



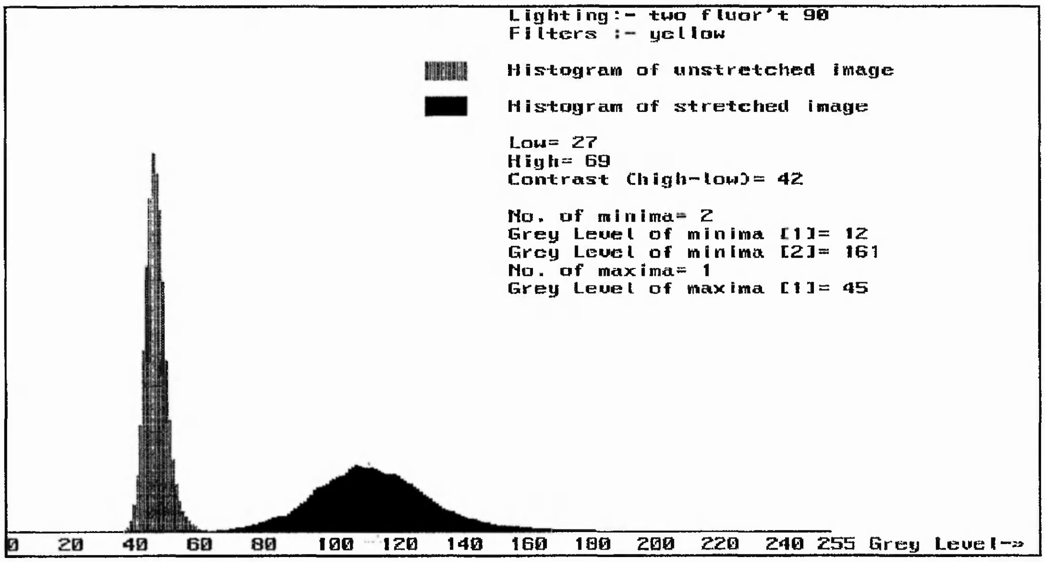
```
Lighting:- two fluor't 90
Filters :- orange

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 38
High= 79
Contrast (high-low)= 41

No. of minima= 3
Grey Level of minima [1]= 17
Grey Level of minima [2]= 62
Grey Level of minima [3]= 166
No. of maxima= 2
Grey Level of maxima [1]= 55
Grey Level of maxima [2]= 66
```

Peaks = 2

Glare = 3

Contrast = 4

Mark very clear, lighter than substrate.



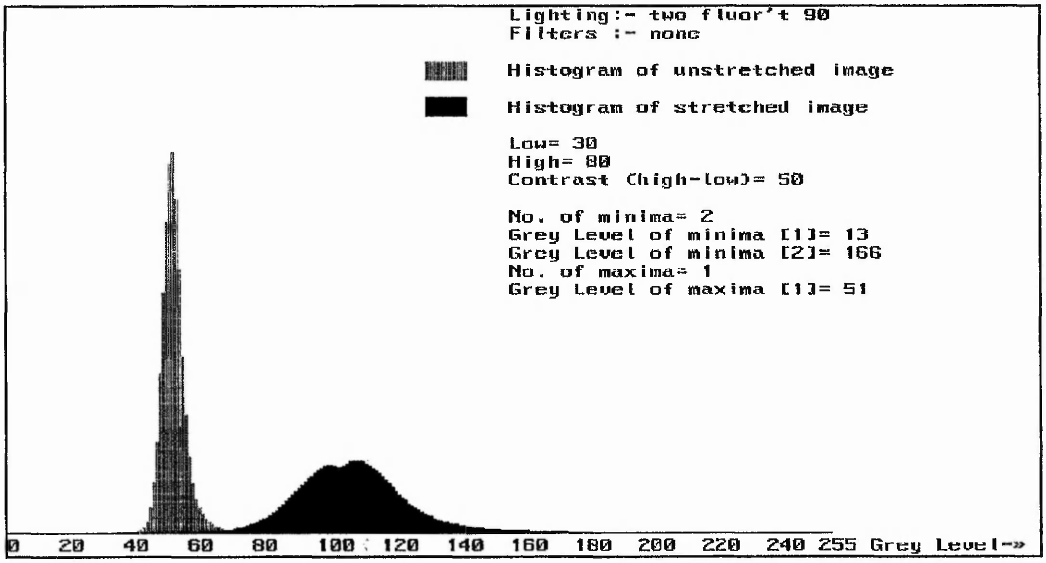
```
Lighting:- two fluor't 90
Filters :- yellow

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 27
High= 69
Contrast (high-low)= 42

No. of minima= 2
Grey Level of minima [1]= 12
Grey Level of minima [2]= 161
No. of maxima= 1
Grey Level of maxima [1]= 45
```

Peaks = 2

Glare = 4

Contrast = 1

Mark not visible high glare.



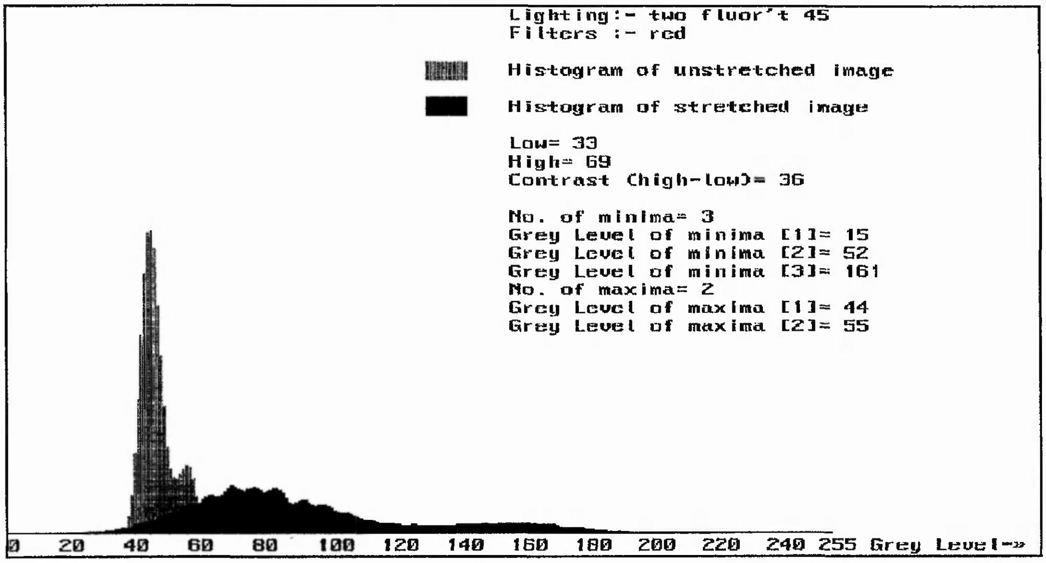
```
Lighting:- two fluor't 90
Filters :- none

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 30
High= 80
Contrast (high-low)= 50

No. of minima= 2
Grey Level of minima [1]= 13
Grey Level of minima [2]= 166
No. of maxima= 1
Grey Level of maxima [1]= 51
```

Peaks = 2

Glare = 4

Contrast = 1

Mark not visible high glare.



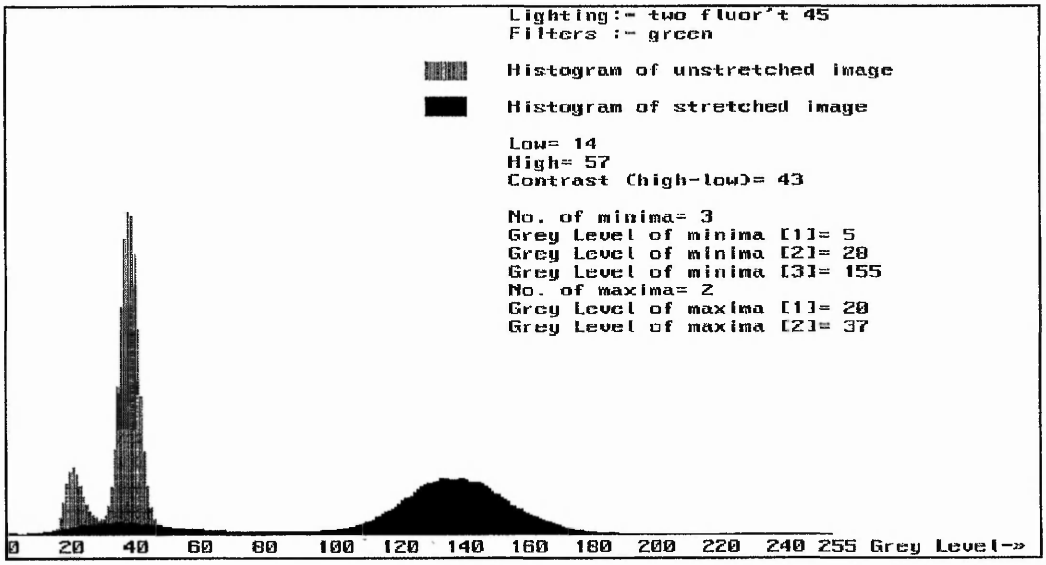
```
Lighting:- two fluor't 45
Filters :- red

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 33
High= 69
Contrast (high-low)= 36

No. of minima= 3
Grey Level of minima [1]= 15
Grey Level of minima [2]= 52
Grey Level of minima [3]= 161
No. of maxima= 2
Grey Level of maxima [1]= 44
Grey Level of maxima [2]= 55
```

Peaks = 2

Glare = 2

Contrast = 4

Mark very clear, lighter than substrate.



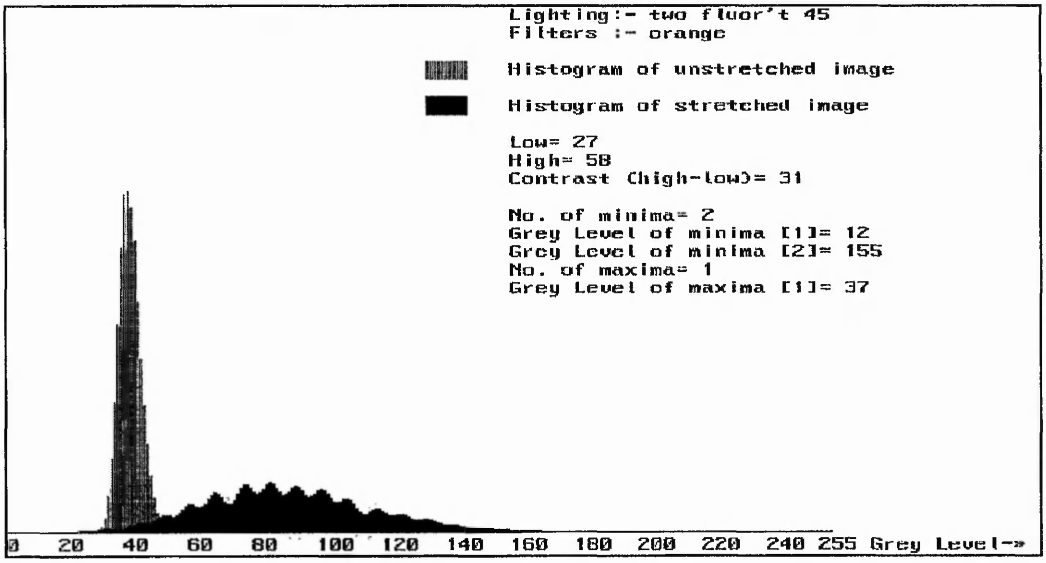
```
Lighting:- two fluor't 45
Filters :- green

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 14
High= 57
Contrast (high-low)= 43

No. of minima= 3
Grey Level of minima [1]= 5
Grey Level of minima [2]= 28
Grey Level of minima [3]= 155
No. of maxima= 2
Grey Level of maxima [1]= 20
Grey Level of maxima [2]= 37
```

Peaks = 2

Glare = 2

Contrast = 5

Mark very clear.



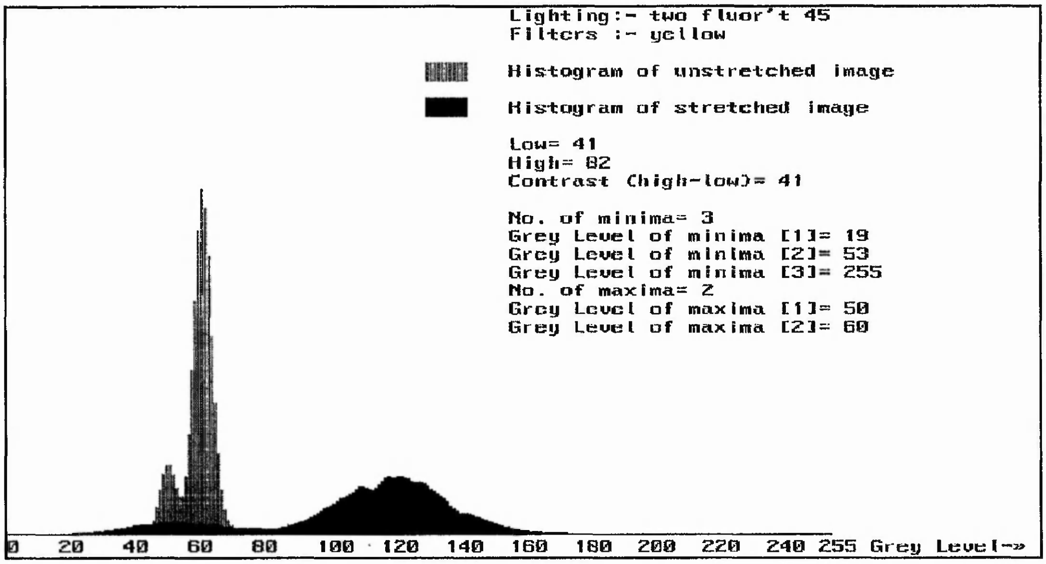
```
Lighting:- two fluor't 45
Filters :- orange

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 27
High= 58
Contrast (high-low)= 31

No. of minima= 2
Grey Level of minima [1]= 12
Grey Level of minima [2]= 155
No. of maxima= 1
Grey Level of maxima [1]= 37
```

Peaks = 1

Glare = 2

Contrast = 2

Mark barely visible to naked eye, lighter than substrate.



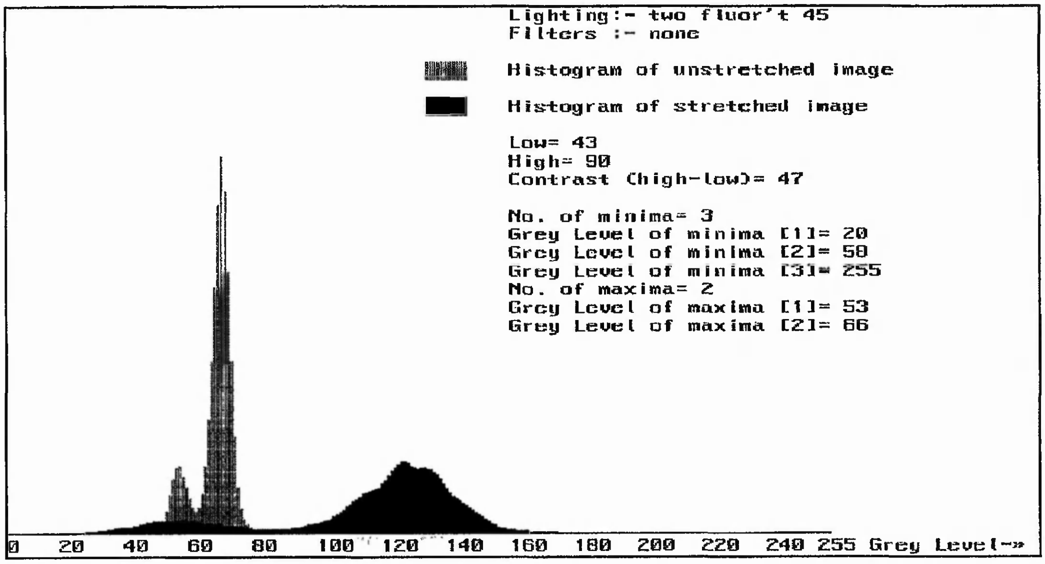
```
Lighting:- two fluor't 45
Filters :- yellow

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 41
High= 82
Contrast (high-low)= 41

No. of minima= 3
Grey Level of minima [1]= 19
Grey Level of minima [2]= 53
Grey Level of minima [3]= 255
No. of maxima= 2
Grey Level of maxima [1]= 50
Grey Level of maxima [2]= 60
```

Peaks = 2

Glare = 2

Contrast = 4

Mark very clear.



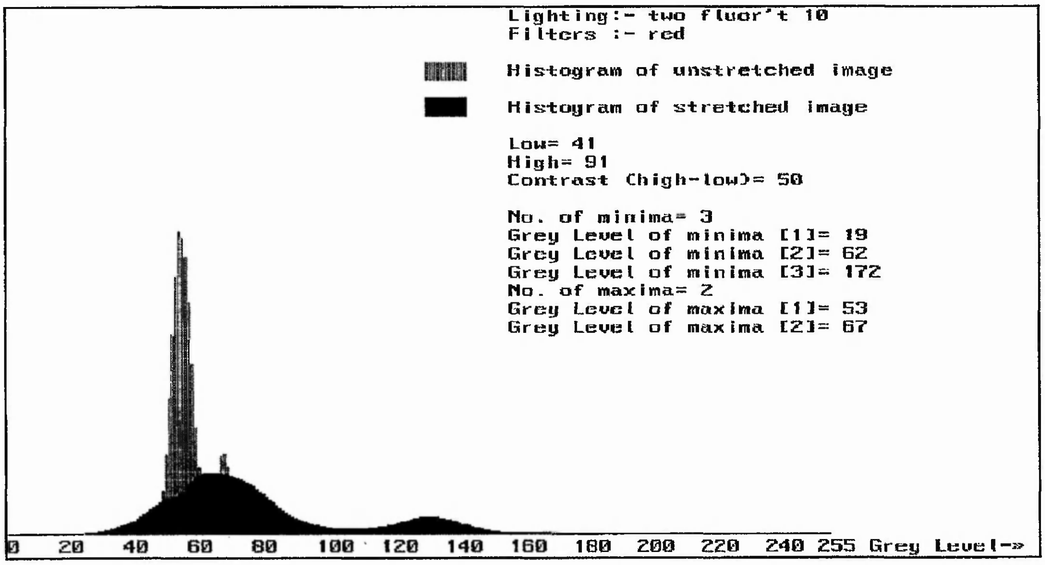
```
Lighting:- two fluor't 45
Filters :- none

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 43
High= 90
Contrast (high-low)= 47

No. of minima= 3
Grey Level of minima [1]= 20
Grey Level of minima [2]= 58
Grey Level of minima [3]= 255
No. of maxima= 2
Grey Level of maxima [1]= 53
Grey Level of maxima [2]= 66
```

Peaks = 2

Glare = 2

Contrast = 4

Mark very clear.



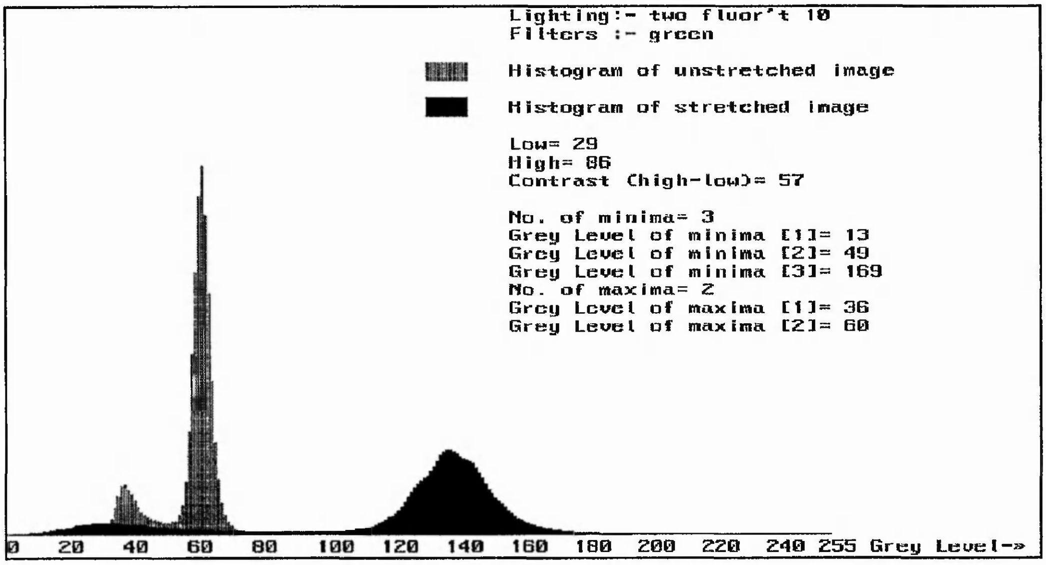
```
Lighting:- two fluor't 10
Filters :- red

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 41
High= 91
Contrast (high-low)= 50

No. of minima= 3
Grey Level of minima [1]= 19
Grey Level of minima [2]= 62
Grey Level of minima [3]= 172
No. of maxima= 2
Grey Level of maxima [1]= 53
Grey Level of maxima [2]= 67
```

Peaks = 2

Glare = 2

Contrast = 3

Mark clear, lighter than substrate.



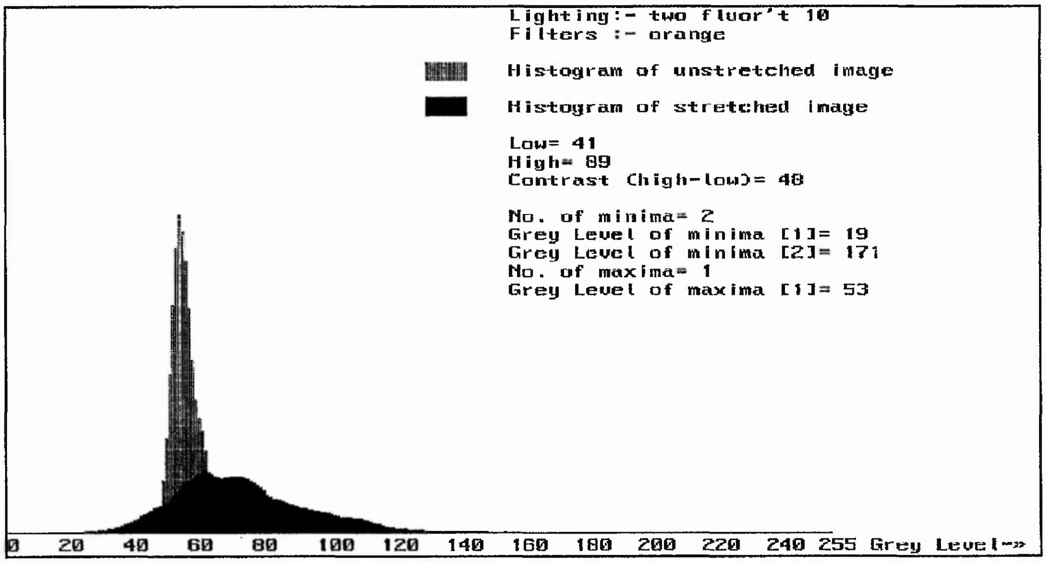
```
Lighting:- two fluor't 10
Filters :- green

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 29
High= 86
Contrast (high-low)= 57

No. of minima= 3
Grey Level of minima [1]= 13
Grey Level of minima [2]= 49
Grey Level of minima [3]= 169
No. of maxima= 2
Grey Level of maxima [1]= 36
Grey Level of maxima [2]= 60
```

Peaks = 2

Glare = 2

Contrast = 5

Mark very clear.



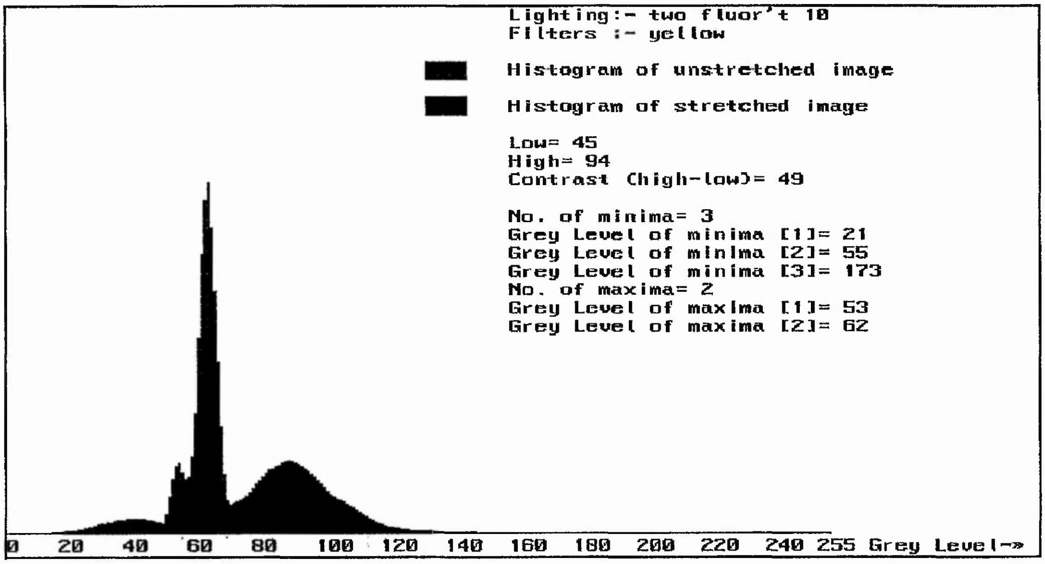
```
Lighting:- two fluor't 10
Filters :- orange

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 41
High= 89
Contrast (high-low)= 48

No. of minima= 2
Grey Level of minima [1]= 19
Grey Level of minima [2]= 171
No. of maxima= 1
Grey Level of maxima [1]= 53
```

Peaks = 1

Glare = 2

Contrast = 2

Mark barely visible to naked eye, lighter than substrate.

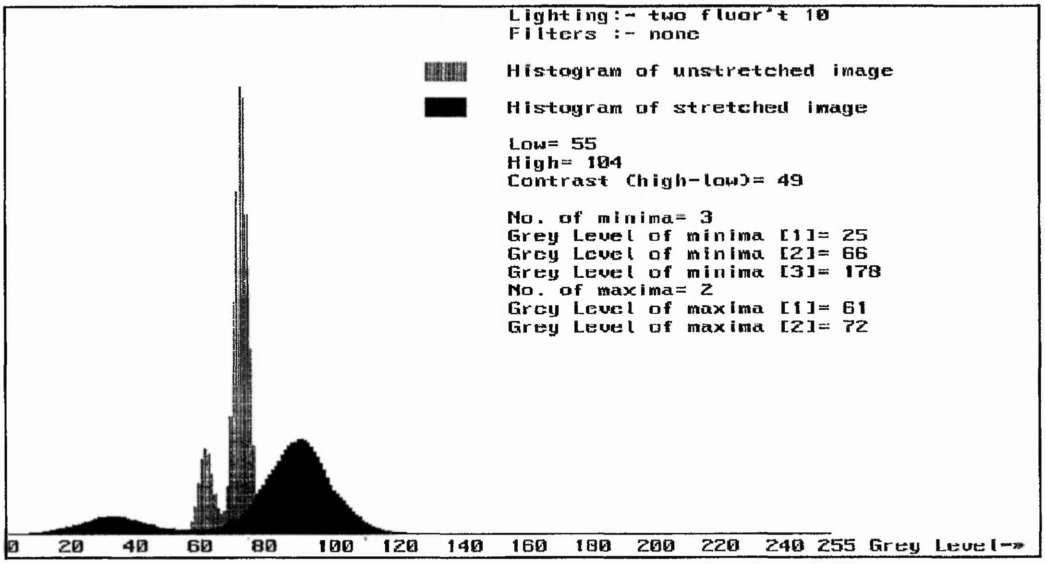


```
Lighting:- two fluor't 10
Filters :- yellow

■  Histogram of unstretched image

■  Histogram of stretched image

Low= 45
High= 94
Contrast (high-low)= 49

No. of minima= 3
Grey Level of minima [1]= 21
Grey Level of minima [2]= 55
Grey Level of minima [3]= 173
No. of maxima= 2
Grey Level of maxima [1]= 53
Grey Level of maxima [2]= 62
```

Peaks = 2

Glare = 2

Contrast = 3

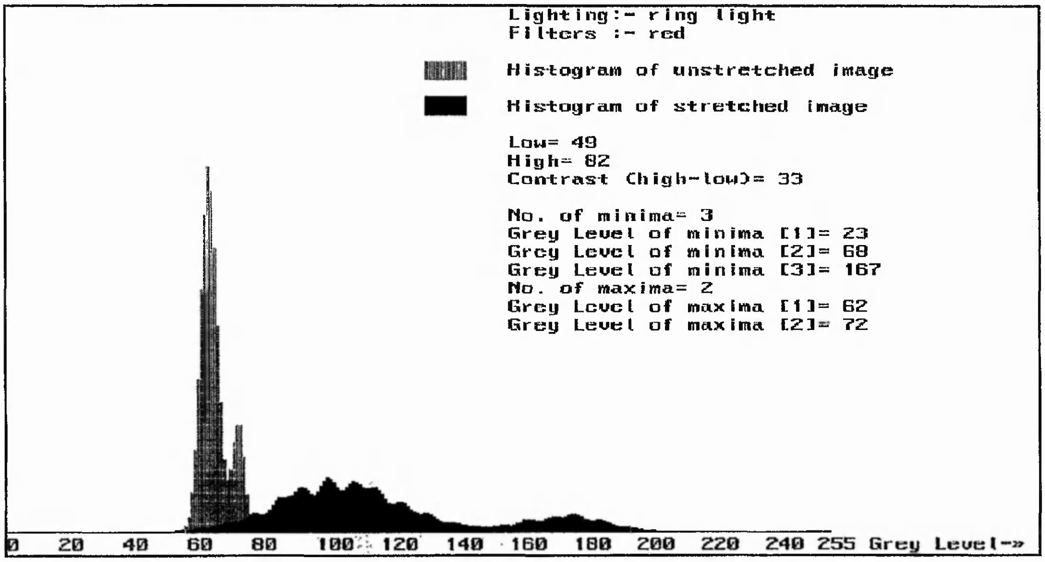


```
Lighting:- two fluor't 10
Filters :- none

▨  Histogram of unstretched image

■  Histogram of stretched image

Low= 55
High= 104
Contrast (high-low)= 49

No. of minima= 3
Grey Level of minima [1]= 25
Grey Level of minima [2]= 66
Grey Level of minima [3]= 178
No. of maxima= 2
Grey Level of maxima [1]= 61
Grey Level of maxima [2]= 72
```

Peaks = 2

Glare = 2

Contrast = 3



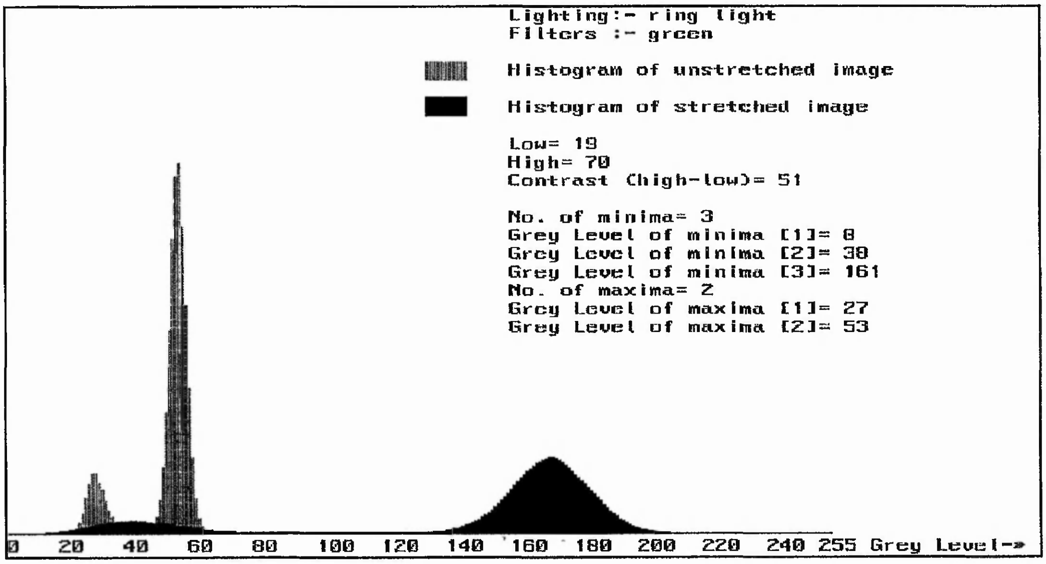
```
Lighting:- ring light
Filters :- red

▥ Histogram of unstretched image

■ Histogram of stretched image

Low= 49
High= 82
Contrast (high-low)= 33

No. of minima= 3
Grey Level of minima [1]= 23
Grey Level of minima [2]= 68
Grey Level of minima [3]= 167
No. of maxima= 2
Grey Level of maxima [1]= 62
Grey Level of maxima [2]= 72
```

Peaks = 2

Glare = 1

Contrast = 3

Lighter than substare.



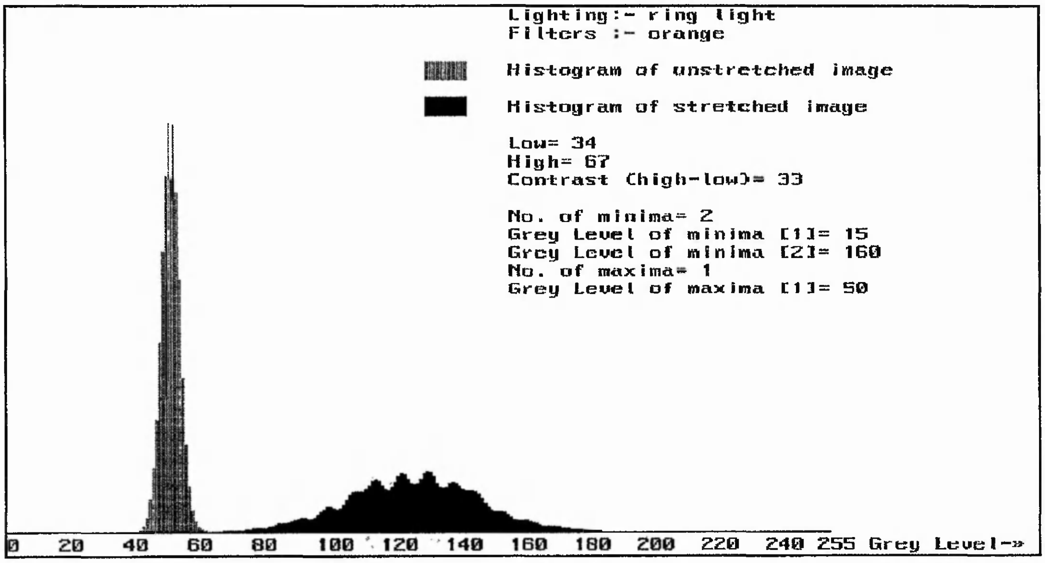
```
Lighting:- ring light
Filters :- green

▥ Histogram of unstretched image

■ Histogram of stretched image

Low= 19
High= 70
Contrast (high-low)= 51

No. of minima= 3
Grey Level of minima [1]= 8
Grey Level of minima [2]= 38
Grey Level of minima [3]= 161
No. of maxima= 2
Grey Level of maxima [1]= 27
Grey Level of maxima [2]= 53
```

Peaks = 2

Glare = 1

Contrast = 5

The best contrast seen.



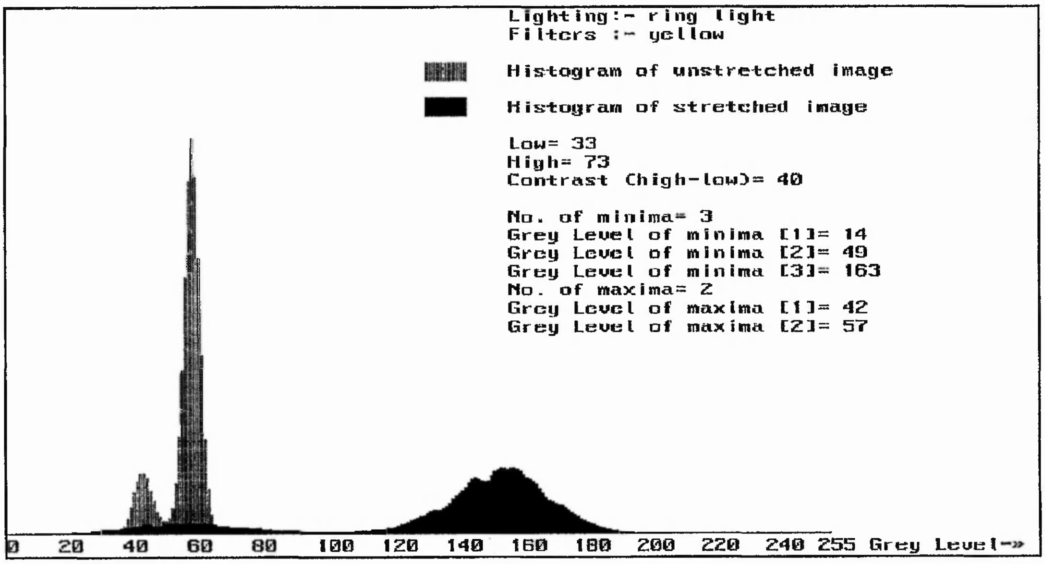
```
Lighting:- ring light
Filters :- orange

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 34
High= 67
Contrast (high-low)= 33

No. of minima= 2
Grey Level of minima [1]= 15
Grey Level of minima [2]= 160
No. of maxima= 1
Grey Level of maxima [1]= 50
```

Peaks = 1

Glare = 2

Contrast = 2

Mark barely visible to naked eye, lighter than substrate.



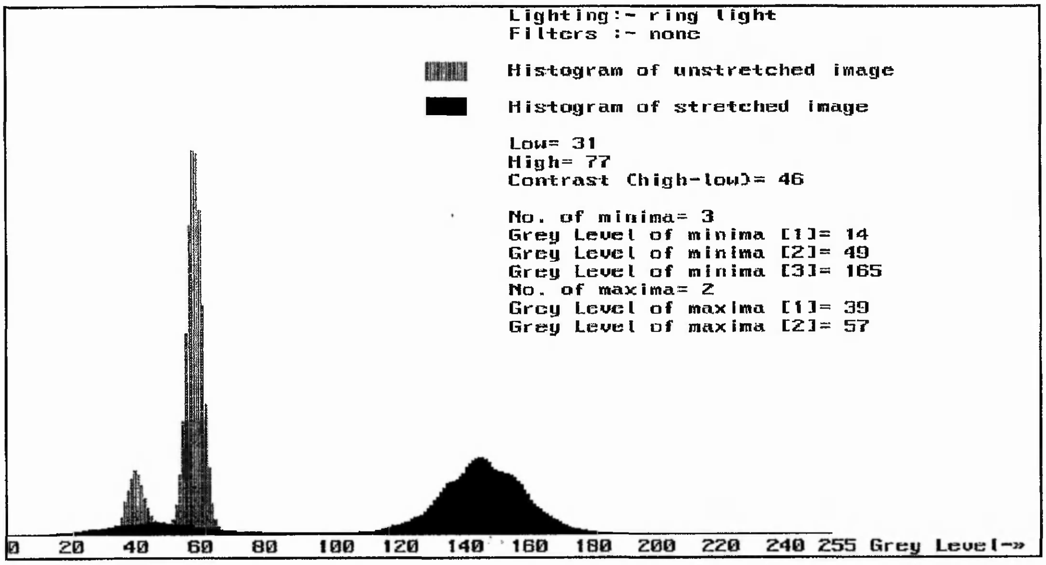
```
Lighting:- ring light
Filters :- yellow

▨ Histogram of unstretched image

■ Histogram of stretched image

Low= 33
High= 73
Contrast (high-low)= 40

No. of minima= 3
Grey Level of minima [1]= 14
Grey Level of minima [2]= 49
Grey Level of minima [3]= 163
No. of maxima= 2
Grey Level of maxima [1]= 42
Grey Level of maxima [2]= 57
```

Peaks = 2

Glare = 1

Contrast = 4

Very clear.



```
Lighting:- ring light
Filters :- none

▥   Histogram of unstretched image

■   Histogram of stretched image

Low= 31
High= 77
Contrast (high-low)= 46

No. of minima= 3
Grey Level of minima [1]= 14
Grey Level of minima [2]= 49
Grey Level of minima [3]= 165
No. of maxima= 2
Grey Level of maxima [1]= 39
Grey Level of maxima [2]= 57
```

Peaks = 2

Glare = 1

Contrast = 4

Very clear.

Appendix 6

Automated Screen Printing Machine Specifications

System Specification

• Printing Specifications

Machine	Sveciamatic SM
Maximum Screen Frame Size	1000 mm x 1000 mm
Squeegee Angle	60°-90°
Screen to Work Distance	0 - 30 mm
Fiducial	Layer 1 ring 1.7 mm inside diameter Layer 2 dot 1.5 mm diameter
Screen	Monofilament Polyester 77T (77 Threads/cm)

• Printing Table Specifications

Work to Table Fixing	Work held by vacuum table after location to alignment pins.
Printing Table Size	930 mm x 800 mm
Table Cycle Times	in 6 seconds. out 4 seconds.

• Opto-Alignment Specifications

Vision Board	Matrox MVP-AT
Camera 2 off	Pulnix TM-565 CCD
Lenses 2 off	Tamron 70-210 mm Zoom
Monitor	12" Green Screen
Alignment System	3 Compumotor Stepper motors controlling X, Y, and Θ.
Magnification	40 X (board to monitor) 2 X (boardon to camera array)
Viewing Field	5 mm x 5 mm (at 40 X magnification)
Camera X, Y Range	300 mm x 600 mm
Cycle time for board/screen	1.5 seconds

position finding	
Centre accuracy	+/- 0.005 mm.
Screen position adjustment time	4 seconds

- **General Specifications**

Machine Cycle Time	30 seconds/component
Processor	386-33Mhz PC

Linked to:

Matrox MVP-AT Vision System with two cameras and monitor,
Saab Programmable Logic Controller,
Stepper Motor Drives.

Appendix 7

Published Project Work

SEVENTH NATIONAL CONFERENCE ON PRODUCTION RESEARCH

HATFIELD 3RD -5TH SEPTEMBER 1991

Use of Machine Vision in the Automation of Printed Circuit Board Manufacture

J Keat, M Howarth, J M Lowe*, M Robinson**

Department of Industrial and Production Engineering

**Department of Electrical and Electronic Engineering

Nottingham Polytechnic

Burton St.

Nottingham

NG1 4BU

ABSTRACT

Use of conducting inks and screen printing has now made it possible to produce multilayer printed circuit boards of increased complexity and reliability. High yield production of such circuitry requires a high degree of accuracy and repeatability in both set up and production, attributes that present screen printing technology is barely able to achieve. One solution to this problem is the possible integration of a machine vision system with an automated screen printing process.

This paper outlines the automation of a standard screen printing machine using a vision system and associated technology. This is achieved with the use of machine vision and developed software, to aid in the automatic registration of the screen to the component. The registration process includes automatic recognition and position location of the registration marks. Using this information the required movement of the screen is calculated and carried out using computer controlled actuators is calculated, and includes compensation for typical process errors such as screen stretch and camera setup.

1. INTRODUCTION

Multilayered printed circuit boards (PCB's), until recently, have been produced by taking standard single or double sided copper etch boards and bonding them together using glass epoxy sheets 'Kear (1987)', with plated through holes providing contact between layers. This production route requires a number of different production processes and results in high reject rates. Poorly fabricated multilayer boards are also prone to failure at the through hole points where improperly cured epoxy degrades after a time. This fault is not often shown up by inspection and this leads to unreliable boards.

Developements in conducting inks and screen printing technology in recent years have made it possible to produce multilayered PCB's of increased complexity 'Carlisle (1988)', but with greatly increased reliability, and at a fraction of the cost of fabricated multilayer PCB's. The application of screen printing to the production of multilayer boards requires fewer production operations giving increased production rates and lower production rejects.

The production of multilayered boards using screen printing involves printing tracks with silver loaded inks insulated from other layers by printed dielectric, it also gives the ability to print resistors and capacitors. The number of layers per board that can be produced has been limited by screen printing technology. Screen printing is widely used by the textile and graphics industries who as a whole tend not to require the precision demanded by the electronics industry, the small number of printing machines that have been produced with the electronics industry in mind are very expensive and in general fail to fulfil the requirements of high accuracy automated multilayer PCB production by screen printing,which demands automatic alignment in order to give the required production rates to be economical 'Schatz (1988)',The object of this project is to adapt a standard semi-automatic screen printing machine to give automated production having an accuracy not normally associated with screen printing.

This paper describes the techniques that were used in the implementation of machine vision to aid in the automatic registration of the screen to the component and the progress that has been made.

2. THE AUTOMATIC SET-UP AND REGISTRATION PROCESS

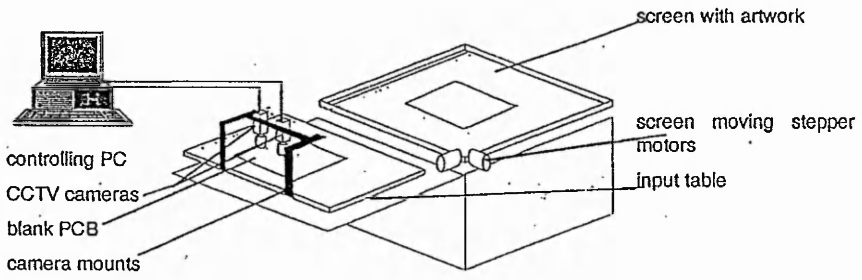
The process works as follows:

(1) Initial Set-up.

A board is placed on the machine table, to which two CCD cameras are mounted. The vision system determines the position of the registration marks to within +/-0.01mm, and then the table moves under the screen. The position of the registration marks on the screen are then found, the relative distance between the marks on the component and the marks on the screen are calculated, and this information is used to move the screen, under stepper motor control, to align with the component. A sample print is taken and the table is returned to its home position, where the position of the printed registration mark to the component is measured. The difference in positions is due to the inherent errors in screen printing, errors such as screen stretch and varying screen tension. These errors have been shown to be highly repeatable for a given set of printing parameters. Once the errors have been quantified the screen is moved to compensate for them, the system is now ready for production.

(2) Registration.

Each time a board is placed on the input table the vision system locates its position and aligns the screen if required.



Sveciamatic Screen Printing Machine

Fig.1. Schematic of System

3. REQUIREMENTS OF THE AUTOMATIC REGISTRATION PROCESS

The registration process needs to cope with a number of problems, not just with the screen printing process, but also with boards supplied by customers, lighting and resolution of vision system.

(a) Problems with the screen printing process.

The problems of screen stretch and varying screen tension and how these were taken into account have already been mentioned. Some work has been carried out into predicting these errors but this has proved not to be sufficiently accurate, and it has proved to be simpler, quicker, and more importantly, more accurate to take a sample print as descibed above.

Problems were also found in finding the position of the printed registration mark accurately in the initial set-up, this was because in some of the prints part of the circular registration marks were missing which meant an algorithm had to be developed to predict where the centre of the circle should be. The algorithm developed used a Hough Transform to achieve this, the advantage in using this transform was that the circle centre could be found from only a small amount of edge information from the circlular registration mark.

(b) Problems with boards supplied by customers.

Some of the boards used in the process come from outside customers and may have an existing copper layer to register to or may be predrilled with the need to register relative to the holes. These boards introduce a number of problems, the two major problems so far have been poor positional repeatability of registration marks and predrilled holes on supplied boards, and the varying size of circular registration marks on the same boards (and in one case a tag on the circular mark).

The poor positional repeatabilty was overcome by giving the system the ability to register to every board that is printed. Normally in screen printing the screen and component are aligned to each other at the start of the process and the screen position remains constant for the whole production run, probably only being adjusted when errors are noticed.

The problem with varying registration mark sizes and the occurrence of different shapes was overcome using the Hough Transform detailed below 'Boyle et al.(1988)' 'Ballard (1980)'. For the Hough Transform to be used the exact size of the circular registration mark need not be known, just the range within which it falls. Any flaw will not affect the results unless it is too large.

(c) Problems with lighting.

Whenever machine vision is used lighting is always an important, usually problematic, factor. This process was no exception. Different inks and screens cause different levels of contrast to occur, but the major problem was illuminating the registration mark on the screen when ink was on it. This was overcome by backlighting the screen so that the registration mark showed up even with a small amount of ink left on the screen.

(d) Problems with resolution.

One of the aims of this project was to print to within a tolerance of +/-0.02mm. This would then require having a vision system resolution of at least 0.01mm per pixel. This level of resolution was difficult to achieve because the cameras had to be mounted at least 350mm from the table due to the design of the printer. Standard CCTV lenses proved inadequate and the desired resolution could only be achieved using a photographic 200mm macro lens with a C-mount to K-mount adapter and extension tubes, to give a resolution of less than 0.01mm per pixel, with a standard imager frame buffer size of 512 x 512 pixels.

4. HOUGH TRANSFORM

The Hough transform is a method of detecting curves by utilising the relationship between points on the curve and the parameters of the curve. This work uses the transform to detect the centre of a circular registration mark. The edge pixel information is known as well as an approximate value of the radius and an area within which the centre should be found. The controlling computer sets up a three dimensional accumulator array set to zero, the three dimensions being radius, and x and y coordinates within the centre area. The range of radius values used is between +/-15% of the approximate value obtained from x and y maxima and minima values, and for each radius value and edge pixel values the parametric equation giving the centre coordinates for a circle is used:-

$(a+R\cos\theta, b+R\sin\theta)\ \theta<=\theta\ <2\pi$ where a and b are the x and y coordinates of the edge pixel and R is the radius.

If the values fall within the centre area of the circle then the relevant position of the accumulator array is incremented. This procedure is carried out for each edge pixel and the radius value is then incremented. The process is repeated for all radius values. When all the radius values have been used the accumulator array is searched to find a maximum value. This maximum value corresponds to the centre of the circle.

```
HOUGH Accumulator array = Accum[r][x][y];
                  Edge pixel = Pixel[x][y];
for(Radius Range){
 for(each Pixel[x][y]){
        (calculate edge pixels centre Pixel[x][y]);
   if(edge pixel in centre range)increment Accum[r][x][y];
   }
}
SEARCH max_val=0;
for(Radius Range){
if(Accum[r][x][y]>max_val)centre=max[x][y];
}
registration mark centre= max[x][y];
```

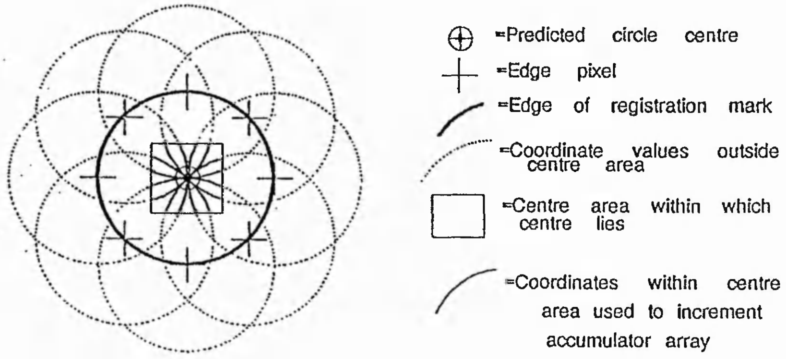

=Predicted circle centre
=Edge pixel
=Edge of registration mark
=Coordinate values outside centre area
=Centre area within which centre lies
=Coordinates within centre area used to increment accumulator array

Fig. 2 Hough Transform

5. FUTURE WORK

The automation of the screen printing will be taken beyond the registration to include automatic inspection of printed boards firstly as a quality assurance procedure and secondly as part of the control process, this inspection process will need to inspect individual layers of a multilayer thick film circuit 'Svetkoff et al. (1983)'. Information from the inspection process will be given to an expert system 'Mahmoud et al. (1988)' which will determine the cause of any faults found. When the cause is identified it will then evaluate which printing parameters need to be adjusted.It will give this information to the controlling computer in order to carry out the necessary adjustments.

6.CONCLUSIONS

The main result of the above work is the development of an automatic screen printing machine that is able to register to individual boards with a high level of accuracy, thus giving a significant improvement over other techniques for the mass production of multilayer PCB's. The ability to register the screen for each board is a facility not found on many screen printing machines even at the very top end of the market.

REFERENCES

R.D.Boyle and R.C.Thomas, "Computer Vision - A First Course", Blackwell Scientific Productions Ltd, 1988.

Ben H.Carlisle, "Screen Printing Promises Smaller Cheaper PCB's", Machine Design December 1988.

M.M.Mahmoud, S.Z.Eid, and A.A.Abou-Elsoud, "A Real Time Expert Control System for Dynamical Processes", IEEE Transactions on Systems, Man, Cybernetics, Vol. 19 No.5 September/October 1988.

David Schatz, "Automatic Alignment Aids Screen Printing", Hybrid Circuit Technology, February 1988.

Fred W.Kear, "Printed Circuit Assembly Manufacturing", Marcel Dekker, Inc. 1987.

Donald J.Svetkoff, John B.Candlish, and Peter W.Vanatta, "High Resolution Imaging of Multilayer Thick Film Circuits",Applied Machine Vision Conference, February 23-24 1983, Memphis, Tennessee, USA.

D.H.Ballard, "Generalizing the Hough Transform to Detect Arbitary Shapes", Readings in Computer Vision- Issues, Problems, Principles & Paradigms, 1980.

Birmingham Polytechnic 8 - 10 September 1992

Screen Print Automation for Printed Circuit Manufacture

M. Howarth*, D.Fulford*, J. Keat*, M. Robinson**.

*Department of Manufacturing Engineering.
**Department of Electrical and Electronic Engineering.
Nottingham Polytechnic.
Burton Street.
Nottingham.
NG1 4BU.

Abstract

Developments in the process for producing printed circuits incorporating both conductive ink and Screen Printing technology, have meant that it is now possible to produce circuits of increased variety and reliability previously not considered possible with other techniques. To assist with this, and fulfil the potential of printing such items, at the required rate, with high precision and reliability, full automatic control is required.

This paper focuses on the field of screen printing machine automation, and outlines the adaptation of a standard machine, using a vision system, a programmable logic controller, and associated sensing technologies. The system has the ability to automatically recognise registration marks and their position, filtering out excess information. The system can then automatically register the printing screen to the substrate being printed on, compensating for traditional process errors, such as screen stretch and camera set up. The system can align for every print taken, preventing registration drift, and can set its own printing parameters.

Introduction

The screen printing of PCB's is a process in its infancy, with the majority of operations being for the application of solder pastes or etch resists. The new found environmental awareness throughout the world, has resulted in legislation restricting and eventually prohibiting, the utilisation of many of the chemicals associated with the process. These include, CFC's for cleaning, heavy metal waste products from the chemical etching of copper clad sheets and lead from solder. In addition, the electronics industry is continually seeking ever tightening specifications and tolerances, in its quest for miniaturisation. This has led to an increase in the incorporation of surface mount technology, and an interest in other processes that can surpass the use of such hazardous chemicals.

Screen printing, has a reputation for producing circuit boards at a high production rate, however, in recent years, it has tended to lag behind in terms of the increased technological requirements of the industry. With the introduction of light-reactive inks and adhesives the process does however have the ability to dispense with many harsh chemicals, and gain greater control of those still in use.

With screen printing it is possible to produce multilayer circuits, at a fraction of the cost of traditionally fabricated PCB's, via the direct printing of the circuit tracks, using a conductive ink. Problems can arise, from the use of non-automatic machinery, particularly from inadequate precision. The accuracy of one layer may be to +/- 0.1 mm, but if this accuracy is sustained for, say, ten subsequent layers, then the top layer will be +/- 1.0 mm out of alignment. A surface mount circuit, would require accuracies of the order of +/-0.01 mm per layer due to the fine and close nature of the tracks. It is therefore apparent that a greater degree of control is required to increase the process accuracy to +/- 0.01mm.

This report, covers the development of a fully automated screen printing machine established by Nottingham Polytechnic's Screen Print Research Team, capable of printing to these high specifications, with a minimum of operator input, to produce high quality printed circuits, at a high production rate. The initial aim of the project was to concentrate on the set up of the machine and to achieve automatic print alignment, however, several further developments have since been achieved, outlined below.

The Automation Process.

The screen printing machine used for the development, is described as being semi-automatic, which implies that it is operator dependant. The aim of any developments to speed the output and increase the accuracy, therefore, had to attempt to reduce operator input to a minimum.

Screen Printing is a process that operates as a result of a complex interaction of a high number of variables, controlling these variables is therefore always going to present a problem, as it is usually performed by a highly skilled operative. The system described, attempts to select parameters that will minimise errors, in addition to correcting those errors that are specified as being uncontrollable.

There are three major stages to this machines development, these are all based around simulating an operators presence.

1) The operators optical analysis must be simulated, this specific task is performed by a vision system. The associated software, has the power to calculate required displacements, and to define the fiducials (or targets) for print registration from a whole array of other information. In addition, sensors situated around the machine gather information to enable the host computer to analyse print data.

2) The operators physical input to the process, must be minimised. This is performed by several devices. A Programmable Logic Controller operates the machine, where an operator would normally be required to press switches. A system of three stepper motors, align the printing screen, which is normally performed by the adjustment of micrometer hand

wheels. Various pneumatic cylinders replace clamping operations performed by hand.

3) The 'thought process' is regulated, with the introduction of an expert system, easing the selection of parameters. The Knowledge Base, for this section, contains information for the selection of printing screens and stencils, and the results of quality testing using Taguchi style techniques, giving information for high quality print production.

The Vision System

The developed automated screen printing machine uses machine vision to recognise and locate registration marks on the substrate and screen. The operation of the vision system can be divided into two distinct categories: image enhancement and image analysis. The basis of image enhancement is the capture of the best possible image and its subsequent processing to remove unwanted image information and to simplify the image to increase the performance of image analysis. It takes the form of hardware (lenses, lights, and filters) and software (noise reduction, contrast stretching, binarising etc.). Image analysis is the process of getting the desired information from the image, in this case the recognition and position of the registration mark.

The system was developed for use with circular registration marks (Keat et al., 1991) but has been improved so that it can cope with virtually any registration mark that falls within the field of view of the vision system. The system is 'taught' to identify a registration by placing an example of the registration mark in front of the vision system. The system takes an image of the mark and uses a Generalized Hough Transform to draw up a look-up table called an R-table (Illingworth and Kittler, 1987)(Ballard, 1981). The R-table classifies the image in terms of a boundary orientation angle for each boundary point with reference to a point within the image (Φ), the distance to that point (r), and the relative angle of that point to the refernce (α). Figure 1 shows the form of the R-table. This process defines a reference point on the registration mark to which subsequent print layers are registered. The system uses the R-table to search for the registration mark, and if it is found it calculates the relevant registration point.

Angle measured from figure boundary to reference point	Set of radii $\{r^k\}$ where $r = (r,\alpha)$
Φ_1	$r_1^1, r_2^1, \ldots, r_{n1}^1$
Φ_2	$r_1^2, r_2^2, \ldots, r_{n1}^2$
.	.
.	.
.	.
Φ_m	$r_1^m, r_2^m, \ldots, r_{nm}^m$

Figure 1. Layout of the R-table (Ballard, 1981).

Machine Control

The machine cycle is controlled by a Programmable Logic Controller (PLC), allowing flexibility to change and alter the operation sequences of the machine. The PLC uses various sensors to feed information back to the host computer of the state of the process (Fulford et al.,1991) These include a displacement transducer on the print head to determine the distance to the print table (snap off), which is adjusted via a D.C. motor, and an inductive sensor, adapted to operate as a shaft encoder to monitor the squeegee position.

The host computer is fed board and screen position information by the vision system in the form of pixels, which it then converts to steps, a form that can be understood by the stepper motor drives, one pixel equals 0.01mm, or approximately 100 steps. From this information it calculates the relative position of screen and board with reference to the machine coordinate system, and feeds the required movement information to each of the three stepper motors. This provides alignment of the screen in X, Y and Θ.

The system can also perform operations that an operator could not, an example is the need to overcome the problem of print screen stretch. As a print is performed, the printing mesh is stretched by the squeegee, both by the force required to overcome the snap off distance, and the frictional force of the squeegee's movement. see fig 2. The amount of stretch in the screen due to the snap off, is a function of the squeegee position and snap off distance, therefore by using the sensors listed, the squeegee speed is calculated and adjustments are made using the stepper motors to remove this displacement. If adjustments are made in the snap height, by the motor attached, the displacements are recalculated, thus making this system much more flexible and accurate than other commercially available systems.

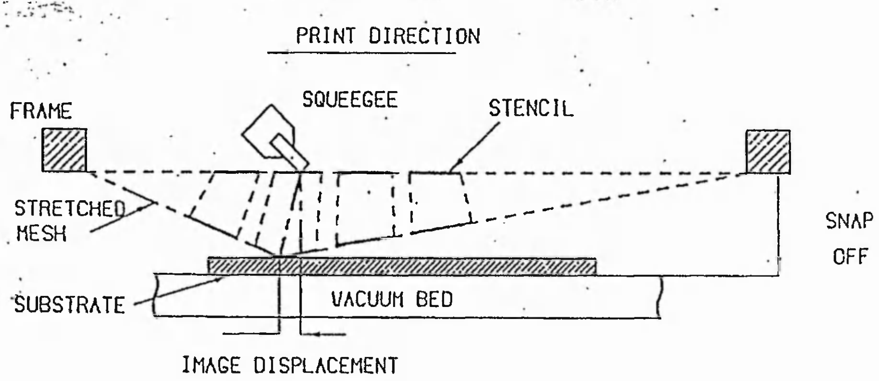


Figure 2. Diagram depicting print screen stretch

Parameter selection is performed by an expert system, containing relevant information as to the capabilities of the process and its variables, by using this, it is possible to remove variability induced by an operator selecting different variations of parameters.

The information in this system is stored in a rule base, consisting of experimental details gathered by Taguchi style experimentation, and from material information from manufacturers.

The Complete System

The final system is the culmination of all the above work, and integrates all of the devices together. The system operates by first taking a test print, this is displaced due to process errors in the system, all subsequent prints are thus offset by this amount.

The system has been used 'in house' to produce multilayer printed circuits required by another research project. The surface mount components were attached to a conductive ink pattern using a conductive adhesive. This four layer print gave a good indication of the effectiveness of the process, and proved its simplicity of use. The design to production time scale and cost is greatly reduced using such a system, giving superior accuracy and precision.

The design to production time scale and cost is greatly reduced using such a system, giving superior accuracy and precision. The production route is also much simplified, the only facilities required, in addition to the machine, are a CAD system with plotter and screen producing facilities. This makes the system ideal to both large and small manufacturers, being far more economically viable than standard multilayer board production. This production technique provides such manufacturers with a much greater degree of flexibility, as it will produce single layer boards and graphics work with equal precision. This removes the restriction on manufacturers of standard multilayer boards of manufacturing a single product.

Future Work

The system has been proven in production at a production rate of 150 boards per hour, this is limited by the electromechanical operating system and the processing speed of the vision analysis hardware. The machines performance has been optimised, and can be increased only to the detriment of the print quality and other hardware. The image analysis process time, however could be decreased by using a more powerful host computer and vision system, or by incorporating parallel processing. The present system has to wait while the computer analyses the image. By incorporating parallel processing the system would be able to carry out image analysis and process control simultaneously.

References

D. Fulford, M. Howarth, and M. Robinson, 1991, "Automating the screen print process for high accuracy, high volume, printed circuit production", *Advances in Manufacturing Technology VI*, pp. 266-70.

J. Keat, M. Howarth, and M. Robinson, 1991, "Use of machine vision in the automation of printed circuit board manufacture", *Advances in Manufacturing Technology VI*, pp. 72-6.

J. Illingworth and J. Kittler, 1987,"The adaptive Hough transform", *IEEE transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 5, pp. 690-9.

D. H. Ballard, 1981,"Generalizing the Hough transform to detect arbitary shapes", *Pattern Recognition 13*, 2, pp. 111-122.