MACHINE VISION TECHNIQUES FOR INSPECTION OF DRY-FIBRE COMPOSITE PREFORMS IN THE AEROSPACE INDUSTRY

A Thesis submitted for the degree of Doctor of Philosophy

by
Thomas A. Mitchell

Department of Manufacturing and Engineering Systems,
Brunel University.

February 1995

ABSTRACT

This thesis presents the results of a three year investigation into machine vision techniques for in-process automated inspection of dry-fibre composite preforms. Efficient texture analysis based techniques have been developed, tested, and implemented in a prototype robotic assembly cell. Industrial constraints have been considered in the development of all the algorithms described.

A single channel texture analysis model is described which can successfully segment images containing only a few textures. The model is based on convolution of the image with small kernels optimised for the task, and is elegant in the sense that it is computationally simple and easily realisable in low cost hardware. A new convolution kernel optimisation algorithm is described. It is demonstrated that convolution kernels can also be optimised to perform as edge operators in simple textured images. A novel boundary refinement algorithm is described which reduces the inspection errors inherent in texture based boundary estimates. The algorithm takes the form of a local search, using the texture estimate as a guiding template, and selects edge points by maximising a merit function. Optimum parameters for the merit function are obtained using multiple training images in conjunction with simple function optimisation algorithms.

Table of Contents.

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

CHAPTER 5

CHAPTER 6

CHAPTER 7

CHAPTER 8

CHAPTER 9

CHAPTER 10

APPENDICES

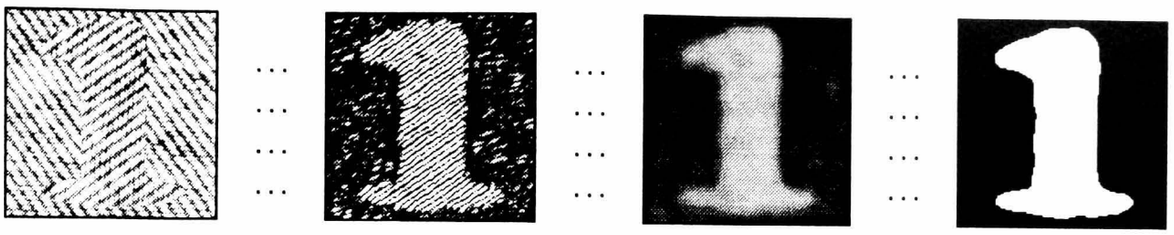
Acknowledgements.

I am indebted to Professor Mansoor Sarhadi for his supervision of the project, for finding the money whenever necessary, and for the continued support and encouragement he provides. Thanks are also due to my colleagues over the long years: Simon Jarvis, Xiao-Qi Chen, Kevin Wilcox, Stephen King, Zhenwen Zhang, Jonathan Chestney, and Gus Dewing.

In addition I gratefully acknowledge the support of the EPSRC, and contributions from the industrial collaborators Dowty Aerospace Propellers Ltd.

"I would hate to tell you what this lousy little book cost me in money and anxiety and time."

[Vonnegut, 1970]



CHAPTER 1

Introduction

1.1 Industrial Automation.

Almost since the advent of the industrial revolution itself, industry ir general and manufacturing in particular has been moving away from labou intensive methods and towards automation. Machines, as the factory owner of the nineteenth century were quick to appreciate, can perform many routine faster and more reliably than their human counterparts. Machines never ge tired or bored, lose concentration, make mistakes, go on strike, or ever demand a pay increase. Machines decrease overheads and production time increase productivity, competitiveness and profitability. As technology has improved, the number of processes which can be automated has increased correspondingly, and in many industries automation has long since ceased to be an option and become mandatory for economic survival. In other industries however, many processes are still highly labour intensive. There are two main reasons for this.

Firstly, the batch sizes involved do not justify the expense required to commission a dedicated machine. Such machines are generally characterised not only by the efficiency with which they perform their appointed task, bu

also by the lack of flexibility inherent in their design. As a result they can produce only a narrow range of components, and the financial expenditure invested in mechanisation can only be recouped if the process in question produces components which are required in very high volumes.

Secondly, there are still many things a human being can do better than any automaton. Manual dexterity, for example, has proven to be extremely difficult for robotic systems. Sensory feedback, especially visual, has demonstrated itself to be almost infinitely more complex than predicted by scientists at the dawn of the computer age. The human operator is still required in such numbers because of his sheer versatility.

These obstacles to automation are, however, beginning to erode. The advent of computers and industrial robots has introduced greatly increased flexibility into assembly lines, reducing the need for expensive dedicated machinery. Robotic assembly cells can be reprogrammed to perform a much wider variety of tasks than any machine. The field of industrial machine vision is widening all the time. Applications are springing up in a variety of industries, ranging from dashboard assembly in cars to quality assurance of fibre optics for telecommunications. As the technology progresses, the result of combining robots, computers, and sensory feedback, will be manufacturing cells which might realistically be called ***flexible manufacturing systems***. Such systems will be cost-effective for small batch sizes as well as mass production. They will be capable of handling a much wider variety of materials and components than is possible today. The advent of such systems will enable automation to become truly widespread.

This thesis presents the results of a three year investigation into machine vision techniques for in-process automated inspection in a particular manufacturing process in the aerospace industry. The components being manufactured are usually described as **advanced composite components**. The manufacturing process is known as **dry-fibre lay-up**, and constitutes the initial and most crucial stage of composite component production. The investigation was carried out as part of a larger multi-disciplinary project aimed at developing enabling technologies for automation of the dry-fibre lay-up

process. The difficulties encountered in this project are indicative of the problems which will have to be overcome before truly flexible manufacturing systems can be produced.

The remainder of this chapter will provide an introduction to the research area in more detail. **Section 1.2** introduces composite components in the aerospace industry. **Section 1.3** describes the current manufacturing process in some detail. **Section 1.4** describes the automated lay-up process adopted for this work. **Section 1.5** provides a brief introduction to machine vision. **Section 1.6** outlines the particular problems faced in this machine vision application. **Section 1.7** defines the criteria for the inspection process. **Section 1.8** provides and overview of the thesis, and indicates how the inspection criteria have been met. **Section 1.9** gives the conclusions of the chapter.

1.2 Advanced Composites in the Aerospace Industry.

The term composite refers to a material which consists of two or more distinct constituent materials. Such materials have a long history in aerospace manufacture. The very earliest aircraft embraced the idea, with structures composed largely of wood, wire and fabric. In the 1930's light aluminium alloys took over and have dominated the aircraft industry to the present day. Over the last 25 years or so however, a new class of composite material has emerged. These are known as **fibre composite** materials, and are becoming increasingly important. The emergence of fibre composite materials can be traced to two key technical developments. The first was the discovery of thermosetting resins, such as phenolics, polyesters, epoxies, etc. The second was the production of glass fibre. Thermosetting resins are useful in very many applications, but for structural use they are often either too brittle or too flexible. To overcome these difficulties filler materials such as wood fibres and asbestos were added. Although moderately successful, the real breakthrough came when polyester resins were combined with continuous glass fibres to produce laminates with attractive mechanical properties, such as high strength-to-weight and stiffness-to-weight ratios. This glass fibre reinforced

polyester laminate was the first **advanced** composite material. In recent years other fibre types have emerged, notably **carbon fibre**, as well as many other resin types. Of all current composites, carbon fibre reinforced epoxy resins are the most significant, accounting for more than 90% of all composites presently specified for aircraft construction **[Middleton, 1990]**. This is in fact the main composite material used by the company sponsoring the work presented in this thesis, Dowty Aerospace Ltd.

1.3 The Manufacturing Process.

There are several manufacturing techniques which may be used to produce composite components. The work presented in this thesis has been directed towards one particular approach which involves resin free lay-up of unidirectional and woven fabrics, known as dry fabrics, with subsequent injection of high performance resin under vacuum. Further processing of the resin injected structure is required to produce the required component. The work of the research group has been concerned only with the first stage of composite manufacture, namely the lay-up process. Resin injection and subsequent processing have not been investigated as part of this project. It should also be emphasized that only "dry" (resin free) fabrics have been considered in this work. A different manufacturing technique uses materials which are pre-impregnated with resin, called **pre-preg** materials. Lay-up of pre-preg materials has not been studied as part of this project.

The manufacture of the moulded blade (lay-up and resin injection) is referred to by Dowty as resin transfer moulding **[M^{c}Carthy,1981]**. The existing lay-up process involves manual stacking of layers of pre-cut carbon fibre sheets, commonly known as **plies**. The warp fibres are oriented to exploit their considerable strength, producing a **preform stack** (see **Figure (1.3.1)**). For more complex components, each layer can consist of between one and three individual plies, with warp fibre orientations at 0°, +/-45°, or 90° to the assembly axis, depending on the component design specifications.

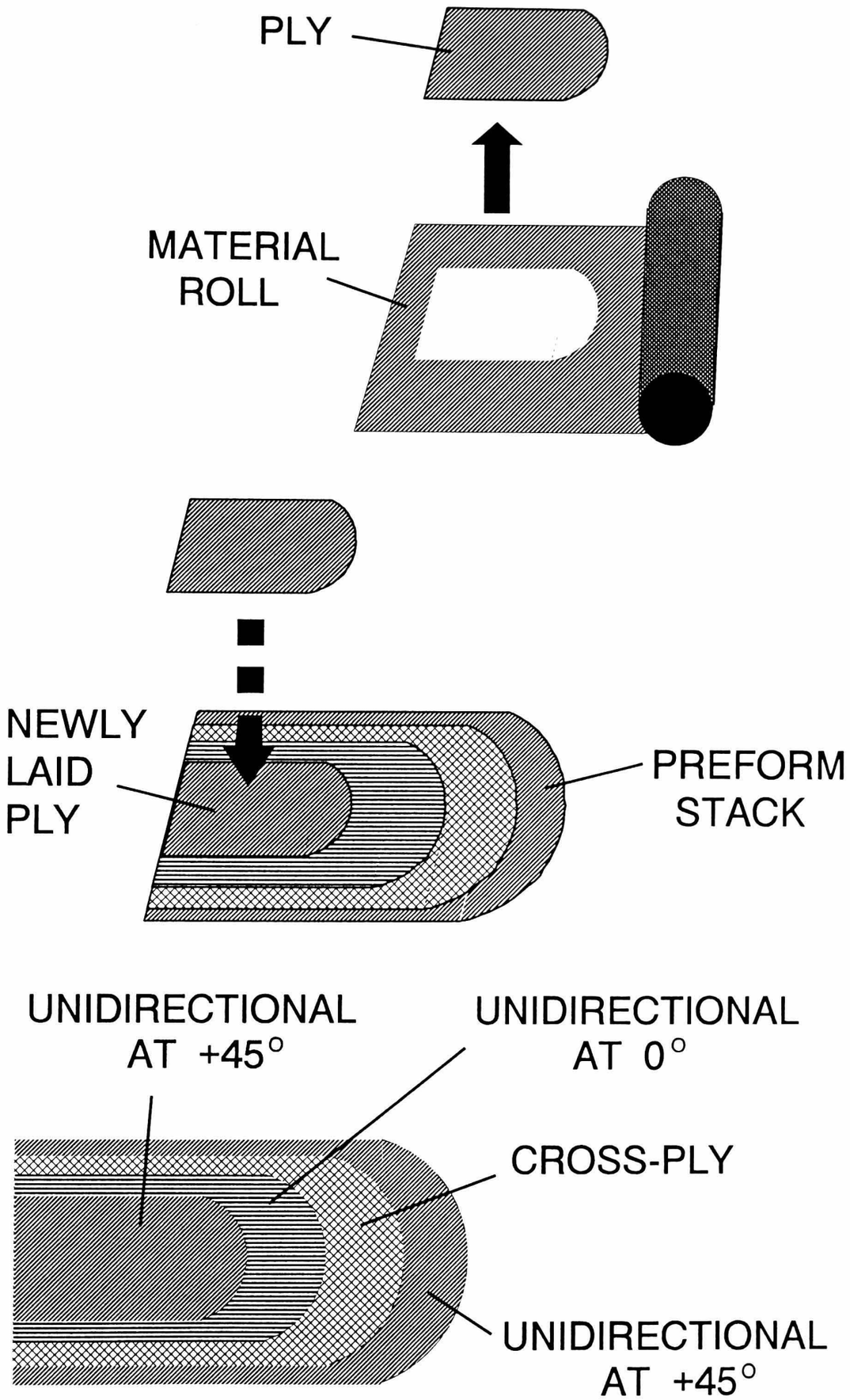


Figure (1.3.1). Each ply is cut from a roll of the appropriate material. The ply is then laid-up at the desired position on the previously laid plies, and forms part of the preform stack. Each ply is cut from a specified material type, with the fibre at a specified orientation.

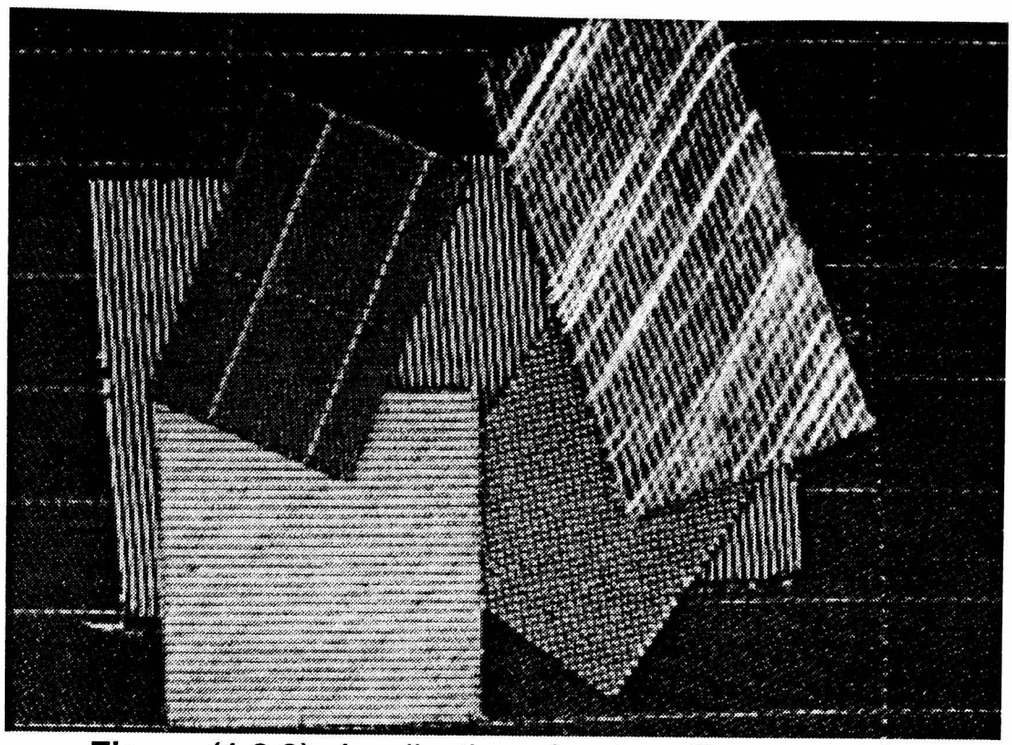


Figure (1.3.2). A collection of carbon fibre materials.

Essentially the 0° plies provide for the direct load in the principal direction, the +/- 45° plies for the torsional loads about this axis, and the 90° plies for the transverse loads. **Figure (1.3.2)** shows an image of a collection of different carbon fibre materials used in the aerospace industry. The background material is known as **woven**, and consist of two orthogonal weaves of carbon woven together. The other pieces have different weft materials, often glass fibre. For very large components such as an aircraft tailplane, ply size can be up to two metres by one metre, and the lay-up positional accuracy on the preform is nominally specified as being ±1 millimetre, although it is doubtful that this accuracy is achieved with current manual lay-up techniques. Correct positioning of a ply on a preform stack is a tedious manual operation involving frequent use of templates, and is further complicated by the requirement that no stack disturbance must occur. To this end, some materials have thermoplastic weft yarns embedded in the fabrics which enable each ply to be heat bonded into the stack prior to laying down the next ply. The existing production technique uses an electric iron for this application. A small amount of pressure is applied to the iron by the operator to cause fusion with the lower ply. The plastic cools after the iron is removed, resulting in a semi-bonded inter-ply adhesion. However, the impurities introduced by the thermoplastic are

unacceptable in some structurally critical component sections. The search for an alternative tacking method suitable for automation is therefore still a research area.

Manual lay-up of preform stacks has been established as the main bottleneck in the composite manufacturing cycle, contributing to the slow turn around times which are currently characteristic of composite manufacture. The lay-up process is tedious, and this results in a relatively high number of wrongly assembled components which are only discovered at later test stages. Defective components of this type must be scrapped, and the expensive nature of composite materials makes this a particularly undesirable occurrence. By automating the lay-up of the pre-form stack, both turn around time and rejection rate can be reduced.

1.4 The Automated Lay-Up Cycle.

One of the first tasks carried out as part of this project concerned the redesign of the lay-up process for automation **[Jarvis,1992]**. This enabled the development of a "proof-of-concept" robotic cell as a framework for investigating the effectiveness of the various techniques under consideration. The cell has evolved over the period of the project, but **Figure (1.4.1)** represents the most recent configuration. The main components of this cell are as follows: ABB IRB3000 articulated robot and controller; FANUC S10 articulated robot and controller; computer controlled cutting table and controller; electrostatic gripping device (EGD); vision system; stitching device; lay-up table. The cell is described in more detail in **Chapter Nine**.

It should be noted that the cutting process has not been investigated until very recently. It had always been assumed that commercially available automated cutting machines could be used to cut dry carbon materials to the required accuracy. It was therefore envisaged that such a machine could be interfaced to the front-end of the process once the other automation problems had been addressed. In fact more recent work in this field has shown that this assumption was not really valid, and some further development work is under way within the research group. The effect of the cutting method on the task of

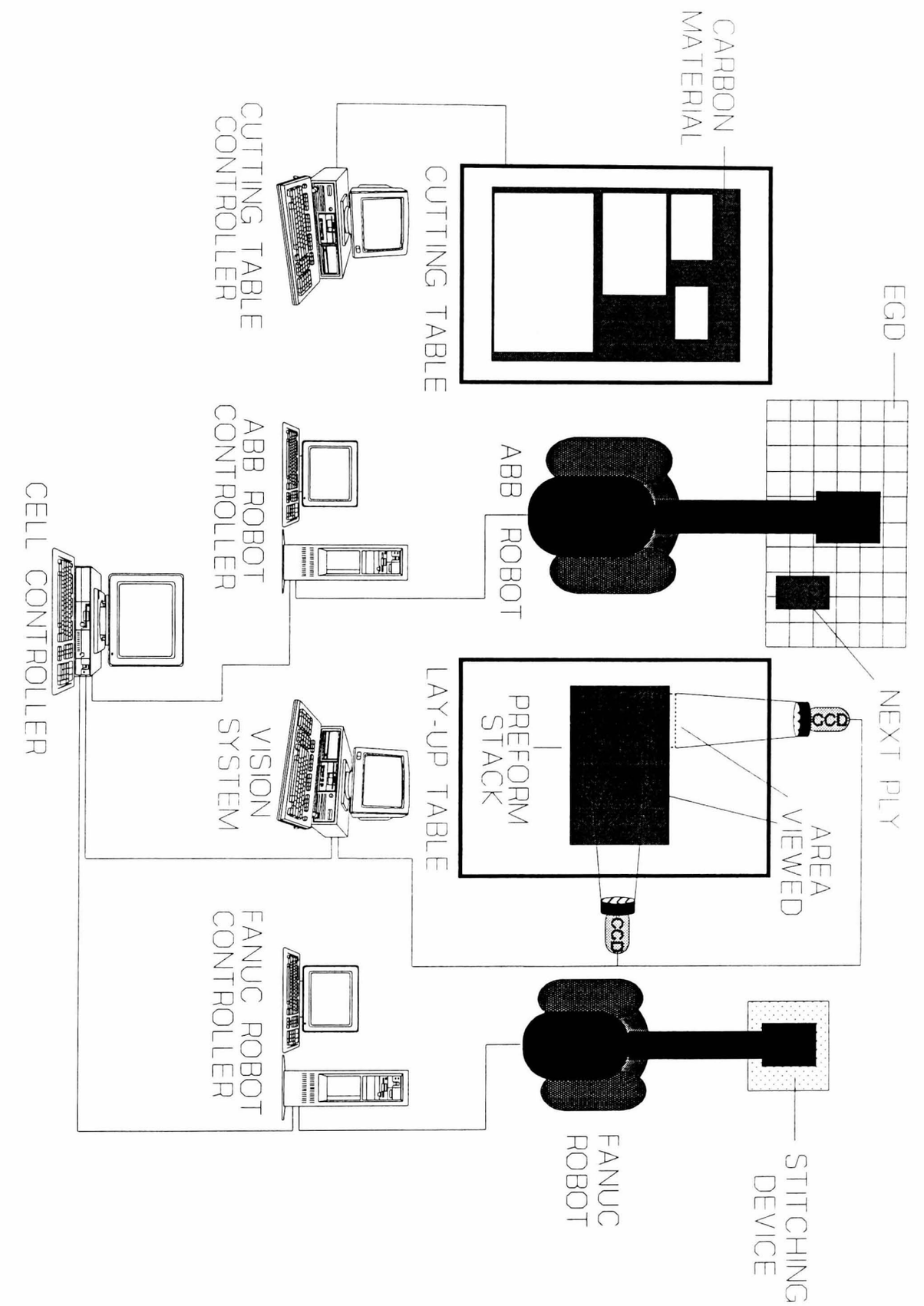


Figure (1.4.1). Lay-out of the robotic assembly cell.

the vision system is considered briefly in **Chapter Nine**. The specification of the automated lay-up cycle, based on the cell hardware illustrated in **Figure (1.4.1)**, is described as follows.

The cutting process leaves the next ply cut at a known position and orientation. The robot moves the gripper to the required position and picks the ply up. It transports the ply to the correct position over the lay-up table and places it. The gripper then moves away to a parking position to await the start of the next cycle. ***At this stage the vision system should detect the position of the ply and compare this information to the position defined in the component design specification***. The result of the inspection is related to the cell controller. If the ply has been laid-up to within the required specification, then it is tacked to the rest of the preform stack to prevent any subsequent undesirable movement. The next cycle can now begin. If, on the other hand, the ply is out-with the positional specification, then it must be re-laid. The task of the vision system is to inspect each newly laid-up ply. This is easily decomposed into three stages:

(1) Determine the position of the ply in the lay-up.
(2) Measure deviation from the design specification.
(3) Report pass or fail to the cell controller.

Stage (3) is trivial. **Stage (2)** is not, as some method of obtaining the design specification in meaningful form must be identified. A simplified means of doing this, suitable for most applications, is demonstrated in **Chapter Nine** with an example component. **Stage (1)** presents by far the most difficult task. The requirement is to find the boundaries between different layers of the same black material. The material itself is "unfriendly" in the sense that its reflective properties make it very difficult to achieve the image quality required for the successful application of machine vision techniques. The accuracy required is high, and the area covered by the plies may be considerable (up to 2 metres by 1 metre). Potentially therefore, an enormous amount of data must therefore be processed for each lay-up, and so very efficient techniques must be

developed.

1.5 Introduction to Machine Vision.

Computer vision systems, as the name suggests, utilise computers to interpret visual imagery of the physical world. The images are generally captured using electronic cameras, and digitised to produce a representation which can be processed by digital computers. Each image is then made up of many thousands of picture elements, or pixels. The task of a computer vision system is often to discriminate the various objects in a scene (image segmentation) and then make some sense of this information (scene interpretation). Such systems are used for research purposes in artificial intelligence, for automatic guidance of vehicles, for target location in military systems, for analysis of satellite imagery, for screening of medical x-rays, and for many other tasks. One of the main fields for computer vision systems is that of industrial inspection. Vision systems for industrial applications differ from other systems in several ways. They tend to have very limited horizons, in that they might well be conceived to inspect only a single design of component. On the other hand, they are often required to accurately measure features of components, or at least to reliably recognise defective components according to some predefined criteria. Perhaps most significantly, they are usually required to process images very quickly, sometimes in fractions of a second. Such systems, perhaps due to their rather dedicated inflexible nature, are often termed **machine vision** systems.

Perhaps the most reported machine vision system has been General Motor's Consight system which used vision to control a robot involved in removing components from a conveyor belt and transferring them to a pre-determined location **[Ward et al,1979]**. Consight employed a linear array camera, binary images, structured lighting, and was only capable of dealing with non-overlapping components. A bright, narrow beam of light was projected across the conveyor belt, and the linear camera appropriately positioned to image the bright line thus created on the belt. When an object appeared on the belt, the line of light reflected on the object and not the belt

and so was not imaged by the camera. The result of this is that when the camera detected light it was viewing the conveyor belt, and when it detected dark there was an object on the belt. As the conveyor belt moved, the image built up line by line, and so did the shape of the object in the vision systems' memory. The position and orientation of the object could then be calculated and used to enable the robot to pick up the component.

Consight is a typical example of machine vision in several ways. Rather than incorporating any high level intelligence regarding the scene, it employed dedicated techniques to enable high speed performance. It had no idea of colour, texture, or even 3-D shape of the object, it only formed a 2-D silhouette from above. It was designed to operate in a particular environment, and would be unable to operate in a different one. Within this environment however, it was very effective.

Most of the early machine vision systems processed only binary images, and there are two main reasons for this. Firstly, hardware limitations made grey-scale processing impractical, and secondly algorithms to process binary images are simpler to develop and faster to process, and can therefore be more sophisticated than grey-scale algorithms. An example of this second point can be found in **[Bolles,Cain,1982]**, where a method known as local feature focus is described. This involves the vision system being trained to recognise objects by the relative placement of features, such as holes and corners, in a thresholded image. The speed with which holes and corners can be detected in a binary image enable quite sophisticated feature matching algorithms to be applied to increase reliability. With grey-scale images it is both slower and more difficult to detect such features.

Binary images are of course only appropriate in a relatively small number of potential machine vision applications. The majority require grey-scale processing at least in the initial stages. One of the first examples of a grey-scale machine vision system in industry was Keysight **[Rossol,1981]**. Keysight had a 64x64 resolution 4-bit framestore, and was designed to inspect valve spring assemblies. This involved quite complex processing, including the application of edge operators, a heuristic thinning algorithm, centroid

calculation and template matching. Several keysight systems were used in production.

Grey-scale processing is more feasible today mainly thanks to the considerable advances in hardware in recent years. For example, the framegrabber used throughout this project has a resolution of 720x512 with 8-bit grey-scale, and in fact 1024x1024 resolution framestores and higher are now commercially available. The processing power has also increased dramatically. Commercially available hardware can perform histogramming, thresholding, image convolution etc. in milliseconds, enabling real-time operation of certain machine vision tasks. The latest general purpose processors (e.g. Intel i860) provide processing power which would have been unthinkable twenty years ago. In common with other areas of computer science, the progress on the software side has been less dramatic. Complex pattern recognition problems, such as texture analysis, object recognition, 3D scene understanding etc. have proven extremely difficult to solve. Machine vision techniques are still far from universally applicable. Generic algorithms have yet to emerge, and the approach adopted to solve a particular problem depends heavily on the nature of that problem.

1.6 Problems Specific to this Inspection Application.

This section will outline the particular problems involved in inspection of dry carbon fibre workpieces, and explain why conventional techniques were not suitable for the application.

The first stage required for in-process ply inspection is essentially that of boundary detection. Perhaps the two most common approaches to this problem are **(a)** application of a thresholding technique, followed by boundary extraction of the objects in the resultant binary image, **(b)** application of edge operator(s) followed by some form of post-processing to select the relevant edge segments and form them into object boundaries.

For inspection of carbon fibre lay-ups, method **(a)** is not feasible. The plies are made of the same material, and so different plies exhibit an almost identical range of grey-levels. A histogram of an image of the lay-up is

essentially uni-modal. Method **(b)**, application of edge operators, is a more generally applicable technique than thresholding, but the results of an investigation into this approach produced equally poor results **[Mitchell, 1990]**. The reason for this is described briefly below.

A text book definition of an edge is "a small area in the image where the local grey-levels are changing rapidly". Similarly, an edge operator is "a mathematical operator (or its computational equivalent) with a small spatial extent designed to detect the presence of a local edge in an image function" **[Ballard and Brown, 1982]**. Edge operators are usually implemented as small two-dimensional convolution masks, like those displayed in **Figure (1.6.1)**. These are referred to as **3x3** masks to indicate their spatial extent.

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad S_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Figure (1.6.1). Sobel 3x3 masks.

Various edge operators were tested on images of carbon fibre lay-ups, but none were found which were able to provide any help in boundary detection. A typical example of the resulting image is shown in **Figure (1.6.2)**. This example demonstrates the problem for inspection of carbon fabrics. Boundary information is effectively masked by the edge segments produced due to the woven nature of the cloth. The problem is that the image exhibits **texture**.

Conventional edge operators, such as those in **Figure (1.6.1)**, have been designed to strongly enhance edge elements in an image. This of course means that any edge elements present in a textured region will also be enhanced and so any boundary between textured regions will be somewhat "swamped" by all the edge information produced by the weave.

The addition of an **edge relaxation** process provided no improvement in performance **[Prager, 1980]**. In an edge relaxation scheme the strength, or **confidence**, of an edge is adjusted based on the strengths (confidences) and position of neighbouring edges.

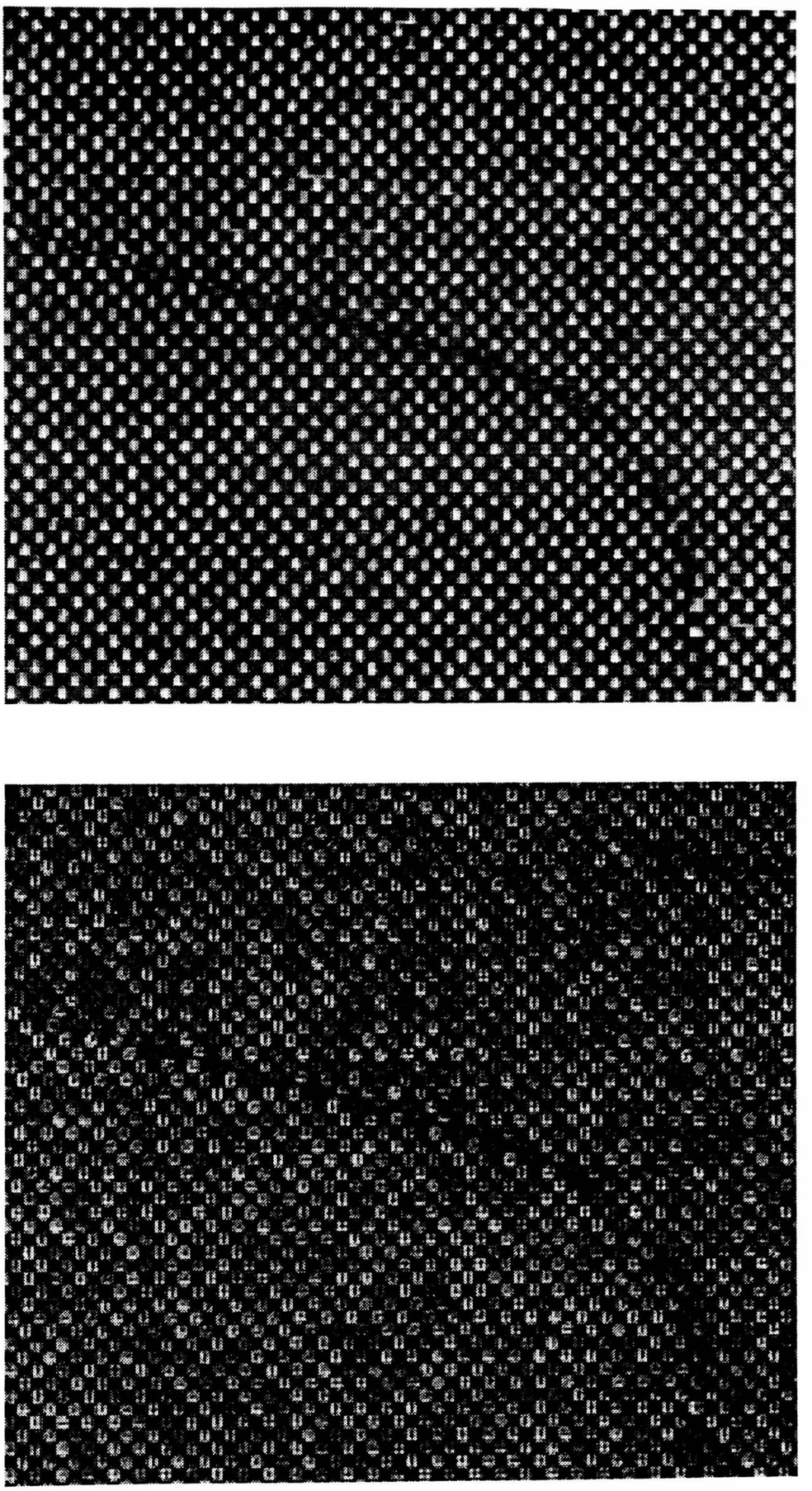


Figure (1.6.2). An image portion showing two pieces of carbon fibre with glass fibre weft, and the result of applying the 3x3 Sobel edge operators shown in **Figure (1.6.1)**.

In essence the confidence of an edge depends not only on the output of an edge operator at that location, but also on the pattern of edges around it. For example, a weak vertical edge between two strong vertical edges would have its confidence increased. Experiments soon showed, however, that edge relaxation is not readily applicable to textured images. The effect is to increase the edge "swamping" effect mentioned above. The edge information generated by the weave is increased equally as much as the boundary information.This is to be expected, since they represent "genuine" edges, not noise.

The conclusions of early work on inspection of carbon fibre workpieces therefore was that detecting the boundary between plies cannot readily be accomplished by the use of conventional edge operators.

1.7 The Area of Research.

The difficulty in finding boundaries between layers of carbon fibre is due to the fact that the images produced do not exhibit areas of (anything like) continuous grey-levels. Instead they show thousands of peaks and troughs corresponding to the weave, which result in a roughly periodic pattern. They are, in short, **textured**. Any boundary detection algorithm employed must therefore be designed to process textured images. Such algorithms are referred to in the computer vision literature as performing **texture analysis**. Texture analysis has been an active research area over the last 20 years or so. In this time many different approaches have been proposed. Since the plies to be inspected exhibit regular texture (the texture is provided by the weave of the cloth which is fairly uniform) and the textures for each component are known *a-priori* (material type and orientation is strictly specified for each layer of a component) then the actual pattern recognition task is greatly simplified, and it should be possible to adopt almost any approach recommended in the literature. For an industrial inspection application however, there are other criteria which must be considered.

- In-process inspection is required, and so processing time must be of the order of a few seconds, using commercially available hardware.

Longer processing time would mean the inspection stage would become the cycle bottle-neck. Specially constructed hardware is undesirable on the grounds of cost and maintainability.

- A high degree of accuracy is required i.e. good localisation of region boundaries. The aim is to find an inspection method which can provide sufficient accuracy of inspection so that ply placement of ±1mm from datum positions can be confirmed.

- The method should be able to cope with a change in the component material easily and quickly.

- The method should be applicable to as many composite inspection tasks as possible.

1.8 The Thesis.

It is the task of finding a texture-based inspection method which satisfies the criteria of **Section 1.7** which forms the bulk of this thesis. The various texture analysis methods developed over the last fifteen years or so have emerged from a wide range of background disciplines. Texture analysis has proven such a difficult pattern recognition problem that no universally satisfactory model has yet been developed (or discovered, depending on your point of view). Fourier analysis, digital filtering, statistical analysis, random field models, fractals, neural models, human based models, and numerous *ad-hoc* models, have all been proposed as techniques for texture analysis. These are fully discussed in **Chapter Two**, which provides an extensive review of the current texture analysis literature. One thing all these techniques have in common however, is that their feature vectors (explained in **Chapter Two**) are usually very large, which entails an enormous amount of processing and memory resources. Optimisation is not yet an issue in the field, since the problem has not yet been "solved". It is a novelty of the work described in this thesis therefore, that a texture analysis model optimised for a specific range

of applications has been developed. The model is described in **Chapter Three**, and is optimised in a dual sense. Firstly the processing and memory resources required are minimised. Secondly, the processing technique itself is optimal in the sense that it can be implemented in hardware on existing commercially available systems. The success of the model is dependant on a means of generating suitable convolution masks. **Chapter Four** describes an existing algorithm to optimise (or train) convolution masks for texture discrimination, and provides modifications to improve the performance for practical applications. From the experience gained in this work, a new algorithm to optimise convolution masks is proposed and studied in **Chapter Five**. The new algorithm is shown to significantly out-perform the previous algorithm. In **Chapter Six**, a novel idea is described using the new optimisation algorithm: optimisation of convolution masks for detection of boundaries in a textured image. The suitability of the new algorithm for the task is demonstrated, and the processing stages required for boundary detection in a textured image are described. All the processing is again convolution based, so that processing time can be kept to a minimum. This new technique of boundary detection can be used successfully in applications where conventional texture analysis cannot i.e. boundary detection between plies of the same material and orientation (and therefore textures). **Chapters Three** to **Six**, therefore, describe in detail the texture analysis models and tools developed to meet the inspection criteria of the application area.

Chapters Seven and **Eight** go on to consider a requirement of the application which is in direct conflict with texture based inspection, and that is the requirement for accurate boundary estimation (i.e. detecting ply edges). Texture analysis techniques are based on the patterns produced by *groups* of pixels, and so determination of boundaries in textured images can only be accomplished with an error of several pixels **[Du Buf et al, 1990]**. Three aspects of the inspection strategy described in this thesis address the problem. Firstly the texture analysis technique employed is convolution based, and such techniques are reported in **[Du Buf et al, 1990]** as providing the most accurate estimates of texture boundaries. Secondly, analytical boundary

models (straight lines and arcs) are employed in an attempt to reduce the effect of errors. **Chapter Seven** describes the implementation of this approach, and details experimental attempts to determine inspection error for typical preform inspection tasks. Thirdly, and most importantly, a novel boundary refinement technique has been developed to improve upon the accuracy of the texture based boundary estimate. This is described in **Chapter Eight**, and takes the form of a localised search for "real" boundary points using the texture based boundary estimate as a guiding template. It is shown that this technique is capable of producing sufficiently accurate estimates of ply position to enable automated in-process inspection of dry-fibre lay-up.

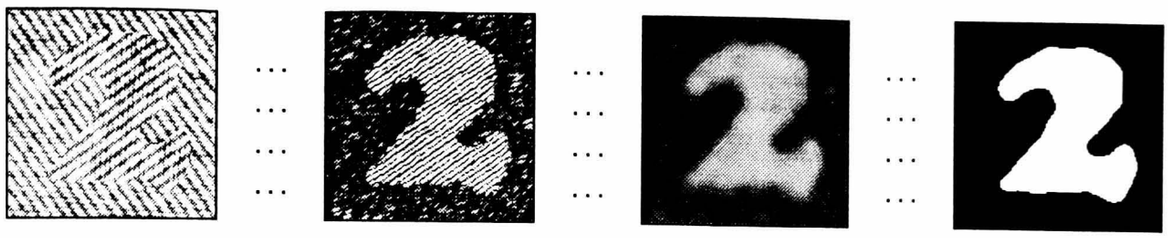
In **Chapter Nine**, the tools developed in the thesis are integrated into the robotic assembly cell. All aspects of integration are considered, and a sample preform design is used to demonstrate practical performance of the cell. The results of this chapter can be seen to justify the approaches adopted in this thesis (discussed more fully below). Finally, **Chapter Ten** provides the conclusions of the thesis, and makes suggestions for further work.

An important aspect of the work presented in this thesis is that both stages of inspection (texture analysis and boundary refinement) have their key parameters optimised by *training*. It is in this way that the flexibility of the inspection system is ensured, enabling the last two criteria of **Section 1.7** to be met. The optimisation criteria for the training stages are practical (empirical) measures of performance for a particular application, rather than theoretically derived measures. Indeed the models themselves (texture analysis and boundary refinement) have little theoretical basis. This differs from many inspection techniques which describe the inspection task in terms of a mathematically based model. Parameters of the model are then extracted analytically (if the equations are mathematically tractable) or set empirically. The success of such techniques depends on how well the adopted model fits the underlying task, and on the choice of parameters. Almost always, significant approximations must be made to derive the basic equations, resulting in loss of practical performance. The approach taken throughout is to develop pragmatic models for the inspection tasks taking into account the

requirements of the application and the limitations of processing resources, and to optimise each stage by training. For the case of in-process inspection of the dry-fibre lay up process, this approach has proven successful.

1.9 Conclusions.

This chapter has attempted to provide a background to the work presented in this thesis. Advanced composite components have been introduced, the current manufacturing process outlined, and the lay-up cycle adopted for automation described. The particular problems faced in the machine vision application have been defined, and an overview of the approach taken to solve these problems has been given. The state of the art in statistical texture analysis is considered in the next chapter.



CHAPTER 2

Texture Analysis Literature Survey.

2.1 Introduction.

This chapter will present the findings of an extensive literature review into texture analysis methods. It is divided into six main sections. The aim of the first section is to explain some of the theory and practice common to many of the methods discussed. The next three sections detail the three most widely used texture analysis methods: co-occurrence matrices, random field models, and multi-channel filtering. The fifth section briefly describes a selection of the many other approaches produced over the years, and the sixth section summarises the main findings of the survey.

2.1.1 Statistical versus Structural.

A strict definition of texture is perhaps not possible, but in the context of image segmentation the following is appropriate: "A region in an image has a constant texture if a set of local statistics or other local properties of the picture are constant, slowly varying, or approximately periodic" **[Sklansky,1978]**.

The perception of texture depends on image resolution, and this is

related to the idea of a **repeating texture primitive**, sometimes called a **texture element** or **textel**. To demonstrate this concept, consider the texture exhibited by a brick wall. From a distance the predominant texture is that formed by the pattern of the bricks, and the obvious primitive would be the brick. Such a structured pattern is sometimes termed a **macrotexture**. From a much closer viewpoint the surface texture of each brick becomes apparent. This is an example of a **microtexture** and it is very much more difficult to find a repeating primitive for such a texture. This has given rise to two different classes of technique for texture analysis. The **structural** model regards the primitives as forming a repeating pattern and describes such patterns in terms of rules (grammar) for generating them. This model is most suitable for describing textures where there is much regularity in the placement of primitive elements and the texture is imaged at high resolution. The **statistical** model usually describes texture by statistical rules governing the distribution and relation of grey-levels. This works well for many natural textures which have barely discernable primitives.

Structural techniques have several drawbacks which preclude use in machine vision applications. Automatic determination of the appropriate primitive is difficult. The processing time required to extract the primitives can be considerable. High resolution images are required where each textel can be represented by enough pixels to define a recognisable structure. This in turn increases the amount of data to be handled and so further increases the processing time required. For these reasons, attention has focused on statistical texture analysis, although some approaches which use a mixture of statistical and structural techniques are also considered.

2.1.2 The Feature Vector.

Statistical texture analysis is closely tied to the idea of pattern recognition. The basic notion of pattern recognition is the **feature vector**. For a given texture, the feature vector **V** is a set of measurements $\{v_1, v_2, \ldots v_m\}$ which attempts to condense the description of relevant characteristics of the texture into a small m-dimensional feature space. By selecting appropriate

characteristics, different texture classes should be mapped to different "zones" of feature space. To achieve image segmentation, a classification rule is applied. For successful segmentation it is essential to choose good features. With well chosen features the simplest of classification rules will work, whilst no classification rule will work if the features chosen are inappropriate. **Figure (2.1.2.1)** illustrates these concepts.

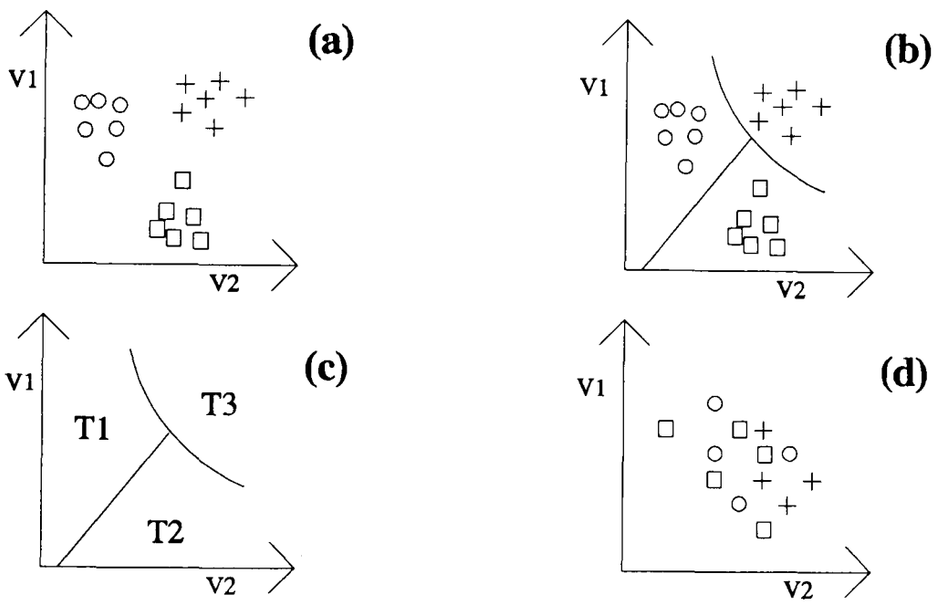


Figure (2.1.2.1). (a) Well chosen features, (b) Choosing a classification rule, (c) Resultant classification paradigm, (d) Badly chosen features, no classification rule is appropriate. Adapted from **[Ballard,Brown,1982]**.

2.1.3 Local Neighbourhoods and Feature Planes.

Texture is a neighbourhood property, in that a single pixel gives no useful texture information. All the methods discussed in this chapter derive measures of texture from groups of pixels. The most common grouping is the 3x3 local neighbourhood shown in **Figure (2.1.3.1)**, although the ordering system used may vary from method to method. A typical texture analysis method would calculate a texture measure over the nine pixels in this 3x3 neighbourhood. The resultant **texture feature** is considered to be attached to the central pixel **X**.

1	2	3
4	**X**	5
6	7	8

Figure (2.1.3.1). The 3x3 local neighbourhood associated with pixel **X**.

The window is moved over the image in such a way that each pixel in the original image is given an associated texture feature. A **feature plane** or **feature image** is one where each pixel **X** is represented by it's corresponding texture feature. If, as is usually the case, more than one texture feature is measured, then there will be one feature plane for each measure.

2.1.4 Second-Order Statistics.

Almost all the methods described in this chapter involve computation based on the **second-order statistics** of the image. The first-order statistics of an image are derived from the intensity histogram e.g. mean, variance, skewness, etc. The second-order statistics describe the relationship between pairs of pixels, both in terms of grey-levels and relative position. The emphasis on second-order statistics is mainly due to a famous result by **[Julesz,1962]**. Julesz found that human subjects could not visually discriminate between textures which differed only in their third and higher-order statistics. This has since been disproved to an extent by **[Caelli,Julesz,1978]** who found some artificial texture pairs with identical first and second order statistics which could be visually discriminated. For the vast majority of workers using natural textures however, the reliance on second-order statistics is a valid one.

A common practice among many workers is to **histogram equalise** the image of texture samples **[Hall et al,1971]**. The result of this is to produce intensity histograms which have the same mean, variance, etc, so that in effect the first-order statistics have been removed, and the texture samples can only be discriminated by second-order statistics. This is regarded as a proper test of the texture analysis method, since classification can then only proceed on the basis of pattern recognition. For a real-world application

however, histogram equalisation is often an unacceptable overhead, and so it is *at least* as important to measure the performance for "real" images.

2.1.5 Texture Samples.

Most of the papers reviewed in this chapter demonstrate the performance of their proposed methodology on test images. Sometimes these images are artificially synthesized textures, sometimes aerial or satellite images, occasionally outdoor scenes, but mainly they are selected samples taken from **[Brodatz,1966]**. This publication, "A Photographic Album for Artists and Designers", has found a niche in the market which its author never intended.

Whilst the practice of using Brodatz textures as test imagery is useful in that it provides a form of benchmark for different texture analysis methods, it is also worrying in that it is very often the only testing ground accorded to a particular method or model.

2.1.6 Classifiers.

Most of the papers reviewed in this chapter will at least mention the classifier used. This might be **maximum likelihood estimate** (MLE), **nearest neighbour** (NN), **maximum a-posteriori probability** (MAP), a **clustering algorithm**, or some other method. In general however, the texture analysis method is independent of the classifier used. For this reason attention will focus on the texture analysis methods rather than the classifiers. In general however, the importance of classifier choice and performance in automated visual inspection is noted. The interested reader is referred to **[Duda,Hart,1973]**.

The remainder of this chapter will address the various statistical texture analysis methods presented over recent years.

2.2 Grey Level Co-occurrence Matrices.

2.2.1 Introduction.

This approach, also known as *spatial grey level dependence* (SGLD), can be thought of as being based on the estimation of the second order joint conditional probability density functions. The first order statistics of an image are concerned with the frequency distribution of the grey levels in the picture i.e. the grey level histogram. The second order statistics can be expressed as:

$$P(i,j,x1,y1,x2,y2) = P(i,j,dx,dy) = P(i,j) \quad (2.1)$$

which is the probability that pixel(x1,y1) has grey-level **i** and pixel(x2,y2) has grey-level **j**. **dx** and **dy** give the relative displacement of the two pixels. From the above definition, it is clear that **P** does not depend on the absolute indices x1,y1, or x2,y2. For a subset of indices **D** specifying a texture region to be analyzed on an input image **I**, the co-occurrence matrix **M** is an estimate of **(2.1)**, and is defined as

$$M(dx,dy,i,j) = \#\{(k,l) \in D,\ I(k,l)=i \ AND \ I(k+dx,l+dy)=j\} \quad (2.2)$$

where **#{x}** denotes the number of occurrences of **x**. A point to note is that the grey-level intensity distribution (grey-level histogram) can be obtained from the co-occurrence matrix by summing the entries in the columns. This means that the first-order statistics are embedded in the second-order statistics, and so any two images which have identical second-order statistics also have identical first-order statistics.

Several studies (e.g. **[Weszka et al, 1976]**, **[Unser, 1983]**) have shown that texture is best discriminated when small values of dx and dy are used, and MAX(dx)=MAX(dy)=1 is probably most common. Taking symmetry into account, the set of displacements would therefore be

$$(dx,dy) = \{(1,-1),(1,0),(1,1),(0,1)\} \quad (2.3)$$

and so four matrices $\mathbf{M_0(i,j)}$ to $\mathbf{M_3(i,j)}$ would be required. For 256 grey-levels then each M_k is a matrix of size 256x256. Matrices of such size are prohibitive both to store and to process, and so several methods have been suggested

to reduce the memory requirements of co-occurrence matrices.

2.2.2 Reducing Memory Requirements

Often the four matrices are added together to produce a single composite matrix. The result of this is a loss of directionality in that the matrix is then invariant under image rotation by 90°, 180°, or 270°, and obviously the suitability of such an operation would depend on the application. The co-occurrence matrix is symmetric (i.e. P(i,j) = P(j,i)), so memory requirements may also be reduced by storing only the upper diagonal part. Perhaps the most common and effective approach is to reduce the matrix size by reducing the number of grey-levels in the input image, typically from 256 grey-levels to 8 grey-levels giving co-occurrence matrices of dimension 8x8 **[Haralick et al,1973],[Weszka,Rosenfeld,1975]**. This is done using *histogram equalization* which transforms the original image histogram into one which is uniformly distributed with a reduced number of equiprobable grey levels **[Hall et al,1972]**. All first order statistics are therefore lost and the texture analysis takes place using only second order statistics.

2.2.3 Features Extracted from Co-occurrence Matrices.

Most frequently co-occurrence matrices are not used as features directly, but features based on them are computed. The aim of these features is to represent some inherent characteristic of the textures. These may be referred to using terms such as homogeneity, coarseness, periodicity, and others, many of which seem to have no precise visual meaning. **[Haralick et al, 1973]** suggested fourteen textural features, commonly referred to as $\mathbf{f}_1,\mathbf{f}_2,..,\mathbf{f}_{14}$, which might be extracted from a co-occurrence matrix. Four of the most commonly used features, $\mathbf{f}_1$, $\mathbf{f}_2$, $\mathbf{f}_3$, and $\mathbf{f}_9$, are given overleaf.

$$f_1 = \sum_i \sum_j \{p(i,j)\}^2 \quad \textit{Angular Second Moment} \qquad (2.4)$$

$$f_2 = \sum_{n=0}^{N_g-1} n^2 \left\{ \sum_{i=1}^{N_g} \sum_{\substack{j=1 \\ |i-j|=n}}^{N_g} p(i,j) \right\} \quad \textit{Contrast}$$

$$f_3 = \frac{\sum_i \sum_j (ij)p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y} \quad \textit{Correlation}$$

$$f_9 = -\sum_i \sum_j p(i,j) \log(p(i,j)) \quad \textit{Entropy}$$

where $\mu_x, \mu_y, \sigma_x, \sigma_y$ *are the means and standard deviations of the rows* (x) *and columns* (y) *of* $p(i,j)$

2.2.4 Choice of Features.

In general, most workers using co-occurrence matrices have employed a process of trial and error to chose the features for their particular application. **[Gotlieb and Kreyszig,1990]** presented a more systematic study of the discriminatory power achievable by using combinations of Haralick's features. Their experiments show that these features are useful for texture discrimination, and that combinations of features, or **composite classifiers**, are more powerful. Gotlieb and Kreyszig selected six of the features detailed by Haralick, chosen to be mathematically representative of all fourteen of his features. They examined the discriminatory power of all sixty-three combinations of these six features (the number of possible permutations of six features is 2^6-1) on thirteen samples of Brodatz textures. They conclude that there is no advantage to be gained by increasing the number of features beyond some small number, the optimal number found to be four in their experiments. This result is interesting, in that it is perhaps not what one might expect intuitively. One further observation was made, namely that it is possible to construct composite classifiers which are particularly good at recognising a specific texture class.

[He,Wang and Guibert,1987] have presented an algorithm which automatically selects the best composite classifier for a given classification problem from a predetermined set of features. Their algorithm was demonstrated using thirty-six features based on co-occurrence matrices (six

features at six different values of dx and dy), eleven features based on spectral peaks in the power spectrum, and one hundred and twenty-six features based on angular distribution in the power spectrum. This gives a grand total of one hundred and seventy-three texture features for each sample image. The algorithm produced a ten-dimensional composite classifier with a mean recognition rate of 95% for ten Brodatz textures. Processing time is not reported, but with such a large initial feature set, this algorithm can only be useful in the set-up stage of a system, selecting the classifiers which will subsequently be used for texture classification. It is interesting to note that using the feature set of **[He,Wang and Guibert,1987]**, which includes features other than those based on co-occurrence matrices, no saturation point of the type discussed by **[Gotlieb and Kreyszig,1990]** is noticeable. The classification rate using only four features, as proposed by Gotlieb and Kreyszig, is about 85%.

2.2.5 Methods Related to Co-occurrence Matrices

[Chen and Pavlidis,1978] have presented a segmentation strategy which makes use of a co-occurrence matrix in conjunction with a **split and merge** algorithm **[Horowitz and Pavlidis,1976]**. In this scheme, a quad tree is used to represent the image **[Samet,1980]**. Starting at an intermediate level **l** in the tree, regions which have approximately uniform texture, as calculated from co-occurrence matrices, are **merged** to create larger regions at level **l-1**. Regions which are non-uniform in texture are split into separate regions at level **l+1**. This operation is applied recursively until the image is fully segmented. "Reasonable" results are claimed for segmentation.

[Haddon and Boyce,1989] have used co-occurrence matrices in a novel way to detect edges in non-textured images. They note that when an image is non-textured, then the co-occurrence matrix exhibits a characteristic structure of peaks corresponding to the regions and boundaries in an image. The matrix itself is segmented, and an inverse mapping performed to enable each pixel in the original image to be classified as either edge or region. A relaxation algorithm is applied iteratively to improve the results. The authors

regard this approach as simultaneously combining segmentation with edge detection.

[Davis,Johns and Aggarwal,1979] have proposed an extension called **Generalised Co-occurrence Matrices** (GCM's) which incorporate more abstract features than grey-levels, such as edge segments. The features will normally require to be represented as vectors. Such a vector for edge segments might be {location(x,y),orientation,magnitude}. The spatial relationship between such features is also modified from the basic dx,dy displacement, and may relate to angular segments relative to a centre pixel or edge. Features are again based on statistics of the matrices. Davis reports that GCM's are useful in discriminating between macrotextures that are not satisfactorily distinguishable using features derived from conventional co-occurrence matrices. Classification accuracy for GCM's was reported as 80+%, compared to 50-57% for conventional co-occurrence techniques, using a dataset of thirty texture samples from five different classes. In **[Davis,Clearman and Aggarwal,1981]** a slightly different feature, called an *extended edge* was used, and compared to edge segment GCM's and grey-level co-occurrence matrices, with variable results.

GCM's represent one of the few attempts to provide some form of continuity between microtexture analysis and macrotexture analysis. Despite this they have attracted little subsequent attention, possibly due to the indifferent results detailed in **[Davis,Clearman and Aggarwal,1981]**.

[Unser,1986] presented a simplification of co-occurrence matrices, based on the associated **sum and difference histograms**. Characterising the grey-level of any pixel as a random variable, then the second order statistics of an image can be considered as the joint probability function of two random variables, as defined in equation **(2.1)**. Unser shows that the sum and difference of two random variables define the principal axes of the second order probability function of a stationary process. He proposes that co-occurrence matrices can therefore be replaced by their associated sum and difference histograms, which can be estimated directly from the image. For an input image **I** the sum and difference associated with the relative displacement

(dx,dy) are defined as

$$s(k,l) = I(k,l) + I(k+dx, l+dy) \qquad (2.5)$$
$$d(k,l) = I(k,l) - I(k+dx, I+dy)$$

The sum and difference histograms with parameters (dx,dy) over the domain **D** are defined in a manner very similar to grey-level co-occurrence matrices.

$$h_s(i,dx,dy) = \#\{(k,l) \in D,\ s(k,l)=i\} \qquad (2.6)$$
$$h_d(j,dx,dy) = \#\{(k,l) \in D,\ s(k,l)=j\}$$

where #{x} denotes the number of occurrences of x. Unser derives features for sum and difference histograms which correspond to the features Haralick derived for co-occurrence matrices. This method gives comparable results to those obtained using co-occurrence matrices, but reduces memory requirements and therefore processing time.

2.2.6 Industrial Applications of Co-occurrence Matrices.

Unlike most of the texture analysis methods described in this chapter, there *have* been some examples of co-occurrence matrices being used in industrial applications. **[Kruger,Thompson and Turner,1974]** presented the results of a feasibility study into automated mass diagnostic screening of pneumoconiosis radiographs. The input image was histogram equalised to eight grey-levels, and displacements of {1,3,5,11} for dx and dy were used. Four Haralick features were used (f3 correlation, f4 variance, f5 inverse difference moment, f9 entropy), and one ad hoc feature similar to variance. The classification accuracy for their dataset was 96.9% compared with a classification accuracy of 93.4% for radiologists over the same dataset.

[Borghesi,Cantoni and Diani] details the results of a feasibility study which used a system based on co-occurrence matrices to recognise defects of pneumatic components. The textures in question were very regular, and defects restricted in their nature. As a result the only feature which had to be calculated from the matrix was an unusual measure relating to the grouping

of values around the main diagonal. This is reported as being "very effective", but no results are given detailing performance, and no indication is given as to whether or not the system was to be implemented.

[Weszka and Rosenfeld,1976] tested a large set of features on samples of an unspecified "industrial material". The samples of material had previously been graded by human inspection. The material was being inspected at an early stage in the production process to try to determine whether it was likely to become of poor quality at a later stage. At the inspection stage this was not easily discernible to the human eye. Weszka and Rosenfeld found that some of the features defined by Haralick could achieve correlations of better than 0.9 between judgements at the inspection stage and defects at the later stage. Only single features were used, not combinations of features. It is interesting to note that this seems to indicate that the judged quality did not correspond to any obvious visual quality of the material. In other words, features measured on co-occurrence matrices do not necessarily relate to any feature detectable to the human vision system. Again, the paper gave no indication regarding the possible implementation of an automated inspection system.

2.3 Random Field Models.

2.3.1 Introduction.

Considerable recent interest in texture has centred on statistical techniques for modelling and processing image data. The focus of much of this work has been the **Markov Random Field** (MRF) model. For this, and related models, texture is considered to be a stochastic, usually periodic, two-dimensional field. Much of the fundamental work relating to MRF's was carried out by physicists investigating lattices, and the nomenclature reflects this. The basic definitions for a MRF are as follows.

For an **M** x **N** rectangular lattice **L** defined as

$$L=\{(i,j):1\leq i\leq M, 1\leq j\leq N\} \qquad (2.7)$$

let **r** = (i,j) index pixel location i,j. Let $\{\mathbf{x}_r\}$ denote a random field, with $\mathbf{x}_r$ the

field at pixel **r**. The term **Markov** is loosely applied to any model where the probability of a pixel $\mathbf{x}_r$ having a particular grey-level does not depend on pixels beyond a small neighbourhood **D** surrounding pixel $\mathbf{x}_r$. This is more succinctly expressed as :

$$p(x_r=x|\ all\ other\ values) = p(x_r=x|\ values\ of\ neighbours) \qquad (2.8)$$
$$or \quad p(x_r=x|\ X_{(r)}) = p(x_r=x|\ X_{(D)})$$

where **x** is a grey level, $\mathbf{X}_{(r)}$ is the random field over the whole lattice except at **r**, and $\mathbf{X}_{(D)}$ is the field in a neighbourhood **D** of **r**. The term $\mathbf{p(x_r=x|X_{(D)})}$ denotes the conditional probability that $\mathbf{x}_r$ will equal **x**, given $\mathbf{X}_{(D)}$.

A **Markov Random Field** (MRF) must satisfy the above neighbourhood condition, as well as two further conditions. The first one is positivity. This simply means that the probability of any pixel having any grey level on the specified grey scale (i.e. 0..255) is non-zero. The second is homogeneity. This says that the conditional probabilities are unaffected by the actual co-ordinates of the pixel, only the neighbourhood values matter.

2.3.2 Binomial Model.

The probability **p(X=x|neighbours)** is **binomial** with parameter θ**(T)** and number of tries **G-1**, where **G** is the number of grey-levels **[Cross,Jain,1983]**. θ is defined as

$$\theta = \frac{\exp(T)}{1 + \exp(T)} \qquad (2.9)$$

For a first-order model, as shown in **Figure (2.3.2.1)**, then **T** has the form

$$T = a+b(0)(t+t') + b(1)(u+u') \qquad (2.10)$$

	u′	
t	x	t′
	u	

Figure (2.3.2.1). First order neighbours used in equation **(2.10)**.

In general, **T** is given by:

$$T = a + \sum_{v \in D} b_v s_v \qquad (2.11)$$

where **D** is the defined neighbourhood, $\mathbf{b_v}$ is the **potential** associated with the pixel pattern $\mathbf{s_v}$, and **v**ε**D [Cohen,Cooper,1987]**.

The binary case, where the lattice variables (or pixels) have range **{0,1}** is a special case of the binomial model, called the **Binary Markov Random Field Model [Cross,1980]**. For a first order binary model, the conditional probability of **x** is given by

$$p(X=x \mid u, u', t, t') = \frac{EXP(xT)}{1 + EXP(T)} \qquad (2.12)$$

The number of parameters needed to describe a model depends on it's order. The model represented in **(2.12)** is specified by three parameters, **a**, **b(0)**, and **b(1)**. The parameter **a** relates to the ratio of "black" pixels to "white" pixels. (In general $\mathbf{a_k}$ denotes the fraction of pixels in an image that assume the value **k**). The parameter **b(0)** controls clustering in the East-West direction, while the parameter **b(1)** controls the clustering in the North-South direction.

2.3.3 Gibbs Random Fields.

Much of the recent research in the field has made use of a **Gibbs Distribution** (GD) to characterise a Markov Random Field **[Derin,Elliott, 1987]**, **[Daily,1989]**, **[Chen,Dubes,1989]**. This type of distribution was introduced by Ising to describe ferromagnetism (the Ising model) **[Ising,1925]**. It has traditionally been used for either Gaussian or binary variables on a lattice in such applications as idealised gas models and crystal lattice models. It finds favour in texture synthesis and segmentation since it allows development of tractable and robust algorithms. For the basic definitions necessary for a Gibbs Distribution on a random field we follow **[Derin,Elliott,1987]**.

Neighbourhood System.

For a lattice **L** as defined in equation **(2.7)**, **n** is a **neighbourhood system** on **L** iff **n(i,j)**, the neighbourhood of pixel(i,j) is such that

$$(i) \quad (i,j) \notin n(i,j) \qquad (2.13)$$

$$(ii) \quad \text{if } (k,l) \in n(i,j), \text{ then } (i,j) \in n(k,l) \;\forall\; (i,j) \in L$$

Figure (2.3.3.1) shows a central pixel **P** and the neighbourhoods to order six. The numbers represent the **order** of the neighbourhood, so that a first order neighbourhood consists of only the pixels that are four-connected to pixel **P**, second order consists of eight-connected neighbours, and higher order neighbourhoods use surrounding pixels at increasing Euclidian distances. An Nth order neighbourhood incorporates neighbourhoods 1..(N-1).

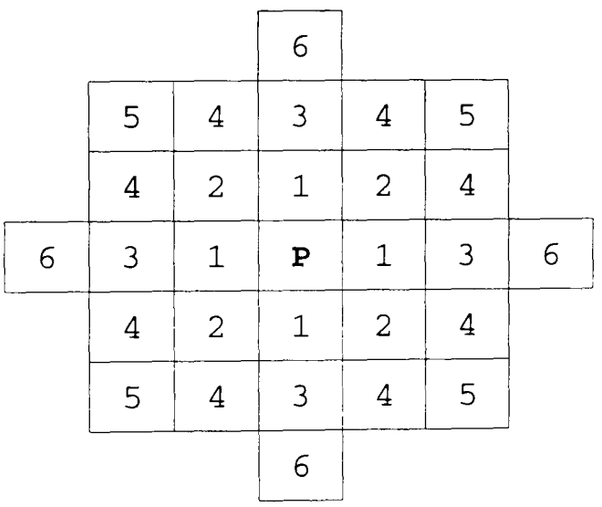


Figure (2.3.3.1). Hierarchal neighbourhood ordering system.

Clique

A **clique** ***c*** is a subset of **L** such that :

$$(i)\ \boldsymbol{c} \text{ consists of a single pixel or} \qquad (2.14)$$

$$(ii)\ \text{for } (i,j) \neq (k,l),\ (i,j) \in \boldsymbol{c},\ (k,l) \in \boldsymbol{c},\ \text{then } (i,j) \in n(k,l)$$

In words, a clique is a set of points that consists of either a single point or has the property that each point in the set is a neighbour of every other point. The cliques associated with a neighbourhood of order two are shown in **Figure 2.3.3.2**.

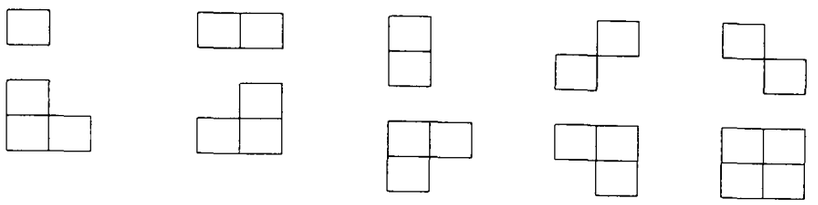


Figure 2.3.3.2. Cliques for neighbourhood of order two.

Gibbs Distribution.

Let **n** be a neighbourhood system defined over the finite lattice **L**. A random field **{X_r}** defined on **L** has **Gibbs Distribution**, or equivalently is a **Gibbs Random Field**, with respect to **n** if and only if its joint distribution is of the form

$$P(x_r = x) = \frac{1}{Z} e^{-U(x)} \qquad (2.15)$$

where

$$U(x) = \sum \beta_c(x)$$

$$\beta_c(x) = \textit{potential associated with clique } c$$

$$Z = \sum e^{-U(x)} \quad \textit{partition function}$$

The partition function **Z** is merely a normalising constant. The only condition on the otherwise arbitrary **clique potential** is that it depend only on the pixel values in clique **c**. The Gibbs Distribution is basically an exponential distribution, but by careful selection of the clique potential function a wide variety of distributions can be formulated. In texture analysis terms it can be used to control, for example, the percentage of pixels in each region type, and the size and direction of pixel clustering.

2.3.4 Gaussian Markov Random Fields.

Another common model is known as the **Gaussian Markov Random Field [Kashyap et al,1982]**, **[Cohen,Cooper,1987]**, **[Jeng,Woods,1989]**, **[Fan,1989]**, **[Manjunath et al, 1990]**. The conditional probability distribution is Gaussian, so that

$$p(x_r=x) = [2\pi\sigma^2]^{-\frac{MN}{2}} |B| \exp\left\{\frac{-[(X-U)^t B (X-U)]}{2\sigma^2}\right\} \qquad (2.16)$$

where **U** is the mean vector of $\mathbf{x}_r$, and **B** is a symmetric **M**x**M** matrix whose diagonal elements are unity and whose off-diagonal elements are β_{r-v} **[Cohen,1986]**. β_{r-v} indicate the clique potential at points **r** and **v**. β_{r-v} is zero if points **r** and **v** are not neighbours. The elements of the mean vector are defined as

$$\mu_r = \mu + \sum_{v \in D} \beta_{r-v}(x_v-\mu)$$

where $\mathbf{x}_v \neq \mathbf{x}_r$ and $\mathbf{x}_v \varepsilon \mathbf{D}$. The Gaussian MRF is parametrised by $\alpha=(\mu,\sigma^2,\beta^t)$

2.3.5 Estimation of Parameters.

In order that MRF's might be useful in texture segmentation and not just texture modelling, some method of estimating MRF parameters from an observed texture is required. One such method is called **Maximum Likelihood Estimation**.

For the Gibbs model a maximum likelihood estimate is complicated by the massive computation required to calculate the normalising constant **Z** in **(2.15) [Cohen,1986]**. One solution, due to **[Besag,1972]**, involves partitioning the lattice (image) into disjoint sets of points called **codings**. Each coding is chosen so that it's points are independent. For MRF's, this means that no two points in a coding can belong to the same neighbourhood system, as defined in equation **(2.13)**. The number of codings required therefore depends on the order of the neighbourhood of the MRF. For a first-order MRF at least two codings are required as shown in **Figure (2.3.5.1)**.

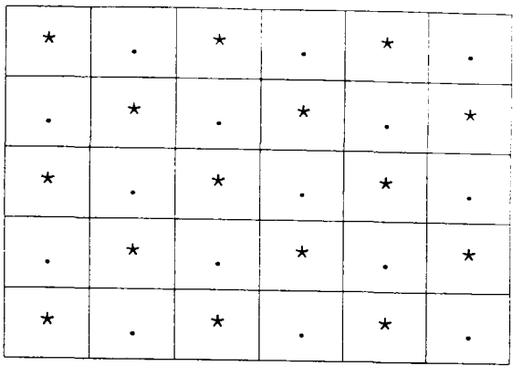


Figure (2.3.5.1). Coding pattern for first-order process showing two codings, * and .

Following **[Cross,1980]**, let **l(i)** be the log likelihood for the i_{th} coding obtained by extending the summation over only those points **X'** which are in coding **i**.

$$l(i) = \sum_{x'} \ln(p(X=x|\ Neighbours\ of\ X)) \qquad (2.17)$$

An estimate of the parameter vector for the $\mathbf{i}^{th}$ coding is obtained by maximising **l(i)**. This is done by finding values of the parameters **a** and β**(j,k)** so that

$$\frac{\partial l(i)}{\partial a} = 0 \quad AND \quad \frac{\partial l(i)}{\partial b(j,k)} = 0 \qquad (2.18)$$

for **j**=1..r and **k**=1,2 where **r** is the order of the process. The system of equations represented by **(2.18)** is non-linear and must be solved by an iterative approach, such as the Newton-Raphson method **[Isaacson and Keller,1966]**. An estimate of the parameter vector is obtained for each coding, and the final estimate is the average value over all the codings.

For the Gaussian MRF of **(2.16)** the normalising constant is more easily obtainable, and so a maximum likelihood estimate can be obtained by maximising **(2.16) [Cohen,1986]**. Again this is non-linear and an iterative solution is needed.

2.3.6 Applications of Random Field Models.

[Cross,Jain,1983] used the binomial model to generate synthetic textures and estimate parameters for natural textures. Image examples are generated using the "Metropolis" algorithm, sometimes referred to as **simulated annealing [Metropolis et al,1953]**, and show how the parameters of the binomial model can be controlled to produce textures which are blurry or sharp, blob-like or line-like. For texture characterisation, a maximum likelihood estimation process is used which requires image areas of at least 64x64 pixels to achieve good parameter estimation. This is obviously inappropriate for accurate segmentation. The authors conclude that microtextures fit the binomial model well, but that regular or inhomogeneous textures do not.

[Cohen,Cooper,1987] use a **doubly stochastic** approach to image modelling, which means that two levels of model are involved. A Gaussian MRF is used to model texture, while a Binary MRF is used to model a-priori information about local geometry of textured image regions. Such a-priori information might be that the image is expected to exhibit boundaries with low (or high) curvature. Two segmentation algorithms are presented. The first, for the case when no a-priori information about region geometry is available, is a hierarchical algorithm using maximum likelihood estimation for segmentation. The second is a recursive relaxation algorithm which searches for a maximum *a-posteriori* likelihood segmentation, using the a-priori information contained in the Binary MRF model. Such a scheme is sometimes called **maximum a-posteriori probability** criteria (MAP). Generally the algorithms perform extremely well on artificially generated MRF's, but markedly less well on natural textures which tend to be non-stationary (non-homogenous).

[Jeng,Woods,1989] also use a doubly stochastic approach and MAP segmentation. In their design both levels are Gaussian MRF models. Two MAP solutions are compared. One, using simulated annealing is very slow but highly parallel in nature, the other, referred to as the **Highest Confidence First** (HCF) algorithm is faster but inherently sequential. Images are presented to illustrate results, but no classification accuracies are reported.

[Fan,1989] presents a hierarchal edge based segmentation algorithm which uses a Gaussian MRF to model texture. The algorithm has two stages. In the first the image is divided into windows. "Mixed" windows, i.e. those containing texture boundaries, are located by application of a **hypothesis test** to the estimated Gaussian MRF parameters of the observed texture. In the second stage a maximum likelihood estimation is applied to the mixed windows to refine the boundary estimate. Both stages are applied hierarchically, from low resolution to high resolution. The results are illustrated with an example image, which shows impressive segmentation of an outdoor scene.

[Manjunath et al, 1990] compare several texture segmentation algorithms using different estimates of MAP, where the texture is modelled as a Gaussian MRF. A fast approximation to MAP is implemented on a neural network, and is compared to simulated annealing in obtaining the MAP estimate. A modification which introduces learning into the network model is also tested. A further algorithm is presented based on **m**aximising the **p**osterior **m**arginal distribution (MPM algorithm). This relates to iteratively examining the classification of a pixel compared to the classification of it's neighbours. Examples showing segmentation of an image containing six Brodatz textures are presented to illustrate results. The segmentation attained by using the MAP estimate of simulated annealing and MPM are particularly impressive.

[Derin,Elliott,1987] present a hierarchal multi-level Gibbsian model to segment noisy and textured images. The model uses second-order Gibbs Random Fields (GRF's) to model texture and region information. Areas of non-texture are modelled as uniform intensity plus Gaussian noise. Segmentation is achieved using a **dynamic programming** algorithm with a MAP criterion **[Bellman, Dreyfus,1962]**. An alternative to coding is detailed, based on histogramming and linear least-squares estimation of the second order neighbourhood. The model presented here requires a-priori information regarding the GRF parameters in order to segment an image. All texture examples are of images generated synthetically by GRF's. Since no images

containing natural textures were tested it is difficult to assess the performance of this model.

[Chen,Dubes,1989] examined MRF estimation procedures and goodness-of-fit of resultant parameters for binary single-texture images. Using the GRF model of **[Derin,Elliot,1987]** and the binomial model of **[Cross,Jain,1983]**, four methods were tested: **Pseudo Maximum Likelihood Estimation** (PMLE); **Least Squares Error Method** (LSQR); **Logit Model Fit Method** (Logit); **Minimum Logit X^2 Method** (Min-X^2). For estimation of parameters, PMLE and Min-X^2 are generally better for both GRF and binomial model, with Min-X^2 performing 10 to 20 times quicker than PMLE. To measure the correspondence between a given texture and the MRF model, a goodness-of-fit statistic was applied. The Min-X^2 method produced the best result for both GRF and binomial model. These results are not readily extendable to grey-scale images.

[Daily,1989] discusses the use of MRF's in colour image segmentation of natural scenes. The most interesting part of this paper is the discussion on the relative merits of different pixel tessellations (rectangular, hexagonal, and triangular).

2.4 Multi-Channel Filtering.

2.4.1 Introduction.

This approach is inspired by the multi-channel filtering theory proposed by cognitive scientists to model the processing of visual information in the early stages of the human visual system. The theory holds that the visual system decomposes the retinal image into a number of filtered images, each of which contains information over a narrow range of frequency and orientation. The theory was first proposed by **[Campbell,Robson,1968]**. They conducted a series of psychophysical experiments to measure the discriminatory power of the human vision system when presented with sinusoidal grating patterns. They found that perception of the gratings was a function of spatial frequency, and concluded that their findings could be

explained by "the existence within the nervous system of linearly operating independent mechanisms selectively sensitive to limited ranges of spatial frequencies". These mechanisms are commonly referred to as **channels**. Experiments by a range of other workers has tended to support the work of Campbell and Robson. **[De Valois et al,1982]** for example, recorded the response of simple cells in the visual cortex of the Macaque monkey to sinusoidal gratings with different frequencies and orientations. The cells were observed to respond only to a narrow range of frequency and orientation.

[Marr,1982], as part of his "primal sketch" model, has suggested a similar multi-channel approach to edge detection.

The multi-channel model is sometimes referred to in the image processing literature as the **bank of filters**, or **linear transform** method. It finds particular favour in digital image processing since each channel can be implemented as a filtering operation using fast convolution hardware. Even with a large number of channels this model is still very much faster than most other approaches to image analysis.

2.4.2 Early Examples of Multi-Channel Filtering for Texture Analysis.

A very simple multi-channel approach was adopted for segmentation of aerial photographs by **[Triendl,1972]**. Two filters were applied in the spatial domain, a 3x3 moving average filter (low-pass), and a 3x3 Laplacian filter (high-pass). An 11x11 moving average filter was used to smooth the resulting feature images before segmentation by a nearest neighbour classifier.

[Faugeras,1978] presented an implementation featuring frequency domain filtering. He used twenty-seven filters (three radial frequencies and nine angular frequencies), and implemented convolution using FFT techniques.

[Granland,1980] presented a variation on the multi-channel filtering approach. He applies a series of edge detectors in a scheme very similar to **template matching** (see **[Davies,1990]**, or **[Ballard,Brown,1982]**) to generate a feature image representing both magnitude and orientation. This feature image is then itself filtered in the same way as the first, with the result that boundaries between areas of different texture are detected. This approach

seems only applicable to regular, line-like textures.

2.4.3 Laws Method.

The multi-channel filtering approach for texture analysis really came to prominence with the work of Laws **[Laws, 1980]**. His model is the basis for many of the publications discussed in the remainder of this section, and so it is presented here in some detail. **Figure (2.4.3.1)** gives a diagrammatic representation of the model.

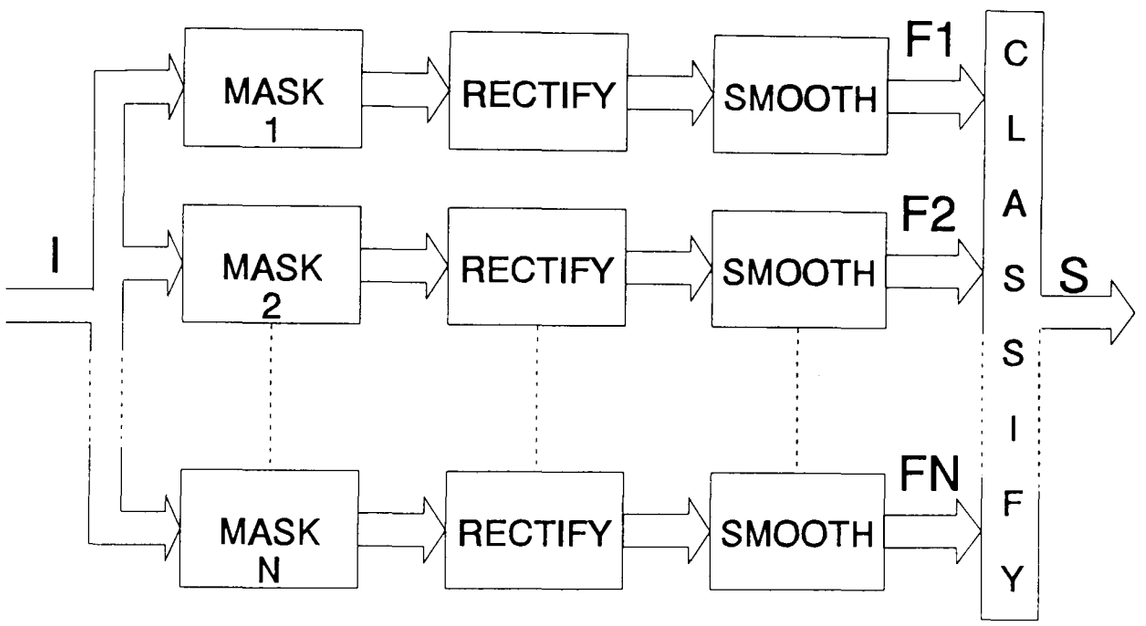


Figure (2.4.3.1). Laws texture analysis model, with input image **I**, feature images **F1**..**FN**, and output segmented image **S**.

Texture is measured by convolving the input image with small centre-weighted filter masks, and then computing statistics within a window around each pixel of the filtered image. Laws developed sets of 3x3, 5x5 and 7x7 masks, and compared results for each. The basic concept is that these masks respond to certain features (frequencies) inherent in the texture. This is reflected in the nomenclature for the masks, using mnemonics representing level, edge, spot, wave, ripple, undulation, and oscillation. His texture segmentation process is as follows.

(1) Each of **N** masks is convolved with the original image to produce **N** new images. Laws developed twenty-five 5x5 masks (**N**=25), but found that almost as good performance was achieved for his dataset using only four (**N**=4).

(2) For each image, **macrostatistics** are computed around each pixel over a larger window size, typically 15x15. The most useful statistic was found to be the variance of pixel values in the transformed image. For zero-sum masks, which all but one of Laws masks are, the filtered images are also approximately zero-sum, and so a fast approximation to variance is the sum of the absolute values. This performs almost as well and has the advantage that it is computationally cheaper. To compute the sum of the absolute values over a 15x15 window for each point in an image is equivalent to convolving a rectified version of that image with a 15x15 mask which has each coefficient equal to one. This is indicated in **Figure (2.4.3.1)** by the two stages, **Rectify** and **Smooth**. The image resulting from rectification and smoothing, Laws calls a **texture energy** image.

(3) The third stage is classification using a nearest neighbour classifier. Each point in the original image now has a feature vector of dimension **N** associated with it, one value for each pixel in the **N** texture energy images. Therefore each pixel now has a value for "edge", "spot", "ripple", etc. Each pixel in the original image is classified as belonging to the texture class which it's features most strongly match. This is **supervised** segmentation, in that a training stage is necessary to establish the feature space occupied by the various texture classes. If the goal is **unsupervised** segmentation (no a-priori knowledge about texture classes), then the classification stage could be changed to use a more sophisticated **clustering** algorithm (see, for example **[Jain,1988]**). The feature extraction stage of Laws method is equally suited to both approaches.

Laws used what he describes as a "worst case" dataset of textured images taken from the ubiquitous Brodatz album **[Brodatz, 1966]**, and compared his method to that of co-occurrence matrices **[Haralick et al,1973]**. His "texture energy" method showed a definite superiority , with 94%

classification accuracy compared to 72% for co-occurrence matrices. It is this result which prompted the popularity of the approach. Until then, no method had been developed which could out-perform the co-occurrence method proposed by Haralick.

Laws four most useful masks are shown (with their corresponding mnemonics) in **Figure 2.4.3.2**.

$$\begin{pmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} \quad E5L5$$

$$\begin{pmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & 16 & -24 & 16 & -4 \\ 6 & -24 & 36 & -24 & 6 \\ -4 & 16 & -24 & 16 & -4 \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix} \quad R5R5$$

$$\begin{pmatrix} -1 & 0 & 2 & 0 & -1 \\ -2 & 0 & 4 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -4 & 0 & 2 \\ 1 & 0 & -2 & 0 & 1 \end{pmatrix} \quad E5S5$$

$$\begin{pmatrix} -1 & 0 & 2 & 0 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 0 & 12 & 0 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & 0 & 2 & 0 & -1 \end{pmatrix} \quad L5S5$$

Figure (2.4.3.2). Laws four most useful masks.

[Pietikainen et al, 1982] found that the power of the masks depends on their general form (edge-like, spot-like etc.) rather than on the specific numerical values used in the masks.

Laws method is heuristic, owing little to theory, and several workers have attempted to study this approach more rigorously. **[Cohen et al, 1989]** arrived at the four 2 x 2 **Hadamard** masks shown in **Figure (2.4.3.3)**, which are derived from the covariance matrix associated with the four variables of a local 2 x 2 neighbourhood **[Harmuth,1972]**.

1	1
1	1

1	1
-1	-1

1	-1
1	-1

1	-1
-1	1

Figure (2.4.3.3). 2x2 Hadamard masks used by some authors as filters in the multi-channel method.

These transform the local neighbourhood into a set of four uncorrelated local

features which measure local average and local derivatives in the horizontal, vertical and diagonal directions. This approach is demonstrated to be superior to the sum and difference histogram method of **[Unser,1986]** discussed in **Section 2.2.5**, but unfortunately no comparison is made with Laws method.

[Boubekraoui et al, 1984] describe a system which uses the Hadamard masks of **Figure (2.4.3.3)**, chosen for the relative ease with which they can be efficiently implemented in hardware. The classification procedure used and the results obtained are unclear not, as is sometimes the case, because of convoluted mathematics, but rather in this instance because of convoluted English.

[Unser,1986] and **[Ade]** compared Laws masks with various transforms which have sound theoretical foundations, such as the discrete sine, cosine, Karhunen-Loeve, or Hadamard transforms. Interestingly, a number of Laws masks are very similar to those which result from application of such transforms to textured images, which is surprising considering that Laws arrived at his filter set in a quite different manner.

[Harwood et al,1983] present an alternative to the texture energy measure proposed by **[Laws,1980]**. Harwood uses **ranked** versions of both mask and neighbourhood. In such a scheme, it is the relative order of the pixel/mask element values which is important, not the actual values. This is best understood by reference to **Figure (2.4.3.4)** which shows Laws mask **L3E3** and the corresponding ranked version.

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 2.5 & 5.0 & 7.5 \\ 1.0 & 5.0 & 9.0 \\ 2.5 & 5.0 & 7.5 \end{pmatrix}$$

L3E3 *RANKED VERSION*

Figure (2.4.3.4). Example of ranking on a 3x3 mask. For a 3x3 local neighbourhood the process is analogous.

A measure of correlation between ranked masks and ranked neighbourhoods is given by **Spearmans rank correlation coefficient [Udny et al, 1968]**. It seems logical to suppose that the process illustrated in **Figure (2.4.3.4)** will reduce the effects of noise in the image, and indeed Harwood reports improved classification results over the texture energy measure. His

implementation of texture classification is not, however, carried out as specified by Laws, and so this result should be treated with some care. More importantly it is not possible to implement this approach as a simple multi-dimensional filtering operation, and so the processing time required would be vastly increased.

[Hsiao,Sawchuk,1989] examine ways to improve the segmentation results obtained by a multi-channel filtering model. Rather than the simple moving window average approximation to variance used by Laws, the use of a **quadrant** filtering method is suggested. This, as the name suggests, implies dividing the moving window into four sub-windows, and calculating the variance of the window as a whole as being equal to the smallest variance of any constituent sub-window. This, it is proposed, reduces the misclassification error at region boundaries where the window will contain more than one texture class. The application of probabilistic relaxation algorithms to the segmented image is examined, and improvements in classification of the order of 1-2% are reported. These algorithms are iterative and computationally expensive, and so it is unlikely that many applications would find this process advantageous.

[Unser,Eden,1988] also modify the moving average approximation used by Laws. They propose an iterative Gaussian smoothing algorithm, which they suggest is less sensitive to nearby edges since it gives greater weight to the central pixels in a window. A further processing stage is used to compress the resultant feature planes into fewer components while retaining the information for optimal discrimination. The individual stages and their effect in segmentation are evaluated in detail, but no comparison to other methods is provided.

[Cano,Minh,1988] have investigated a more complex multi-channel model than that described by Laws. Their model incorporates multi-resolution multi-channel filtering. The features measured are first, second and third-order statistics of the filtered images at each resolution. The masks used may be Hadamard, Laws or others. Little mention is made of texture classification, but the model is demonstrated to be able to **synthesize** microtextures well and

structured textures less well. This is a similar result to that reported by **[Cross,Jain,1983]** in **Section 2.3.6**, which used Markov Random Fields for texture synthesis.

[Michel et al,1989] present a texture segmentation system based upon multi-channel filtering in which they have attempted to make the segmentation algorithm independent of the masks used. This approach is very similar to the method described in **[Chen,Pavlidis,1978]** (see **Section 2.2.5**), which uses a split and merge algorithm in conjunction with a co-occurrence matrix to segment images. In this case the homogeneity of a region (whether it contains more than one texture class) is determined from the output of a bank of filters. For the composite texture images used as test data, the segmentation appears very good.

[Benke,Skinner,1987] introduce the idea that in cases where only two textures appear in an image then a single channel can discriminate them. Such an approach is obviously only suitable for a limited number of applications. An algorithm to train convolution masks to discriminate between two textures is presented. The algorithm is iterative and uses a Monte-Carlo approach. A potential application is described in **[Skinner et al,1990]**, where the algorithm is used to generate masks which will detect defects in radiograph images of ordinance fuses. The position and nature of each possible defect is known a-priori, and so this is an ideal application for automated visual inspection. In **[Benke et al,1988]** the algorithm is used to generate masks which are used in psychophysical experiments. Human subjects were asked to sort images of fourteen textures according to the order of features such as "blob-likeness", "linearity" and "regularity". Masks were generated so that the variances of the filtered images correlated with the human perception. The authors feel that, if the features used in the human vision system for texture discrimination can be determined, then convolution masks can be generated which strongly correlate with these features. It is, however, the first part of this process which is presenting all the problems.

2.4.4 Frequency Domain Methods.

Some authors have chosen to implement the multi-channel filtering model in the frequency domain. One of the earliest examples, already mentioned in **Section 2.4.2**, is due to **[Faugeras,1978]**. In this implementation twenty-seven filters (three radial frequencies and nine angular frequencies) are used. Convolution is implemented using FFT techniques.

[Ikonomopoulos,Unser,1984] demonstrates a similar approach, using bandpass filters selected to span the domain. A maximum likelihood procedure is used for classification. Using four Brodatz textures, the classification results obtained for window sizes of 7x7, 15x15, 31x31, and 61x61 were 76%, 88%, 98%, and 99% respectively.

[Coggins,Jain,1985] also applied filtering in the frequency domain. Rather than texture energy, they use a novel feature based on measurement of the grey-level histogram of each filtered image. Using eight Brodatz textures and a nearest neighbour classifier, results of 98% and 91% classification accuracy were obtained using window sizes of 64x64 and 32x32 respectively. This feature was incorporated into a segmentation scheme which used a clustering algorithm to discriminate texture regions. From the accompanying images it is difficult to assess the performance.

[Monte et al,1988] have developed method for isolation of a texture class in an image. This is similar in concept to the approach of **[Benke, Skinner,1987]** detailed in **Section 2.4.3**, but with the difference that the design of filter for texture discrimination is carried out in the frequency domain. The application here is analysis of aerial photographs, where it is useful to highlight a particular texture, i.e. forest areas. This paper describes a simple system using binary filters which provide only limited performance. An extension to grey-scale has not, as yet, been published.

2.4.5 Eigenfilters.

[Ade,1983a] introduces the idea of actually deriving the filter masks from the texture being analyzed. From a homogenously textured region he calculates the **eigenvectors** of the corresponding covariance matrix and uses

these as the coefficients for his masks, which he calls **eigenfilters**. This process is sometimes referred to as the **Karhunen-Loeve** transform **[Unser,1986]**.

For a 3x3 neighbourhood, the covariance matrix **C** is defined as

$$C = \frac{1}{N-1} \sum_{(i,j)\in I} (b(i,j)-m)(b(i,j)-m)^T \qquad (2.19)$$

$$\text{where} \qquad m = \frac{1}{N} \sum_{(i,j)\in I} b(i,j)$$

where **(i,j)** is the coordinate pair of the central pixel, **N** is the number of pixels in the area **r**, and **b(i,j)** indexes the nine components of the local 3x3 neighbourhood (see **Figure (2.1.3.1)**). The covariance matrix fully determines the joint second order distribution of grey-levels in a Gaussian process **[Ade,1983a]**. For a 3x3 neighbourhood the covariance matrix **C** has size 9x9. For matrix **C**, the eigenvalue problem is posed as

$$(C - \lambda I)e = 0 \qquad (2.20)$$

where **I** is the 9x9 identity matrix, λ is an eigenvalue, and **e** is an eigenvector (for an introduction to eigensystems, see **[Kreysig,1983, p.345]**). Equation **(2.19)** has solutions $\mathbf{e}_i$ with eigenvalues $\lambda_i>0$ (**i**=1,2,...9). Since covariance matrices are symmetric, these solutions are easily obtainable by numerical methods such as the **householder reduction [Press et al,1990]**, and so the eigenvalues and, more importantly, eigenvectors can be extracted.

These eigenfilters have some appealing properties. They are derived without loss of information from the covariance matrix, which as mentioned above fully describes the second order statistics generally thought to represent essential texture information. The eigenfilters themselves are orthogonal, and therefore completely span the feature space. Laws filters are not orthogonal. Eigenfilters are, therefore, theoretically attractive. Unfortunately this does not readily translate to practical applications.

In explaining this, it is important to realise the difference between this approach and that of Laws. The multi-channel filtering method, as realised by Laws, uses an empirical set of filters to extract features from textures which are subsequently used for classification purposes. The filter set might be considered a common metric by which all textures can be measured.

Eigenfilters on the other hand, are themselves extracted from the texture, so that each texture will produce a *different* set of eigenfilters. This method, therefore, does not produce any common metric with which to *classify*, and therefore segment, texture. What it does do is produce some means with which to *characterise* a texture class. It has therefore found application only in areas relating to defect detection, and not texture segmentation. **[Ade,1983b]** details some experiments involving defect detection of textile web. The process is as follows. In the training stage the 3x3 eigenfilters are derived from a "good" sample of web. For defect detection, each input image is filtered by each of the nine eigenfilters producing nine feature images. These feature images are used as input to a classifier. Pixels which are not sufficiently close to the feature space defined by the "good" sample of web are classified as defect. Ade reports some problems and limitations of his procedure.

[Dewaele et al,1990] present a similar system, but introduce the idea of using convolution masks which are referred to as **sparse matrices**. A sparse matrix is a matrix with relatively few non-zero elements, compared to the more common **contiguous** form of convolution matrix (mask) in which (generally) all elements are non-zero. The position of the non-zero elements within the sparse matrix are determined by estimation of the **period** of the texture being processed. A sparse texture (texture elements relatively few and far apart) will result in a "long" period and therefore a sparse convolution mask. A high frequency texture on the other hand will result in a "short" period and therefore a contiguous mask. Two methods are suggested for estimation of the period, one using measurement of the co-occurrence matrix, the other an iterative comparison between different autocorrelation window sizes. The non-zero elements are ascertained by the eigenfilter technique, but with the difference that the covariance matrix, as defined in equation **(2.19)**, now has **b(i,j)** indexing the pixel pattern determined by the period. Improved performance is reported over contiguous convolution masks. This approach is only useful for regular textures, since random textures do not exhibit periodicity.

2.4.6 Models Based on Features of the Human Visual System.

[Laws,1980] set out the results of an empirical study into a multi-channel filtering model for texture analysis. Another approach is to use models more directly related to the human visual system, and test their suitability for texture analysis.

[Wermser,Liedtke,1982] presented a model strongly influenced by theories regarding the human visual cortex. A novel method of determining the resolution of the features detected is employed. The input image is processed by three median filters of size 5x5, 9x9, and 19x19. The output of each median filter is processed by four 5x5 masks corresponding to "line" edges at orientations of 0°, 45°, 90°, and 135°. In total, twelve filtering operations are performed for each input image. This approach is similar to that suggested by **[Marr,Hildreth,1980]**, except that Marr and Hildreth used different resolution Gaussian masks to blur features, and Laplacian masks to detect edges.

[Bergen,Adelson,1988] have demonstrated that simple low-level mechanisms based on multi-channel filtering can successfully explain many aspects of human texture discrimination. This is a controversial area at the moment, as it contradicts the theory related by **[Julesz,Bergen, 1983]**, which introduced an entity called a **texton** which they claim to be the "fundamental elements in pre-attentive vision and perception of textures". Textons are somewhat loosely defined as "elongated blobs (e.g. rectangles, ellipses, or line segments) with specific properties, including colour, angular orientation, width, length etc". The theory is that the human vision system pre-attentively (without effort) discriminates textures based on the variation of texton features extracted from those textures. The work of **[Bergen,Adelson,1988]** casts some doubt on this assumption.

There have been a profusion of papers in recent years relating to the use of **Gabor** functions in image processing **[Gabor,1946]**. The response of an even-symmetric Gabor filter in the spatial domain is given by

$$h(x,y) = \exp\left\{-\frac{1}{2}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right]\right\}\cos(2\pi u_0 x) \qquad (2.21)$$

and in the frequency domain by

$$H(u,v) = A\left(\exp\left\{-\frac{1}{2}\left[\frac{(u-u_0)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2}\right]\right\} + \exp\left\{-\frac{1}{2}\left[\frac{(u+u_0)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2}\right]\right\}\right) \quad (2.22)$$

where

$$\sigma_u = \frac{1}{2\pi\sigma_x} \qquad \sigma_v = \frac{1}{2\pi\sigma_y} \qquad A = 2\pi\sigma_x\sigma_y$$

The interest is generated by the suggestion that Gabor signals can be used to represent the response profiles of receptive fields found in the mammalian visual cortex **[Pollen,Ronner,1983]**, **[Daugman, 1980]**. **[Caelli,Moraglia,1985]** have carried out psychophysical experiments to determine the discriminability of textures composed of Gabor signals over a range of frequencies. **[Perry,Lowe,1989]** have used Gabor-like filters in a multi-channel model to segment images. The results, using some rather *ad hoc* features, were reasonable. **[Jain,Farrokhnia,1991]** presented a multi-channel model using up to twenty-eight Gabor filters. A post-processing stage is used to reduce the number of feature planes used in classification. In this system it seems that the filters are not approximated by convolution masks, but rather the equation of **(2.21)** is applied to the image data with the appropriate parameters to give different frequency and orientation responses. Whilst this will provide a more accurate transformation, the processing advantage of hardware convolution is sacrificed. **[Iwama,Maida,1989]** present a texture segmentation algorithm that looks for textons in the feature plane images produced by a bank of Gabor filters. This combines the two viewpoints of **[Bergen,Adelson, 1988]** and **[Julesz, Bergen,1981]**. **[Gopal et al, 1990]** use the feature images produced by a bank of Gabor filters to calculate the texture gradient of the input image.

Variations on the theme are presented by **[Malik,Perona,1989]** who use difference of Gaussian (DOG) and difference of offset of Gaussian (DOOG) filters, and by **[Caelli,1988]** who presents a model where the response profile of the filters is allowed to adapt to the incoming image signal. The approach of Caelli is suitable for neural network implementation. **[Stone,1990]** also present an adaptive filtering method using DOG filters as the initial response profiles. The objective in this case is to derive the shape of textured objects by estimating the texture gradient of planar surfaces.

2.5 Other Texture Analysis Methods.

2.5.1 Introduction.

This section will give a brief summary of the many other methods which have been applied to the subject of texture analysis. For a more complete review, the reader is referred to **[Haralick,1979]** or **[Gool et al,1985]**, although these are now relatively old. Perhaps the sheer volume of literature on texture analysis over the last decade is the reason that no more recent survey has been forthcoming.

Some of the papers mentioned here are representative of areas of research which are substantial in their own right. Methods based on fractals, for example, have received considerable treatment in recent years. Neural networks is another obvious example. In general however, the approaches discussed here have not as yet received the attention accorded the three methods discussed in **Sections (2.3)-(2.2)**, and as a result have yet to reach a stage where proper comparison can take place.

2.5.2 Mathematical Morphology.

Mathematical morphology (MM) is the study of shape, and a morphological transformation transforms the given input image into another form which is more expressive in certain respects. The definitive text on MM is **[Serra,1982]**, and an introduction can be found in **[Haralick et al,1987]**. For binary texture analysis, MM involves the **erosion** of the image by structuring elements, sometimes called **partition filters**. This process can be regarded as a kind of structural filtering. The texture features are extracted from the eroded image by counting the number of pixels set to 1 after filtering with a particular element i.e. the number of matches with the structuring element. For grey-scale images, the features are based on matrices relating to the grey-level after partition filters operations, such as **dilation**, **erosion**, **closing**, and **opening [Serra,1982]**. For an example of this type of operation on Brodatz textures refer to **[Fouques,Cohen,1989]**. They present a class of partition filters which are controllable with respect to both orientation and spatial frequency. Examples show how they can be used to homogenise regions

whose structure corresponds to their characteristics, while leaving others untouched.

2.5.3 Fractals.

The major text on fractals is **[Mandelbrot, 1983]**. The definition of a fractal is generally taken as "a set with its Hausdorff dimension strictly greater than its topological dimension". In essence this means moving beyond the conventional notion of dimensions such as 1,2,3.... to the rather less tangible concept of fractional dimensions. Fractals have recently been used in a variety of image processing applications, and a survey of these can be found in **[Ait-Kheddache,1988]**. In particular fractals have been used to characterise textures **[Pentland,1984]**, **[Kaneko,1989]**, **[Pelag et al,1983]**, **[Jardine, Whitworth,1990]**. The fractal dimension of a texture can be estimated from an image in several ways. **[Pentland,1984]** presents a method of estimating it from the Fourier power spectrum using linear regression. A test for the suitability of a texture to be modelled as a fractal is described in **[Jardine,Whitworth,1990]**. This is as follows: if a logarithmic plot of standard deviations of intensity differences for pixel pairs versus their Euclidean separation in the image is linear, then the texture is considered to be fractal. Such a relationship indicates self-similarity at all scales. The authors found that natural textures in general did not accurately fit the fractal model, but that the approximation still produced reasonable results. **[Pelag et al,1983]** have used a model based on more geometric considerations to determine the fractal nature of textures. It has been shown however that the fractal dimension itself is not sufficient to fully describe a texture **[Ait-Kheddache,1988]**.

At present the use of fractals in image processing, as in many other fields, is still in it's infancy.

2.5.4 Multi-Dimensional Edge Detection.

This approach attempts to segment an image by finding the boundaries between textured regions rather than classifying the regions themselves. The theory is that texture features change abruptly near boundaries between

different textures. The approach is intuitively simple, and consists of applying (possibly multidimensional) edge operators to the feature space created by whichever method is employed for measurement of texture features.

An example of such an operation is detailed in **[Xiaohan et al,1991]**. Eight directional measures are computed for each pixel in the original image, and so the feature space is eight dimensional. These measures bear some resemblance to the way in which co-occurrence matrices are constructed. Edge detection is performed as **non-local-maximum-suppression** over each feature image. This is basically an adaptive thresholding procedure. The image is "cleaned" by the removal of small edge segments in a process that requires a-priori knowledge of expected boundary lengths. The remaining edge segments mapped back to the original image as boundaries.

The features used by **[Khotanzad,Chen,1989]** are based on a random field model. The parameters of the random field are estimated for six different local pixel patterns (the **cliques** discussed in **Section 2.3.3**), and so the feature space is six dimensional. Edge detection is achieved by extending the Sobel operator to operate in windows defined over different dimensions.

2.5.5 Neural Networks.

Currently one of the most active research areas is the study of neural networks. Any conference on computer applications now has numerous papers on the topic, if not a separate neural network session. Texture analysis, since it can be considered a branch of pattern recognition, would seem an obvious target for such research, and indeed many workers are now investigating this approach. A description of the various neural models and learning algorithms proposed is beyond the scope of this thesis, and so the interested reader is referred to **[Lippman,1987]** or **[Aleksander,Morton,1990]**. The discussion here is limited to an indication of how neural network models are being applied in texture analysis.

The approach taken by most workers is to use a conventional technique to extract texture features (co-occurrence, Laws, MRF, etc.), but to use a neural network to learn the optimal classification of these features. Such an

approach is detailed in **[Patel,Stonham, 1991]**. The original images are thresholded, and then features extracted based on the co-occurrence matrix for four-connected neighbours. A **single layer** neural network is used to classify textures based on these features. A supervised training stage is required to "teach" the various texture classes to the network. Using eight Brodatz textures, a classification rate of 95% is achieved. **[Patel,Stonham,1992]** extend this model to grey-scale images. Again the features extracted are co-occurrence measures for four-connected neighbours, but in this case the four co-occurrence values "attached" to each pixel are sorted into descending order, or **ranked**. This is an application of the method suggested by **[Harwood et al, 1983]** discussed in **Section 2.4.3**. These features are classified by a **multi-layer perceptron** neural network model, which is trained using a **back-propagation** algorithm. The performance of the model is demonstrated by segmentation of images taken from a potash mine face.

[Visa,1990] presents a model which uses a **self-organising** neural network to classify textures based on measurement of co-occurrence features. A self-organising network performs in an analogous way to a clustering algorithm, and so unsupervised segmentation is possible. The performance and the dataset are unclear.

[Kasparis et al,1990] use texture features which are based on **Hough Transform** descriptors **[Hough, 1962]**. A back-propagation network is again used for classification, and the performance of the system tested using different noise levels. The conclusion of this paper is that neural networks perform classification better than a previously published **linear associative memory** classifier **[Eichmann,Kasparis,1989].**

A different approach to the use of neural networks in texture analysis is taken by **[Zhang,Sarhadi,1992]**. In this model the neural network is actively involved in the feature extraction stage, rather than just acting as a classifier.

2.5.6 The Texture Spectrum.

[He,Wang,1990], **[He,Wang,1991]** have proposed a **texture spectrum**

approach to texture analysis based on the statistics of the local 3x3 neighbourhood. Fundamental to this is the idea of the **texture unit**. Let the local 3x3 neighbourhood be represented by a nine-dimensional vector $\mathbf{V}=\{\mathbf{V}_0,\mathbf{V}_1,\ldots,\mathbf{V}_8\}$ where $\mathbf{V}_0$ represents the intensity of the central pixel and $\mathbf{V}_i$ is the intensity of the neighbourhood pixel **i**. The corresponding texture unit is an eight-dimensional vector $\mathbf{TU}=\{\mathbf{E}_1,\mathbf{E}_2,\ldots,\mathbf{E}_8\}$. The values for $\mathbf{E}_i$ are determined by the formula

$$E_i = \begin{cases} 0 & if\ V_i < V_0 \\ 1 & if\ V_i = V_0 \\ 2 & if\ V_i > V_0 \end{cases}$$

where the element $\mathbf{E}_i$ occupies the same position as the pixel **i**. An example of the transformation from image values $\mathbf{V}_i$ to texture unit values $\mathbf{E}_i$ is given in **Figure (2.5.6.1)**.

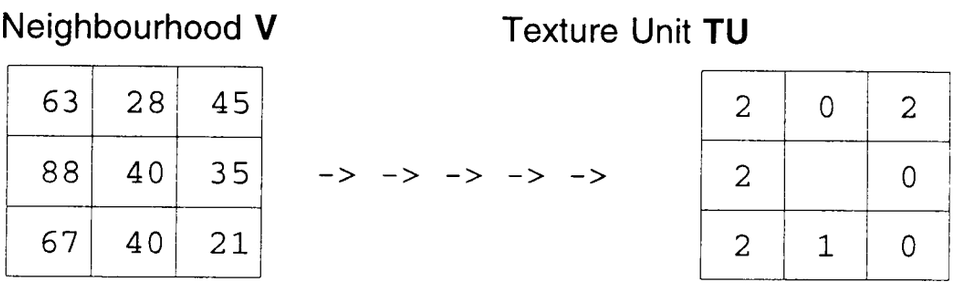


Figure (2.5.6.1). Example illustrating how a texture unit is derived.

As each element of **TU** has one of three possible values, the combination of all eight elements results in $\mathbf{3^8 = 6561}$ possible texture units in total. The frequency histogram of these texture units is termed the **texture spectrum**. Texture classification and segmentation is not carried out on this spectrum directly, but on features derived from the spectrum, such as symmetry of the peaks, symmetry of the spectrum as a whole, etc. In this way it is rather reminiscent of co-occurrence matrices, and indeed the texture spectrum might be considered as an approximation to a co-occurrence matrix.

2.5.7 Methods Based on Pyramidal Structures.

[Brzakovic et al,1990] describe **TEXIS**, which is a TEXture Inspection System. The function of the system is defect detection and classification, and it uses a series of *ad hoc* procedures towards this end. One of the constituent parts of the system is a **pyramid linking scheme** (**[Tanimoto,Pavlidis,1975]**) which segments the image into defects and generic texture. The bottom level of the pyramid is the input image of size $2^n x 2^n$, each subsequent level is a square array which has half the dimension of it's predecessor, and so the top layer is a 1x1 array. Each intermediate layer is derived from the layer below by averaging over a 2x2 area. In such a scheme, texture homogeneity is examined at different image resolutions, in a manner similar to the split and merge algorithm of **[Chen,Pavlidis,1978]** discussed in **Section 2.2.5**. This system uses a different approach, in that the strengths of the links between the layers of the pyramid are updated iteratively, and it is the strengths of these links which determine the homogeneity of a texture area. These links represent the degree of match between "father" and "son" nodes. The performance of the system is discussed in relation to the inspection of wood samples, but it is not possible to determine the performance of the texture analysis stage, due to the uncertain effects of the many other stages.

[Spann,Wilson,1985] use a segmentation scheme based on homogeneity estimation at different levels of a **quad-tree [Samet, 1980]**. A quad-tree is similar to a pyramid, but differs in that not all levels are fully realised **[Ballard,Brown,1982]**. The segmentation method proceeds in the following top-down fashion. At a level **k**, boundary pixels are determined as being those different from any of their neighbours (remembering that at level **k** the averaging effects of the quad-tree will have removed any texture or noise information and left only areas of approximately constant grey-level). A smoothing process is then carried out and isolated pixels removed. This process is then repeated at level **k-1**. The classification introduced at level **k** is valid at level **k-1**. Pixels in level **k-1** are classified as non-boundary if their father in level **k** was non-boundary. The boundary regions of level **k-1** are classified in such a way that the width of a boundary is reduced by a factor of

two in each step down the quad-tree. The result is a one pixel wide boundary at the image level. The images accompanying this paper show impressive segmentation results. It does seem however that there must exist a difference in first-order statistics between the respective areas for this method to work. It is therefore not really appropriate for textured images, but rather for non-textured images where the noise level might render other methods ineffective.

2.5.8 Miscellaneous Methods.

This section will describe some of the more idiosyncratic approaches to texture analysis. This is not to diminish the possible importance of such methods either past or future, merely to indicate that they do not easily fit into any of the previous categories. In many cases these approaches have been developed to cope with only certain classes of texture analysis problem.

Fourier analysis of textured images is one of the oldest approaches. This technique is intuitively attractive, and has the added advantage of being well understood, due to extensive previous work carried out by workers in other fields. The specific case of using Fourier analysis to implement the multichannel model has already been discussed in **Section 2.4.4**, but many more add-hoc approaches have been published. A typical early example of this is given in **[Bajcsy,1973]**. The frequency domain representation of an image, or image region, is partitioned into **bins**. Measurements are made using two types of bin, **radial** and **angular**. The radial bins form concentric circles around the frequency origin, whilst the angular bins dissect the domain into regular segments, similar to a pie chart representation. Features can be defined using either or both of these representations. Methods such as these produced only limited success, and as a result largely fell out of favour as more powerful methods, such as co-occurrence matrices, were developed.

[Tan,Constantinides,1989] present a method that might be considered as a hybrid of structural and statistical texture analysis, and which certainly bears some resemblance to the ideas of textons **[Julesz,Bergen,1981]**. In conventional structural texture analysis, primitives and their placement rules are determined for a texture. The problem arises that even quite regular

textures, once they have been corrupted by noise, sampling non-linearities, and various other natural phenomenon, often do not strictly follow the placement rules. This paper presents an approach in which the placement rules are not represented as a grammar, as is more normal, but rather are represented statistically. Thus both structural and statistical information is used. The primitives are extracted by various thresholding techniques, and are represented by a set of measures relating to area, perimeter, compactness, eccentricity, orientation, colour, and contrast. The placement of these primitives is described by first order statistical measures, namely the mean and variance of their frequency histogram. The performance is demonstrated on fourteen Brodatz textures. Generally the results are somewhat less than might be expected from one of the principal statistical techniques, such as Markov Random Fields or Multi-Channel Filtering.

[Mitchell et al,1977] introduced the idea of using the relative frequency of local extremities in grey-level as texture features. Such **maxima** and **minima** of the image were extracted in one dimension in the direction of the scan. The number of maxima and minima were measured over a range of threshold levels, and this data was used to characterise texture. The results obtained were reported to be "almost as good" as co-occurrence matrices.

[Ramponi,Sicuranza,1989] differ from the vast majority of workers, in that they attempt to discriminate different textures on the basis of fourth-order statistics rather than second-order statistics. The fourth-order statistics they use are arrived at heuristically, and it is not clear why they are chosen. Only four texture classes are used for testing, and the results are not presented in a clear manner, so that it is difficult to assess the performance. Nevertheless, it is an interesting result that at least some natural textures can be discriminated by higher-order statistics.

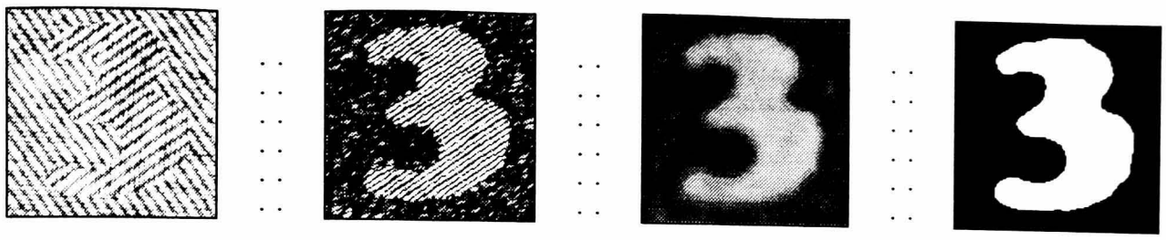
[Kadar,Liebman,1988] suggest an approach to segmentation rooted in statistical measures of variance. The main measure tested is the **two-way-ANOVA** (ANalysis Of VAriance), which is commonly used in statistical packages such as SPSS. In the words of the authors, "Both global and local robust statistical mask texture extractor/operators/object-detectors are

developed...". Despite this tantalising promise, no evidence of any results for texture analysis are to be found. Application of the ANOVA does, however, seem to perform a degree of edge detection.

[Dinstein et al] have developed a fast algorithm to discriminate between regions which are textured, and regions which are not. This algorithm works by calculating the range of grey-levels in a small (3x3,5x5,etc) moving window. For textured regions this is generally much higher than for regions of approximately uniform grey-level.

2.6 Conclusions.

This chapter has presented the results of a comprehensive survey of current statistical texture analysis methods. Broadly speaking, the methods described split into two main categories. On the one hand there are methods which attempt to model the underlying texture process, and describe textures on the basis of extracted model parameters e.g. random field or fractal models. On the other hand there are methods which do not explicitly model the texture, but rather attempt a description based on measurements of certain pertinent features e.g. co-occurrence methods, multi-channel filtering, mathematical morphology, texture unit etc. It is not possible at this stage to determine which of these quite different approaches will in the end prove superior. Comparison of results for different methods is difficult, mainly due to the fact that a precise definition of texture has so far proved illusive. Certainly a detailed and comprehensive comparative study of the (apparently) most promising methods would help to illuminate the situation. Such speculation is however, superfluous to this thesis. The interest here does not lie in producing the ultimate texture analysis system, but rather in developing an appropriate approach for the task in hand. The knowledge gained in this literature survey in conjunction with the application criteria outlined in **Chapter One** have enabled the development of the techniques described in the following chapter.



CHAPTER 3

The Single Channel Model for Texture Analysis

3.1 Introduction.

The aim of this chapter is to introduce and explain the texture analysis model developed for this work. The approach adopted is based on grey-scale convolution, and is a variation on the multi-channel filtering paradigm described in **Section 2.4**. The model was not however derived directly from the multi-channel filtering approach, but rather evolved from experiments aimed at determination of carbon fibre weave pattern and orientation. A prototype system based on binary filtering was initially developed and tested, and is explained in the early sections of the chapter. The progression from this work to the final texture segmentation model is detailed, and the influence of hardware considerations on the design is indicated. The suitability of the model for the application is demonstrated, and the effect of various parameters on segmentation considered.

3.2 A Binary Filtering Approach to Texture Analysis.

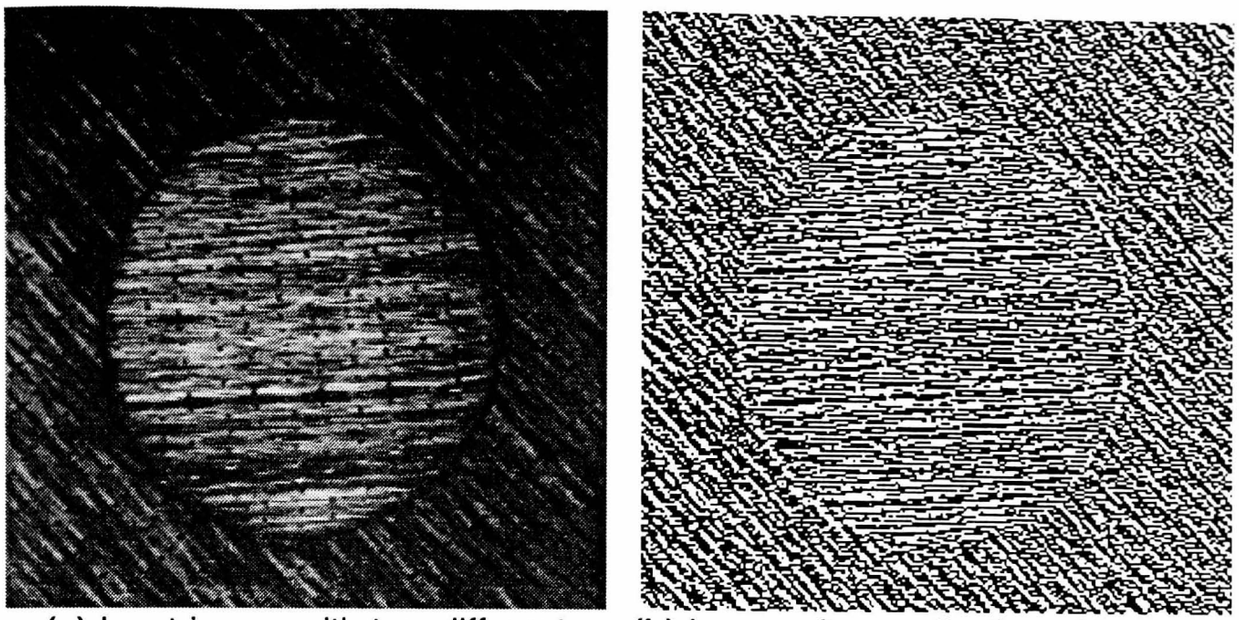
Images of carbon fibre plies exhibit texture due to the woven nature of

the material. Plies cut from different material types have a different weave and therefore a different texture. Plies of the same material type but at different orientations can also be differentiated on the basis of texture, if texture descriptors which are not rotationally invariant are used. For these reasons, the preliminary investigation into texture based segmentation of images of carbon fibre focused on determination of weave pattern and orientation. Initial work was directed towards developing a means of enhancing the weave structure in images of carbon fibre, so that subsequent processing stages could perform segmentation based on this weave information. The first method developed for this purpose was based on a hill-climbing operator, called the **MAX-MIN** operator. Although this operator produced some visually interesting results, it was not efficient enough at enhancing the weave information. It is, however, described in **Appendix A** for the sake of completeness.

A better means of enhancing the weave information is provided by the Laplacian operator. The Laplacian is a second order derivative of the image function. Equation **(3.1)** gives the formal definition where **f** is the image function, and also shows a convolution mask commonly used as a discrete approximation.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \qquad (3.1)$$

This discrete Laplacian is a very effective line and spot enhancer. This is easily demonstrated by reference to **Figure (3.2.1)**, where application of the Laplacian to an input image of different orientations of unidirectional material can be seen to have extracted much of the weave information. Image (b) has been thresholded to produce a binary representation of the weave structure inherent in the respective cloths.



(a) Input image with two different orientations of unidirectional carbon.

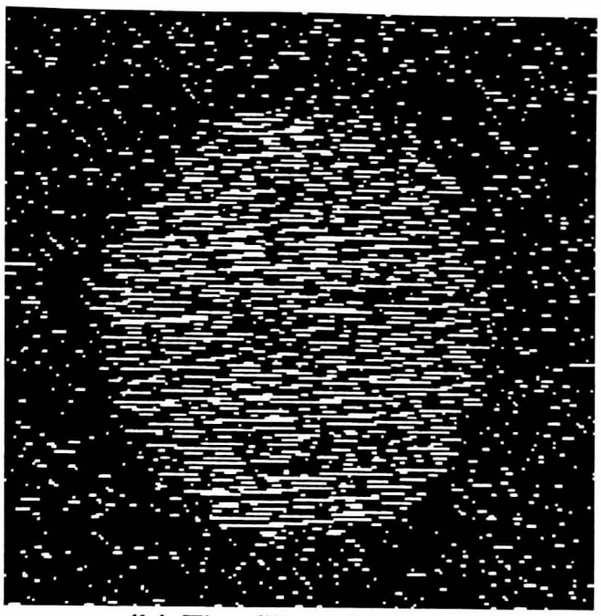
(b) Image after application of Laplacian operator and threshold.

Figure (3.2.1). Extracting the weave information from carbon fibre using the Laplacian operator.

To segment the binary image into regions corresponding to different weave types/orientations, binary filtering was employed. This might be considered either as an adaptation of thinning **[Rosenfeld, Kak, 1982]** or as a binary morphological operation **[Serra, 1982]**. In essence, a filter is applied which will delete (erode) all pixel groups which do not conform to a particular pattern (weave structure). For example, by filtering image (b) of **Figure (3.2.1)** with the binary filter shown in **Figure (3.2.2a)** the weave information of the 0° material is retained but the weave information of the -45° material is eroded, as shown in **Figure (3.2.2b)**. For the unidirectional cloth shown in **Figure (3.2.1)**, the binary filter is simply a binary line at an orientation which matches the texture to be retained. Other textures in the image, which do not exhibit weave structure at the appropriate orientation, are substantially eroded. The remaining stage, segmentation of the image into distinct regions, can be accomplished by counting the number of white pixels in the local vicinity of each pixel. In this example, a high number of white pixels means the pixel probably corresponds to an area of the 0° material, a low number of white pixels means the pixel probably corresponds to an area of the -45° material.

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0

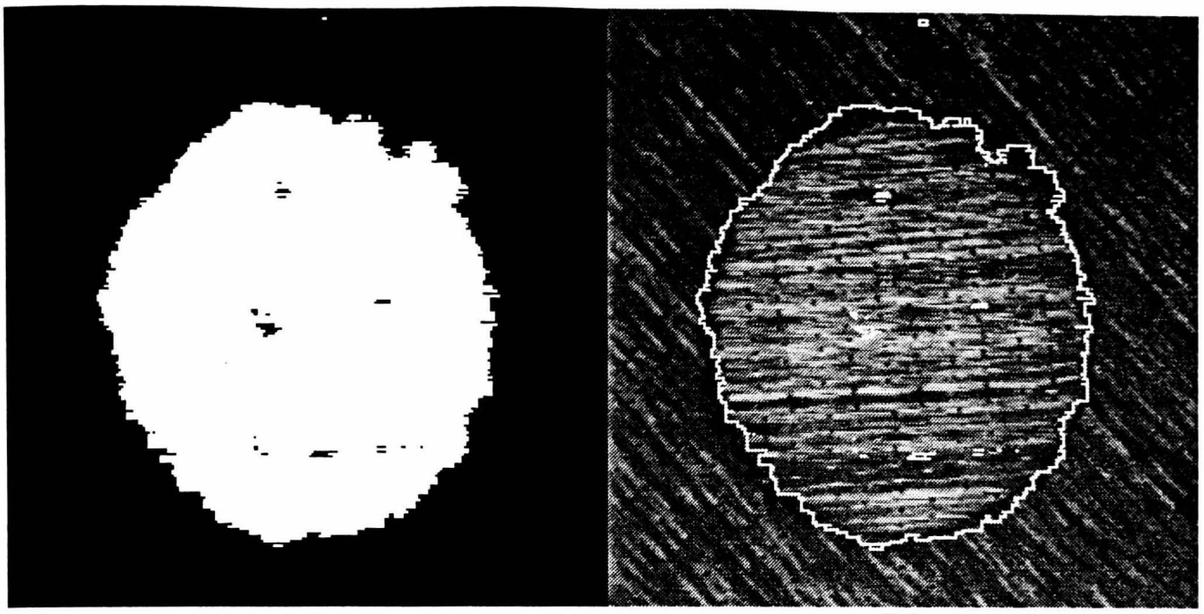
(a) Binary filter.



(b) The filtered image.

Figure (3.2.2). The binary structure image shown in **Figure(3.2.1)** is filtered with the binary filter shown in (a) to produce the image shown in (b).

This process of counting white pixels in a binary image can be implemented as a moving average filter followed by an appropriately set threshold. A moving average filter is efficiently implemented in image processing by convolving an image with a filter which has each coefficient equal to 1, and normalising the result according to the size of the filter. This process is analogous to the smoothing operation described in **Section 2.4.3** as part of Laws texture analysis model. Laws and other workers used a 15x15 smoothing filter, and after some experimentation a 15x15 filter size was also adopted in this work. The decision as to whether a pixel corresponds to one material type or another can now be implemented by thresholding the smoothed image. The result of smoothing and thresholding the image of **Figure (3.2.2b)** is shown in **Figure (3.2.3a)**. The boundary between the segmented regions can be extracted and overlaid on the original image to illustrate the segmentation achieved. This is shown in **Figure (3.2.3b)**. A block diagram of the processing stages which have produced the segmented image shown in **Figure (3.2.3a)** is shown in **Figure (3.2.4)**.



(a). (b).

Figure (3.2.3). (a) is the result of smoothing and thresholding the image shown in **Figure (3.2.2b)**. (b) shows the boundary extracted from (a) overlaid on the original image.

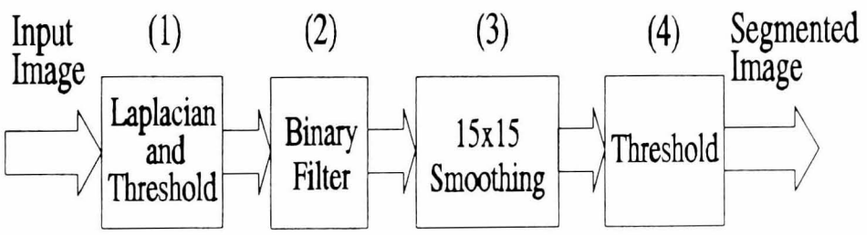


Figure (3.2.4). Block diagram of the binary filter texture analysis method.

Each stage is briefly described below.

(1) Input image is convolved with Laplacian operator, and the output thresholded to produce a binary structure image.

(2) Binary filter is applied to remove unwanted weave information.

(3) 15x15 moving average filter is applied to smooth features.

(4) Threshold applied to segment image into foreground and background.

When dealing with materials exhibiting more complex spatial patterns, such as woven material or unidirectional material with glass fibre weft, the problem of finding the appropriate binary mask to separate textures must be addressed.

This question is particularly relevant for an industrial inspection system, where the product material may well change from time to time. It is obviously desirable to find appropriate masks without painstakingly testing ad-hoc guesses, and so a more formal method was developed.

3.3 Supervised Training of Binary Filters.

The objective is to produce a mask which will discriminate between the weave structures of two plies. Given an image displaying areas of both, the first operation is to apply the Laplacian operator and threshold at an appropriate level (chosen to retain weave structure and minimise noise). Two regions in the binary image are selected, **A** and **B**, as **training regions**, one from each of the relevant texture areas. The criterion for the filter to be generated is that it should have minimal effect on region **A**, but will maximally erode the edge information in region **B**. The task is therefore to establish the most commonly occurring pattern in **A** which does not commonly occur in **B**. By filtering with a mask representing this pattern, separation of the two texture areas can be achieved. The determination of the most commonly occurring pattern is achieved as follows.

For every pixel **p** which is *set* in the training region **A**, examine the surrounding neighbourhood $\mathbf{N_p}$ (a 5x5 matrix centred on **p**). For every pixel set in $\mathbf{N_p}$, the corresponding entry in a 5x5 accumulator array **Acc** is incremented. That is,

$$\sum_{\forall p \in A} \sum_{i=-2}^{i=2} \sum_{j=-2}^{j=2} Acc(i,j) = Acc(i,j) + N_p(i,j) \qquad (3.2)$$

where

$$N_p(i,j) = \begin{cases} 1 & \text{for pixel } p \text{ on} \\ 0 & \text{for pixel } p \text{ off} \end{cases}$$

The accumulator array, once it has been normalised, is a measure of the relative frequency at which binary patterns occur within region **A**. This array is thresholded in such a way as to remove all but the **M** most frequently occurring pixels. **M** is typically chosen as 5, but may be increased if the weave structure image is particularly "busy". **Figure (3.3.1)** shows the normalised accumulator array **Acc** constructed from a sample of woven material, and the

corresponding thresholded version, $\mathbf{Acc_T}$.

36	15	0	0	1
11	9	0	2	1
34	62	100	62	34
2	2	0	10	11
1	0	0	15	38

Acc

1	0	0	0	0
0	0	0	0	0
0	1	1	1	0
0	0	0	0	0
0	0	0	0	1

$\mathbf{Acc_T}$

Figure (3.3.1). The normalised accumulator array extracted from a training region of woven, and the thresholded version.

The thresholded accumulator array shown in **Figure (3.3.1)** is not necessarily a representation of the best mask for separation of the two training regions **A** and **B**. It represents the **M** most frequently occurring pixels of region **A** within a 5x5 neighbourhood, but it is by no means guaranteed that they *all* occur at the same time in any one neighbourhood with any great frequency. Usually the most common pattern (which is what the algorithm is trying to find) will be a subset of $\mathbf{Acc_T}$. To find the best mask for region separation all the possible subsets from the thresholded accumulator array $\mathbf{Acc_T}$ must be generated. One constraint is that the centre pixel must always be set, and so the total number of possible mask permutations is $\mathbf{2^{(M-1)}}$. With **M** set to 5, this means that 16 different masks will be generated.

The algorithm therefore generates each mask, applies it to the two training regions **A** and **B**, and measures the separation achieved. The mask giving the best result is chosen. Separation is measured as

$$Ratio = \frac{pixels\ set\ in\ A}{pixels\ set\ in\ B} \qquad (3.3)$$

Figure (3.3.2) shows a screen-dump taken whilst generation and testing of the masks is taking place.

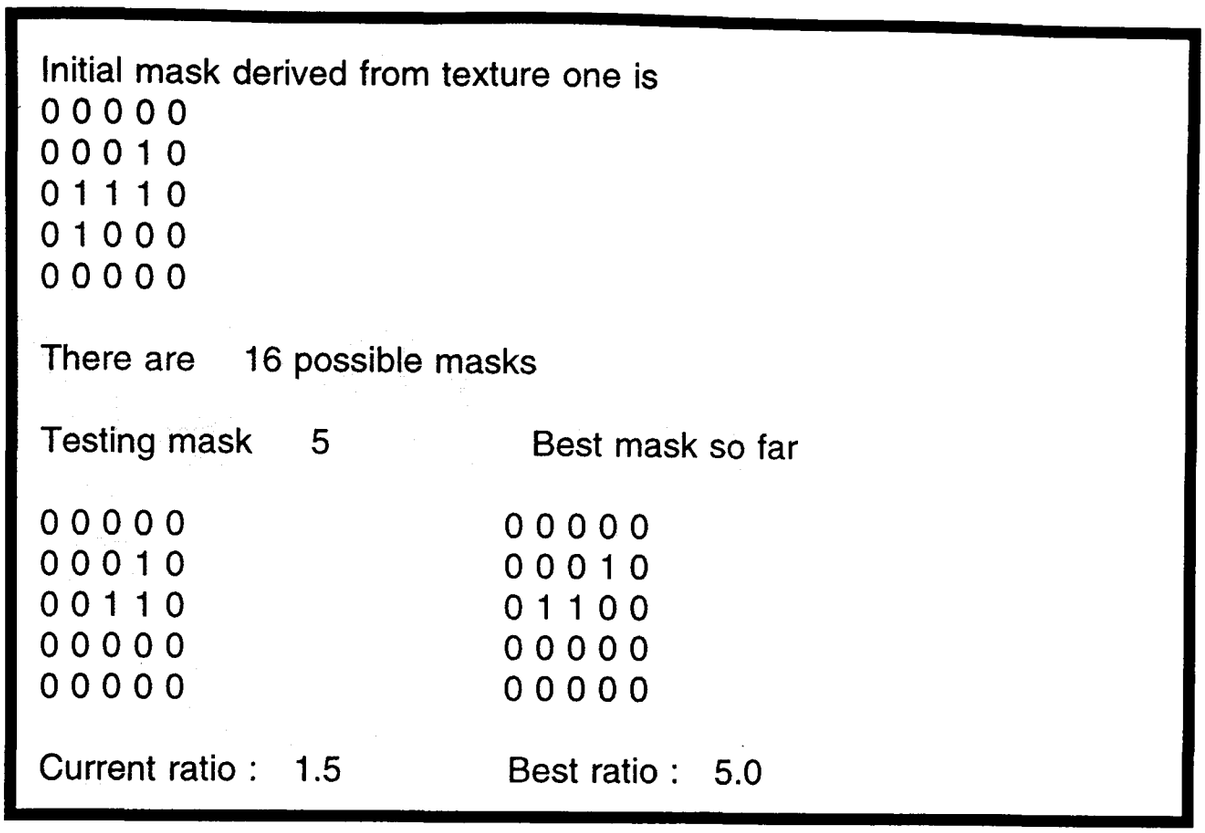
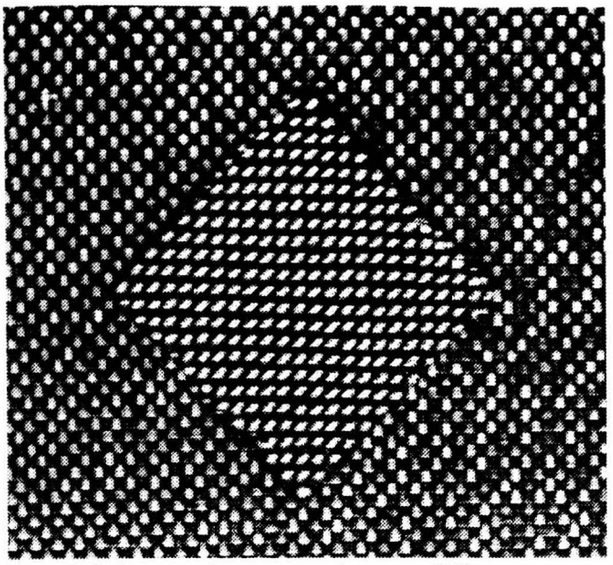


Figure (3.3.2). Screen dump taken whilst generating a binary filter.

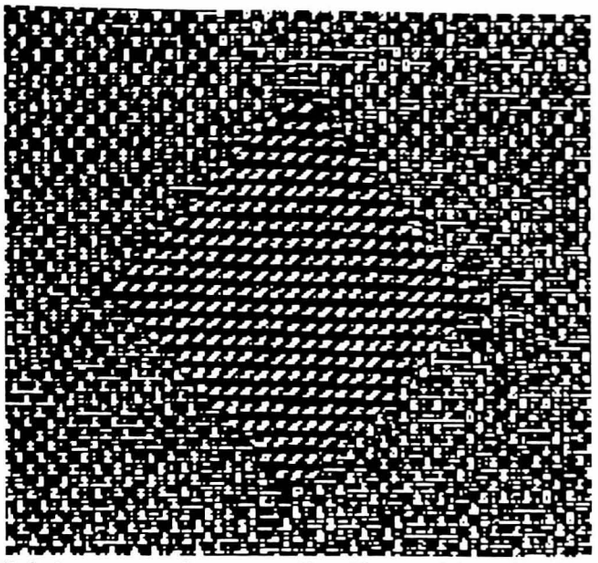
3.4 Assessment of Binary Filtering Method.

Whilst the binary filtering method is capable of segmenting images of carbon fibre, the limitations are considerable. The segmentation achieved is poorer than would be required. The method is highly susceptible to lighting variations. The method is inherently very sensitive to noise, since it involves thresholding at various points. It is not generally applicable to all material types, in that textures where the Laplacian operator does not produce significant pattern differences cannot be segmented.

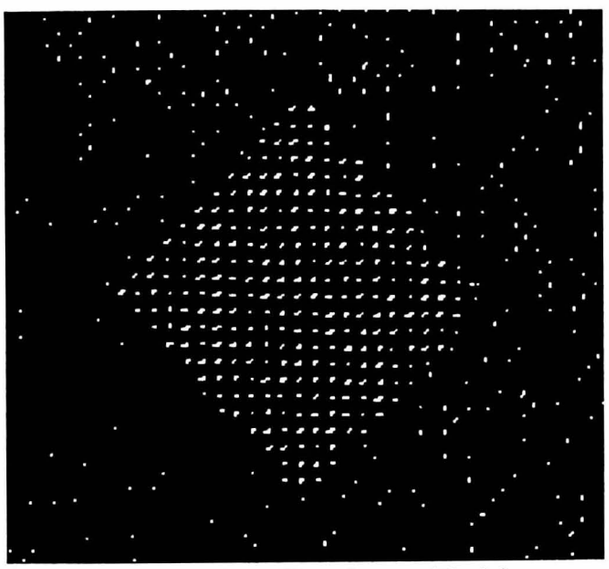
Figure (3.3.3) demonstrates the results achievable using this method to "train" binary filters. The training image contains two orientations of unidirectional material with glass fibre weft. This material is currently very heavily used in the manufacture of various composite aerospace components. The dominant texture from this material arises from the contrast between the white glass fibre and the black carbon fibre.



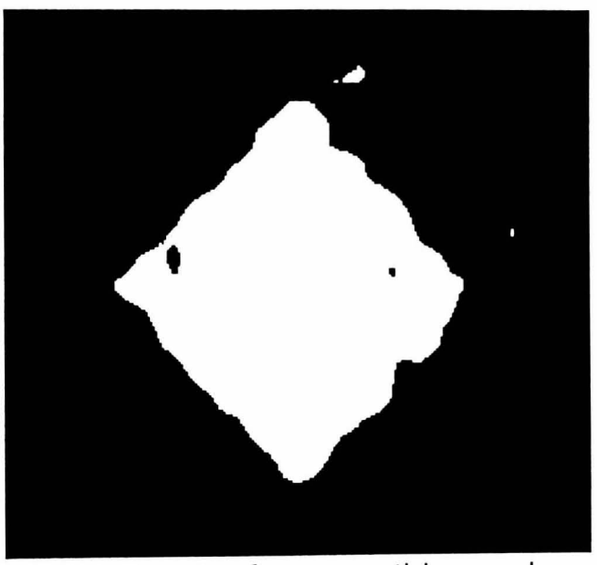
(a) Input image of two different orientations of material.



(b) Image after application of Laplacian operator and thresholding.



(c) Image after filtering with binary mask shown in (e) below.



(d) Result after smoothing and thresholding.

0	0	0	0	0
0	0	0	1	0
0	1	1	1	0
0	1	0	0	0
0	0	0	0	0

(e) Binary mask used to filter image (b).

Figure (3.3.3). An image sequence demonstrating the binary filter texture analysis method. The material in (a) is carbon fibre with glass fibre weft. The binary filter used is shown in (e).

Such limitations, it was hypothesised, could be overcome if the filtering stage could be extended to grey-scale rather than binary. The main problem with binary methods, is that the thresholding operation drastically reduces the information content of the image. Often this is an advantage, but in this particular case a great deal of useful information is also lost. Grey-scale filter masks could utilise this information to improve the feature plane separation of different textures.

3.5 Hardware Considerations.

The extension to grey-scale should also be related to hardware considerations. It was recognised at an early stage in the project that to achieve the desired response time, any computationally intensive algorithms which are required will have to be readily implementable in hardware. Any algorithm can be implemented in hardware given adequate time and resources. For a project such as this however, time and resources are limited, and so it is desirable to avoid dedicated *hard-wired* hardware solutions. One of the main criterion for system design therefore, was that the algorithms developed should be implementable in hardware using only *readily available commercial hardware*, and should not require any special circuitry. The elegance of the proposed design lies in the fact that it is wholly implementable using only convolution, and hardware convolution is standard in most image processing boards available today.

Based on the idea that all texture analysis will be convolution-based, a pipeline processing board has been developed which can perform fast convolution, and which can interface to the framegrabber. The details of the design and construction of this board can be found in **[King,1994]**. A salient feature of the board is its ability to perform convolution up to kernel sizes of 9x7. For texture analysis, a symmetric neighbourhood attached to the central pixel is desirable. The investigation is therefore restricted to square, odd-ordered neighbourhoods, namely 3x3, 5x5, and 7x7. This is a reasonable range of mask size to investigate, since few image processing boards allow

for convolution kernels other than at these dimensions. The 15x15 moving average filter is implemented as a two pass 15x1 and 1x15 operation. This is only possible for a limited class of filter, such as low-pass or Gaussian. Fortunately these are sufficient for the requirements of this application.

This auxiliary processing board greatly facilitated the development of the filtering model from binary to grey-scale. One 3x3 image convolution on the framegrabber board takes about 12 seconds, whereas on the pipeline card it takes 30 milliseconds. More dramatically, to carry out a 15x15 moving average filtering operation takes about four minutes on the framegrabber, but only 60 milliseconds on the pipeline card.

3.6 Binary Filtering and Grey-Scale Filtering.

To test the hypothesis that grey-scale filtering provides an improvement over binary filtering, the binary filters generated by the process detailed in **Section 3.3** were translated into grey-scale representation. That is, rather than produce a binary image which is processed with binary masks, grey-scale masks are applied to a grey-scale image. Note that since it is no longer required to produce a binary image, then the Laplacian filtering and thresholding operations are no longer required. The texture segmentation method is therefore as illustrated in **Figure (3.6.1)**. An example of how a binary filter is represented as a grey-scale mask is illustrated in **Figure (3.6.2)**.

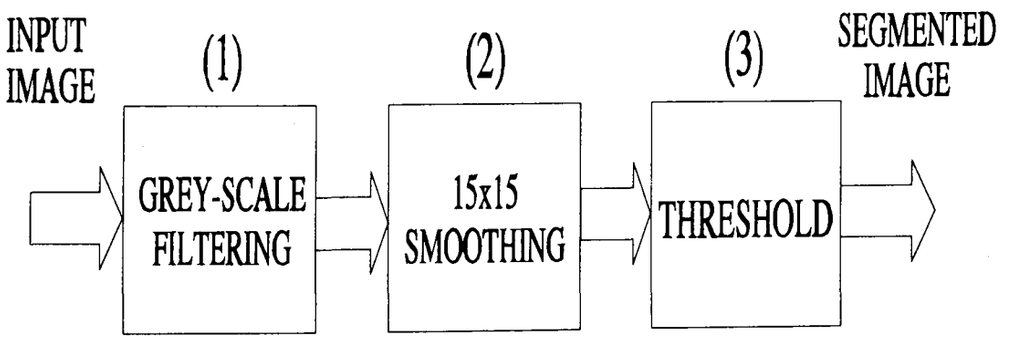


Figure (3.6.1). Texture Segmentation using a grey-scale mask.

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0

0	0	0	0	0
-2	-2	-2	-2	-2
4	4	4	4	4
-2	-2	-2	-2	-2
0	0	0	0	0

Figure (3.6.2). A binary filter and one possible grey-scale interpretation.

Note that the mask shown is only one of a number of possible grey-scale interpretations for the binary mask.

Figure (3.6.1) can be considered a streamlined version of the multi-channel model depicted in **Figure (2.4.3.1)**, and is referred to in this thesis as the **Single Channel Model** for texture analysis. The individual stages of the process represented in **Figure (3.6.1)** are illustrated by the image sequence of **Figure (3.6.3)** which uses the same input image as **Figure (3.2.2)**, and so allows easy comparison. This image sequence is representative of experiments which demonstrated that a single convolution mask can be used for texture segmentation of carbon fibre images. In fact the texture segmentation shown in **Figure (3.6.3)** is surprisingly good, considering the way the mask was derived. A heuristic derivation from a binary mask generated for a subtly different purpose, that of binary pattern erosion, would be thought unlikely to produce an optimal grey-scale texture filter. The reason the mask is successful is that the material in question exhibits a very simple linear primitive. The linearity of the material means that near optimum segmentation results can be achieved by application of a simple edge operator matched to the orientation of the material to be discriminated. Essentially the mask shown in **Figure (3.6.2)** is a horizontal line detector, and so the response from the weave at 0° is stronger than the response from the weave at 45°. For materials which exhibit more complex spatial patterns, finding an appropriate mask would obviously be much more difficult.

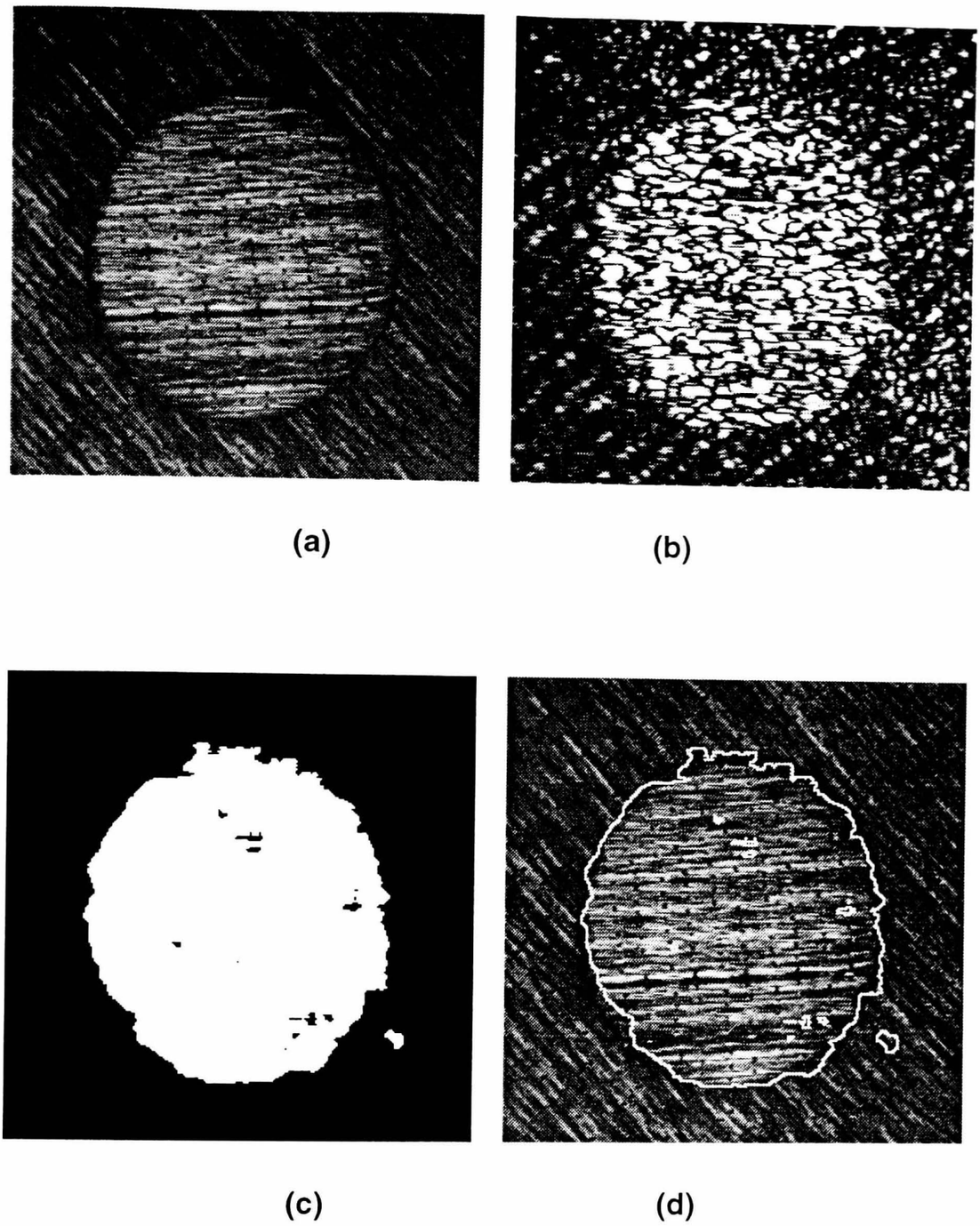


Figure (3.6.3). Image sequence demonstrating grey-scale filtering using the mask shown in **Figure (3.6.2)**. The individual images are as follows.

(a) Input image showing a circular piece of unidirectional cloth at 0° lying on a background of unidirectional cloth at -45°.

(b) Image after filtering with grey-scale filter shown in **Figure (3.6.2)**.

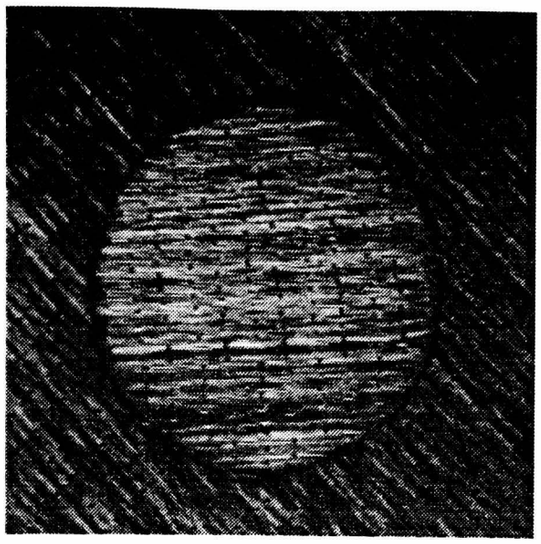
(c) The result after smoothing and thresholding.

(d) The extracted boundary from **(c)** overlaid on the original image **(a)**.

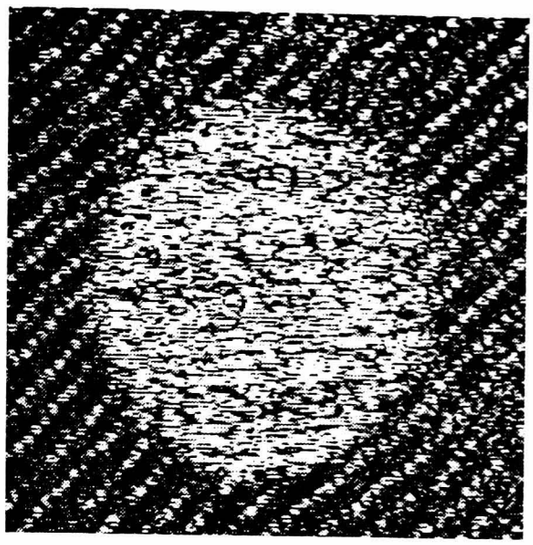
What is required is a mask generation algorithm for grey-scale masks analogous to that described in **Section 3.3** for binary masks. As mentioned in **Section 2.4.3**, such an algorithm has already been developed **[Benke,Skinner,1987]**. This algorithm will be studied in some detail in **Chapter Four**. Improvements will be demonstrated and a new algorithm, called the **basis** algorithm, will be detailed in **Chapter Five** and shown to perform more efficiently. For the purposes of this chapter however, the mechanics of mask generation are not of paramount importance. It is sufficient for the presentation of the single channel model that a mask optimisation method exists. This widens the scope of the method to include textures where the choice of filter for discrimination cannot be derived by an ad-hoc heuristic.

The image sequence of **Figure (3.6.4)** demonstrates the segmentation achieved using a mask trained to discriminate unidirectional at 0° from unidirectional at -45°. The mask is shown in **(e)**, and was produced using the basis algorithm. It is constrained to be zero-sum, and have the maximum value of any one element equal to +/-100. As can be seen from inspection it has a strong horizontal structure.

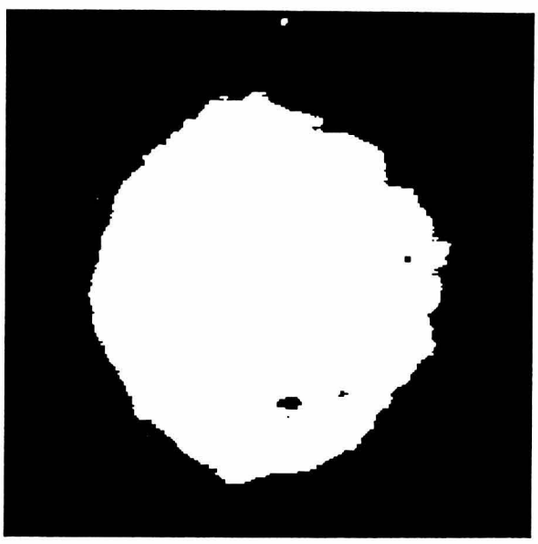
The image sequence of **Figure (3.6.4)** shows little improvement in segmentation over that achieved in **Figure (3.6.2)**, and so for this specific image, little has been gained by using an optimised grey-scale mask. This suggests that the mask used previously was near optimal for the task of discriminating between the two textures. This result only occurred because the structure of the texture in the image is very simple i.e straight line primitives at different angles. For more complex textures however, mask training provides the only way of finding suitable masks for texture discrimination. The examples presented in the following section illustrate this point.



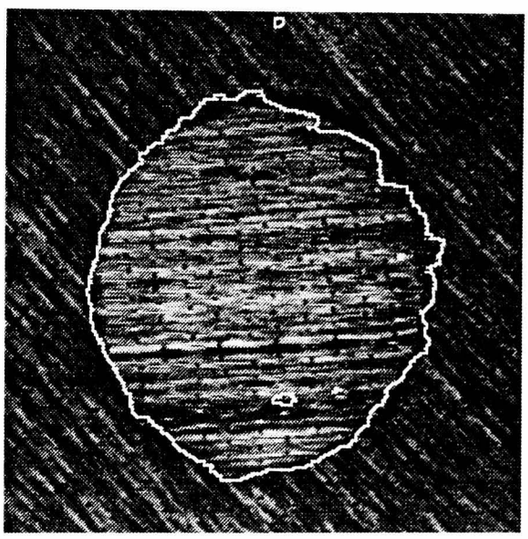
(a)



(b)



(c)



(d)

-16	-27	-82	-56	-24
44	100	99	99	42
48	31	37	-30	-34
-42	-99	-99	-99	-44
14	50	63	20	5

(e)

Figure (3.6.4). Image sequence demonstrating grey-scale filtering using the mask shown in **(e)**. The mask was produced using the basis algorithm detailed in **Chapter Five**.

3.7 Texture Segmentation Using the Single Channel Model.

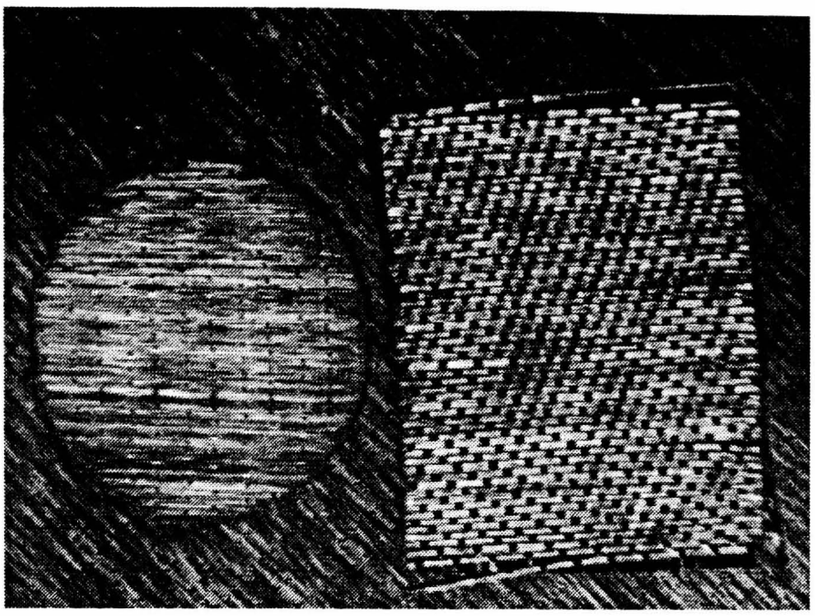
This section will present some examples of texture segmentation using the single channel model. The examples chosen demonstrate the suitability of this method for visual inspection of carbon fibre materials, as well as the applicability to other materials. The relevance of appropriate mask size for a particular texture set is addressed.

Example (1). Image **(a)** in **Figure (3.7.1)** shows three pieces of carbon fibre. The image resolution is 400x300 pixels. The background cloth is unidirectional at -45°, the circular piece is unidirectional cloth at 0°, and the rectangular piece is woven material. This image is chosen to be representative of a lay-up situation where it might be required to check the position of the woven piece. Such a task requires a mask which can discriminate woven texture from the *two* other textures in the image. The mask used is shown in **(e)**, and was produced by the basis algorithm. It seems to be a bar detector oriented at about 60° from the horizontal.

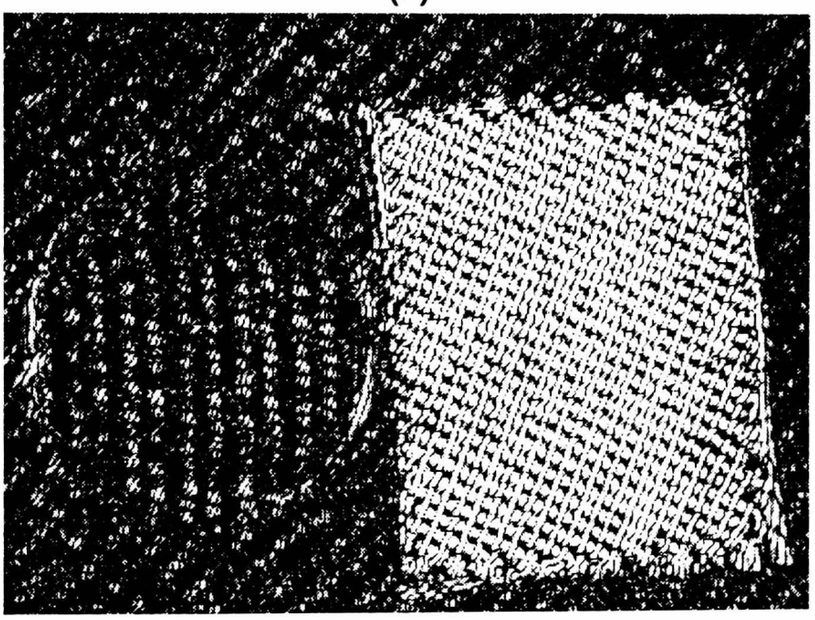
As can be seen from the image sequence of **Figure (3.7.1)**, the woven is successfully detected using this mask. Some points to note are :

- The segmentation is substantially better than that achievable using the binary filter method (not illustrated here).
- It would be difficult to heuristically derive a grey-scale mask which could produce this segmentation.
- The segmentation result, whilst not perfect, would be sufficient for many applications.

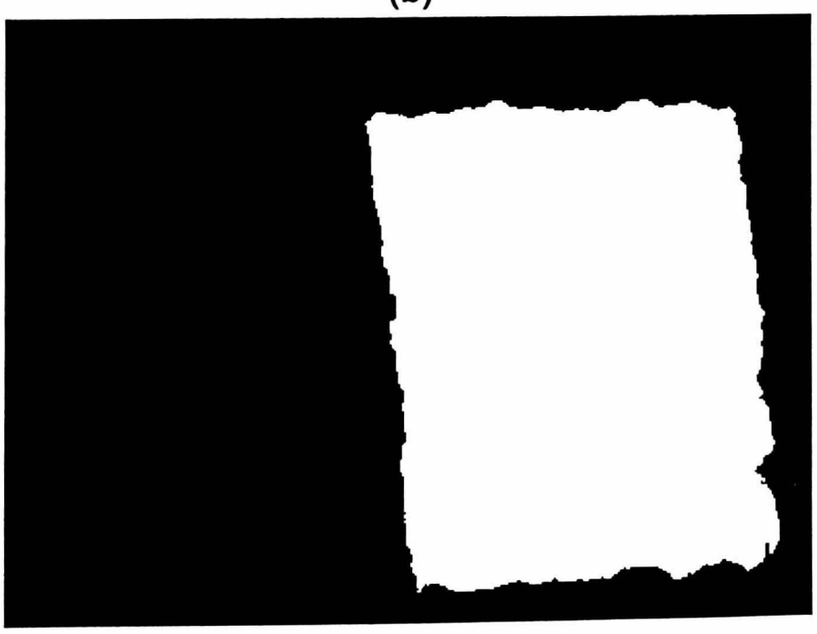
As regards the loose fibres evident at the boundary of the woven material, such defects are not evident in a real application where the cutting process is much more reliable, and the material in much better condition than in this example.



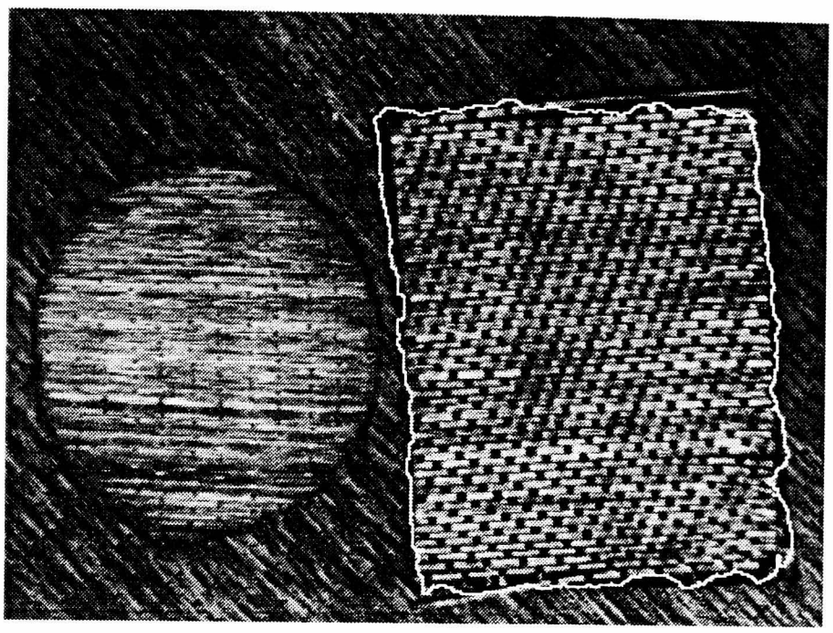
(a)



(b)



(c)



(d)

-50	-41	22	42	20
-55	10	77	13	-33
-65	15	100	6	-63
-29	7	69	3	-59
23	50	26	-39	-49

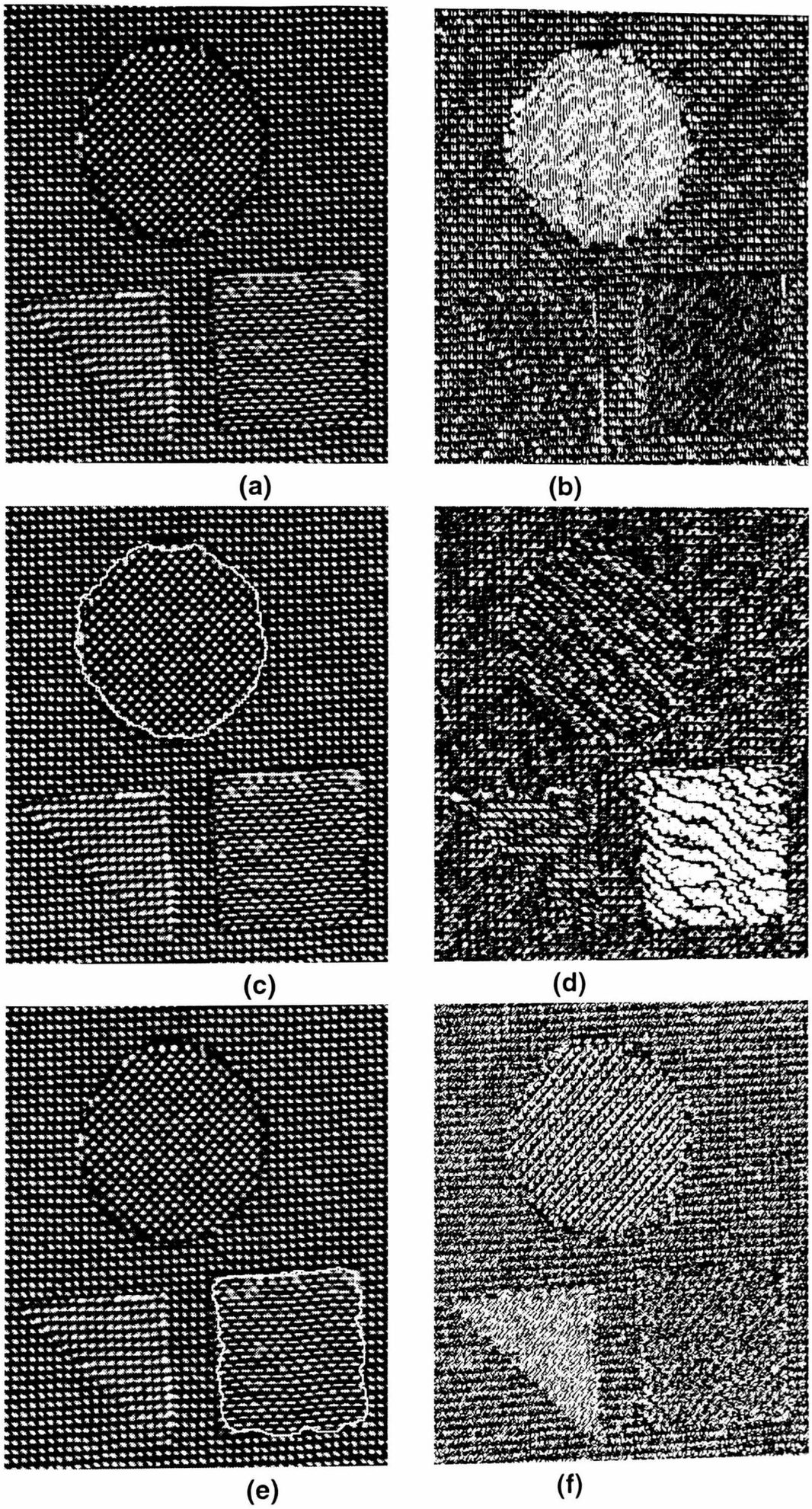
(e)

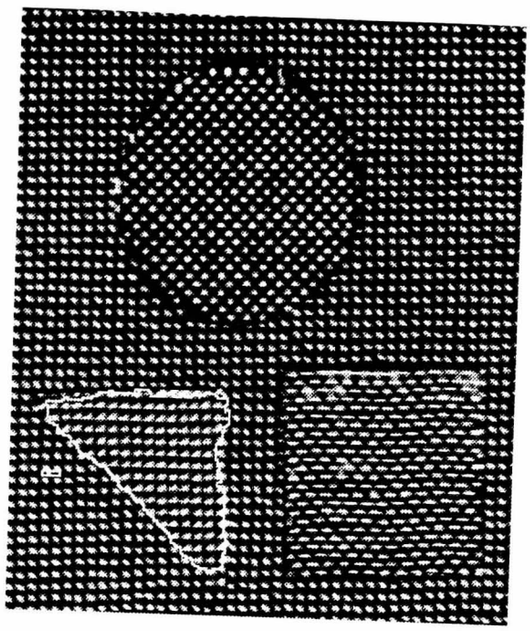
Figure (3.7.1) continued from previous page. Image sequence demonstrating discrimination of cross-ply carbon material from two orientations of unidirectional. The mask used is shown in **(e)**, and was produced using the basis algorithm detailed in **Chapter Five**.

Example (2). Image **(a)** in **Figure (3.7.2)** shows four pieces of carbon fibre with a glass fibre weft. The image resolution is 250x300 pixels. As can be seen from the images, the dominant texture characteristic is due to the contrast between the white glass fibre and the black carbon fibre. Different orientations of material appear quite similar to the human eye, although the different pieces (circle, triangle, square) can be distinguished from the background. The orientations of the pieces are illustrated in **(j)**. Note that this is only an illustration, not a template. The image shows real pieces of carbon which have been manually laid-up.

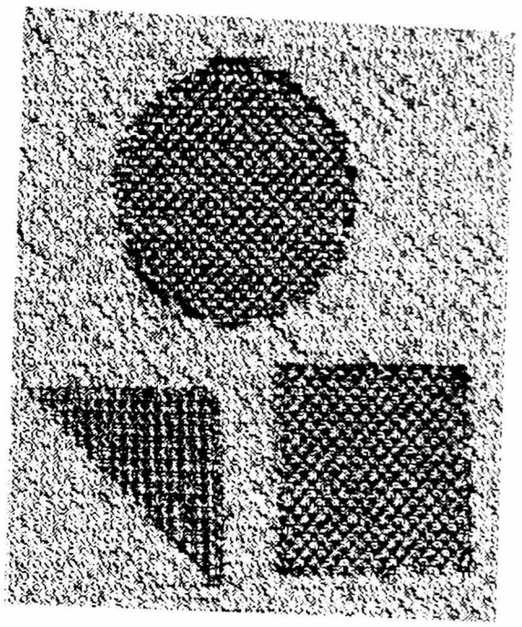
Image (a) *in* ***Figure (3.7.2)*** *represents all the orientations of unidirectional carbon fibre used in composite component manufacture, and as such is an important test image.* The image sequence of **Figure (3.7.2)** shows how any particular orientation of material can be detected by the application of the appropriate mask. Each mask has been trained to maximise the texture energy of a given orientation whilst minimising the texture energy of the other three. For each orientation the filtered image and the resultant extracted boundary are shown. This is an impressive demonstration of the power of a single convolution mask, and goes a long way towards proving the suitability of this texture analysis method for inspection of carbon fibre materials. The masks which provide the discrimination are shown in **Figure (3.7.3)**.

It is worth noting here that in real applications the texture analysis mask need never encompass as many as four textures. In fact for most inspection tasks the objective is merely to detect the boundary between the foreground and background ply, so only two textures need be distinguished.

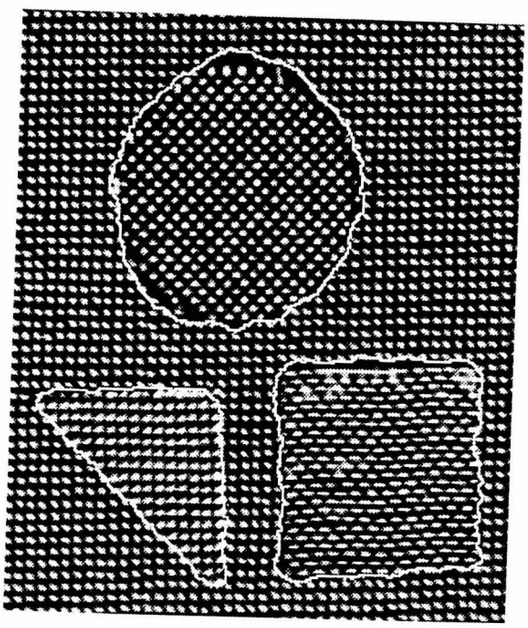

(a)
(b)
(c)
(d)
(e)
(f)



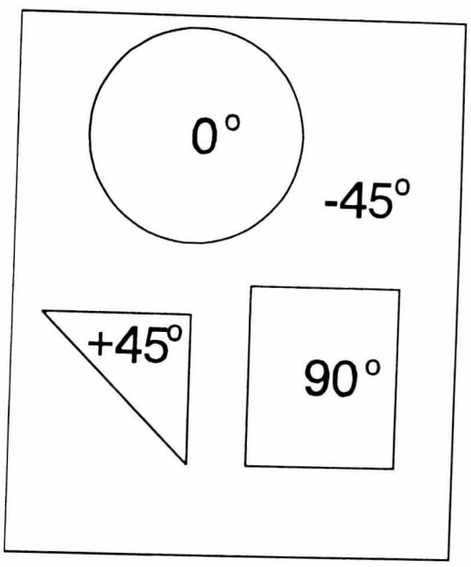
(g)



(h)



(i)



(j)

Figure (3.7.2) continued from previous page. Demonstrating the selectivity achievable by using trained masks to discriminate the different orientations of fibre shown in image **(a)**. An indication of the respective fibre orientations is given in **(j)**. The masks used are shown in **Figure (3.7.3)**.

13	-17	-5	25	-18
68	-81	-32	100	-54
47	-59	-5	89	-71
45	-49	-3	50	-40
31	44	13	30	-33

(b) 0°

7	9	22	6	4
5	-52	-41	-31	3
9	5	97	5	18
-17	-7	-44	-26	-14
0	15	0	31	4

(d) 90°

-5	100	-65	-44	-2
50	-17	-68	20	27
10	-81	33	24	10
-8	-2	16	71	-62
-22	52	37	-83	9

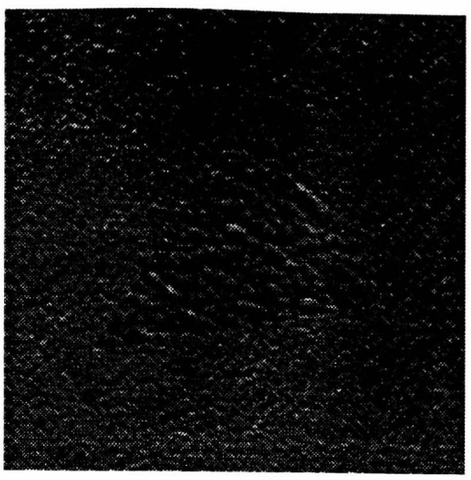
(f) 45°

27	17	-76	12	29
10	34	-2	-46	8
-34	-45	97	3	-41
-3	0	-35	24	12
29	-11	-49	11	29

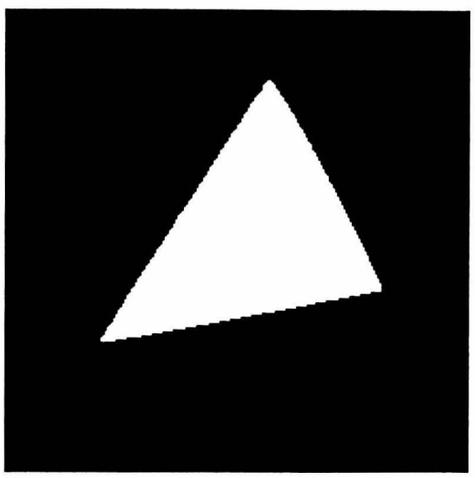
(h) -45°

Figure (3.7.3). The masks used in **Figure (3.7.2)**. The caption below each mask corresponds to the caption of the relevant image.

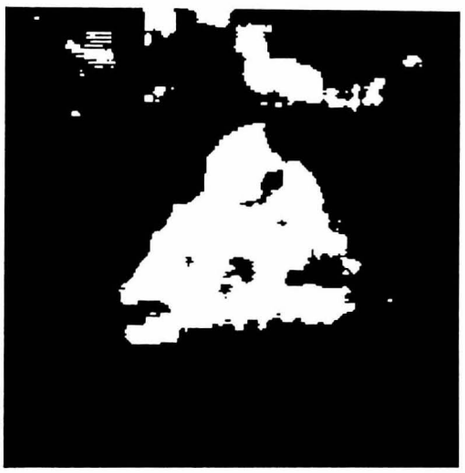
Example (3). The next example demonstrates the use of a single convolution mask on textures which are less regular than the man-made materials common in industrial inspection. Image **(a)** of **Figure (3.7.4)** shows an image in which one leather sample is overlaid on another. The leathers in question have a different grain, and therefore a different texture. The template used to produce the overlay is shown in image **(b)**. **Figure (3.7.4)** is the first image in this thesis that does not exhibit "real" boundaries, but rather is a composite of textures. By reference to the template the segmentation achieved by the texture analysis process can be measured. Image (a) in **Figure (3.7.4)** represents a difficult task for segmentation, in that the respective textures are irregular. Such textures are often difficult to separate in feature space, and the result can be poor segmentation especially near region boundaries.



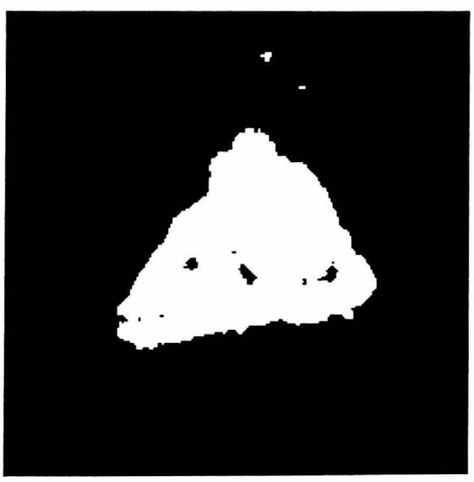
(a)



(b)



(c)



(d)

(e)

Figure (3.7.4). Example demonstrating how segmentation improves with larger mask size for non-regular textures. **(a)** Original image with one leather texture overlaid on another **(b)** Template used to generate overlay **(c)** Segmentation by 3x3 mask **(d)** Segmentation by 5x5 mask **(e)** Segmentation by 7x7 mask.

The results shown in **Figure (3.7.4)** are good, and demonstrate again the power of convolution based texture segmentation. Some notable properties of the method can be detailed by reference to these images. Images **(c)**, **(d)**, and **(e)** show the segmentation achieved using mask sizes of 3x3, 5x5, and 7x7 respectively. A mask size of 5x5 exhibits a large improvement in segmentation over that achieved by 3x3, while a further increase to 7x7 provides only a marginal improvement. There are two processes which contribute to this result.

- If the mask size is smaller than the spatial extent of the texture elements which make up the texture, then poor discrimination results. This accounts for the large errors in image **(c)** where 3x3 masks have been used.
- For irregular or random textures, the smoothing implicit in larger masks will reduce the effects of any inhomogeneous areas of texture and so provide slightly better segmentation. This accounts for the smoother segmentation shown in image **(e)**.

One more segmentation feature is evident from the image sequence of **Figure (3.7.4)**, namely the "rounding-off" of region corners. This effect is discussed more fully in the next section.

3.8 Segmentation Accuracy.

There are two distinct types of segmentation error: misclassification of pixels within a texture region, and misclassification of pixels at or near a boundary between texture regions. In the examples presented so far both types of error have been evident. For an application such as ply inspection where the individual texture regions are comparatively large, then inter-regional errors are not particularly important. Small "erroneous" regions can be discarded on the basis of area or perimeter, and so segmentation errors of this type are not critical. Conversely, since the detection of boundaries is the prime aim, then misclassification of pixels at or near a region boundary is

much more of a concern. The discussion of segmentation accuracy is therefore presented here only in terms of boundary accuracy. The effects of region curvature, mask size, smoothing operator, and secondary smoothing are all investigated.

In order to measure boundary accuracy, it is preferable to use composite images where the position of the texture boundaries are known exactly, and so error evaluation can be carried out more precisely. The images used to determine boundary accuracy have been generated by overlaying one texture on top of another, using a template to delineate the respective regions. Each image sequence shows the composite image, the template used to generate that image, and the segmentation achieved by texture analysis. The difference between the boundaries of the template and of the segmented image constitute boundary errors.

The first set of images use denim as the texture, with two orientations of denim providing two different textures. Denim was chosen since it can be segmented using 3x3 masks as well as 5x5 and 7x7, and so allows comparison between different mask sizes. Three templates have been used to generate the images, each chosen to exhibit different amounts of boundary curvature. The templates themselves were digitised from hand-cut paper shapes, and so are not geometrically accurate. The image sequences of **Figure (3.8.1)** show the three composite images used for the experiment, the templates used to generate them, and the segmentation results achieved using a 3x3 mask. Images **(a)** through **(c)** show the segmentation of an image created using a circular template, images **(d)** through **(f)** using a "C" shaped template, and images **(g)** through **(i)** using a star shaped template.

The threshold in the texture segmentation process has been automatically chosen each time so that the number of pixels above the threshold is as near as possible to the number of pixels in the template. This simple device results in near-optimal thresholding, and produces a segmentation result which is visually very close to the template.

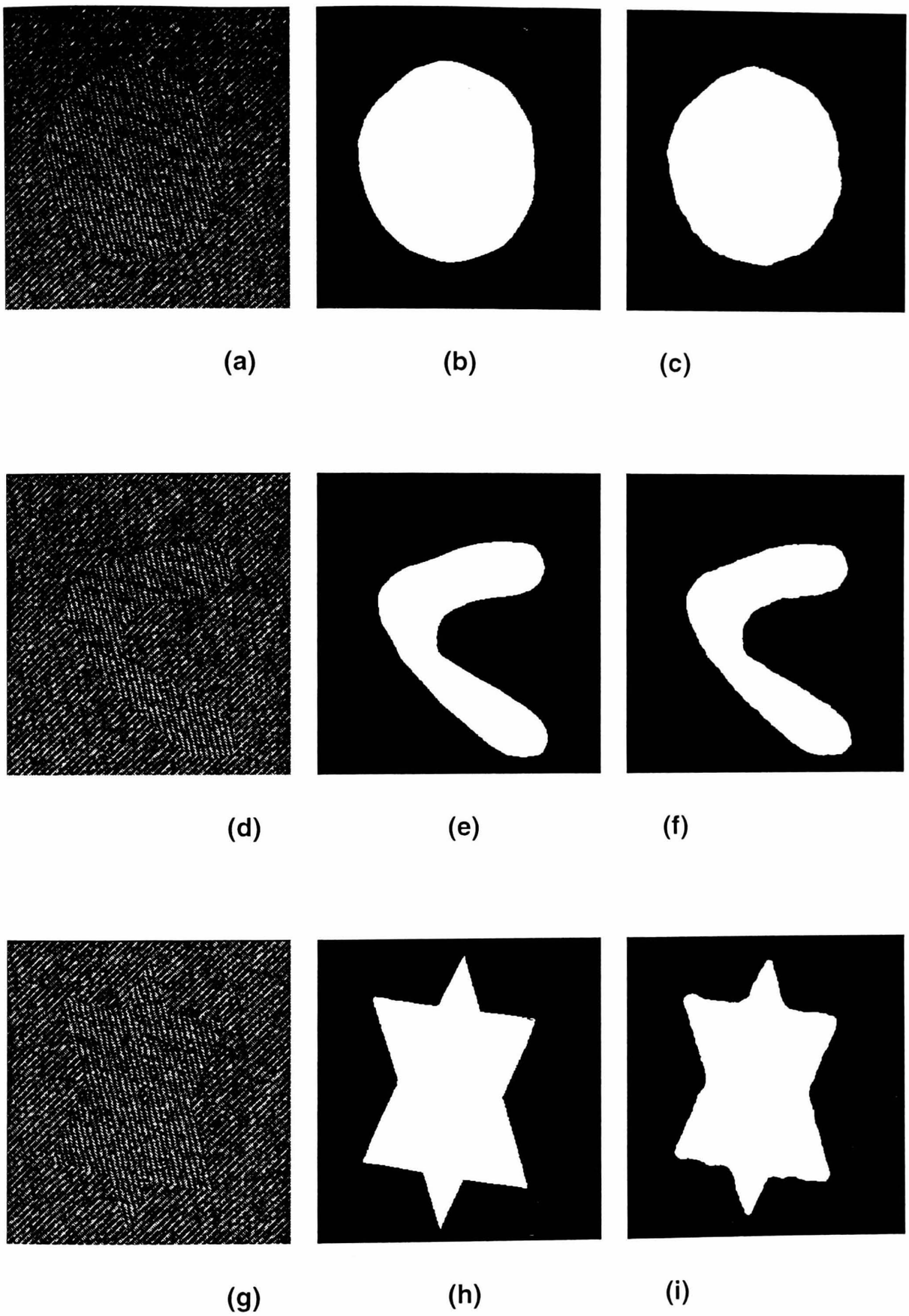


Figure (3.8.1). The texture images used to measure boundary accuracy, the templates used to generate the texture overlay, and the segmentation achieved using a previously trained 3x3 mask.

The images in **Figure (3.8.1)** serve only to illustrate the segmentation for one particular 3x3 mask. A more quantitative assessment of the effect of mask size and boundary curvature is presented in **Table (3.1)**, which details the mean boundary error and maximum boundary error for each mask size on each image. All errors are measured in pixels, and not as Euclidean distances. From the table the effects of curvature and mask size can be considered.

IMAGE	BOUNDARY ERRORS					
	3x3		5x5		7x7	
	MEAN	MAX	MEAN	MAX	MEAN	MAX
CIRCLE	0.85	4	0.86	4	0.75	3
"C"	0.87	4	0.9	3	0.8	3
STAR	1.52	10	1.63	10	1.52	10

Table (3.1). Table of boundary errors incurred by different mask sizes on the three composite images shown in **Figure (3.8.1)**. All the figures quoted are in pixels.

Regarding the effect of **boundary curvature**, the results for the circle image and the "C" image are very similar. This suggests that the quite high curvature exhibited by the "C" image does not have a significant effect on segmentation. The extreme curvature of the star image does, however, have a detrimental effect on segmentation. This is evident from the boundary errors, and from inspection of image **(i)** in **Figure (3.8.1)** which shows the "rounding off" of the star corners. Such an effect is common to all region based analysis methods. The extreme points of the star in **Figure (3.8.1)** represent only a few pixels, and so the dominant texture in the neighbourhood is that of the background material. This effect is also evident in the human vision system. Where we *can* detect the presence of "texture corners", it is probably due to the extrapolation of detected boundaries which allow us to deduce the existence of corners.

The effect of **mask size** seems to be somewhat less linear than might have been expected. It has already been mentioned in the previous section that too small a mask size will result in poor segmentation, and that larger

masks provide more smoothing and so generally better segmentation. With regards to boundary errors, it might have been expected that this extra smoothing implicit in the larger masks would result in edge attenuation, and so increase boundary errors. 3x3 masks might therefore be expected to provide more accuracy. From the data in **Table (3.1)** it seems that this is not so. The most accurate boundaries are provided by 7x7 masks, with 5x5 masks producing the least accurate. This non-linear effect is representative of the unpredictable results found in other experiments carried out within this project. The conclusion is that the best mask size for segmentation accuracy in the single channel model is ***texture dependent***. It is tempting to hypothesise that the mask closest in size to the texture primitive will provide best results, but in the absence of a suitable method to determine primitive size, then this must remain speculation.

A second set of experiments were carried out to determine the effect of the **smoothing filter** on boundary error. The segmentation feature proposed by Laws is the absolute sum of values in a 15x15 neighbourhood. This corresponds to filtering a rectified image with a 15x15 mask where each element is set to 1. For ease of reference this mask is referred to from now on as **low-pass**. Another possibility is to use a **Gaussian** filter, which as the name suggests has a Gaussian spatial profile. Such a mask has several attractive properties. It is smooth and localised in both the spatial and the frequency domains, and so is least likely to introduce any changes which were not present in the original image **[Marr,1982]**. It is far more isotropic (rotationally invariant) than the low-pass mask, and isotropy reduces the distortion of edges in the filtered image. Equally importantly, a 15x15 Gaussian mask can be decomposed into two one-dimensional Gaussian operators with dimensions 1x15 and 15x1 **[Niblack,1985]**. This means that such a mask can be implemented on the pipeline processing card as a two pass process.
In choosing the Gaussian mask, the method of **[Davies,1987]** is followed, which states that the optimum value of sigma is directly proportional to the linear dimension of the neighbourhood. Since the mask is being implemented as two one-dimensional filters, then the isotropy cannot be properly

maximised, and so the criterion to maximise is taken as the accuracy of the Gaussian as implemented in a 15x15 convolution mask. For a 15x15 neighbourhood the corresponding value of sigma is 3.6. Using this Gaussian as a smoothing mask rather than the low-pass, the results shown in **Table (3.2)** are obtained.

IMAGE	BOUNDARY ERRORS					
	3x3		5x5		7x7	
	MEAN	MAX	MEAN	MAX	MEAN	MAX
CIRCLE	0.75	3	0.79	3	0.73	3
"C"	0.82	3	0.83	4	0.77	3
STAR	1.2	10	1.32	10	1.27	10

Table (3.2). Table of boundary errors incurred by different mask sizes on the three composite images shown in **Figure (3.8.1)**. The smoothing is performed by a Gaussian mask. All the figures quoted are in pixels.

These figures are an improvement over the results shown in **Table (3.1)** using the low-pass smoothing mask. Even from inspection of the smoothed images, it appears that the region boundaries are more clearly defined. The improvement is especially noticeable with reference to the star image, where the mean boundary error has decreased by approximately 20% for each mask size. The main reason is that the Gaussian mask produces less rounding of corners than the low pass mask. This is because the Gaussian mask is centrally weighted. As a result, less blurring occurs, and so small regions are less likely to be smoothed out of existence.

In addition to the results of these tests, many images taken from preform lay-ups have been processed using both Gaussian and low-pass smoothing. The opinion of human observers has been that Gaussian smoothing produces less edge attenuation. The conclusion both from controlled experimentation and subjective observation therefore, is that Gaussian smoothing is preferred over low-pass smoothing.

A third set of experiments has been carried out to test the effect of what

is referred to here as **secondary smoothing**. This entails a second smoothing and thresholding stage after the initial segmentation, the result of which is to smooth the region boundaries. The full segmentation process is then as represented in **Figure (3.8.2)**. Stages **(4)** and **(5)** are identical to stages **(2)** and **(3)**. The effect this additional smoothing process has on boundary accuracy can be found in **Table (3.3)**.

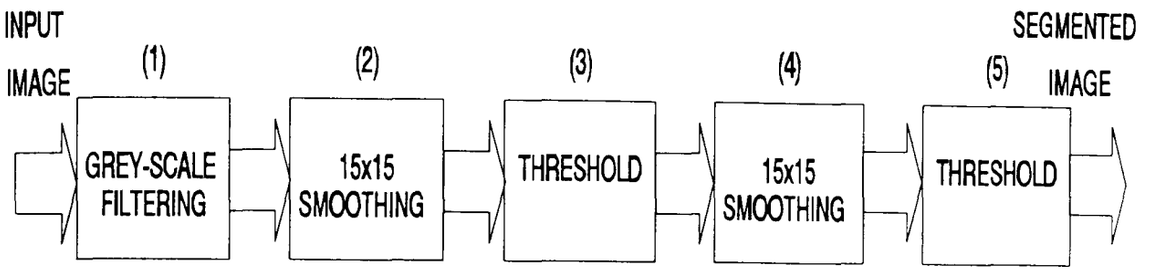


Figure (3.8.2). Texture segmentation using grey-scale filtering with a secondary smoothing stage.

IMAGE	BOUNDARY ERRORS					
	3x3		5x5		7x7	
	MEAN	MAX	MEAN	MAX	MEAN	MAX
CIRCLE	0.71	3	0.70	3	0.66	3
"C"	0.74	3	0.78	4	0.72	3
STAR	1.35	10	1.5	10	1.4	10

Table (3.3). Table of boundary errors incurred by different mask sizes on the three composite images shown in **Figure (3.8.1)**. Smoothing is carried out using a Gaussian filter. A secondary smoothing stage is carried out using the same mask. All the figures quoted are in pixels.

These figures provide an interesting result when compared to those of **Table (3.2)**. For the circle and "C" images, where the region boundaries are smooth with modest curvature, secondary smoothing results in a decrease in boundary errors of between 6% and 11%. For the star image where the region boundaries have areas of extreme curvature (corners), secondary smoothing results in an *increase* in boundary errors of between 10% and 13%. The variations obtained are similar for Gaussian smoothing and normal low-pass

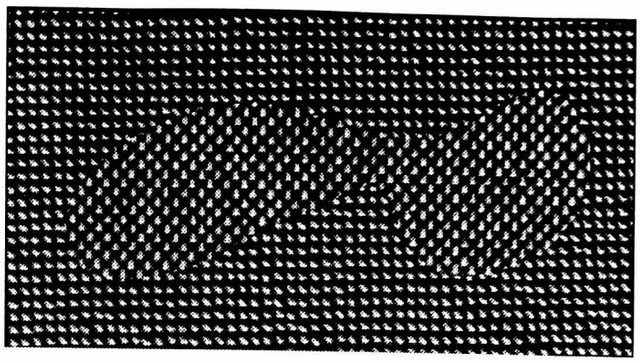
smoothing (not reported here), although Gaussian always exhibits the best result. This suggests that secondary smoothing may be a useful function in certain applications where the regions are expected to exhibit smooth boundaries with no points of extreme curvature.

The above experiments relating to mask size, region curvature, smoothing operator, and secondary smoothing, have been repeated on an image where the texture is unidirectional carbon fibre material with glass fibre weft, shown in **Figure (3.8.3)**. This, as has been mentioned previously, is a prime material in many composite manufacturing processes. The image sequence of **Figure (3.8.3)** shows the composite texture image and the generating template, as well as showing two images to illustrate the effect of secondary smoothing on the extracted region boundary. The particular example shown is using a 3x3 mask which produces a particularly ragged boundary with this texture if secondary smoothing is not effected. The table of **Table (3.4)** details the results using this image.

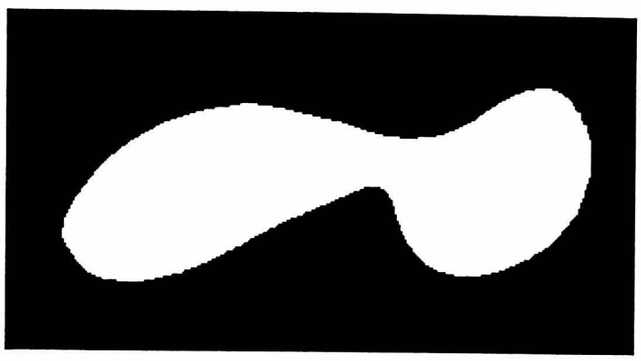
The results in this table show that for this texture, boundary errors decrease as mask size increases. Again this is against the intuitive idea that small masks should produce better accuracy, and seems to reinforce the concept that the mask should be matched to the texture in question.

This table gives a clear indication of the effect the smoothing stage can have on boundary accuracy. The overall decrease in boundary errors obtained by using secondary Gaussian smoothing rather than one low-pass smoothing operation ranges from 10% for 7x7 masks to 30% for 3x3 masks. The maximum boundary error, whilst not quite so well behaved, also tends to reduce with secondary smoothing.

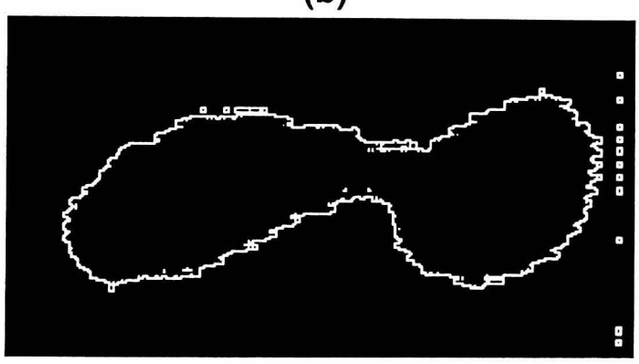
It is, however, important to remember that these results all relate to composite images exhibiting no "real" boundaries, and so should be interpreted with care. Images with real boundaries often have edge areas which exhibit a pattern which is a product not only of the local texture on either side of the boundary (as in composite images), but also of the spatial pattern caused by the physical edge itself.



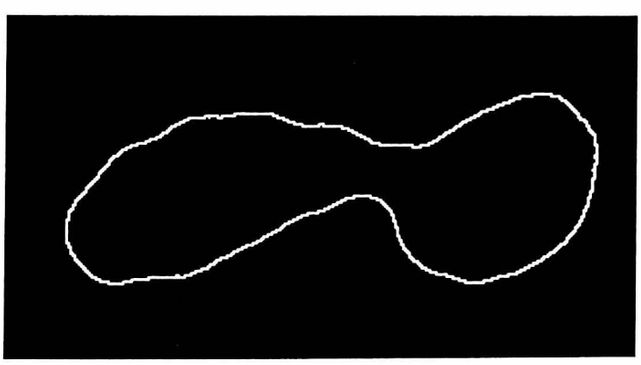
(a)



(b)

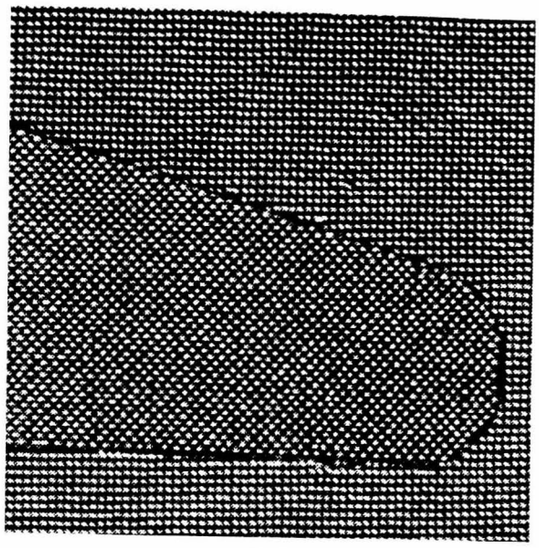


(c)

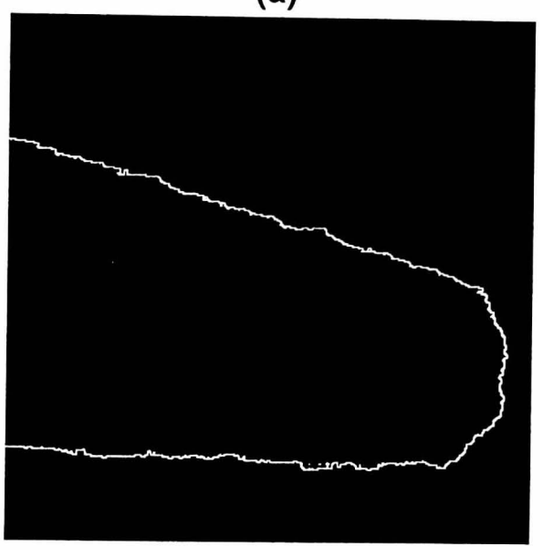


(d)

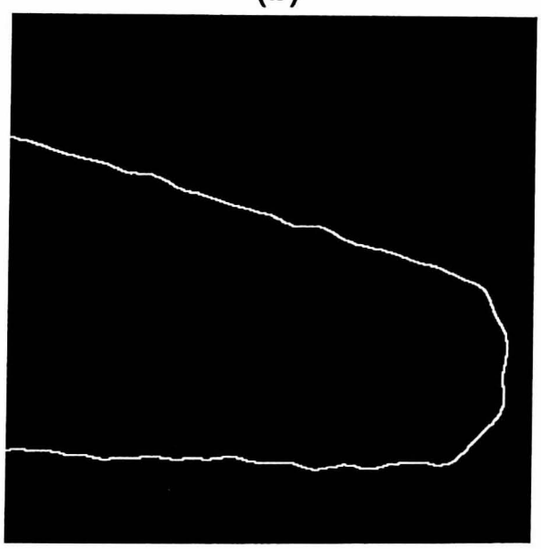
Figure (3.8.3). Demonstration of effect of secondary smoothing. **(a)** shows an image with an area of unidirectional cloth at 0° overlaid on a background of unidirectional cloth at 45°. **(b)** Template which made the overlay **(c)** Boundary extracted from segmentation using 3x3 mask. **(d)** Boundary extracted using same mask but with secondary smoothing.



(a)



(b)



(c)

Figure (3.8.4). Demonstration of effect of secondary smoothing on an image with real edges.**(a)** An image of two plies taken from the robotic lay-up cell. **(b)** Boundary extracted from segmentation using 5x5 mask. **(c)** Boundary extracted using the same mask but with secondary smoothing.

Secondary smoothing does produce "cleaner" boundaries on such images (see **Figure (3.8.4)**) but the effect on boundary accuracy is not necessarily beneficial. This last point is addressed in more detail in **Chapter Seven**.

IMAGE	SMOOTHING	BOUNDARY ERRORS					
		3x3		5x5		7x7	
		MEAN	MAX	MEAN	MAX	MEAN	MAX
CARBON	Low-pass	2.01	7	1.29	6	1.13	5
	Low-pass 2nd	1.58	5	1.17	7	1.09	6
	Gaussian	1.67	7	1.19	5	1.12	4
	Gaussian 2nd	1.44	6	1.08	5	1.02	4

Table (3.4). Table of boundary errors incurred by different mask sizes on the composite image shown in **Figure (3.8.3)**. Data is given using low-pass smoothing, secondary smoothing with low-pass, Gaussian smoothing, and secondary smoothing with Gaussian. All the figures quoted are in pixels.

The results presented in this section, although derived from a small dataset, are consistent with the performance of the single channel model for texture segmentation. The observations that can be made are summarised in the following section.

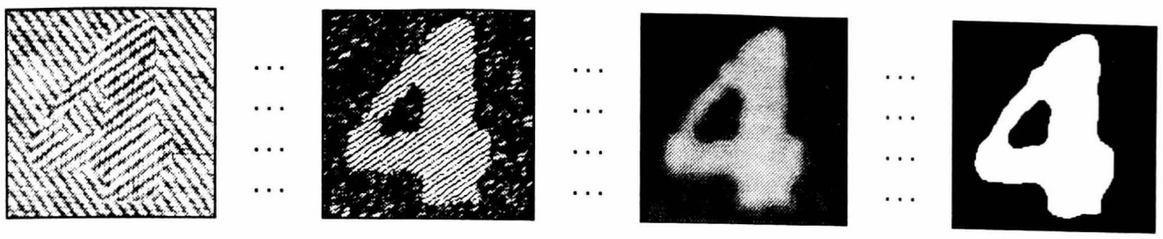
3.9 Conclusions.

This chapter has presented the single channel texture analysis model, and examined it in some detail. The main points are as follows:

- The method is elegant, in the sense that it is easily realisable in hardware.
- For regular textures, successful discrimination can be achieved between at least four textures in any one image.
- Irregular textures, whilst more difficult, can also be discriminated. Larger masks may be required, and the number of textures which may be discriminated in any one image may be lower than for regular textures.
- Segmentation accuracy is generally good, assuming an appropriate

mask is used.

- The method seems appropriate for the task of automated inspection in composite lay-up.
- In general, segmentation improves with mask size. Boundary accuracy on the other hand seems to be a non-linear function involving both mask size and texture primitive size.
- Region boundaries exhibiting extreme curvature result in segmentation inaccuracies.
- Gaussian smoothing provides more accurate boundaries than low-pass smoothing.
- In applications where region boundaries are smooth with modest curvature, secondary smoothing can increase accuracy.



CHAPTER 4

Convolution Mask Optimisation.

4.1 Introduction.

This chapter will present a method of optimising, or training, convolution masks for the task of texture discrimination. The flexibility this introduces enables the single channel model to be applicable in a much wider range of texture analysis problems than would otherwise be the case. Mask training means that a convolution-based texture analysis inspection system can be easily modified to cope with a change in the material or product to be inspected. For an industrial inspection system, such flexibility is essential.

4.2 The Monte-Carlo Approach.

As already mentioned in **Section 2.4.3**, an algorithm to optimise convolution masks for texture analysis has already been published **[Benke,Skinner,1987]**. The algorithm takes the form of a **heuristic random walk**, and uses a **Monte Carlo** approach for mask optimisation. Such an algorithm makes random choices of location in a parameter domain believed to contain the maximum, and the location giving the highest function value is taken as an approximation to the optimum. If such a process is carried out

iteratively, then successively better approximations can be obtained. For complex problems where the function to be considered is multivariate, non-linear, and/or multi-modal, then analytic solutions are frequently not applicable and the Monte Carlo approach is often the only solution available. The task of optimising a convolution mask for texture discrimination presents just such a problem. In this case the function is a combination of texture energies, with the variables being the mask elements. The function may well be non-linear and/or multi-modal.

In the algorithm described in **[Benke,Skinner, 1987]** four major constraints are placed on the search domain.

- Firstly, the convolution masks are required to have integer elements. This constraint is purely for computational convenience, since the vast majority of image processing hardware requires kernel coefficients to be integer rather than floating point.
- The mask elements must sum to zero. This constraint ensures that the masks' response is zero over a non-textured (uniform intensity) area of image. It also means that texture analysis is performed only on the basis of second order statistics and higher.
- The masks are normalised so that the maximum magnitude of any of their elements is equal to some arbitrary integer, **X**. Such normalisation is necessary since for any given mask, an infinite number of equivalent masks can be constructed by scaling the values uniformly. Usually **X** is chosen as 100, which gives more than acceptable dynamic range.
- In order to reduce the number of free variables, the masks are constrained to symmetry about the vertical axis.

The Monte Carlo algorithm of **[Benke,Skinner,1987]**, applied to the task of discriminating between two areas of texture, is represented in **Figure**

```
begin
    /* initialise maximum texture energy to zero
    E_max := 0

    /* get initial best mask
    loop for num.tries
        generate.random.mask(rmask)                    /* generate random mask
        filter.image(image1,image2,rmask)              /* filter image with random mask
        measure.texture.energy(image2,A,B,E_A,E_B)     /* measure texture energy in A and B
        if MAX(E_A,E_B) > E_max                        /* select mask producing highest energy
            E_max := MAX(E_A,E_B)
            best.mask := rmask
        endif
    endloop

    /* measure texture ratio for initial best mask
    filter.image(image1,image2,best.mask)              /* filter image with best mask
    measure.texture.ratio(image2,A,B,R_max)            /* measure ratio of texture energies
                                                       /*   between regions A and B

    /* main loop
    loop i = 0 for num.iterations                      /* loop for required # of iterations
        generate.random.mask(rmask)                    /* generate random mask
        filter.image(image1,image2,rmask)              /* filter image with random mask
        measure.texture.ratio(image2,A,B,R_r)          /* measure ratio of texture energies
                                                       /*   between regions A and B
        learned.mask := weighted.average(best.mask,R_max,rmask,R_r)  /* produce learned mask as
                                                       /*   weighted average of the best
                                                       /*   mask and the random mask
        filter.image(image1,image2,learned.mask)       /* filter with learned mask
        measure.texture.ratio(image2,A,B,R_l)          /* measure ratio of texture energies
                                                       /*   between regions A and B
        if (R_r > R_max) AND (R_r > R_l)
            R_max := R_r                               /* select mask producing highest
            best.mask := rmask                         /*   ratio of texture energies
        elseif (R_l > R_max) AND (R_l > R_r)
            R_max := R_l
            best.mask := learned.mask
        endif
    endloop
end
```

Figure (4.2.1). Pseudo-code for the Monte Carlo algorithm of [Benke,Skinner,1987]. The function MAX(x1,x2) returns whichever is the greater of x1 and x2.

(4.2.1). This algorithm can be implemented in a similar framework to that described in **Chapter Three** for the training of binary masks. Two training areas **A** and **B** are user selected, and the algorithm attempts to produce a mask which will maximally discriminate them. Note that two image stores are required, one to hold the training image and one to receive the filtered image.

The first step in the algorithm is to produce an initial mask, chosen to maximise the texture energy in one of the training regions. This mask is applied to the training image, and the discrimination achieved between the two

training regions is measured. This mask is now regarded as the **current best mask**. The algorithm then attempts to improve upon this mask in an iterative loop. At each iteration, a new "guess" is made at the optimum mask. In practice this means the generation of a **random** mask constrained to zero-sum, scale, and vertical symmetry. The discrimination achieved by this new mask is measured. A third mask is now produced called the **learned mask**. This mask is a weighted average of the current best mask and the random mask. This mask is also applied to the training image, and the discrimination achieved measured. The final step in the loop is to compare the discrimination achieved by each mask, and to choose the mask giving the best discrimination as the current best mask for the next iteration. This process is repeated for a pre-determined number of iterations, and so the discrimination of the current best mask is iteratively improved. **[Skinner et al,1990]** report that an optimum mask is usually found within about 1000 iterations. A point to note is that the algorithm can be parametrised to produce filters which maximise output energy for a particular training area whilst minimising output energy for the other, since this may sometimes be required. Alternately, the algorithm can be left to allow the characteristics of the textures in the training regions to determine automatically which will have higher texture energy and which lower. The latter case will result in optimal discrimination, while the former case may not. For the purposes of this chapter, the algorithm will not be used to force a particular region to high texture energy, but rather will be left to produce optimum discrimination. This algorithm is now examined in more detail.

The initial mask is produced by generating a number of random masks, and choosing the one which maximises the texture energy in either training region **A** or **B**. Each random mask must satisfy the constraints with regard to scale, symmetry and zero-sum. In order to simplify the equations relating to random mask generation, the symmetry constraint is relaxed. This is for clarity only.

To generate a **kxk** random matrix **r** with elements which sum to zero would be a time consuming trial and error process. It is optimised by producing only $\mathbf{k^2-1}$ random elements, and choosing the last element to be

equal to the negative sum of the other mask elements. This enforces the zero-sum constraint. The last element in the mask, **r(k-1,k-1)**, will therefore have a different probability distribution from the other elements, but this is of negligible importance. The elements for a **kxk** random mask **r** are therefore derived as follows

$$r(i,j) = \begin{cases} 0 & if\ \ i=j=(k-1) \\ RND(1) & OTHERWISE \end{cases} \qquad i,j=0..k-1 \qquad (4.1)$$

$$r(k-1,k-1) = -\sum_{i=0}^{i=k-1}\sum_{j=0}^{j=k-1} r(i,j)$$

where **RND(1)** is a function which produces a uniform distribution of pseudo-random real numbers between -1 and 1. The mask normalisation required to scale **r** so that the maximum magnitude of any one element is **X**, can be defined as

$$r(i,j) = ROUND\left(\frac{r(i,j)*X}{MAX(r(i,j))}\right) \qquad i,j=0..k-1 \qquad (4.2)$$

where **MAX(r(i,j))** represents the maximum magnitude of any element of **r**, and **ROUND(n)** returns the nearest integer to real number **n**. Equations **(4.1)** and **(4.2)** therefore define the generation of the random mask **r** with integer elements, suitably constrained to scale and zero-sum.

For an image function **I** with spatial domain **D**, convolution with **r** to produce a filtered image **F** is defined as

$$F(i,j) = \sum_{p=0}^{p=k-1}\sum_{q=0}^{q=k-1} r(p,q)*I(i'+p,j'+q) \qquad \forall i,j\in D \qquad (4.3)$$

$$where\ i'=i-k/2, j'=j-k/2$$

Note that as a standard function of many image processing cards, including the one used in this work, the values **F(i,j)** would be automatically rectified, so that

$$F(i,j) = |F(i,j)| \qquad \forall i,j\in D \qquad (4.4)$$

Fortunately this is in keeping with the idea of texture energy as proposed by

[Laws,1980]. It is assumed throughout this thesis that all filtered images have been rectified as in **(4.4)**.

For a filtered image **F**, the corresponding texture energy image **F'** is given by calculating for each pixel **(i,j)** the mean of the rectified values in a window centred on **(i,j)**. This can be considered a moving average smoothing operation on **F**, so that

$$F'(i,j) = \frac{1}{W^2} \sum_{p=-\frac{W}{2}}^{p=\frac{W}{2}} \sum_{q=-\frac{W}{2}}^{q=\frac{W}{2}} F(i+p, j+q) \qquad \forall\, i,j \in D \qquad (4.5)$$

where **W** is the window size of the smoothing operator. **W** is set by Laws and most other workers to 15.

The texture energy of any area is obtained by summing **F'** over that area, so that for the training region **A** with spatial domain $\mathbf{D_A}$, the total texture energy **E** is defined as

$$E = \sum\sum F'(i,j) \qquad \forall i,j \in D_A \qquad (4.6)$$

An equally effective approximation to the total texture energy in a region can be obtained if the smoothing stage indicated by equation **(4.5)** is omitted. This approximation is only valid if the region in question is significantly larger than **W**, the window size of the smoothing filter. With **W**=15, and the training areas with dimensions of at least 64x64, the criterion is satisfied for this application. The total texture energy in region **A** can therefore be taken directly from the filtered image **F**, so that equation **(4.6)** becomes

$$E = \sum\sum F(i,j) \qquad \forall i,j \in D_A \qquad (4.7)$$

For a random mask **r** applied to the training image, the texture energy in each training region is measured, and the largest chosen thus

$$E_r = \begin{cases} E_A & if \quad E_A > E_B \\ E_B & if \quad E_B > E_A \end{cases} \qquad (4.8)$$

where $\mathbf{E_A}$ and $\mathbf{E_B}$ are the texture energies in region **A** and region **B** respectively, and $\mathbf{E_r}$ is the texture energy associated with mask **r**. To choose the initial best mask **b**, **N** random masks $\mathbf{r_0}$ to $\mathbf{r_{N-1}}$ are generated, and the one with the largest associated texture energy selected, so that

$$b = r_i \quad which\ maximises \quad E_i \qquad i=0..N-1 \qquad (4.9)$$

where $\mathbf{E_i}$ is determined for each mask $\mathbf{r_i}$ using equation **(4.8)**. **N** is usually chosen, rather arbitrarily, as 10. This gives reasonable performance.

At this stage it is useful to summarise the process of choosing an initial best mask. A number of random masks are generated as defined in equations **(4.1)** and **(4.2)**. The random mask giving the maximum texture energy in either training region, measured by equations **(4.7)** and **(4.8)**, is chosen as the best initial mask according to equation **(4.9)**. This mask becomes the current best mask **b**.

Referring back to **Figure (4.1)**, the next step is to measure the texture discrimination achieved by this mask. The discrimination achieved by a mask is measured as a ratio of texture energies. Assuming as before that the textures themselves are to be allowed to determine which region gravitates towards higher texture energy, then the texture ratio for mask **b** is denoted as $\mathbf{R_b}$, and defined as

$$R_b = \begin{cases} \dfrac{E_A}{E_B} & if \quad E_A \geq E_B \\ \dfrac{E_B}{E_A} & if \quad E_A < E_B \end{cases} \qquad (4.10)$$

where $\mathbf{E_A}$ and $\mathbf{E_B}$ are the texture energies of regions **A** and **B** respectively. Note that to force one or other area to high texture energy, simply use that regions' texture energy as the numerator in the texture ratio equation of **(4.10)**. The region chosen as numerator is iteratively directed towards higher texture energy, whilst the region chosen as denominator is iteratively directed towards lower texture energy.

Now consider the main loop of the algorithm. A new random mask **r** is

generated using equations **(4.1)** and **(4.2)**. The mask is applied to the training image in the way defined in equation **(4.3)** and the texture energy of each region measured as in equation **(4.7)**. The texture discrimination between the two training regions for this mask is denoted as $\mathbf{R_r}$, and this is measured by application of equation **(4.10)**.

The next stage is the generation of the learned mask **l**. This is calculated as a weighted average of the current best mask **b** and the random mask **r**. The weighting attached to each mask is taken as the texture ratio achieved by that mask. In this way the learned mask is weighted more strongly in favour of the mask with better discriminatory power, but will also be significantly affected by the weaker mask. This will produce a mask which is different from either of the constituent masks, and which may achieve greater discrimination. The learned mask **l** is therefore calculated as

$$l(p,q) = R_b b(p,q) + R_r r(p,q) \qquad p,q=0..k-1 \qquad (4.11)$$

where **k** is the mask size. **The power of this heuristic random walk algorithm lies in this concept of a learned mask.** If this mask were not generated, the number of iterations taken to improve discrimination using only random guesses would be prohibitive. By combining two masks in proportion to their power of discrimination, it is hoped to produce a mask which reflects the best features of both constituent masks. If applied in an iterative algorithm then this is a surprisingly effective approach. Since the learned mask will be significantly different from the current best mask, the problem of finding only local minima is also avoided.

The learned mask **l** must also be normalised as in equation **(4.2)**. It is then applied to the training image as in equation **(4.3)**, and the corresponding texture ratio calculated as in equation **(4.7)**.

From the three masks now defined, the current best mask, the random mask, and the learned mask, the one giving the highest texture ratio is chosen as the current best mask for the next iteration.

Formally this is represented as

$$b = \begin{cases} b & R_b > R_r \text{ AND } R_b > R_l \\ r & R_r > R_b \text{ AND } R_r > R_l \\ l & R_l > R_b \text{ AND } R_l > R_r \end{cases} \qquad (4.12)$$

where $\mathbf{R_b}$, $\mathbf{R_l}$, and $\mathbf{R_r}$ are the texture energies associated with masks **b**, **l**, and **r** respectively. In this way the discriminatory power of the current best mask is iteratively increased. The process is repeated for a set number of iterations.

4.3 Performance of the Benke and Skinner Algorithm.

The algorithm detailed in the previous section has been implemented on the pipeline card, and so is able to run at about 10 iterations per second for 3x3 masks, 8 iterations per second for 5x5 masks, and 5 iterations per second for 7x7 masks **[King,1994]**. The algorithm has been extensively tested on many textures, with generally good results. It is neither necessary nor desirable to reproduce these results here, but instead the trends observed and the improvements implemented will be detailed. A more complete demonstration of the power of this method is given in **Chapter Five**, when an improved version of the Benke and Skinner algorithm is compared against a new algorithm. For the purposes of this and the following section, results can be effectively explained with reference to a single suitably chosen dataset of textures, namely samples of carbon fibre at four different weave orientations. Similarly only the results for the optimisation of 5x5 masks are detailed, since the extension to other mask sizes provides no additional information. It is useful in this section to illustrate the performance of the algorithm using graphs, the first of which can be found in **Figure (4.3.1)**.

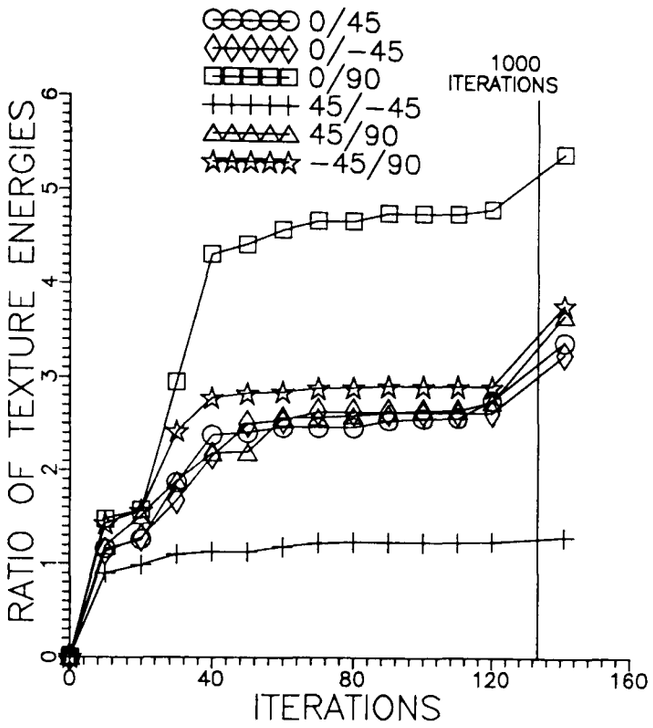


Figure (4.3.1). Graph showing iterative improvement of masks trained using the Benke and Skinner algorithm. The test data is different orientations of carbon fibre material. The last point is the texture ratio achieved after 1000 iterations.

The orientations which each mask has been trained to discriminate are indicated by the legend on the graph. Since the Monte Carlo method uses random masks then each run will progress differently, and so the average, or **expected**, performance is depicted. Each set of data on the graph therefore represents the performance of the algorithm averaged over 10 runs. The final point plotted represents the texture ratio achieved after 1000 iterations. This is perhaps not as clear as it might be on the graph (i.e. the appearance of the "160" on x-axis is misleading), but this is due to limitations of the software package used to produce the graphs. The graph is presented in this format (rather than over the full 1000 iterations) to allow the optimisation rate in the early stages to be easily observed.

This graph effectively illustrates a weakness in the Benke and Skinner algorithm. Four of the masks generated follow a similar path, and achieve a texture ratio of between 3 and 4 after 1000 iterations. One mask performs much better with a texture ratio greater than 5. The remaining mask cannot achieve any significant ratio at all.

These discrepancies are a direct result of the symmetry constraint imposed, which requires that the masks generated are symmetric about the vertical axis. The result of this is that the performance of the masks depends on the symmetry exhibited by the textures in question. Where vertical symmetry is reflected in *both* training textures, then a near-optimum mask can be produced. This is the case with the fibre orientations of **0°** and **90°** which results in a ratio of greater than 5. Where *one* of the textures does not exhibit this type of symmetry, then a reasonable but far from optimal mask can still be produced. This is the case with any texture pair which includes one orientation of either **45°** or **-45°**. If *neither* texture exhibits vertical symmetry, as is the case when trying to discriminate **45°** from **-45°**, then the mask is not able to produce any kind of discrimination at all.

4.4 Improving The Benke and Skinner Algorithm For Automated Inspection.

This section will present two modifications to the original algorithm which have been shown to improve performance. They are detailed in the following two sub-sections.

4.4.1 Removal of Symmetry Constraint.

Obviously the symmetry constraint is detrimental to the objective of finding optimum masks for texture discrimination. Benke and Skinner proposed the symmetry constraint to reduce the number of free variables in the mask, and therefore speed up the convergence to an optimal mask. They justified this by claiming that natural textures do not in general exhibit an "innate left-right asymmetry". This assumption is obviously invalid for industrial inspection. The constraint that masks exhibit symmetry must therefore be relaxed.

Figure (4.4.1.1) shows a graph generated using masks with no symmetry constraint, using the same texture dataset used to produce the graph of **Figure (4.3.1)**. The results illustrated in **Figure (4.4.1.1)** show an overall improvement in discrimination over the results of **Figure (4.3.1)**. Removal of the symmetry constraint has enabled the masks to produce

greater feature-plane separation, with only a relatively slight decrease in the rate of convergence. Specifically, the carbon samples at **45°** and **-45°** can be readily discriminated by antisymmetric masks. This graph also shows that some texture pairs are easier to discriminate than others. The question of how to quantify the discriminability of textures is one which awaits a better understanding of texture.

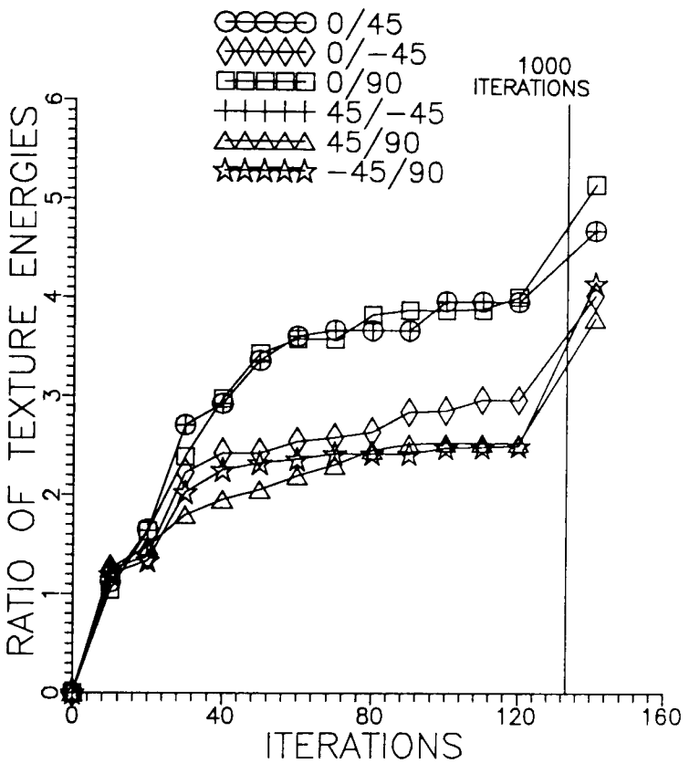


Figure (4.4.1.1). Graph showing iterative improvement of masks trained using the Benke and Skinner algorithm with no symmetry constraint. The test data is different orientations of carbon fibre material. The last point is the texture ratio achieved after 1000 iterations.

4.4.2 A New Optimisation Criterion.

Experiments with the Benke and Skinner algorithm have suggested that the ratio of texture energies is not the most appropriate measure of texture discrimination, when the means of discrimination itself is to be a simple threshold. For reliable thresholding it is required to separate the texture classes in the feature-plane as much as possible. Maximising the texture energy ratio introduces a subtly different criterion to the one required. To maximise a ratio, the most effective way is to minimise the denominator. This corresponds to minimising the texture energy of one of the training regions,

whilst keeping the texture energy of the other comparatively high. The masks which are chosen will therefore be the ones which minimise the texture energy in one or other of the regions. This will generally provide texture discrimination, but not *optimal* texture discrimination, since the measure of texture ratio becomes highly non-linear for low texture energies. To illustrate this, assume that the texture energy of one training region, region **A**, is static at a value equivalent to an average pixel grey-level of 200. For the second region, region **B**, reducing the average grey-level increases the texture ratio in a non-linear manner. This can be appreciated by reference to **Figure (4.4.2.1)**, which shows how the ratio of texture energies varies non-linearly when the average grey-level of the second training area decreases.

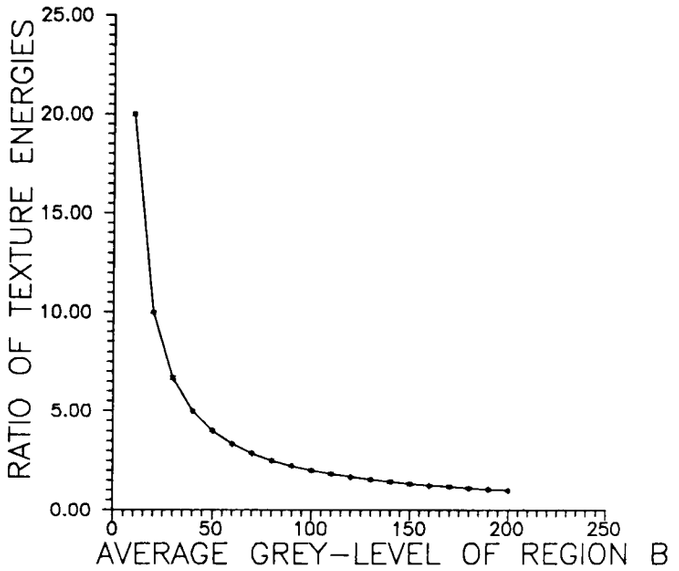


Figure (4.4.2.1). Non-linear increase of texture ratio when the texture energy in region **B** decreases. The value in region **A** is assumed to be constant, and equal to 200.

From this graph it is obvious that a greater increase in texture energy ratio can be achieved by minimising the grey-level of region **B** than by increasing the grey-level of region **A**. As a result of this, the algorithm will tend to produce masks which minimises the grey-level of one of the regions. This will produce a high texture ratio, *but does not guarantee maximum discrimination.* For example, consider a mask which when applied to two training regions **A** and **B** results in average grey-levels of 200 and 50 respectively. The corresponding texture energy ratio would be 4, whilst the separation of the peaks on the grey-level histogram, which is a more important feature for thresholding, would

be 150. A second mask might result in an average grey-level of 150 and 30 respectively. The corresponding texture energy ratio for this mask would be 5, but the distance between the two peaks on the grey-level histogram would be only 120. The algorithm as defined would therefore choose the mask which is producing less feature-plane separation.

A more appropriate criterion for mask optimisation would therefore be one which measured the **histogram separation** of texture classes in the feature-plane. This can be approximated by the difference in mean grey-level of the respective texture regions after convolution with the relevant mask. The mean grey-level for region **A** with spatial domain $\mathbf{D_A}$, is denoted $\mathbf{G_A}$ and defined as

$$G_A = \frac{1}{N}\sum\sum F(i,j) \qquad \forall i,j \in D_A \qquad (4.13)$$

where **N** is the number of pixels in $\mathbf{D_A}$. The histogram separation **S** for two regions **A** and **B** is therefore

$$S = G_A - G_B \qquad (4.14)$$

This measure is obviously linear over its full range, which the texture energy ratio measure defined in equation **(4.10)** is not. It also has the capacity to be either positive or negative, and so equation **(4.12)** which defines the choice of best mask for the next iteration must be adapted thus

$$b = \begin{cases} b & |S_b| > |S_r| \ AND \ |S_b| > |S_l| \\ r & |S_r| > |S_b| \ AND \ |S_r| > |S_l| \\ l & |S_l| > |S_b| \ AND \ |S_l| > |S_r| \end{cases} \qquad (4.15)$$

where $\mathbf{S_b}$, $\mathbf{S_l}$, and $\mathbf{S_r}$ are the histogram separations associated with masks **b**, **l**, and **r** respectively. The learned mask is still calculated in a similar manner to that defined in equation **(4.11)**, but the histogram separation term replaces the texture energy ratio term. The resulting equation is

$$l(p,q) = S_b b(p,q) + S_r r(p,q) \qquad p,q=0..k-1 \qquad (4.16)$$

The Benke and Skinner algorithm using histogram separation as the maximisation criterion performs at least as well, and in most cases noticeably better, than the version using ratio of texture energies. The graph of **Figure (4.4.2.2)** shows the results when applied to the test dataset of textures.

The histogram separation measure has several attractive features. As already stated, it is a linear measure of the property to be maximised, namely feature-plane separation. It is readily extendable to the task of discriminating between more than two texture classes, as shall be seen in **Section 4.4.3**. It also results in faster convergence in the sense that a near optimal mask is produced in relatively few iterations. This is illustrated by reference to **Figure (4.4.1.1)** and **Figure (4.4.2.2)**.

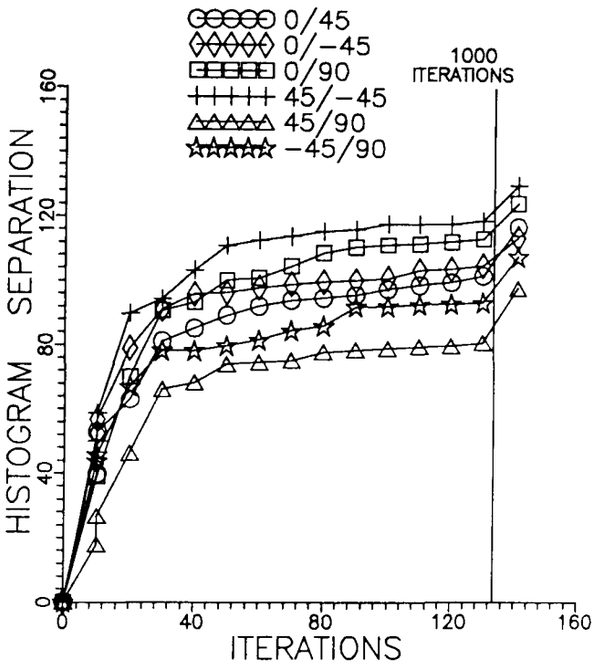


Figure (4.4.2.2). Graph showing iterative improvement using Monte Carlo algorithm with histogram separation. The test data and graph format are the same as in previous graphs.

In the former case, using ratio of texture energies, the ratio is increasing at a relatively slow, but consistent, rate. The performance after 1000 iterations is significantly greater than after 120 iterations. In the latter case, using histogram separation, there is a much sharper initial improvement in mask performance (i.e. within the first 50 iterations). There a proportionally smaller improvement in the result at 1000 iterations over the result at 120 iterations than when using the ratio of texture energies.

It is difficult to produce a more rigorous argument to support the assertion that using histogram separation results in faster convergence than using ratio of texture energies, since the two different measures of performance cannot be directly compared. However, experience with the algorithm does indicate that the new measure does produce faster convergence. A possible reason for this appears to be the increased dynamic range offered by the histogram separation measure. This can vary from a value greater than 100, to a value less than -100. When this measure is used in equation **(4.16)** to calculate the learned mask, then a mask giving good separation may be weighted by perhaps 10 times as much as a mask giving weak separation. This means much stronger proportional weighting for good masks than takes place using the ratio of texture energies. The result is faster optimisation. An additional factor is that histogram separation may be negative, and so this introduces another degree of freedom into the calculation of the learned mask.

4.4.3 Extension to More than two Texture Classes.

Adopting the histogram separation measure as the criterion for mask optimisation means that the algorithm of Benke and Skinner can be adapted to deal with more than two texture classes. For the sake of clarity only the extension to three classes is detailed, since the extension to more is analogous.

When attempting to discriminate only two texture classes, it was noted that the algorithm itself could be allowed to determine which texture energy to maximise and which to minimise. With three textures the situation is slightly different. What is required in this case is to discriminate a particular texture from the other two. The algorithm must therefore be 'told' which texture is to be discriminated. This is most easily implemented by assuming without loss of generality that the first texture region, region **A** contains the texture to be discriminated. Using this assumption then the only change required to extend the algorithm to deal with three textures is that equation **(4.14)** now becomes

$$S = \begin{cases} G_A - G_B & \text{if } |G_A - G_B| < |G_A - G_C| \\ G_A - G_C & \text{if } |G_A - G_C| < |G_A - G_B| \end{cases} \qquad (4.17)$$

where $\mathbf{G_A}$, $\mathbf{G_B}$, and $\mathbf{G_C}$ are the mean grey levels of regions **A**, **B**, and **C** respectively. Equation **(4.17)** defines the histogram separation as being the *minimum* separation between the region to be discriminated **A**, and the other two regions. By choosing this as the criterion to maximise, the inter-class errors between textures is minimised, and so ensuring optimum segmentation.

Equation **(4.17)** has been implemented in the Benke and Skinner algorithm, and the results for different texture classes observed. In general the performance is good, and for regular textures good discrimination can be achieved with three or four texture classes. This has already been illustrated by **Figure (3.7.2)** in the previous chapter. For more irregular textures the performance degrades, but is generally still acceptable. In the absence of a much more elaborate texture model, the only way to check if one texture class can be easily discriminated from other classes is to experiment.

The performance of the algorithm in optimising masks for discrimination between three textures is illustrated in **Figure (4.4.3.1)**.

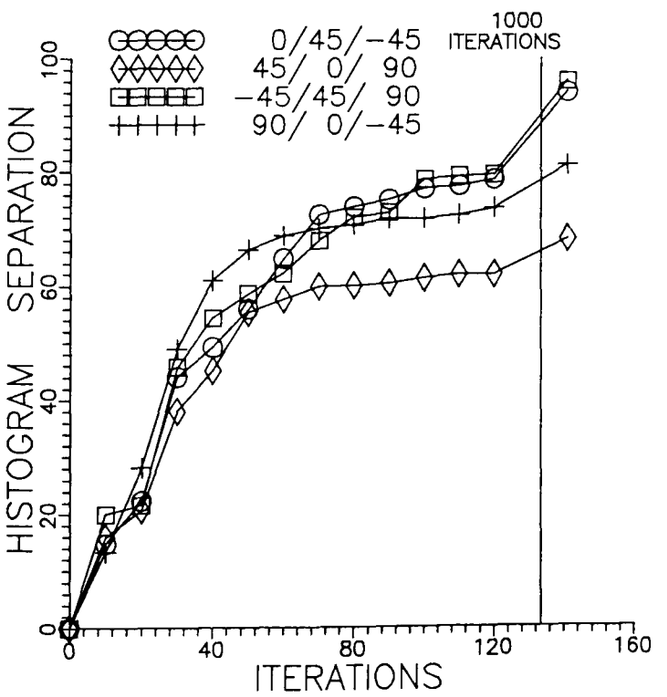


Figure (4.4.3.1). Graph showing performance of Benke and Skinner algorithm using histogram separation as the criterion to discriminate three texture classes. The format is the same as in previous graphs.

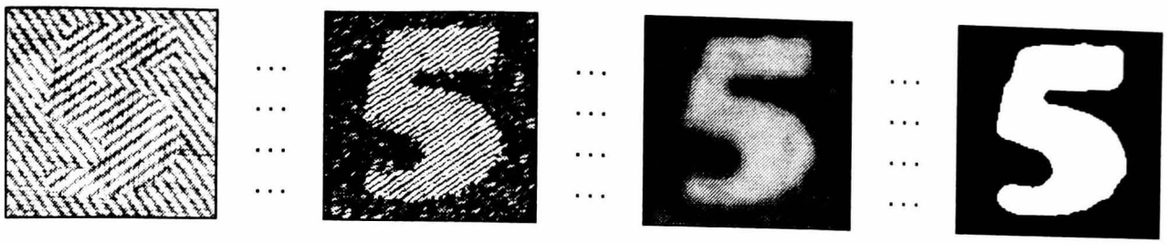
The dataset is again different orientations of carbon fibre, and the legend indicates the various orientations used for each test. The first orientation listed is discriminated from the two which follow. From this graph it can be seen that the optimisation of the masks proceeds at a slower rate than when discriminating between only two texture classes of the same dataset. From this it might be deduced that the task of discriminating two regular textures presents a multi-modal function with many peaks, and so optimisation to at least a local minimum is fast. With three or more textures, or with more irregular textures, the function seems to exhibit far fewer, or perhaps narrower peaks. The algorithm requires more iterations (guesses) to optimise a mask for such data.

4.5 Conclusions.

The algorithm presented by **[Benke,Skinner,1987]** has been implemented and extensively tested. By dropping the symmetry constraint the classes of texture which can be optimally discriminated is increased. A new optimisation criterion has been introduced, and has been shown empirically to produce faster and better discrimination. This criterion is also directly extendable to the discrimination of more than two texture classes, and the revised algorithm has been found to successfully optimise masks for such tasks.

The algorithm does however still exhibit some restrictive properties. Firstly, the number of iterations required for mask optimisation is sometimes prohibitive. This is not the case using the carbon fibre dataset used to illustrate results in the previous two sections, but can be true when using materials exhibiting more complex texture primitives. If a system has to be trained to inspect a large number of such textures then the training time would be considerable. Secondly, because the algorithm uses a Monte Carlo approach then the number of iterations required for optimisation is unpredictable. A fixed number of iterations for optimisation is preferable.

As a result of these drawbacks a new algorithm has been developed to optimise convolution masks. This is presented in the following Chapter.



CHAPTER 5

A New Algorithm for Convolution Mask Optimisation.

5.1 Introduction.

This chapter will detail a new algorithm for convolution mask optimisation. It will be shown that the algorithm is more effective and more efficient than the Benke and Skinner algorithm described in **Chapter Four**.

5.2 The Basis Algorithm.

The task of producing a convolution mask optimised for a specific texture analysis criteria might be considered as a **combinatorial optimisation** problem. Solving such a problem amounts to finding the "best" or "optimal" solution among a finite or countably infinite number of alternative solutions **[Aarts, Korst, 1989]**. Two approaches to such a problem can be identified. **General algorithms** are applicable to a wide range of problems, and so can to a certain extent be called problem independent. The heuristic random walk algorithm described in the previous chapter is, with suitable modifications, applicable to a wide range of problems and so is an example of a general algorithm. **Tailored algorithms** use problem-specific information and their

application is therefore limited to a restrictive set of problems. In this chapter a new tailored algorithm which uses information specific to the texture analysis task is presented, and shown to provide better performance for this application than the heuristic random walk algorithm of Benke and Skinner.

The strength of the Benke and Skinner algorithm lies in the concept of constructing a learned mask. This mask combines the current best mask, which has good discrimination, with a newly guessed mask, and as a result may produce better discrimination. The weakness of this approach is that the guesses are entirely random. With no criteria for the guessed masks, the probability of making a good guess is comparatively low, and so long sequences of poor guesses are likely. This means that the number of iterations required for optimisation is unknown, unpredictable, and possibly prohibitive.

To consider the problem in a slightly different way, it seems reasonable to hypothesize that the masks required in the single channel model are sensitive to some feature or combination of features inherent in the training textures. If, instead of combinations of random masks, linear combinations of masks which have been chosen for their sensitivity to different texture features are produced, then fast generation of a mask capable of near optimum texture discrimination might result. An algorithm working on this theory would therefore be a tailored algorithm, using knowledge of the underlying problem to provide faster optimisation.

An algorithm based on this idea has been developed, and is presented in **Figure (5.2.1)**. The algorithm produces a mask which is a weighted average of a set of previously determined masks. These masks are referred to as **basis** masks, and the new algorithm has been termed the **basis** algorithm. The nomenclature in **Figure (5.2.1)** has been selected to be as meaningful as possible, but for the sake of brevity in the following equations simpler terminology is adopted. The basis algorithm is explained as follows.

Let the number of **kxk** basis masks be **n**, and each of the **n** basis masks be referred to as $\mathbf{m}_i$, where **i=0..n-1**. In the loop to find the best initial mask, each of the basis masks $\mathbf{m}_i$ is applied to the training image **I** with spatial

domain **D** to produce a filtered image **f** thus

$$f(x,y) = \left| \sum_{p=0}^{p=k-1} \sum_{q=0}^{q=k-1} m(p,q) \cdot I(x'+p, y'+q) \right| \quad \forall x,y \in D \qquad (5.1)$$

$$where\ x' = x - \frac{k}{2},\ y' = y - \frac{k}{2}$$

The mean grey-level for region **A** with spatial domain $\mathbf{D_A}$, is denoted $\mathbf{G_A}$ and defined as

$$G_A = \frac{1}{N} \sum \sum f(x,y) \qquad \forall x,y \in D_A \qquad (5.2)$$

where **N** is the number of pixels in $\mathbf{D_A}$. The histogram separation between the training regions **A** and **B** achieved by each basis mask is denoted as $\mathbf{S_i}$, and is measured thus

$$S_i = G_A - G_B \qquad (5.3)$$

The current best mask **b** is chosen as the basis mask giving best separation, such that

$$b = m_i\ ,\ S_b = S_i\ \ where\ |S_i| \geq |S_j| \qquad j=0..n-1 \qquad (5.4)$$

In the main loop, a learned mask **1** is generated as a weighted average of the current best mask **b** and a basis mask $\mathbf{m_i}$. The weighting attached to the current best mask is the histogram separation achieved by that mask, $\mathbf{S_b}$. The weighting attached to the basis mask has two factors, $\mathbf{S_b}$ and $\mathbf{A_j}$. The first, $\mathbf{S_b}$, is the histogram separation achieved by the basis mask. The idea of a second weighting factor, $\mathbf{A_j}$, has been derived from a popular optimisation technique called **simulated annealing**. Annealing is the physical process of heating up a solid until it melts, followed by cooling it down until it crystallizes into a state with a perfect lattice structure (and therefore minimum free energy) **[Aarts, Korst, 1989]**. The ground state of the solid (i.e. the lattice) is obtained only if the maximum temperature is sufficiently high and the cooling is done sufficiently slowly.

```
begin
    /* initialise maximum histogram separation to zero
    S_max := 0

    /* get initial best mask
    loop i = 0 for num.basis.masks
        filter.image(image1,image2,basis.mask(i))          /* filter with basis mask
        measure.histogram.separation(image2,A,B,S(i))      /* measure separation achieved
        if abs(S(i)) > abs(S_max)                          /*    between regions A and B
            S_max := S(i)
            best.mask := basis.mask(i)                     /* select mask with best
        endif                                              /*      discrimination
    endloop

    /* main loop
    loop j = 0 for num.amplitude.scalars                   /* iteratively vary amplitude
        loop i = 0 for num.basis.masks                     /* for each basis mask
                                                           /* calculate learned mask
                                                           /*  A(j) is amplitude scalar
                                                           /*  (see text)
            learned.mask := weighted.average(best.mask,S_max,basis.mask(i),S(i),A(j))

            filter.image(image1,image2,learned.mask)       /* filter with learned mask
            measure.histogram.separation(image2,A,B,S_i)   /* measure separation achieved
            if abs(S_i) > abs(S_max)
                S_max := S_i                               /* select mask with best
                best.mask := learned.mask                  /*      discrimination
            endif
        endloop
    endloop
end
```

Figure (5.2.1). Pseudo-code for the basis algorithm. The function abs(x) returns the absolute value of x.

The name simulated annealing has been given to a class of optimisation algorithms which model the annealing process in order to perform function minimisation. From an initial state, the function is *perturbed* i.e. a new value in the vicinity of the initial state is obtained. If this new value is less than the current (initial) value, then it becomes the current minimum. The process is repeated iteratively to determine the minimum of the function. As described so far the process is very similar to that used in the Monte Carlo based heuristic random walk algorithm detailed in the previous chapter. For simulated annealing however, there is some control over the size of the perturbation area. Initially the possible area of perturbation will span the entire range of the function. This is equivalent to the maximum temperature of the liquid in the annealing process, where molecules are free to interact as they wish. After so many iterations however, the possible area of perturbation will decrease

slightly, so the new function "guesses" are more limited in their range. This is analogous with reducing the temperature in annealing. As the temperature falls, the molecules will begin to form the low energy lattice structure. If the temperature falls sufficiently slowly, then the ground state of the solid will be reached, and meta-stable states (local minima in terms of the functional analogy) will be avoided. In simulated annealing therefore the possible area of perturbation is reduced in a controlled manner in an attempt to find the global minimum of the function.

It is this idea of controlling the area of perturbation which has led to the use of a second scaling factor in the algorithm described here. It might be regarded as producing a "fine tuning" effect to increase the overall discrimination. The scaling factor is denoted as $\mathbf{A}_j$, where **j=1..M**. The term $\mathbf{A}_j$ is varied iteratively downwards from 1 to some lower bound greater than 0. This facilitates the fine tuning which may be required to accurately approximate the optimum mask. The learned mask in loop **j,i** is therefore calculated as

$$l(p,q) = S_b\, b(p,q) + S_i\, A_j\, m_i(p,q) \qquad p,q=0..k-1, \qquad (5.5)$$

where **k** is the mask size. The value of **M** can range from 1, in which case the weighting of the basis masks is not scaled at all (since $\mathbf{A}_1 = 1$), to perhaps 5, in which case four additional scaling loops are performed. The former will provide a rough approximation to the optimum in very few iterations, the latter a better approximation but in many more iterations. The optimum value of **M** is determined empirically in **Section 5.4.1**. Note that the histogram separation achieved by each basis mask, $\mathbf{S}_i$, is already known, since this was recorded in the initial loop. As a result of this only one image convolution is required per loop for the basis algorithm compared to two for the Benke and Skinner algorithm.

The learned mask is normalised so that the maximum magnitude of any one element is **X**, thus

$$l(i,j) = ROUND\left(\frac{l(i,j)*X}{MAX(|l(i,j)|)}\right) \qquad i,j=0..k-1 \qquad (5.6)$$

where **MAX(r(i,j))** represents the maximum magnitude of any element of **I**, and **ROUND(r)** returns the nearest integer to real number **r**. S_I, the histogram separation for the learned mask is measured as defined in equation **(5.3)**. The best mask for the next iteration is chosen as

$$b = \begin{cases} b & if \ |S_b| > |S_1| \\ 1 & if \ |S_1| > |S_b| \end{cases} \qquad (5.7)$$

The total number of iterations required for the optimisation of a mask by the basis method is therefore equal to the number of iterations required to calculate the initial best mask (the number of basis masks) plus the number of iterations required in the main loop (the number of basis masks times the number of scalars).
Hence

$$\#\ iterations = n(M+1) \qquad (5.8)$$

5.3 Possible Basis Mask Sets.

The success of the basis algorithm will obviously depend on an appropriate basis mask set being chosen. The masks sets tested are

- an augmented set of Laws masks
- sampled Gabor filters
- eigenfilters

Each of these mask sets has been generated and tested for 3x3, 5x5, and 7x7 masks. All the masks have been normalised as in equation **(5.6)** so that the maximum magnitude of element is equal to 100. The details specific to the generation of each of the above masks sets are examined in the following three sections.

5.3.1 Augmented Laws Filter Set.

The Laws filter set is an obvious candidate for basis masks. **[Laws,1980]** defined 3x3, 5x5, and 7x7 masks for texture discrimination, and no-one has yet published a mask set claiming better performance. For this

application one mask from each set must be discounted, namely L3L3, L5L5, and L7L7, since these masks are not zero-sum. In addition, it is advantageous when dealing with synthetic materials (commonly encountered in machine vision tasks) to enlarge the set by adding another two masks for each spatial resolution. These masks are chosen to have a strong diagonal element to them, and enable more effective masks to be produced for textures exhibiting a similar structure (e.g. carbon fibre cloth at ±45°). Whilst these masks are redundant in the sense that the Laws mask sets are independent and complete (the 3x3 and 5x5 sets at least), they do offer an advantage in terms of performance with the basis algorithm. The additional masks have been constructed by rotating other Laws masks, E3L3, E5L5, and E7L7 . The 5x5 form of the masks added are shown in **Figure (5.3.1.1)**. Note that these masks have yet to be normalised, and are simply the Laws mask E5L5 rotated by -45° and 45° respectively.

0	-2	-1	-4	-6
2	0	-8	-12	-4
1	8	0	-8	-1
4	12	8	0	-2
6	4	1	2	0

6	4	1	2	0
4	12	8	0	-2
1	8	0	-8	-1
2	0	-8	-12	-4
0	-2	-1	-4	-6

Figure (5.3.1.1). Additional masks in the augmented Laws filter set. The masks shown are rotated versions of E5L5.

The number of masks in the augmented Laws set is therefore 10 for 3x3 masks, 26 for 5x5 masks, and 37 for 7x7 masks. For reasons of brevity these masks are referred to from now on simply as the Laws basis set.

5.3.2 Sampled Gabor Filters.

Gabor filters have enjoyed a high profile in the texture analysis literature recently **[Caelli,Moraglia,1985]**, **[Perry,Lowe,1989]**, **[Jain,Farrokhnia,1991]**. This is mainly due to psychophysical experiments which have suggested that the response profile of simple cells in the human visual cortex can best be approximated by Gabor signals **[Pollen,Ronner,1983]**, **[Daugman,1980]**. This would seem to be a good justification for their inclusion in these experiments,

but there is a problem here. The small size of the masks, with the largest being 7x7, means that the sampled versions cannot realistically be called Gabor filters. The approximations are just too coarse. There is however a second reason to use Gabor functions, and that is that the functions themselves are easily controllable in terms of frequency and orientation. They therefore provide a convenient means of generating a set of filters whose frequency and orientation can be specified. This is ideal for the requirement of finding good basis mask sets.

The response of an even-symmetric Gabor filter in the spatial domain is given by

$$h(x,y) = \exp\left\{-\frac{1}{2}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right]\right\}\cos(2\pi u_0 x) \qquad (5.9)$$

where $\mathbf{u_0}$ is the frequency of a sinusoid along the x-axis, and σ_x and σ_y are the space constants of the Gaussian envelope along the x and y axes respectively. To obtain different orientations a rigid rotation on the axes of the function is effected. That means the parameters of the function are transformed, so that

$$x' = x\cos(\theta) - y\sin(\theta) \qquad (5.10)$$
$$y' = x\sin(\theta) + y\cos(\theta)$$

where θ is the angle of rotation, so that equation **(5.9)** is parametrised by **x'** and **y'** rather than **x** and **y**. The function can be sampled as a 3x3, 5x5 or 7x7 mask. To ensure that each mask is zero sum, the mean is calculated and subtracted from every element before normalisation.

Table (5.1) shows the frequencies, orientations and Gaussian parameters which have been used to generate the functions sampled in these tests. The reason for the different values of σ_x and σ_y is that the extreme value (σ_x=σ_y=100) produces step "edges" in the sampled masks, compared to the smoother more ramp-like "edges" produced when σ_x and σ_y have values more appropriate to the mask size. It seems likely that this will have some effect on the power of discrimination of the masks generated. Therefore two sets of

Gabor masks will be tested for each mask size. The choice of frequencies and orientations are empirical, chosen after experimentation.

Name of Mask Set	σ_x	σ_y	Frequencies in cycles/mask			Orientations in degrees			# Masks
			Low	High	Step	Low	High	Step	
Gabor31	100	100	0.5	1.5	0.5	0	135	22.5	21
Gabor32	0.73	0.73	0.5	1.5	0.5	0	135	22.5	21
Gabor51	100	100	0.5	2.5	1.0	0	135	22.5	21
Gabor52	1.16	1.16	0.5	2.5	1.0	0	135	22.5	21
Gabor71	100	100	0.5	3.5	1.0	0	135	22.5	28
Gabor72	3.6	3.6	0.5	3.5	1.0	0	135	22.5	28

Figure (5.1). Parameters for each of the sampled Gabor masks.

5.3.3 Eigenfilters.

Eigenfilters were introduced by **[Ade,1983a]** and discussed in section **(2.4.5)**. From a training region of texture the eigenvectors of the corresponding covariance matrix are calculated, and these are used as the coefficients in convolution masks. These eigenfilters are derived without loss of information from the covariance matrix, and so fully describe the second order statistics of the sample texture. It is reasonable to consider then that they might be useful in texture discrimination.

A method to derive the eigenvectors from the texture covariance matrix has been implemented based on the **Householder Reduction [Press et al, 1990]**. Experiments have been performed with eigenfilters from either or both training regions. It is important to remember that since the eigenfilters have to be extracted from the texture samples to be discriminated, then there is the additional overhead incurred in calculating the eigenfilters to consider, an overhead which increases exponentially with mask size. For 3x3 or 5x5 masks this is not significant, but for 7x7 or larger neighbourhoods the processing time is of the order of tens of seconds.

5.4 Performance of the Basis Algorithm.

The main parameters which have to be investigated in the basis algorithm are the choice of basis mask set, and the optimum number and value of scalars. It is convenient to tackle the latter problem first, since this will simplify the task in determining the best basis mask set. This is a reasonable approach, since experiments have shown that the optimum number and value of scalars are largely independent of the basis mask set used. The Laws basis set is therefore used throughout the following section.

5.4.1 Scalars.

To determine the optimum number of scalars, the texture dataset depicted in **Figures (5.4.1.1a)** and **(5.4.1.1b)** is used, where the second dataset is the histogram equalised version of the first. The texture pairs used are, from top to bottom, two different orientations of carbon fibre with glass weft (the orientations are 0° and -45°), two different grains of leather, wood grain and packing foam, and herringbone and cotton. The last two textures are taken from **[Brodatz, 1966]**. These textures have been chosen to cover a range of texture regularities and texture element sizes. They are also all textures which might be encountered in automated visual inspection.

The results for different scalars are shown for each mask size on each texture in **Figure (5.4.1.2a)** for non-equalised textures, and in **Figure (5.4.1.2b)** for the equalised versions. Two sets of results are shown for each mask size. The first shows the optimisation rate with four additional scaling loops, the second with only one additional scaling loop. The values of the scalars are

$$A = [0.7, 0.5, 0.3, 0.1] \quad \textit{and} \quad A = [0.5] \qquad (5.11)$$

for four scaling loops and one scaling loop respectively. These values have been found to perform well empirically.

To interpret the data in **Figures (5.4.1.2a)** and **(5.4.1.2b)** it is necessary to refer back to the basis algorithm shown in **Figure (5.2.1)**. Remembering that there are 10 3x3 Laws basis masks, 26 5x5 masks, and 37 7x7 masks,

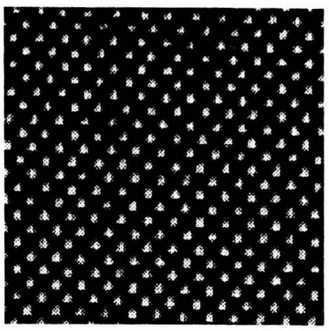
then the loop to choose the initial mask takes 10, 26, or 37 iterations depending on mask size. The main loop to optimise the best mask takes another 10, 26, or 37 iterations. Each subsequent scaling loop therefore takes another 10, 26, or 37 iterations.

From the graphs it is clear that the basis algorithm produces a mask with good discrimination after the first optimisation loop. The first scaling loop produces a further small improvement, and all further scaling loops produce only very slight improvements. The texture pair which benefits most from scaling is the carbon fibre pair.

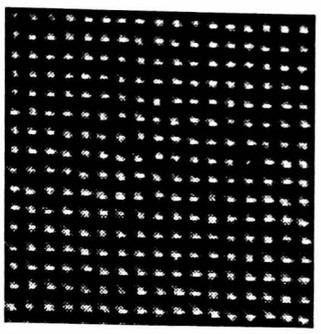
In general, therefore, scaling provides only minor improvements in discrimination, although for some textures this can be useful. It is concluded therefore that one scaling loop is adequate. The scaling value of 0.5 has been selected since this was shown empirically to provide the best performance.

5.4.2 Comparison of Different Basis Mask Sets.

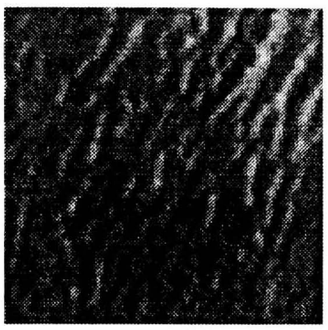
To compare the performance of the basis algorithm for different mask sets the texture dataset shown in **Figure (5.4.2.1)** is used. The discrimination will be measured between pairs of textures. Discrimination between three textures is treated in the next section in the comparison with the Monte Carlo algorithm. The texture pairs of **Figure (5.4.2.1)** are, from top to bottom, two different orientations of carbon fibre with glass fibre weft (the orientations are 90° and 45°), two different orientations of carbon fibre with thermoplastic weft (the orientations are 90° and -45°), two different grains of leather, histogram equalised images of leather and water from **[Brodatz,1966]**, and histogram equalised images of raffia and wood from **[Brodatz,1966]**.The results for this dataset using 3x3, 5x5, and 7x7 masks are shown in **Figures (5.4.2.2a)**, **(5.4.2.2b)**, and **(5.4.2.2c)** respectively. The mask sets tested are the Laws basis set described in **Section 5.3.1**, the two Gabor basis sets described in **Section 5.3.2**, and two eigenfilter sets which have been extracted from the texture samples. The set denoted as **EIGEN1** consists of the eigenfilters extracted from the first texture sample of each pair only, whilst the set denoted as **EIGEN2** consists of the eigenfilters extracted from both texture samples.



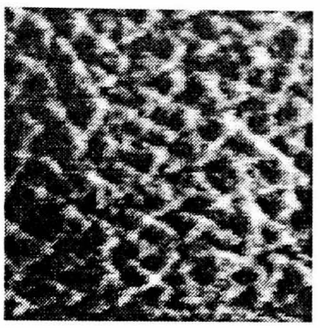
(a) Carbon fibre at 0°.



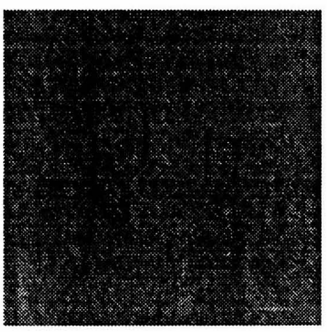
(b) Carbon fibre at -45°.



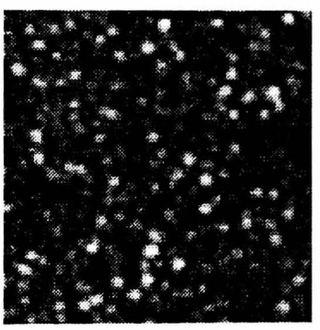
(c) Leather grain #1.



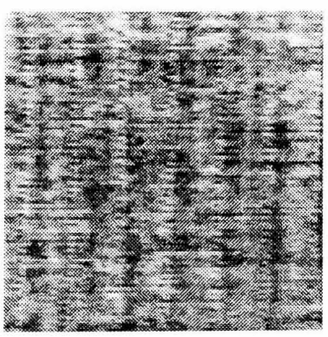
(d) Leather grain #2.



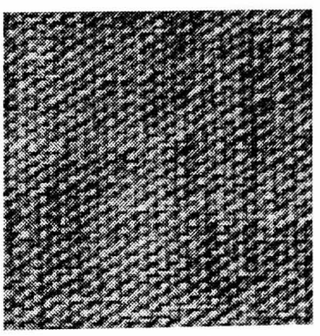
(e) Wood grain.



(f) Packing foam.

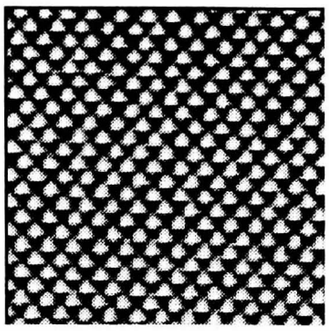


(g) Herringbone weave.

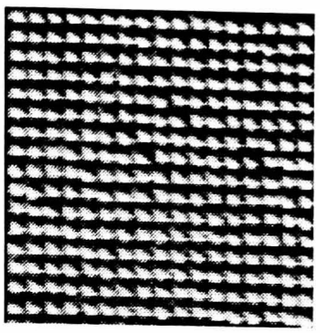


(h) Cotton fabric.

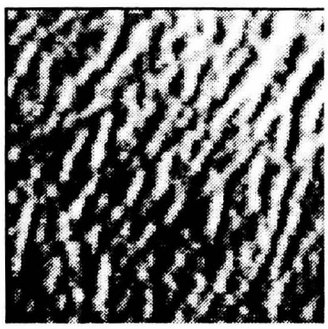
Figure (5.4.1.1a). Texture dataset used to determine the optimum number of amplitude scalars. Results for these textures are presented in **Figure (5.4.1.2a)**.



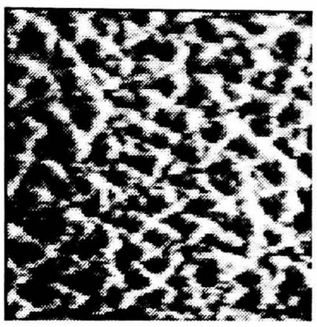
(a) Carbon fibre at 0°.



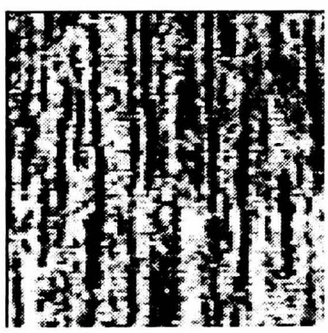
(b) Carbon fibre at -45°.



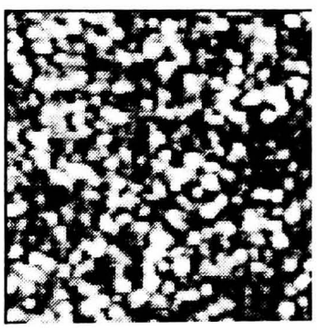
(c) Leather grain #1.



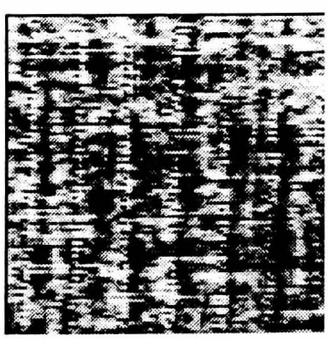
(d) Leather grain #2.



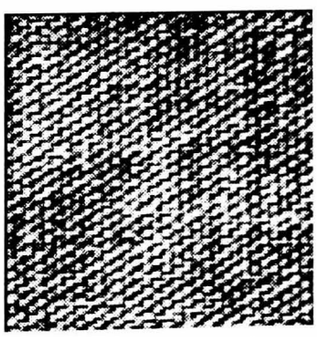
(e) Wood grain.



(f) Packing foam.

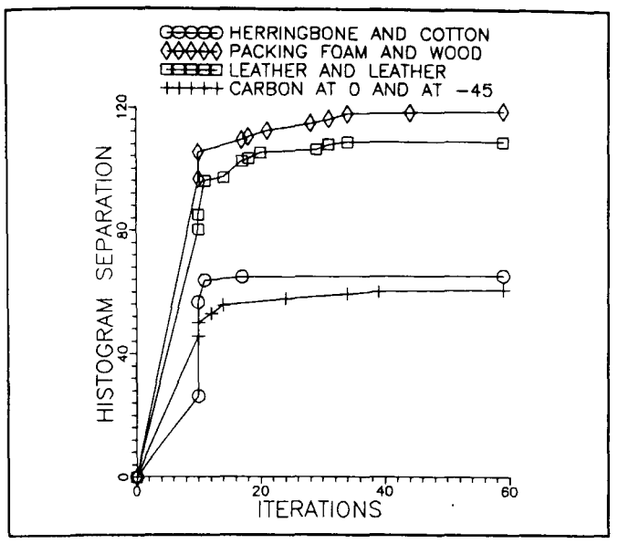


(g) Herringbone weave.

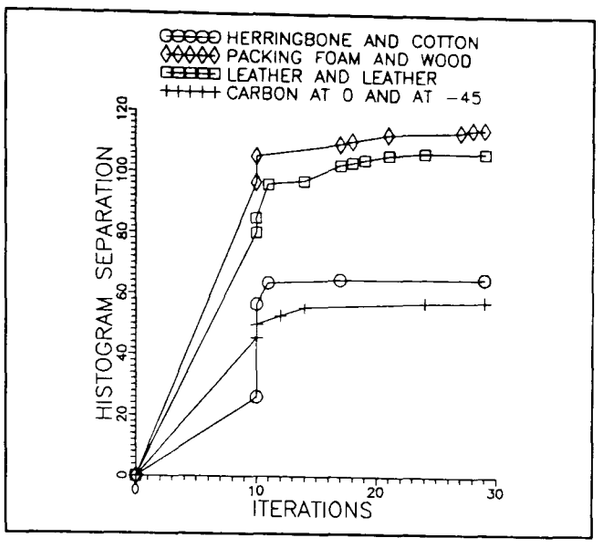


(h) Cotton fabric.

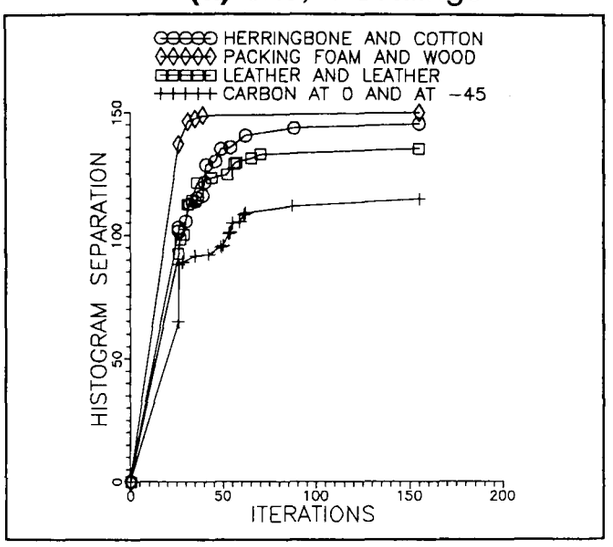
Figure (5.4.1.1b). Histogram equalised version of the texture set of the previous page. Results for these textures are presented in **Figure (5.4.1.2b)**.



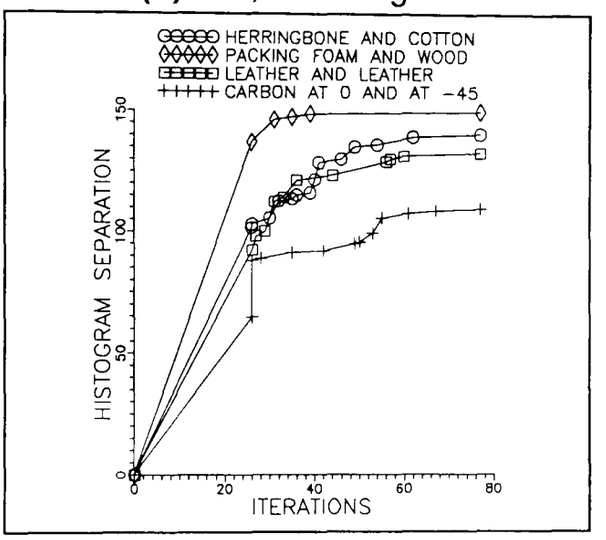
(a) 3x3, 4 scalings



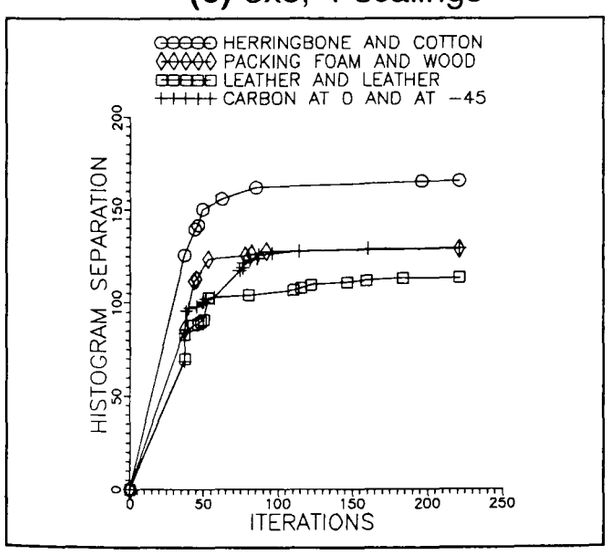
(b) 3x3, 1 scaling



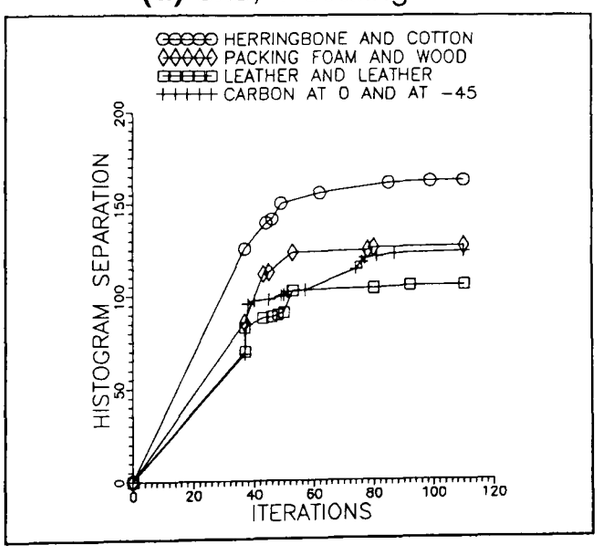
(c) 5x5, 4 scalings



(d) 5x5, 1 scaling

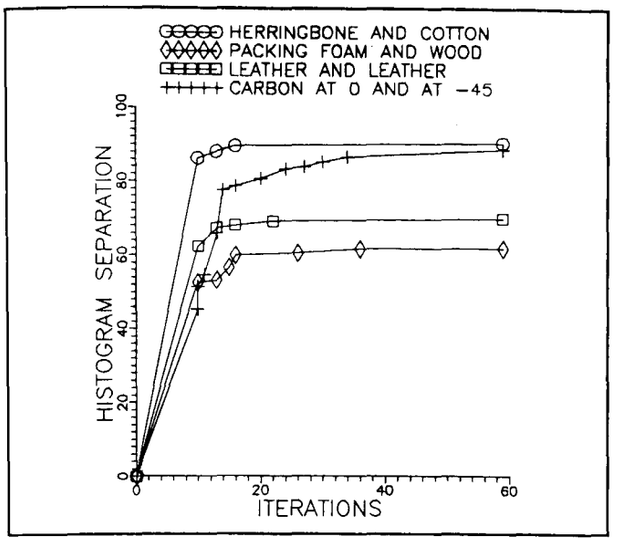


(e) 7x7, 4 scalings

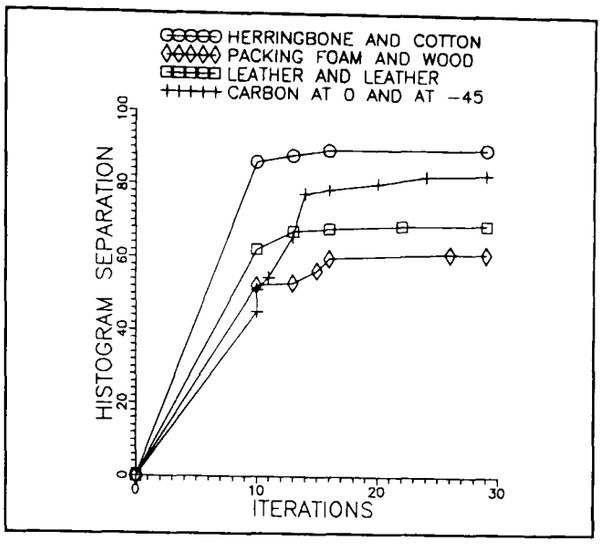


(f) 7x7, 1 scaling

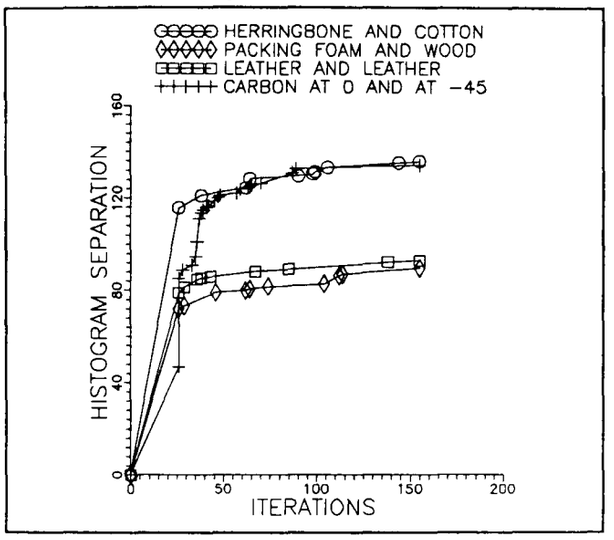
Figure (5.4.1.2a). Graphs showing that one amplitude scaling loop is near optimum for all mask sizes. Texture samples from **Figure (5.4.1.1a)**



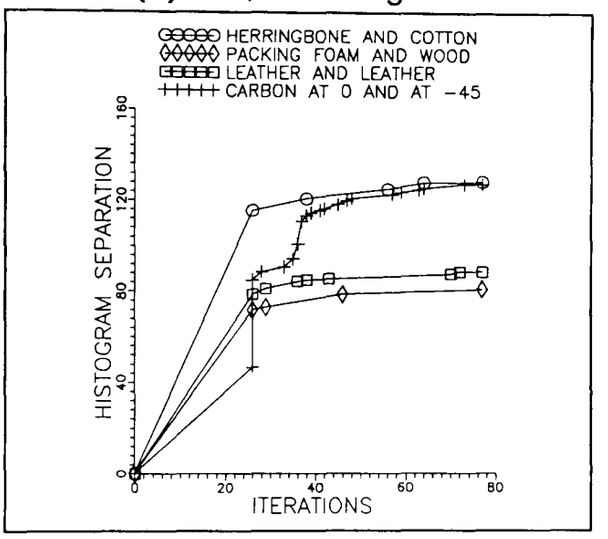
(a) 3x3, 4 scalings



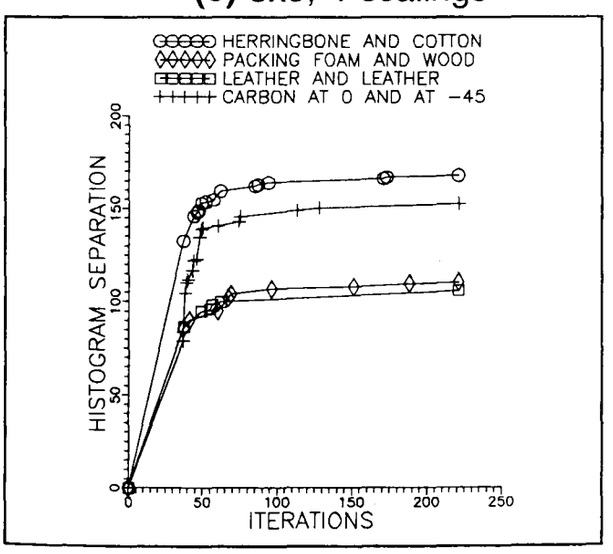
(b) 3x3, 1 scaling



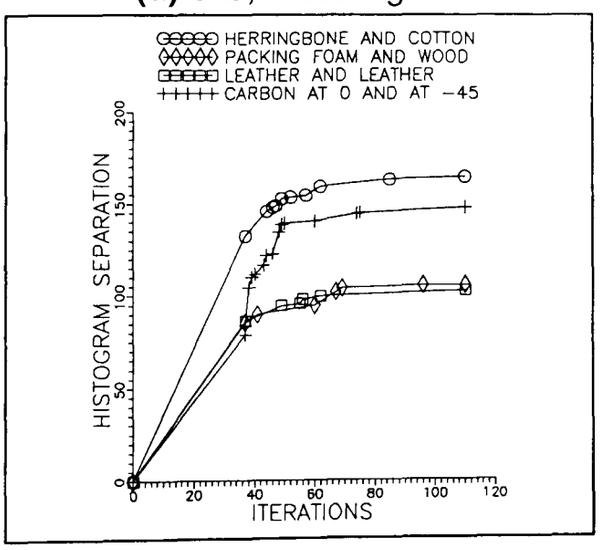
(c) 5x5, 4 scalings



(d) 5x5, 1 scaling

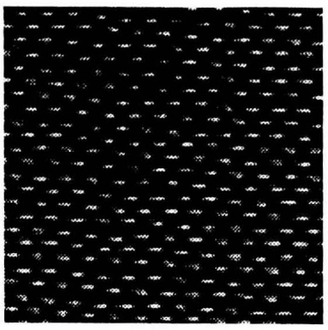


(e) 7x7, 4 scalings

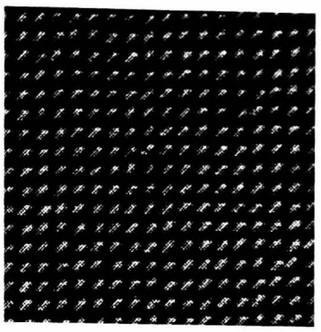


(f) 7x7, 1 scaling

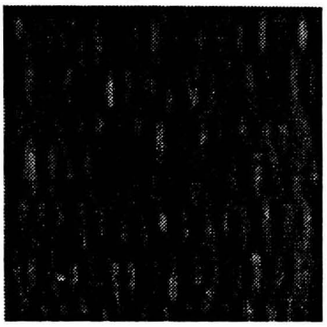
Figure (5.4.1.2b). Same format as previous page, but this time the texture samples have been histogram equalised (shown in **Figure (5.4.1.1b)**).



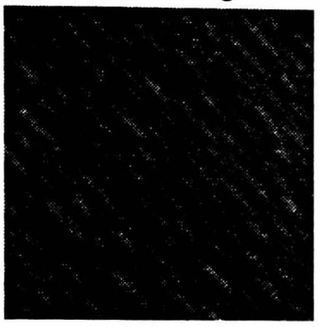
(a) Carbon & glass at 90°.



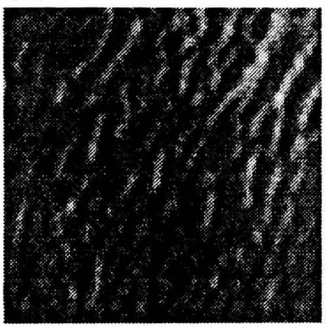
(b) Carbon & glass at 45°.



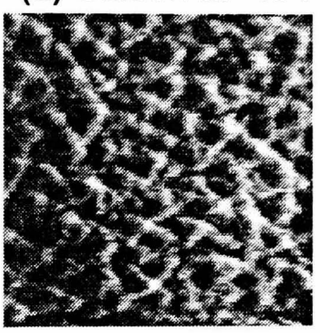
(c) Carbon at 90°.



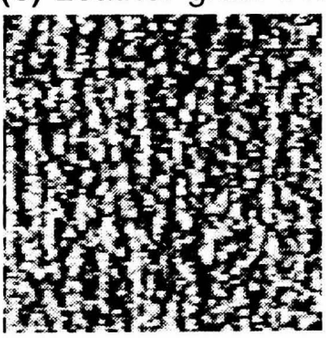
(d) Carbon at -45°.



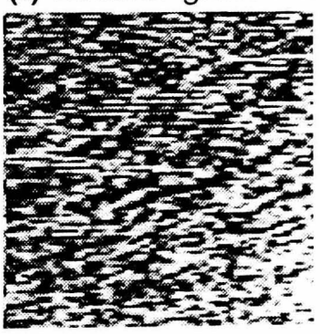
(e) Leather grain #1.



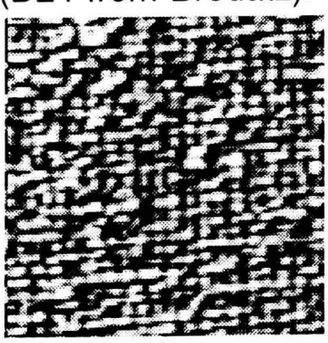
(f) Leather grain #2.



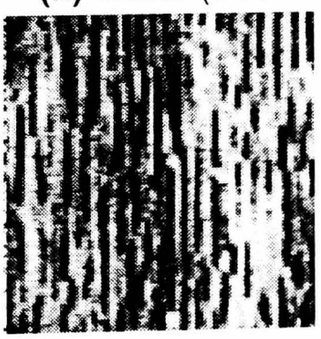
(g) Leather (D24 from Brodatz).



(h) Water (D38 from Brodatz).

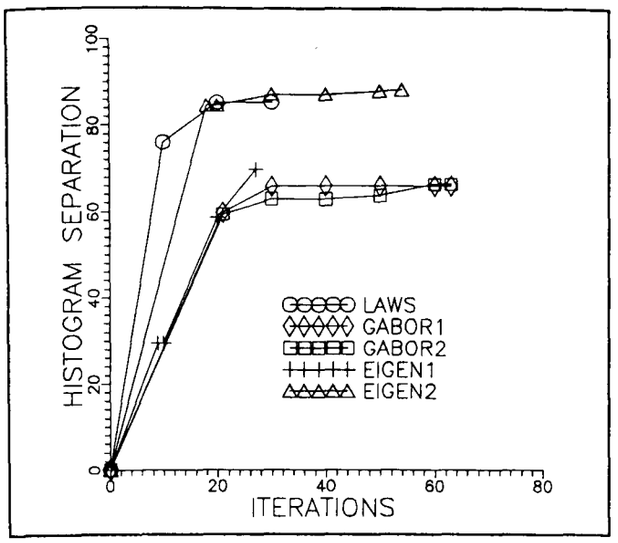


(i) Raffia (D84 from Brodatz).

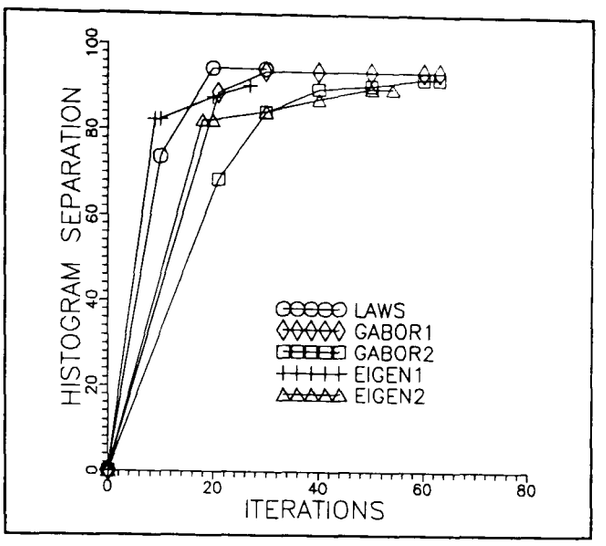


(j) Wood (D68 from Brodatz).

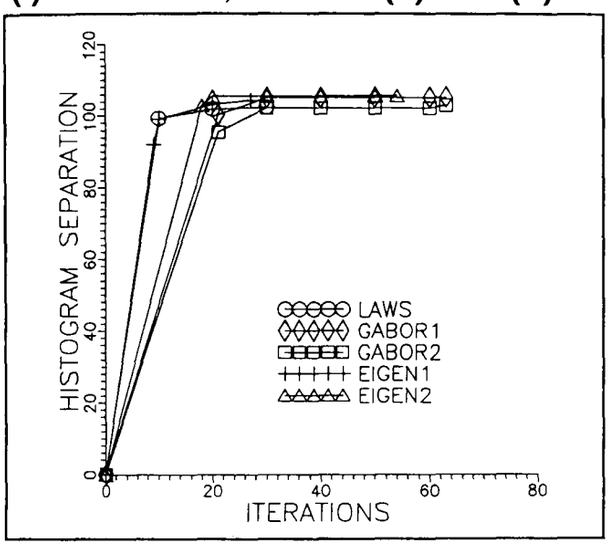
Figure (5.4.2.1). Dataset used to compare performance of various sets of basis mask. Images **(g)**, **(h)**, **(i)**, and **(j)** have been histogram equalised.



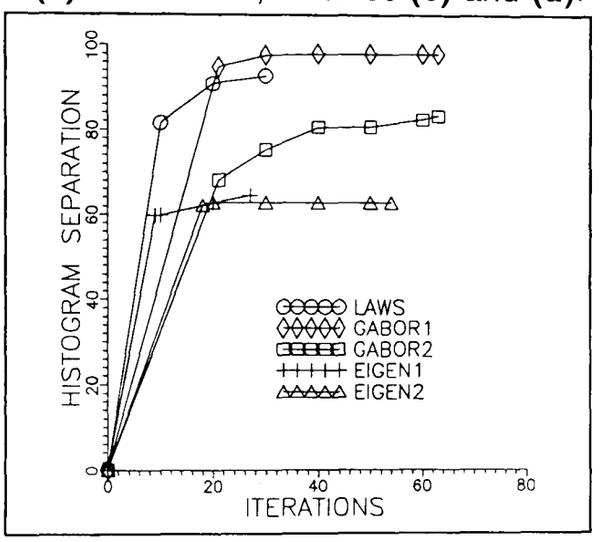
(i) 3x3 masks, textures **(a)** and **(b)**.



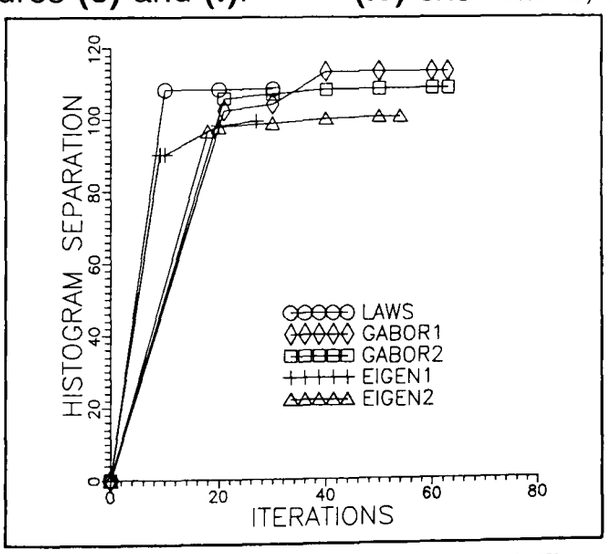
(ii) 3x3 masks, textures **(c)** and **(d)**.



(iii) 3x3 masks, textures **(e)** and **(f)**.

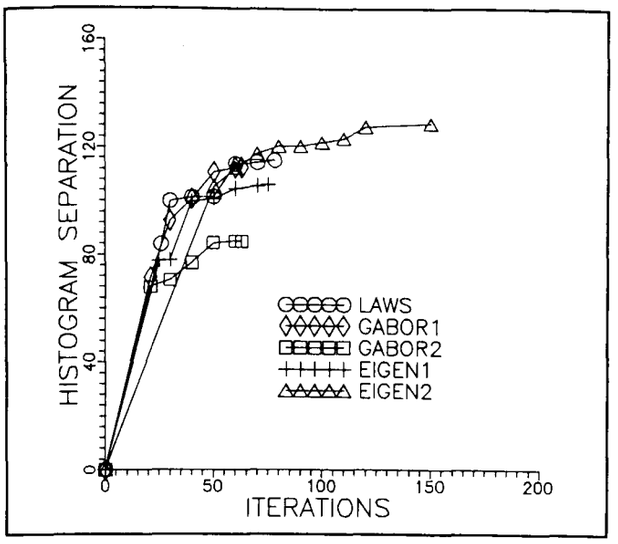


(iv) 3x3 masks, textures **(g)** and **(h)**.

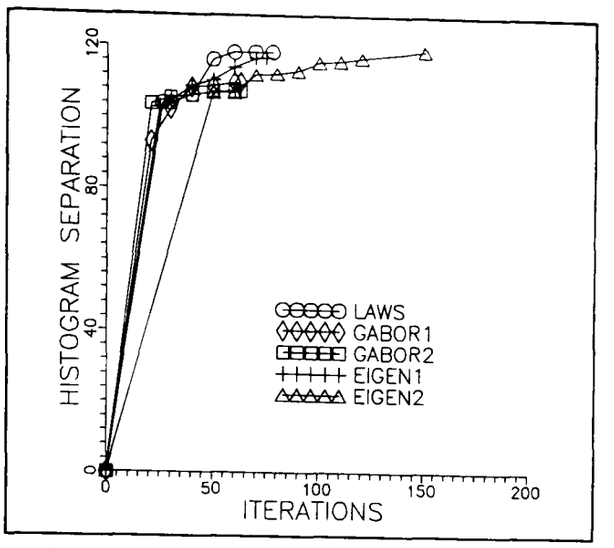


(v) 3x3 masks, textures **(i)** and **(j)**.

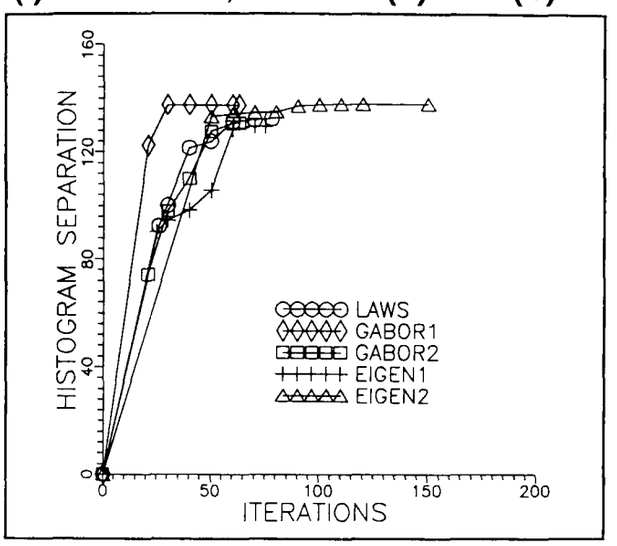
Figure (5.4.2.2a). Optimisation rate for different 3x3 basis mask sets. The dataset is shown in **Figure (5.4.2.1)**.



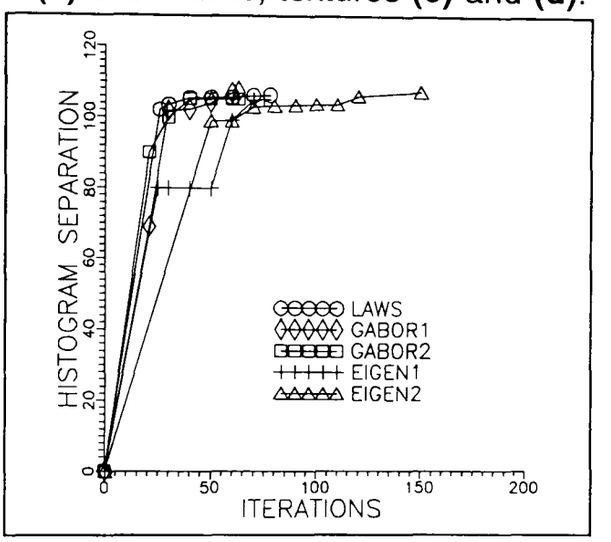
(i) 5x5 masks, textures **(a)** and **(b)**.



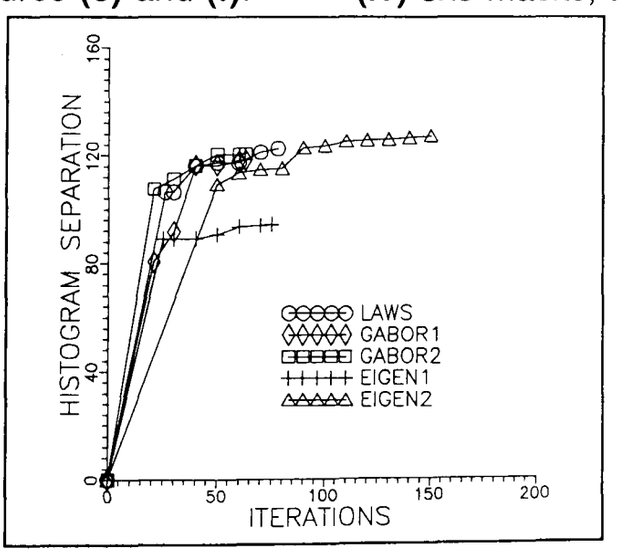
(ii) 5x5 masks, textures **(c)** and **(d)**.



(iii) 5x5 masks, textures **(e)** and **(f)**.

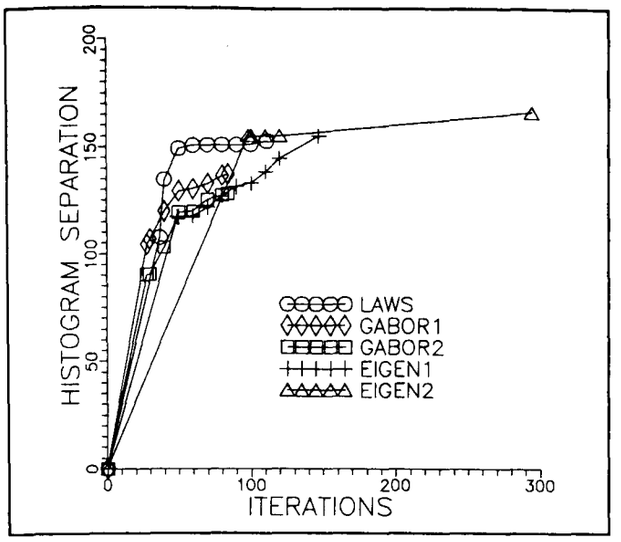


(iv) 5x5 masks, textures **(g)** and **(h)**.

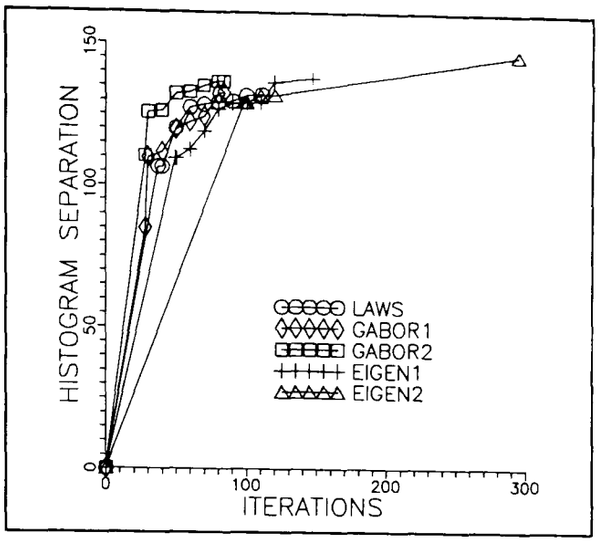


(v) 5x5 masks, textures **(i)** and **(j)**.

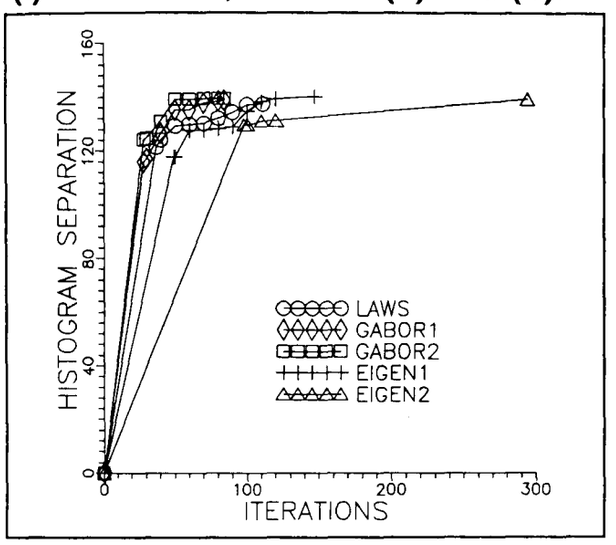
Figure (5.4.2.2b). Optimisation rate for different 5x5 basis mask sets. The dataset is shown in **Figure (5.4.2.1)**.



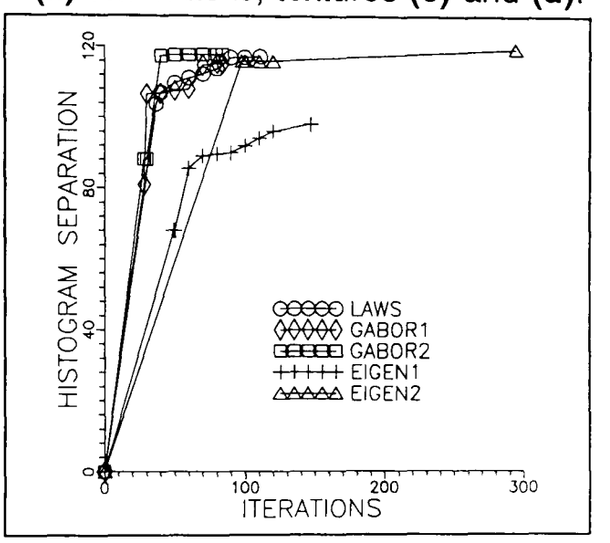
(i) 7x7 masks, textures **(a)** and **(b)**.



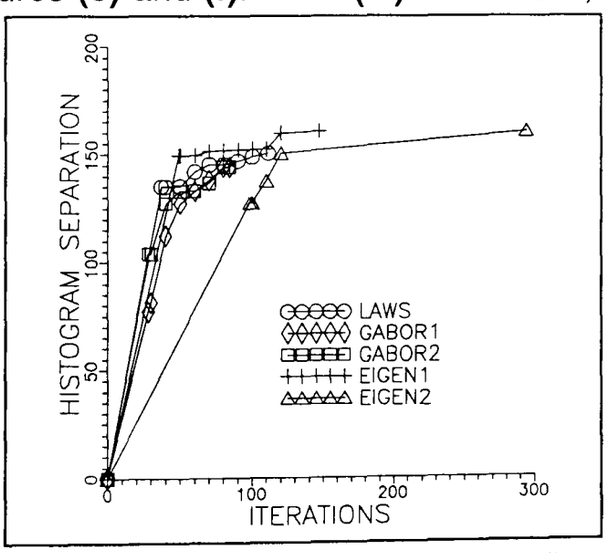
(ii) 7x7 masks, textures **(c)** and **(d)**.



(iii) 7x7 masks, textures **(e)** and **(f)**.



(iv) 7x7 masks, textures **(g)** and **(h)**.



(v) 7x7 masks, textures **(i)** and **(j)**.

Figure (5.4.2.2c). Optimisation rate for different 7x7 basis mask sets. The dataset is shown in **Figure (5.4.2.1)**.

The number of masks for **EIGEN1** is therefore 9 3x3 masks, 25 5x5 masks, and 49 7x7 masks, and the number of masks for **EIGEN2** is 18 3x3 masks, 50 5x5 masks, and 98 7x7 masks. The performance of each mask set can be summarised as follows.

- The Laws basis set produces consistently high discrimination in relatively few iterations
- The Gabor sets are slightly less consistent in the discrimination produced. Of the two Gabor sets, **GABOR1** with $\sigma_x=\sigma_y=100$ performs slightly better on some textures.
- **EIGEN2**, the basis set consisting of eigenfilters extracted from both texture samples, provides the best discrimination for all texture samples with 5x5 and 7x7 masks. More iterations are required, due to the high number of masks in the basis set. **EIGEN1**, with eigenfilters extracted from only one texture sample, performs inconsistently.

In summary, in this section the performance of potential basis mask sets for the specific application of discriminating two textures has been investigated. The conclusion is that the Laws basis set is preferable when discriminating two textures. The number of iterations required is modest, and the difference between the discrimination achieved by the Laws set and the eigenfilter set is generally very small.

5.5 Comparison of the Basis and Benke and Skinner Algorithms.

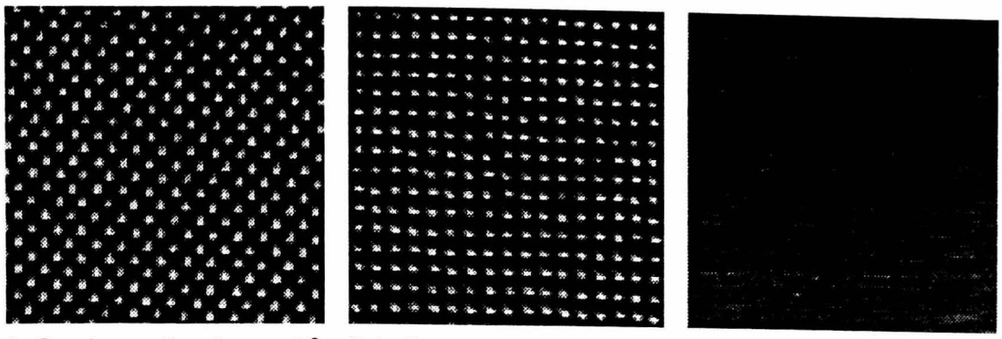
In this section the performance of the basis algorithm and the improved Monte Carlo algorithm developed in **Section 4.4** is compared. Both algorithms will use histogram separation as the optimisation criterion. The basis algorithm will use one scaling loop, with the scaling value equal to 0.5. Results are presented for discrimination of two and three textures. For two textures, the basis algorithm will use the Laws basis set, whereas for three textures performance of Laws, Gabor, and eigenfilter basis sets is compared against the performance of the Benke and Skinner algorithm.

The texture dataset used to compare the performance for discrimination

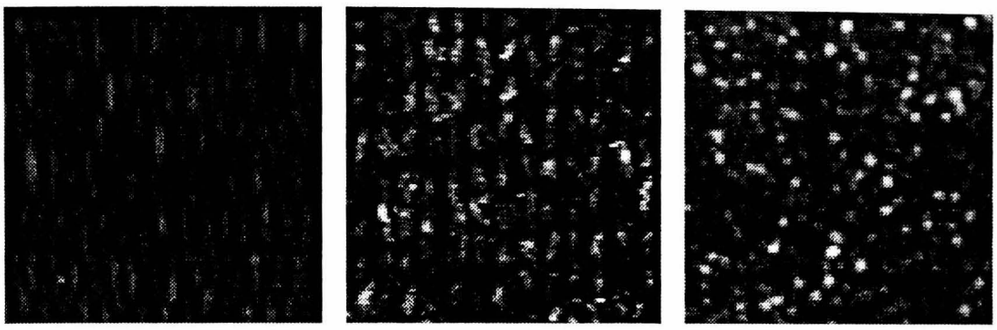
of two textures is shown in **Figure (5.5.1)**. The texture pairs are **(a)** and **(b)**, carbon fibre with glass weft, orientations 0° and -45°, **(c)** and **(d)** carbon fibre with thermoplastic weft, orientations 0° and 90°, **(e)** and **(f)** wool and packing foam, **(g)** and **(h)** two different grains of leather, **(i)** and **(j)** histogram equalised images of raffia and leather taken from **[Brodatz,1966]**, and **(k)** and **(l)** histogram equalised images of water and wood taken from **[Brodatz,1966]**. The results for 3x3, 5x5, and 7x7 masks are shown in **Figures (5.5.2a)**, **(5.5.2b)**, and **(5.5.2c)** respectively.

These results show a definite superiority of the basis algorithm over the Benke and Skinner algorithm, especially for 5x5 and 7x7 masks. Only for textures **(a)** and **(b)** with 3x3 masks does the Monte Carlo algorithm produce a higher discrimination, and in that case it seems that 3x3 masks are too small to produce satisfactory discrimination for those textures anyway. For 5x5 and 7x7 masks on the same textures the basis algorithm is markedly superior.In general it seems that not only is the basis algorithm faster, which might have been expected, but also that the discrimination achieved is higher than that produced using the Monte Carlo algorithm. These results were of course produced using only 1000 iterations of the Monte Carlo algorithm, and given more iterations it is likely that the discrimination would be improved. The number of iterations required however is unknown and may well be prohibitive. It is concluded that in the discrimination of two textures the basis algorithm is in all cases preferable, and that the augmented Laws basis set is appropriate.

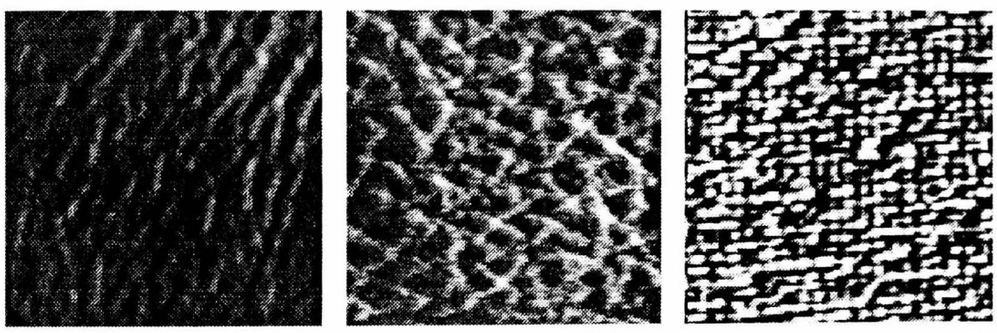
For the discrimination of three textures the performance of the Monte Carlo algorithm is compared with the basis algorithm using the augmented Laws basis set, the Gabor basis set with $\sigma_x=\sigma_y=100$, and the basis set comprising eigenfilters extracted from all three texture samples to be discriminated. Data is only presented for 5x5 masks, since this seems to be representative of results obtained using other mask sizes. The texture dataset is not shown, since it comprises textures already presented in **Figures (5.4.1.1a)**, **(5.4.2.1)**, and **(5.5.1)**.



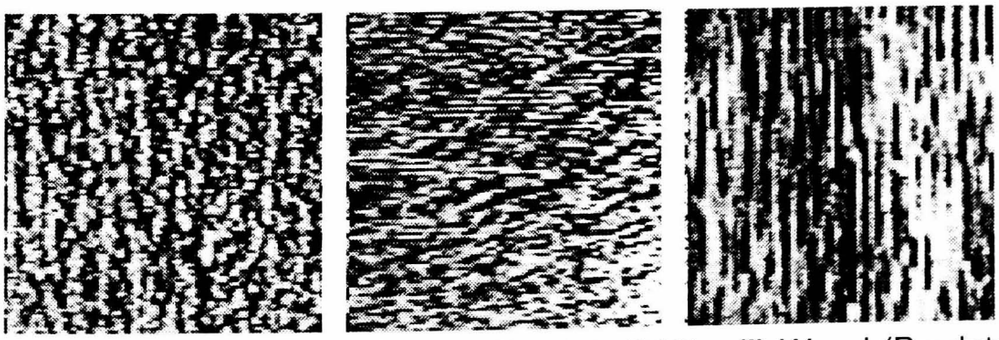
(a) Carbon & glass 0°. **(b)** Carbon & glass -45°. **(c)** Carbon 0°



(d) Carbon 90°. **(e)** Wool. **(f)** Packing foam

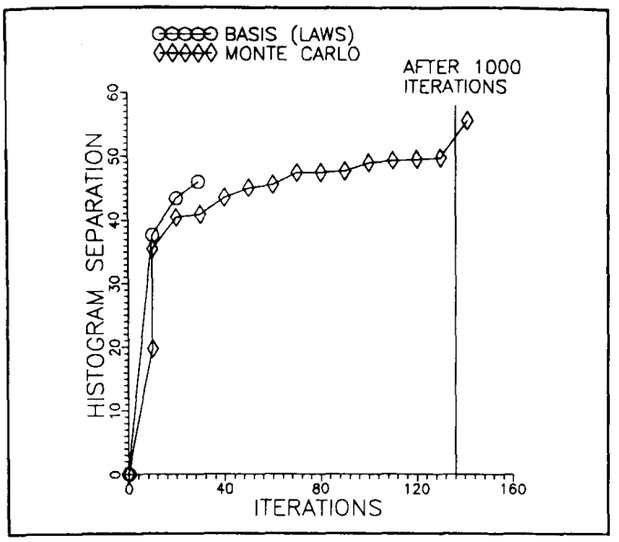


(g) Leather grain #1. **(h)** Leather grain #2. **(i)** Raffia (Brodatz D84)

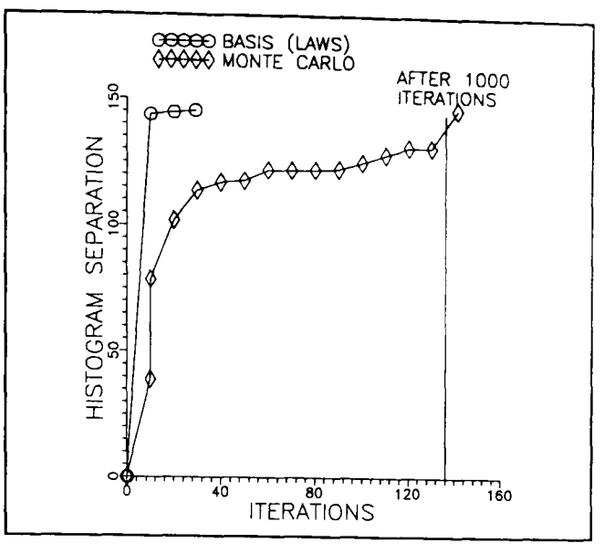


(j) Leather (Brodatz D24). **(k)** Water (Brodatz D38). **(l)** Wood (Brodatz D68).

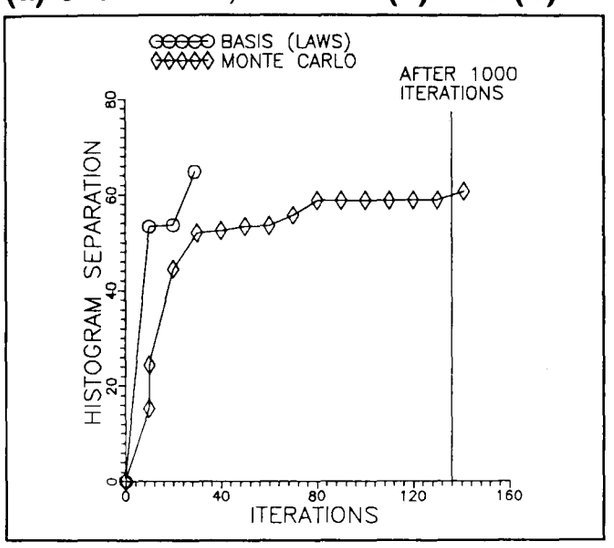
Figure (5.5.1). Texture dataset used to compare performance of basis and Monte Carlo algorithms.



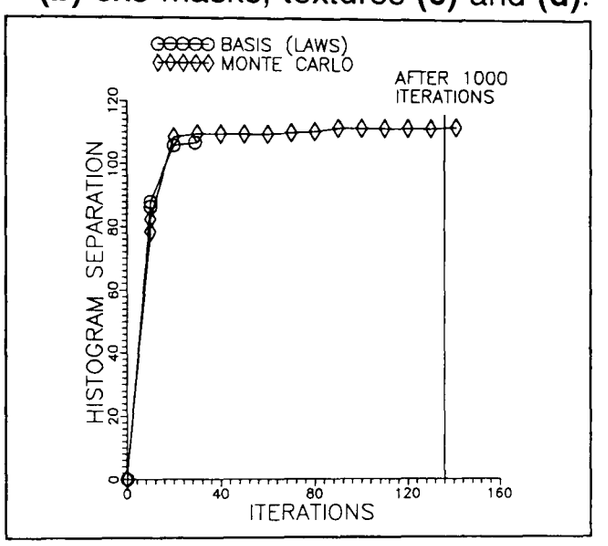
(a) 3x3 masks, textures **(a)** and **(b)**.



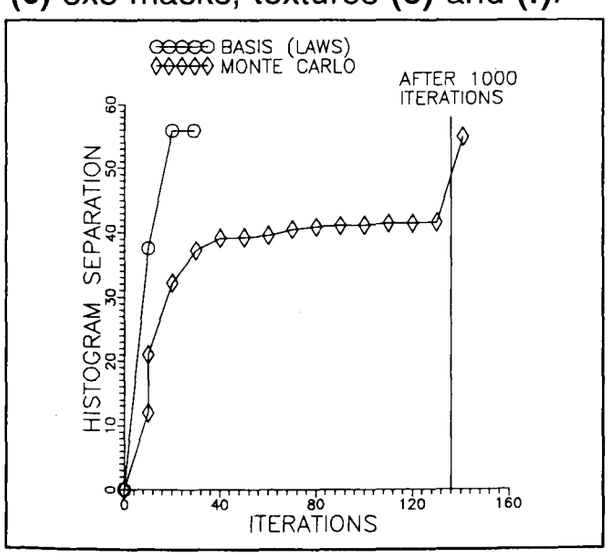
(b) 3x3 masks, textures **(c)** and **(d)**.



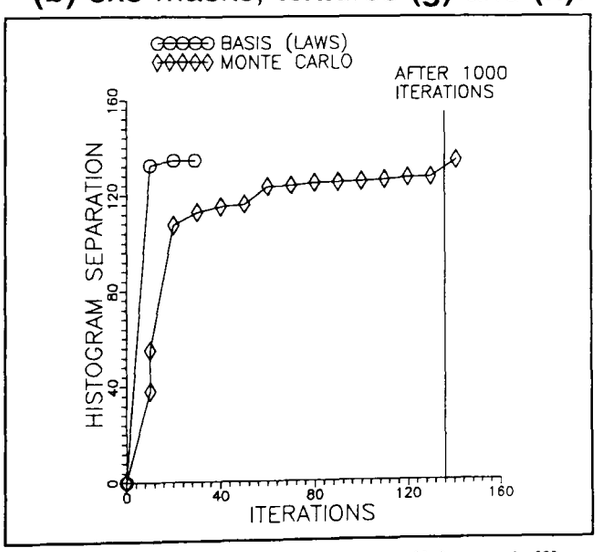
(c) 3x3 masks, textures **(e)** and **(f)**.



(b) 3x3 masks, textures **(g)** and **(h)**.

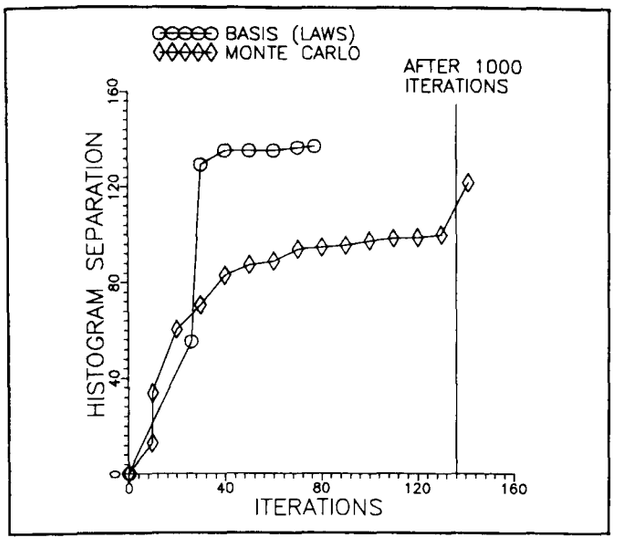


(e) 3x3 masks, textures **(i)** and **(j)**.

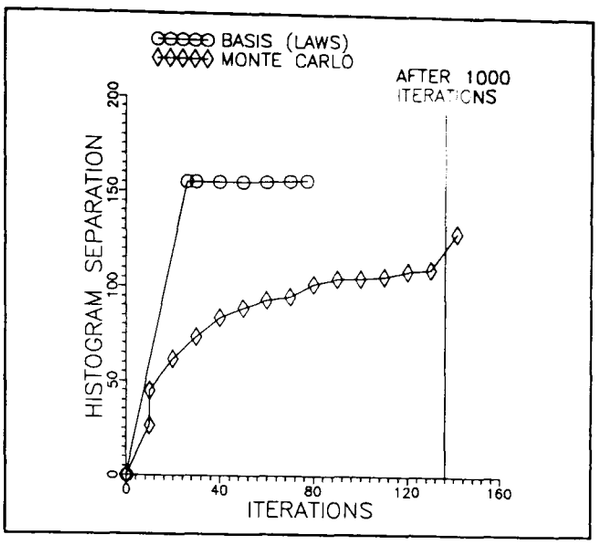


(f) 3x3 masks, textures **(k)** and **(l)**.

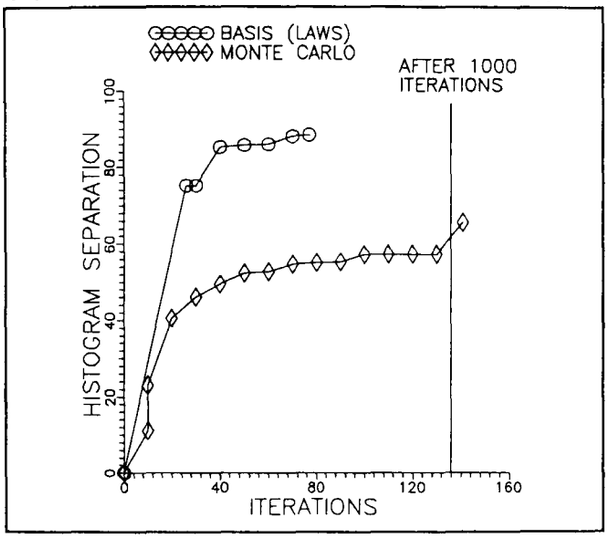
Figure (5.5.2a). Comparison of optimisation rate for basis and Monte Carlo algorithms using 3x3 masks. The dataset is from **Figure (5.5.1)**.



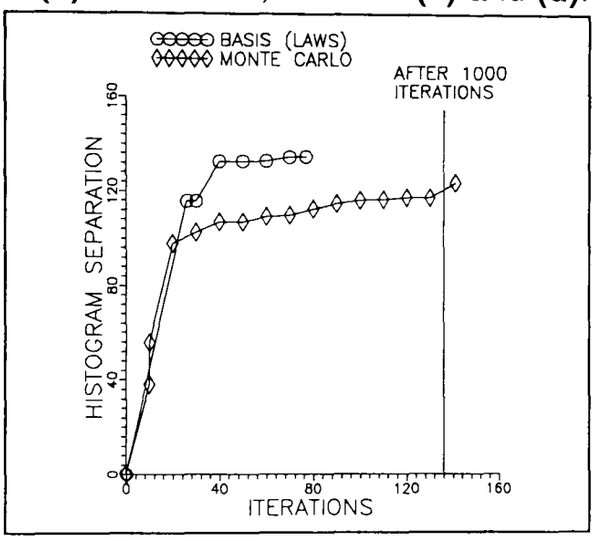
(a) 5x5 masks, textures **(a)** and **(b)**.



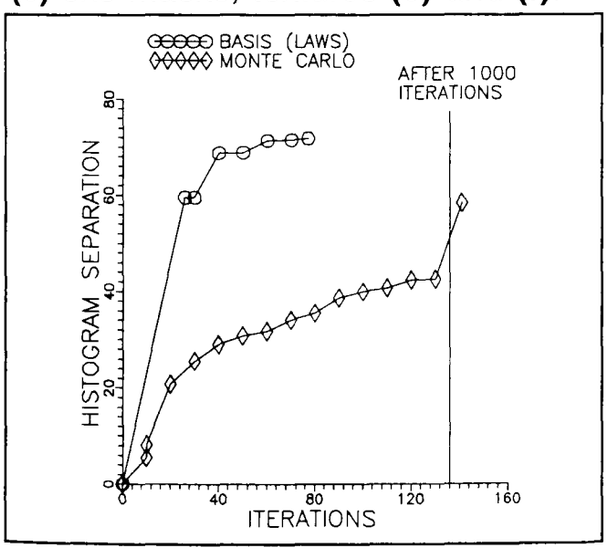
(b) 5x5 masks, textures **(c)** and **(d)**.



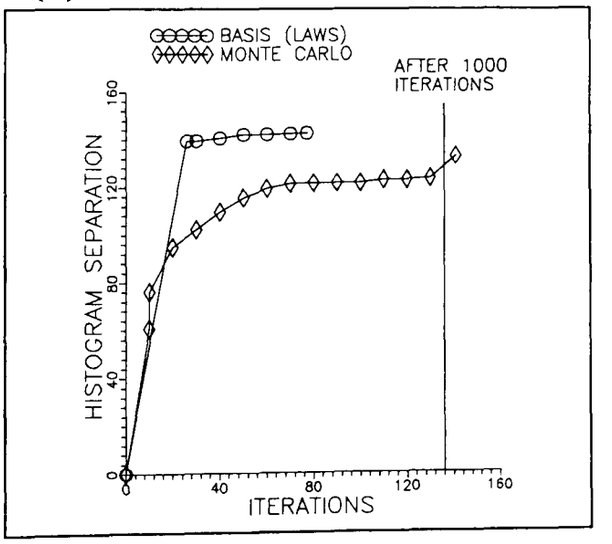
(c) 5x5 masks, textures **(e)** and **(f)**.



(b) 5x5 masks, textures **(g)** and **(h)**.

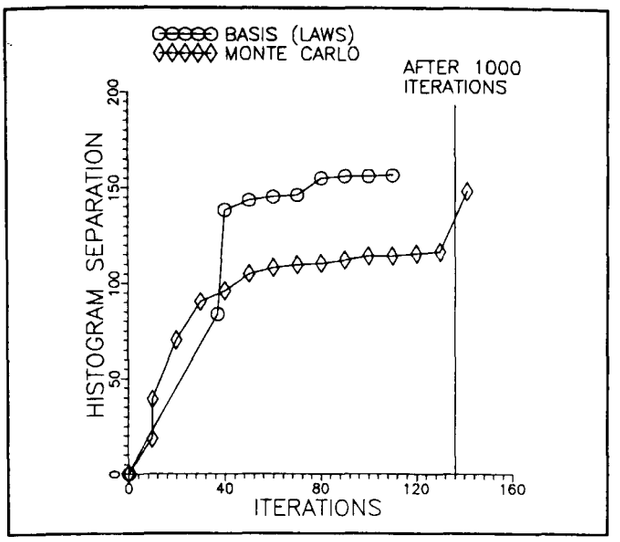


(e) 5x5 masks, textures **(i)** and **(j)**.

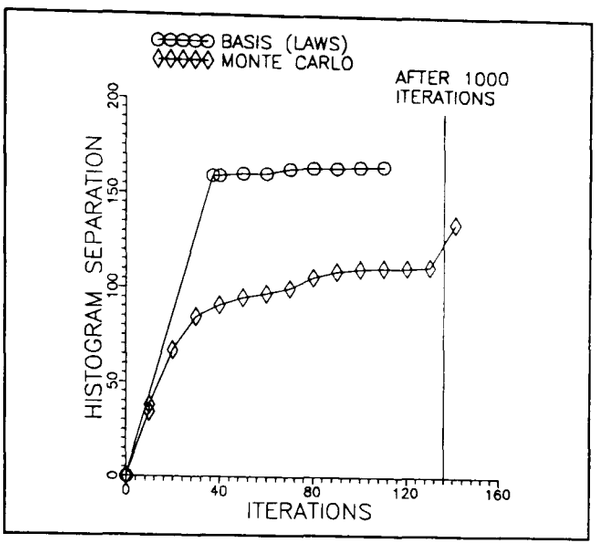


(f) 5x5 masks, textures **(k)** and **(l)**.

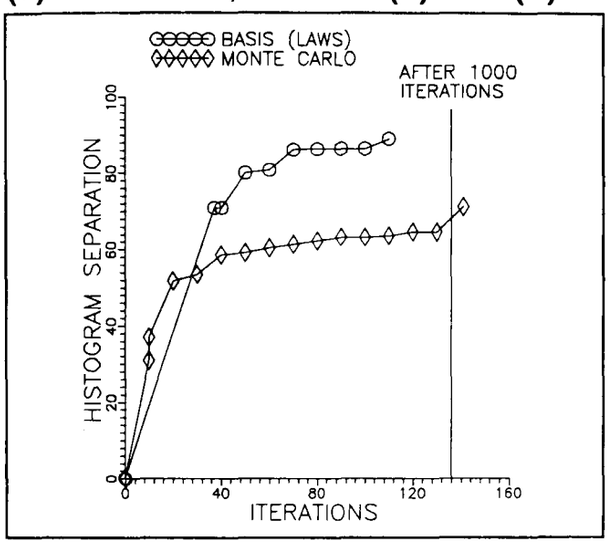
Figure (5.5.2b). Comparison of optimisation rate for basis and Monte Carlo algorithms using 5x5 masks. The dataset is from **Figure (5.5.1)**.



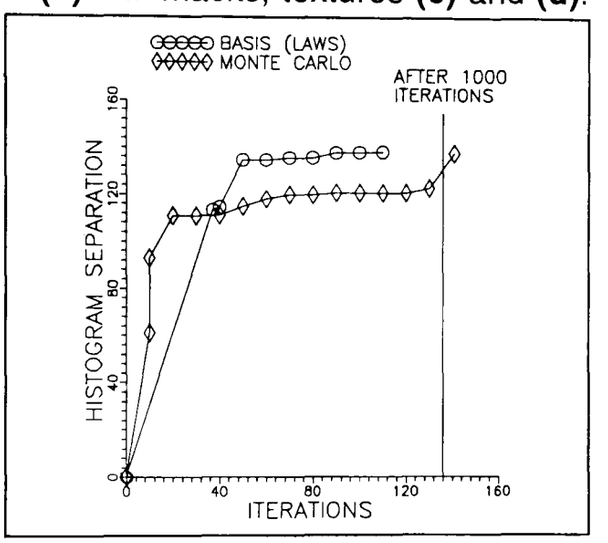
(a) 7x7 masks, textures **(a)** and **(b)**.



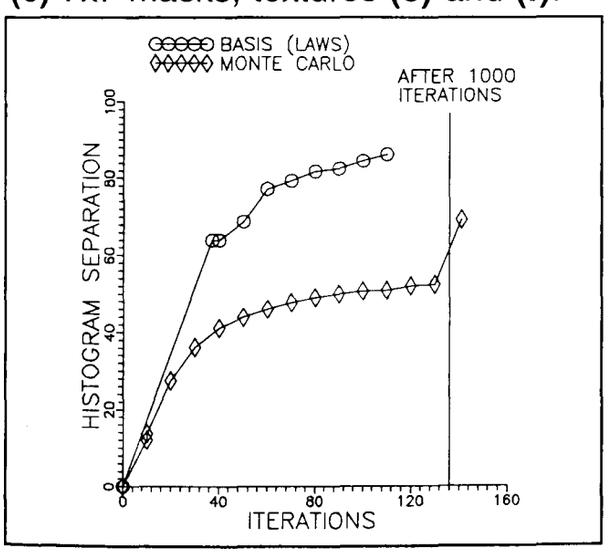
(b) 7x7 masks, textures **(c)** and **(d)**.



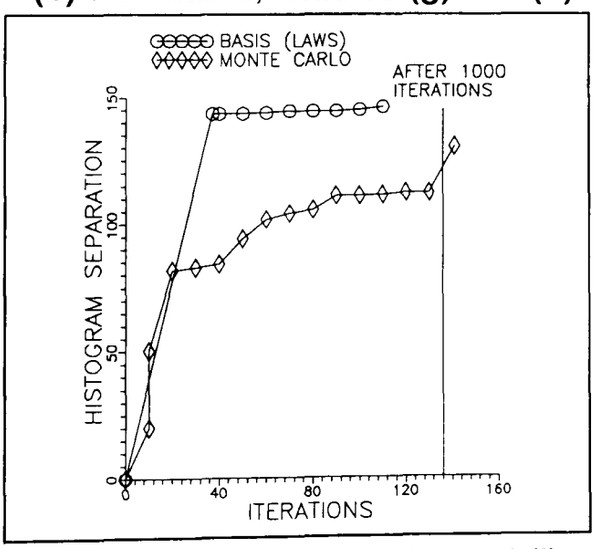
(c) 7x7 masks, textures **(e)** and **(f)**.



(b) 7x7 masks, textures **(g)** and **(h)**.

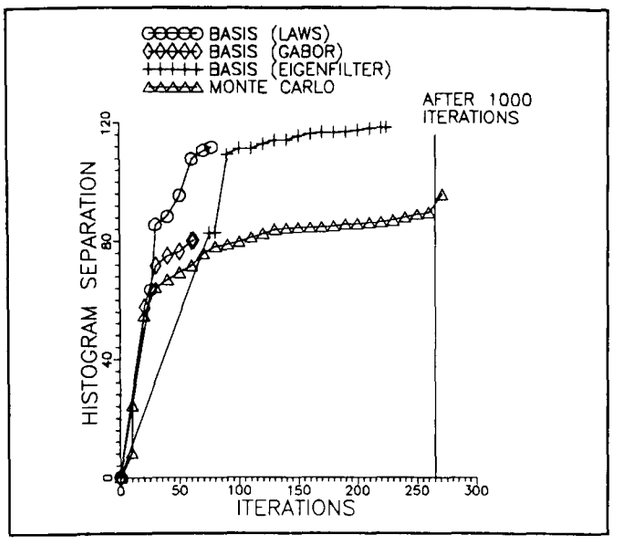


(e) 7x7 masks, textures **(i)** and **(j)**.

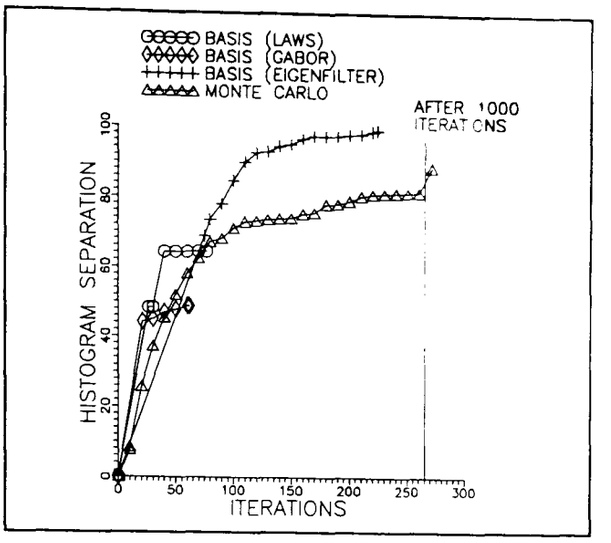


(f) 7x7 masks, textures **(k)** and **(l)**.

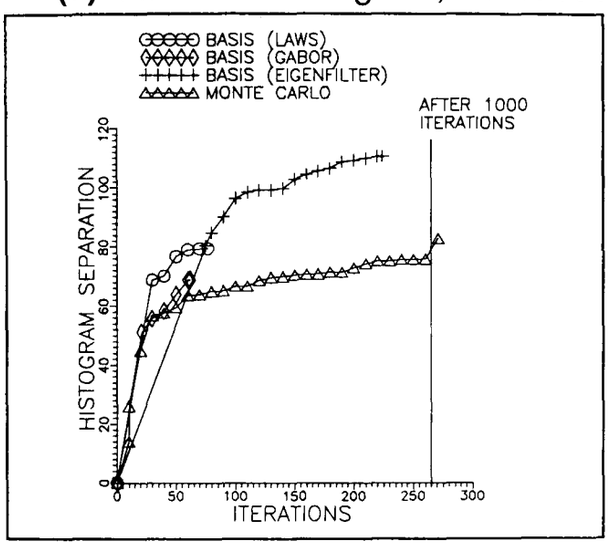
Figure (5.5.2c). Comparison of optimisation rate for basis and Monte Carlo algorithms using 7x7 masks. The dataset is from **Figure (5.5.1)**.



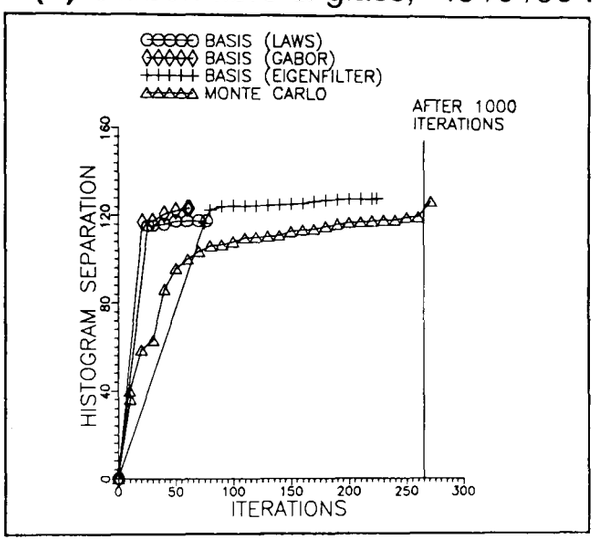
(a) Carbon fibre & glass, 0°/-45°/90°.



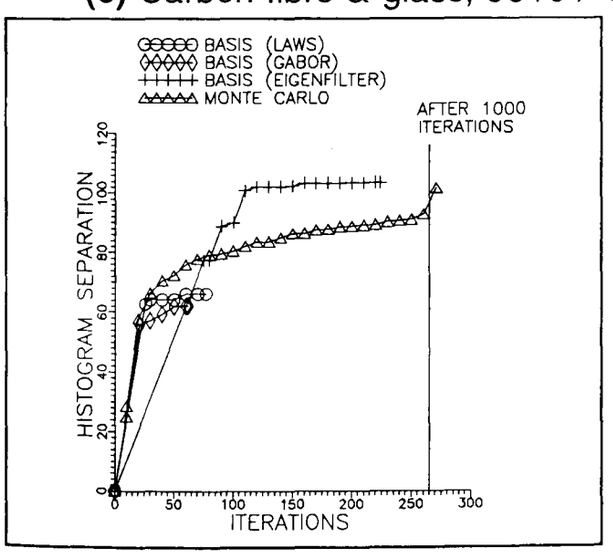
(b) Carbon fibre & glass, -45°/0°/90°.



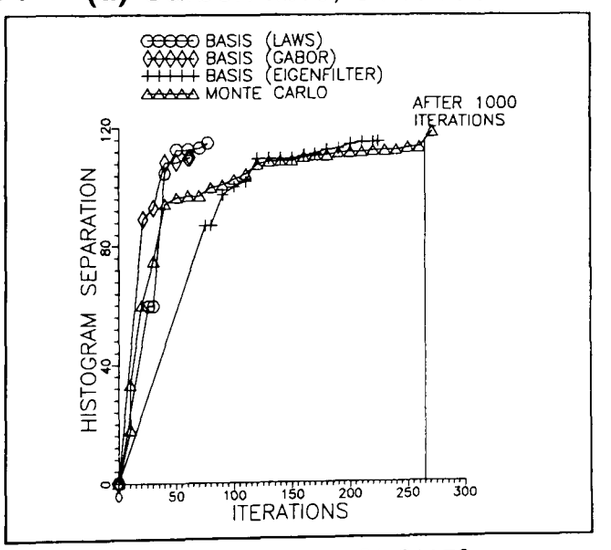
(c) Carbon fibre & glass, 90°/0°/-45°.



(d) Carbon fibre, 0°/45°/90°.

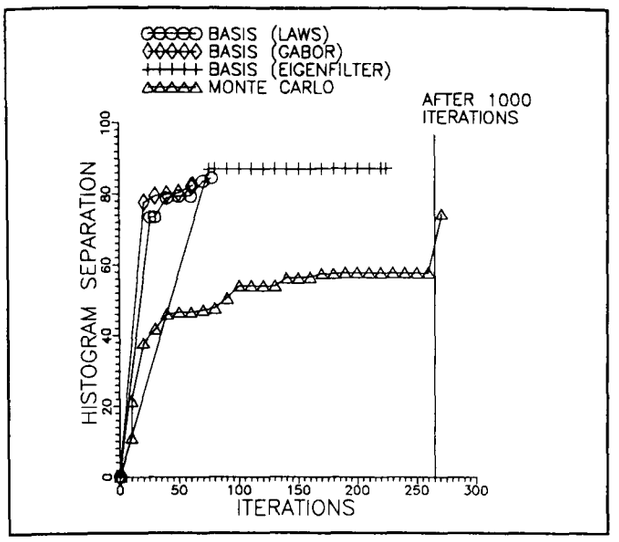


(e) Carbon fibre, 45°/0°/90°.

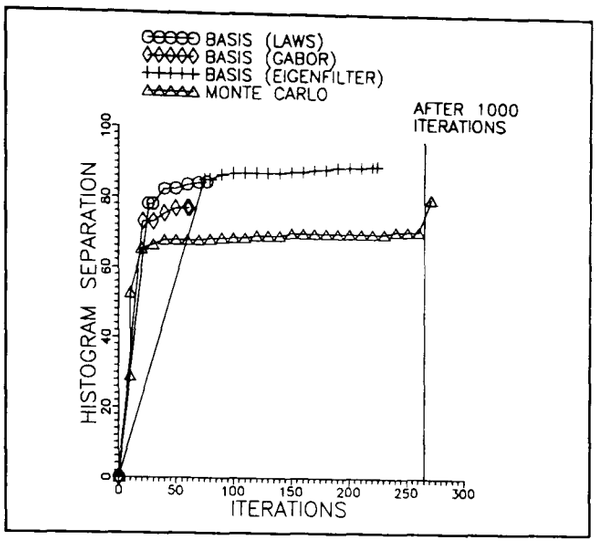


(f) Carbon fibre, 90°/0°/45°.

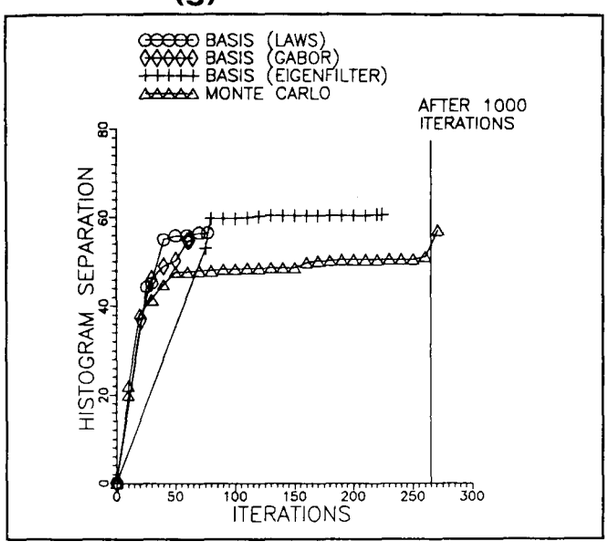
Figure (5.5.3). Discrimination between three textures. The first texture in each list is the texture discriminated. All data is for 5x5 masks only.



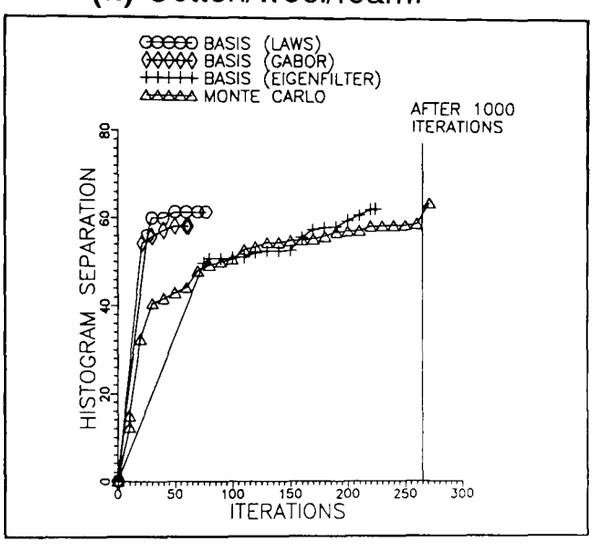
(g) Wool/cotton/foam.



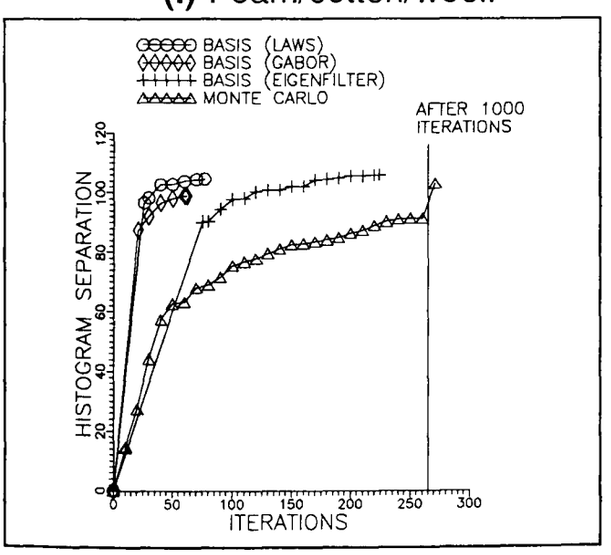
(h) Cotton/wool/foam.



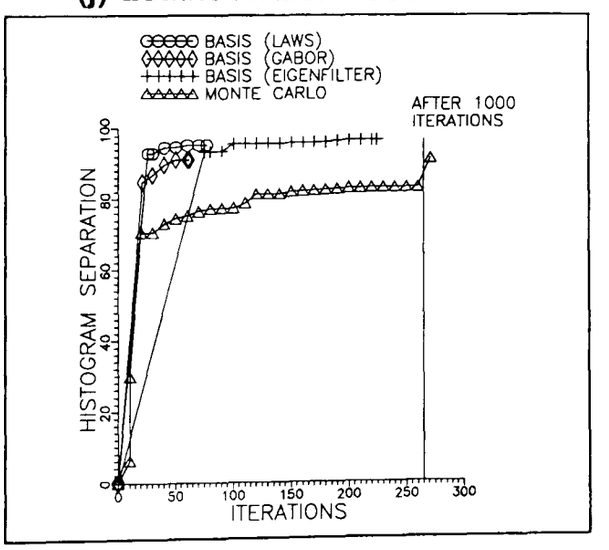
(i) Foam/cotton/wool.



(j) Leather/water/wood.



(k) Water/leather/wood.



(l) Wood/leather/water.

Figure (5.5.3) continued from previous page. Discrimination between three textures. The first texture in each list is the texture discriminated. All data is for 5x5 masks only.

The triples of texture to be discriminated are carbon fibre with glass weft, orientations 0°, -45°, and 90°, carbon fibre with thermoplastic weft, orientations 0°, 45°, and 90°, wool cotton and packing foam, and leather water and wood. The last texture triple are taken from **[Brodatz,1966]** and have been histogram equalised. The results obtained using this dataset are shown in **Figure (5.5.3)**. These results can be summarised as follows.

- The Laws basis set performs disappointingly for some orientations of carbon fibre. In graph **(e)** for example, when attempting to discriminate carbon at 45° from carbon at 0° and 90°, the histogram separation achieved is only just over 60, compared to 100 for the Benke and Skinner algorithm after 1000 iterations. For the other textures the Laws set performs at least as well and generally better than the Benke and Skinner algorithm.

- The performance of the Gabor basis set is generally worse than the Laws basis set

- The best performance is provided by the eigenfilter basis set. The discrimination achieved is consistently greater, and sometimes significantly greater, than that achieved by the Benke and Skinner algorithm. The high number of masks (75) in this basis set means that more iterations are required than for Laws or Gabor filter sets, but this is still a significant improvement over the performance offered by the Benke and Skinner algorithm.

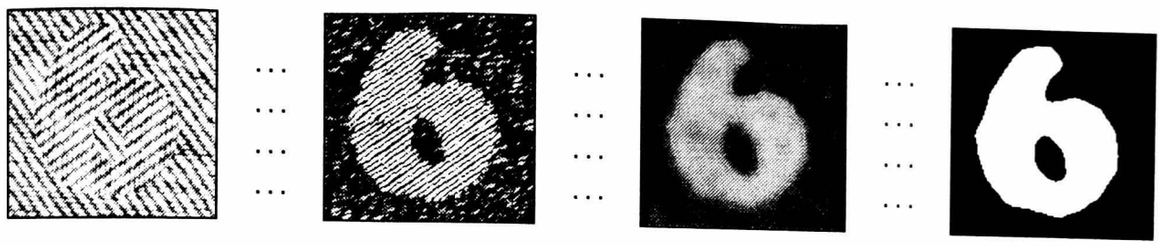
It must be concluded that the Laws basis set is not entirely appropriate for discrimination of three or more texture classes. It has been used successfully in many experiments, most involving carbon fibre samples, but occasionally it is not able to produce the required discrimination. The Laws basis set comprises relatively few filters, and is heuristicly derived. It is perhaps unreasonable therefore to expect it to provide all the basis filters

PAGE
MISSING
IN
ORIGINAL

masks.

- When optimising a mask to discriminate between two textures, the Laws basis set is preferred. When optimising a mask to discriminate between three or more textures, best results are obtained using a set of eigenfilters.

The work contained in this chapter is complementary to the work already detailed in **Chapter Four**, and completes the investigation into texture analysis for automated inspection of composite materials. It has been shown that a single convolution mask can be used to discriminate a required texture from a limited number of background textures. The problem of how to optimise a mask for such a purpose has been examined in some detail with satisfactory results. The following chapter considers an alternative method of boundary detection which can be used when there is no texture information to help.



CHAPTER 6

Optimising Edge Operators.

6.1 Introduction.

The objective of this project is to develop tools which can be used in the automated visual inspection of composite components. The preliminary investigation described in **Chapter One** found that boundary detection could not be accomplished by thresholding, or by the application of conventional edge operators such as the Sobel masks. As a result of this texture analysis techniques have been investigated. Such techniques, as demonstrated in **Chapter Three**, can distinguish between different orientations of carbon fibre. Inspection of dry-fibre lay-ups will, however, often require boundary detection between plies of the same material and orientation, and so the boundary cannot be found by any conventional texture analysis method. For such a problem segmentation is obviously inappropriate, and a method of detecting the boundaries directly must be developed.

Gradient edge operators, such as the Sobel operators, perform poorly on textured images. The edge information at ply boundaries is swamped by the edge information contained in the weave. What are required are edge

operators which are sensitive to edge information at ply boundaries but insensitive to the edge information contained in the texture. Considering the work detailed in the previous chapter, it seems logical to attempt to train convolution masks for the purpose using the basis algorithm.

6.2 Edge Detection in Textured Images.

There are two issues which need to be addressed when performing edge detection in textured images. How to obtain convolution masks suited to the task, and how to most effectively apply them in practice? The following two sections will detail the approach adopted in this work.

6.2.1 Modification of the Training Algorithm.

In fact only superficial modifications need to be made to enable the basis training algorithm to produce masks optimised for edge detection. The "front end" needs to be modified to allow the sizes of the training regions to be user-selectable. This allows one training region to be positioned only over a portion of the "edge" to be detected, whilst the other contains a representative sample of the background texture which the generated mask should be as insensitive to as possible. It is also useful to have a third training region, which can be used in a variety of ways. With three training regions a mask can be trained to enhance two different edge profiles, whilst suppressing the third region which is again background texture. Such a mask might be optimised to enhance edges at varying orientations. Alternately the third region can be used to minimise the effect of a particular feature in the background texture. These points are illustrated in the examples provided in **Section 6.3**.

Using variable sized training regions does not require any modifications to the basic equations of the algorithms, since the optimisation criterion defined in equation **(5.4.2.3)** is calculated on the basis of mean grey-level. This is another advantage of adopting histogram separation as the optimisation criterion.

6.2.2 Applying Edge Operators to Textured Images.

The conventional approach to edge detection in non-textured images is represented in **Figure (6.2.2.1)**. This assumes, for simplicity, that only a single edge operator is required. The input image is convolved with the edge operator, and the result thresholded at a predetermined level to produce a binary edge image. This is illustrated in **Figure (6.2.2.2)** (although note that two edge operators have been used in this example). Depending on the application, a further edge linking process may be performed on the binary edge image to remove "noise" and form edge segments into object boundaries.

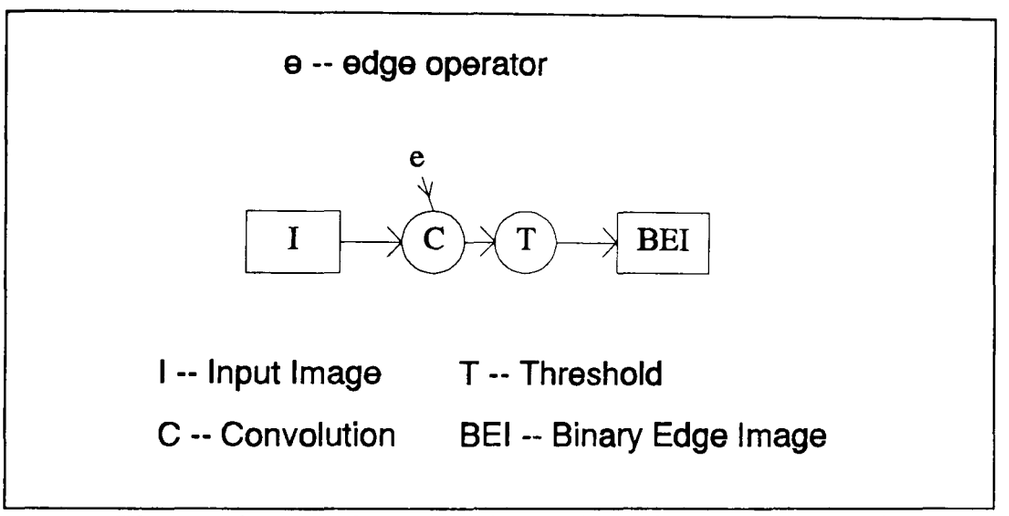
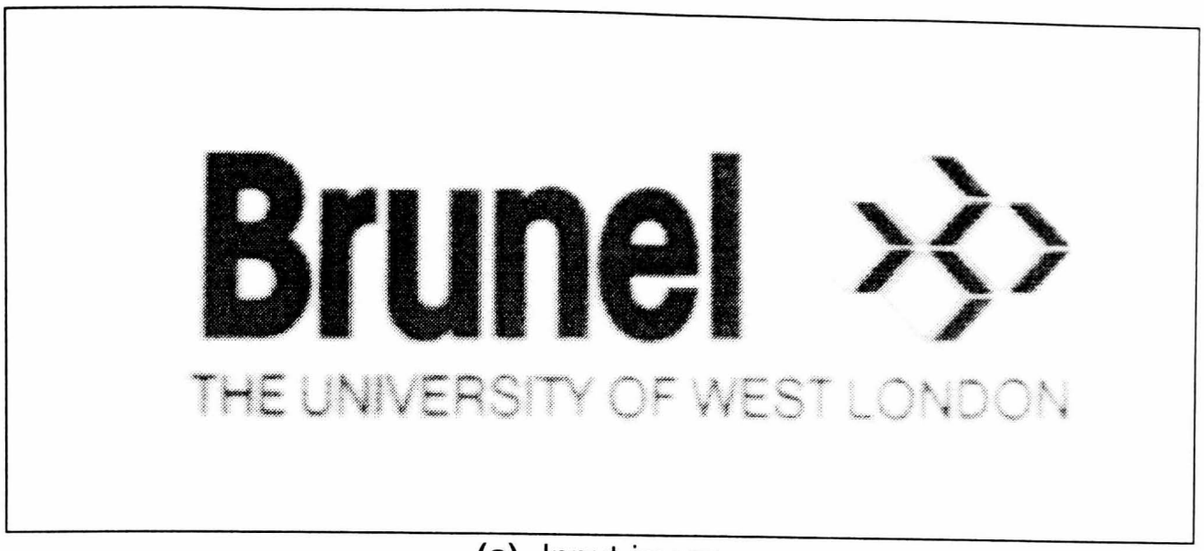
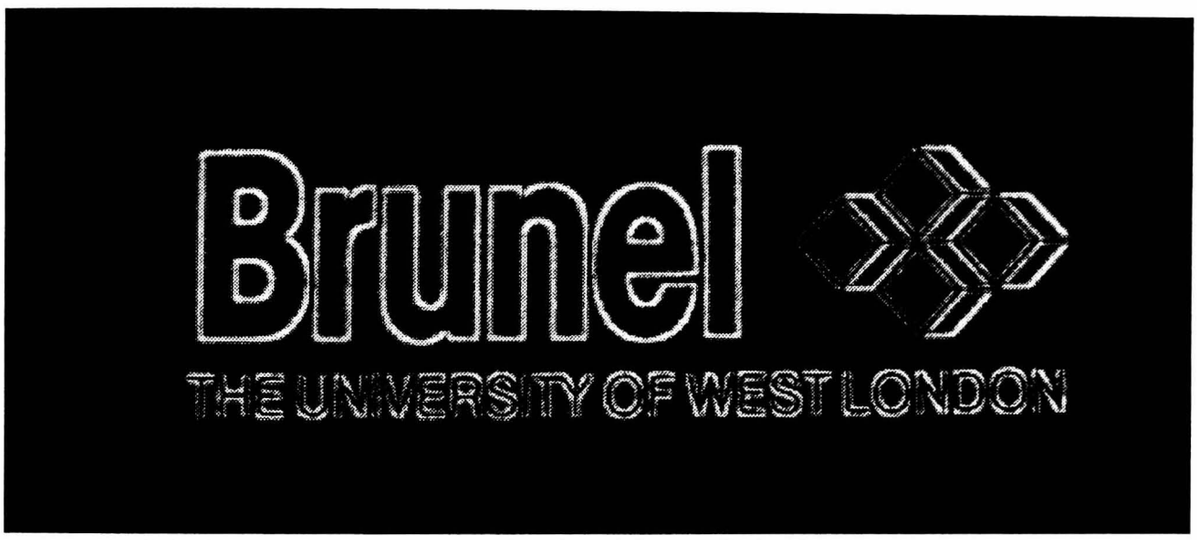


Figure (6.2.2.1). Detecting edges in non-textured images.

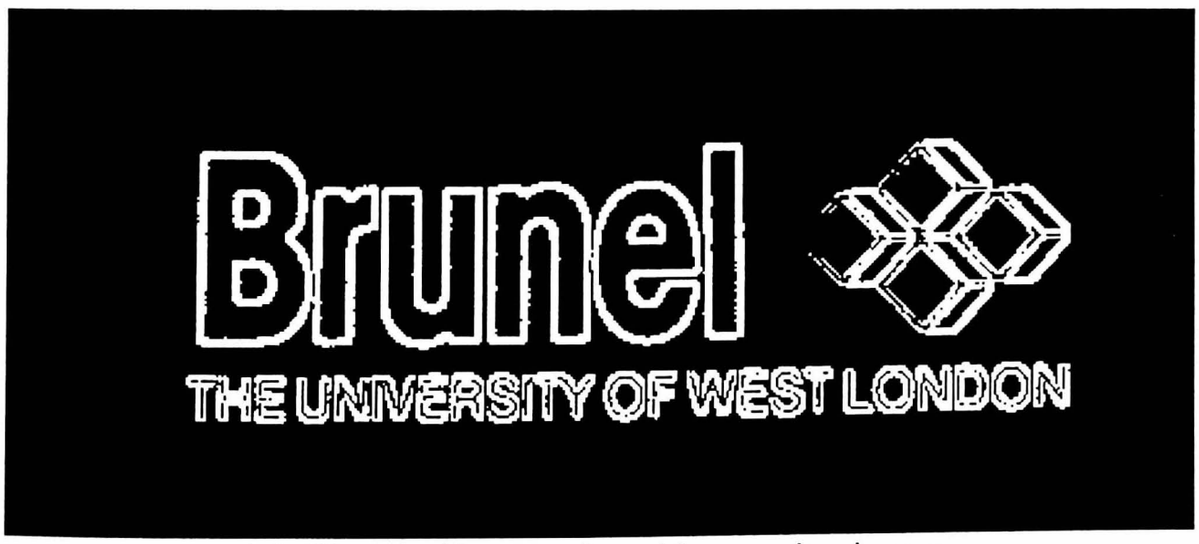
When dealing with textured images the scheme represented in **Figure (6.2.2.1)** is not generally adequate. This is demonstrated in **Figure (6.2.2.3)** with an image of corduroy material. The response of the edge operator to edges inherent in the texture background increases the "noise" of the filtered image to such an extent that it is difficult to adequately discriminate boundary and non-boundary pixels using a simple threshold. Note that this image shows a fairly low contrast texture background. If the texture was visually more distinct, then it would prove impossible to extract the object boundaries using conventional edge operators.



(a). Input image.

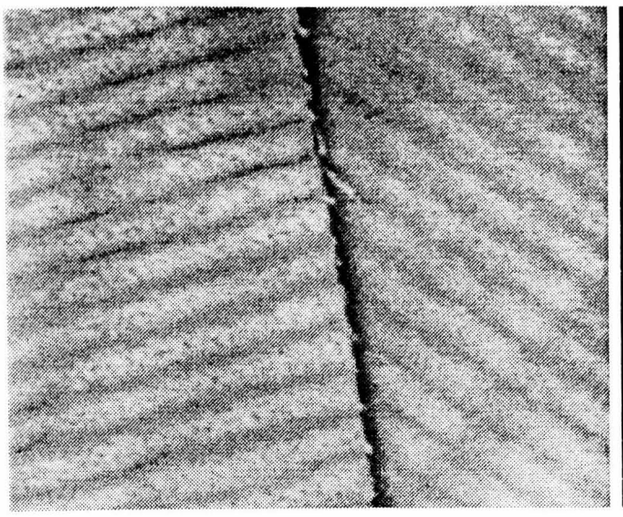


(b). Magnitude image obtained from Sobel edge operators.

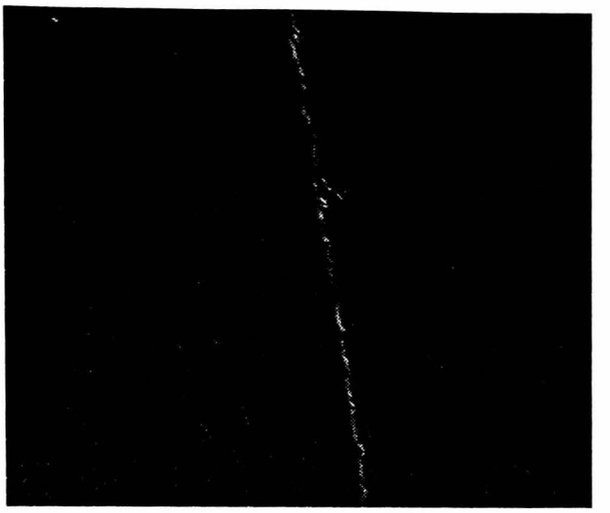


(c). Threshold to produce binary edge image.

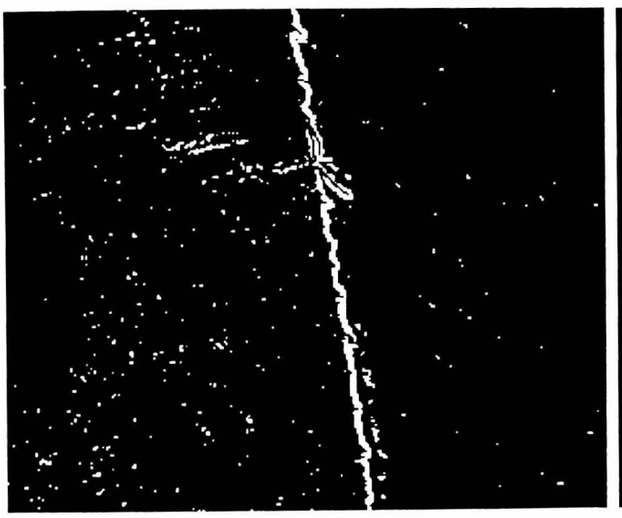
Figure (6.2.2.2). Edge detection in a non-textured image.



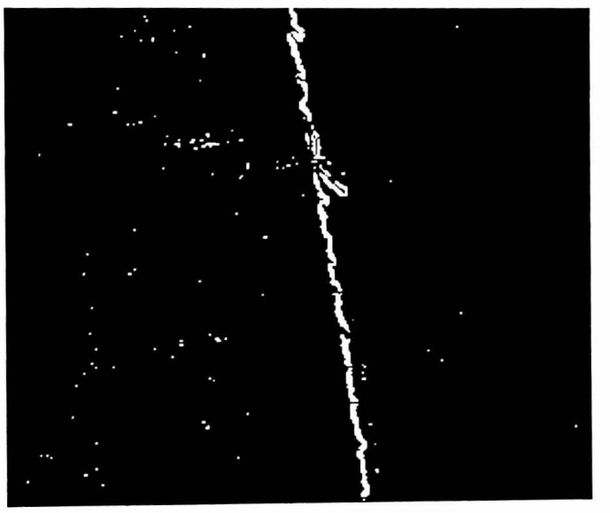
(a). Input image.



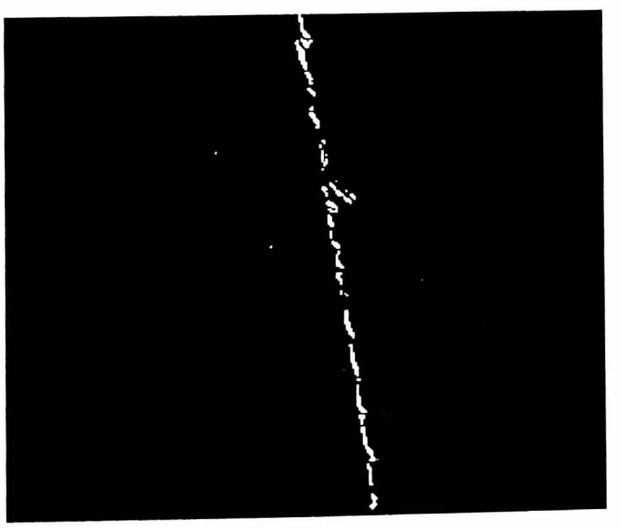
(b). Sobel magnitude image.



(c). Thresholded at 50.



(d). Thresholded at 60.



(e). Thresholded at 70.

Figure (6.2.2.3). Edge detection in a textured image.

This effect of "edge noise" arising from textured images is evident even with the trained edge operators presented in **Section 6.3**. Such an effect is unavoidable with real images, since texture is never entirely homogenous. To overcome this problem, the scheme represented in **Figure (6.2.2.4)** has been adopted.

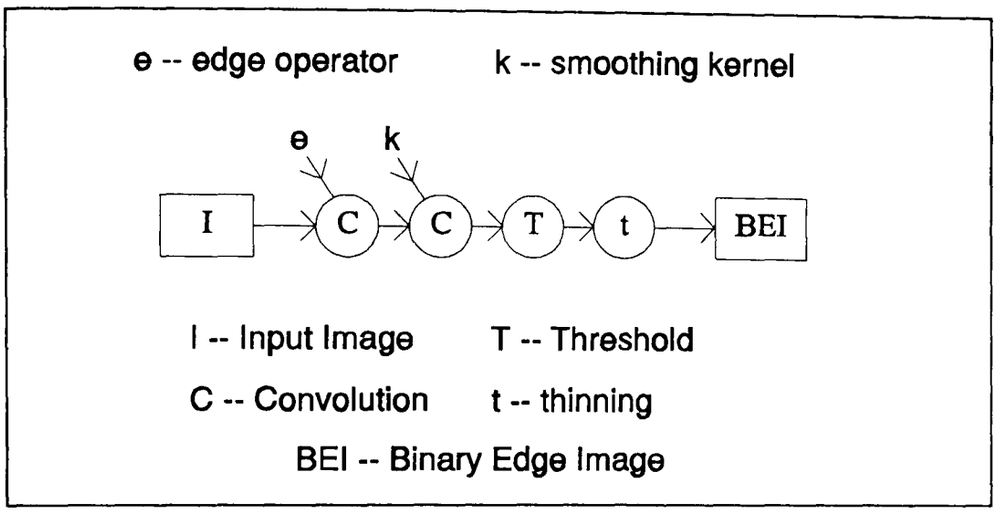
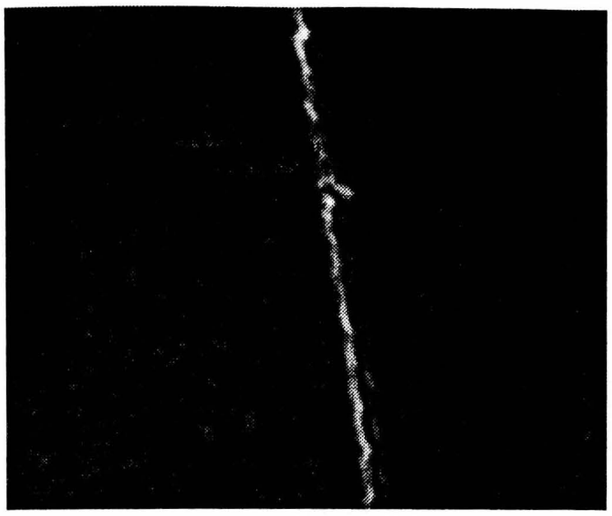


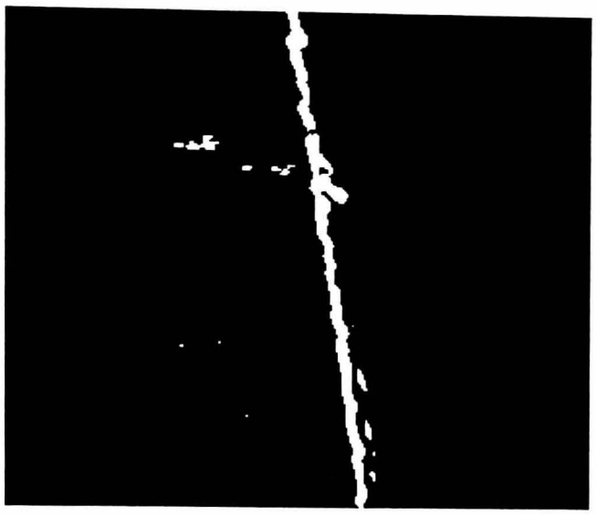
Figure (6.2.2.4). Image processing operations for detection of edges in textured images using optimised convolution masks.

The input image is convolved with the edge operator as before. The resultant image is then smoothed using a Gaussian kernel before being thresholded. This image is then thinned to produce a one pixel wide binary edge image.

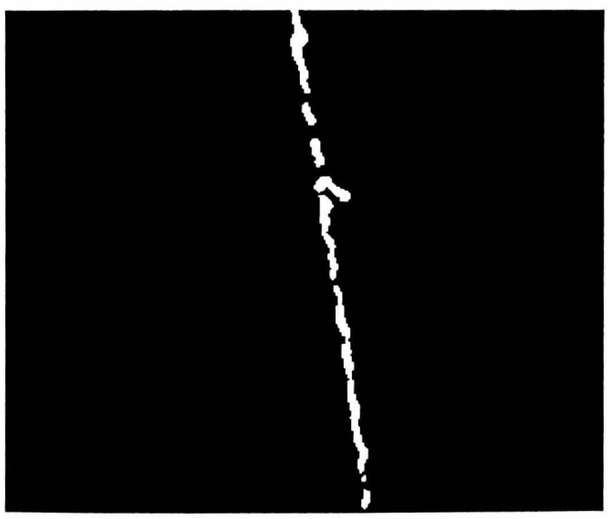
This process is demonstrated in **Figure (6.2.2.5)** using a 5x5 smoothing kernel and **Figure (6.2.2.6)** using a 15x15 smoothing kernel. The effect of smoothing the edge image is to reduce the strength (grey-level) of isolated or weaker edge points but maintain the strength of boundary points where the neighbouring pixels are also strong edges. Once this process has been carried out pixels at or near the boundary can be discriminated from non-boundary pixels fairly well. The resulting (thresholded) image exhibits very thick boundaries due to the attenuation introduced by smoothing, and so a thinning operation is performed to produce pixel wide boundaries. The resulting image is significantly tidier in its representation of the major boundaries present in the original image, and the edge fragmentation is much less than without the extra operations.



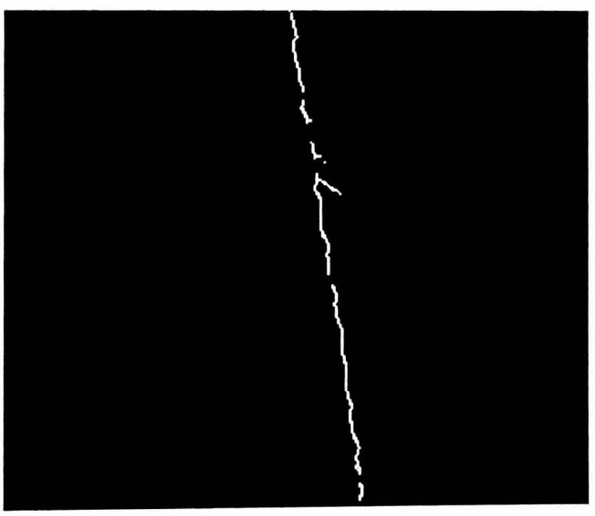
(a). Image (b) of **Figure (6.2.2.3)** after smoothing with a 5x5 Gaussian kernel.



(b). Thresholded at 40.

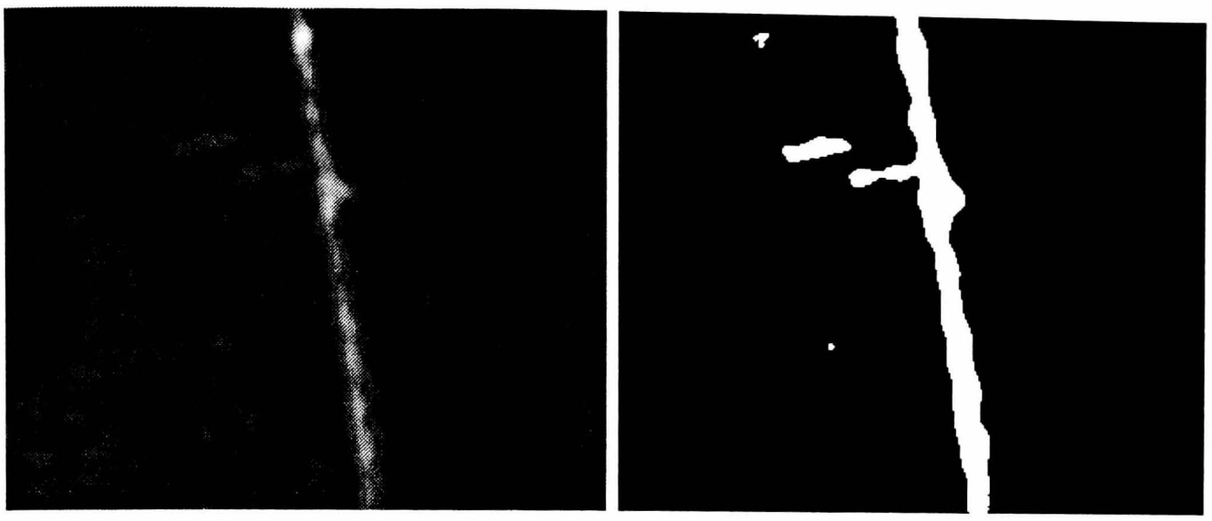


(c). Thresholded at 50.



(d). Image (c) thinned.

Figure (6.2.2.5). The effect on boundary detection of smoothing with a 5x5 kernel.



(a). Image (b) of **Figure (6.2.2.3)** after smoothing with a 15x15 Gaussian kernel.

(b). Thresholded at 30.

(c). Thresholded at 35.

(d). Image (c) thinned.

Figure (6.2.2.6). The effect on boundary detection of smoothing with a 15x15 kernel.

The processing time required for these extra operations (smoothing and thinning) is not a significant problem since both procedures have been implemented using convolution **[King, 1994]**. The process illustrated in **Figure (6.2.2.4)** takes between 0.5 and 1 second, depending on the number of thinning operations required.

Obviously the smoothing and thinning stages are likely to introduce some error into the edge detection process, and so this should really be considered as a means of obtaining a moderately accurate boundary estimate, rather than a means of accurately identifying individual edge points.

6.3 Results.

This section will present the results as a series of example applications for the mask optimisation tool. The performance will be compared to the that achieved using conventional edge operators, such as the rather *ad-hoc* operators shown in **Figure (6.3.1)**.

$$\begin{pmatrix} 2 & 1 & 1 & 2 & 1 & 1 & 2 \\ 3 & 2 & 3 & 4 & 3 & 2 & 3 \\ 4 & 5 & 6 & 8 & 6 & 5 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -4 & -5 & -6 & -8 & -6 & -5 & -4 \\ -3 & -2 & -3 & -4 & -3 & -2 & -3 \\ -2 & -1 & -1 & -2 & -1 & -1 & -2 \end{pmatrix} \quad \begin{pmatrix} 2 & 3 & 4 & 0 & -4 & -3 & -2 \\ 1 & 2 & 5 & 0 & -5 & -2 & -1 \\ 1 & 3 & 6 & 0 & -6 & -3 & -1 \\ 2 & 4 & 8 & 0 & -8 & -4 & -2 \\ 1 & 3 & 6 & 0 & -6 & -3 & -1 \\ 1 & 2 & 5 & 0 & -5 & -2 & -1 \\ 2 & 3 & 4 & 0 & -4 & -3 & -2 \end{pmatrix}$$

Figure (6.3.1). 7x7 horizontal and vertical edge operators.

For each example these masks are applied to the relevant image, the magnitude image produced according to the following equation:

$$g = \max(|g_x|, |g_y|) \qquad (6.1)$$

where $\mathbf{g}_x$ and $\mathbf{g}_y$ are the respective responses of the masks.

Visual Inspection of Denim.

Image **(a)** of **Figure (6.3.2)** shows an image with two pieces of denim,

a circular piece and a larger background piece. The pieces have been aligned to present virtually identical textures. The boundary of the circular piece is more visible at some points than others, due to uneven lighting. If the edge operators shown in **Figure (6.3.1)** are applied to this image, then the result is the image shown in image **(b)** of **Figure (6.3.2)**. The edge operators have managed to enhance the boundary of the circular piece, but there is a very high proportion of background noise contributed by the texture. The best result that can be achieved after this image has been smoothed (15x15 Gaussian), thresholded, and a thinning operation applied is shown in image **(c)**. Note that this result is better than anything achievable using standard 3x3 or 5x5 edge operators.

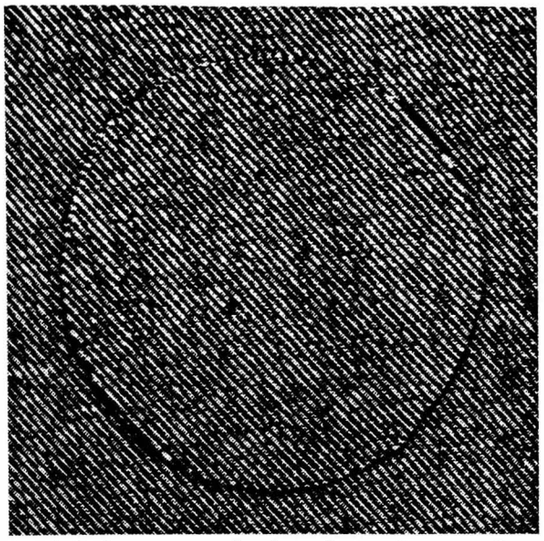
In an attempt to improve upon this performance, the masks shown in **Figure (6.3.3)** were generated using the basis algorithm, with a set of eigenfilters extracted from an image of denim as the basis set.

$$\begin{pmatrix} -94 & 13 & 38 & -20 & -47 & 71 & 31 \\ -49 & 51 & 33 & 6 & -54 & -3 & -14 \\ 2 & -58 & 80 & -4 & -21 & -33 & 19 \\ 74 & 22 & -50 & 45 & 13 & -85 & -77 \\ 96 & 100 & -5 & -63 & 57 & -85 & -57 \\ 60 & 53 & 3 & -42 & 16 & -8 & -68 \\ -33 & 48 & 10 & -18 & 0 & 22 & 25 \end{pmatrix} \quad \begin{pmatrix} 100 & 99 & -11 & -45 & -63 & -73 & 34 \\ 24 & 56 & 81 & 21 & -43 & -29 & 26 \\ 25 & -66 & 89 & 22 & -19 & 72 & -35 \\ -52 & 35 & -13 & -43 & -97 & 55 & -80 \\ 29 & 5 & -2 & 1 & -59 & -88 & -60 \\ 19 & 50 & 38 & 6 & -17 & -82 & 37 \\ 8 & 14 & 99 & 66 & -7 & -78 & -49 \end{pmatrix}$$

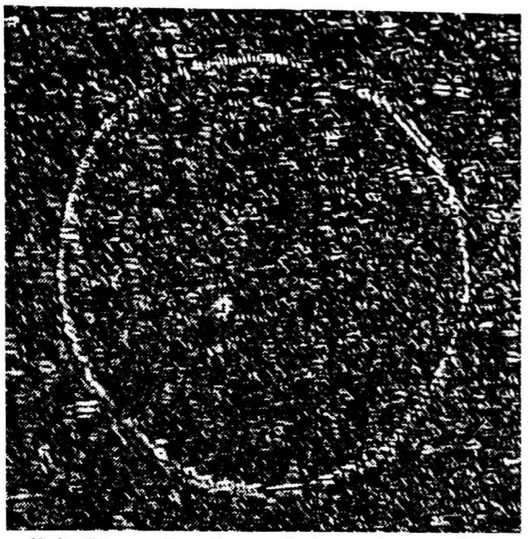
Figure (6.3.3). 7x7 masks optimised to detect horizontal and vertical edges of denim material.

The left-most mask has been trained over horizontal edges in the image and the right-most mask over vertical edges. Both have been optimised to minimise their response to the background texture. The result of applying these masks to image **(a)** in **Figure (6.3.2)** and combining the results into a single magnitude image is shown in image **(d)** of **Figure (6.3.2)**.

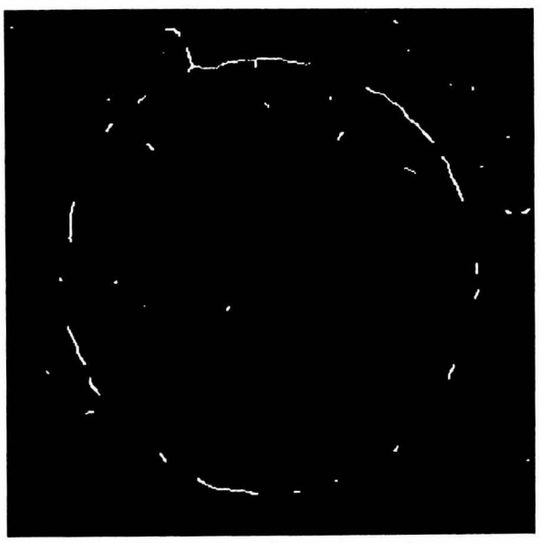
The ratio of edge to background noise is considerably better than that achieved using the standard edge operators. The result after smoothing, thresholding and thinning is shown in image **(e)**.



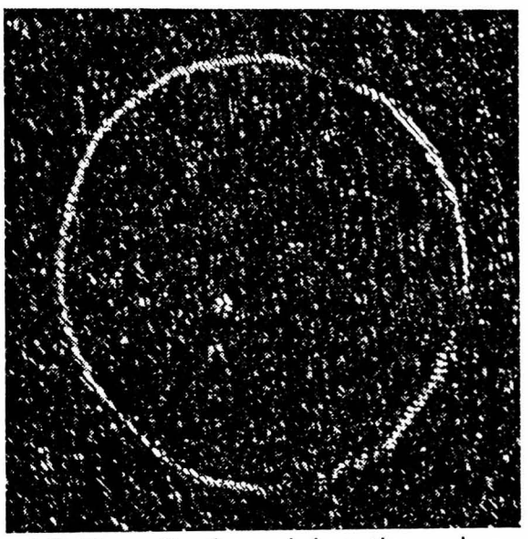
(a) Image of circular piece of denim on a similar background.



(b) Result of applying the edge operators of **Figure (6.3.1)** to (a).



(c) Result of smoothing, thresholding, and thinning (b).



(d) Result of applying the edge operators of **Figure (6.3.3)** to (a).

(e) Result of smoothing, thresholding, and thinning (d).

Figure (6.3.2). Comparison of edge operators on denim image.

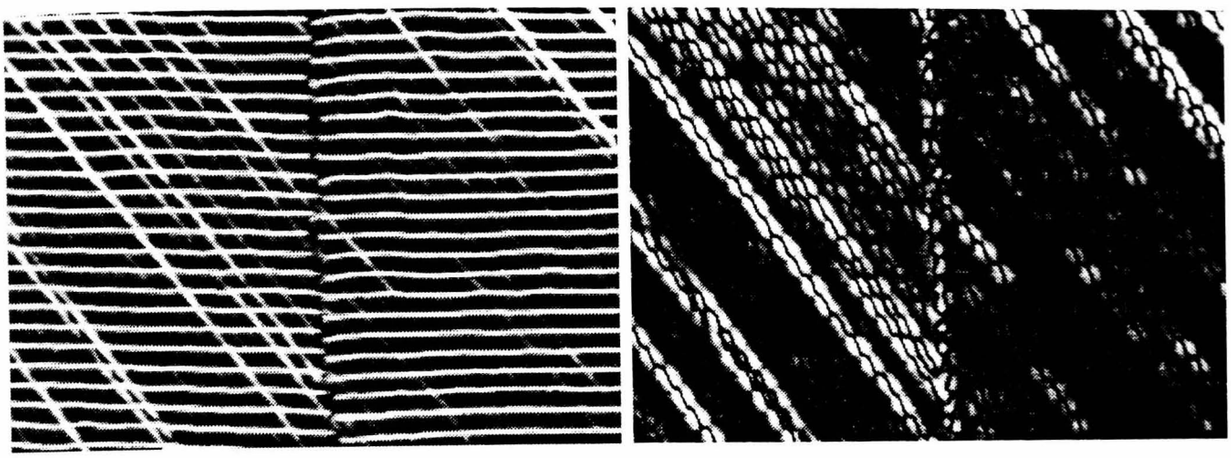
Whilst this result is not perfect, it is a considerable improvement in performance over that offered by conventional edge operators, and would probably suffice for many applications.

Inspection of Carbon Fibre After the Cutting Process.

This example will demonstrate an application where the use of three training regions proves to be particularly advantageous. Image **(a)** of **Figure (6.3.4)** shows an image of two pieces of carbon fibre. The fabric has just been cut, and the boundary between the pieces lies approximately down the centre of the image. This particular material is used in the construction of aircraft fuselage, and is a much thicker material than is normally used for most simple components. The horizontal lines are glass fibre weft. The diagonal lines are also glass fibre, and are intended to help hold the material together. If it were required to find the vertical boundary between these pieces after cutting, then the vertical edge operator in **Figure (6.3.1)** might well be used. The result of applying this mask is shown in image **(b)**. The boundary has in fact been enhanced, and the inherent symmetry of the mask has made it insensitive to the glass fibre weft. The diagonal glass fibre, however, is detected, with the result that the information required to determine the position of the vertical boundary cannot be extracted. Orientation information does not help, since the edges detected relate to the break in the glass fibre weft, and so provide no coherent boundary direction information.

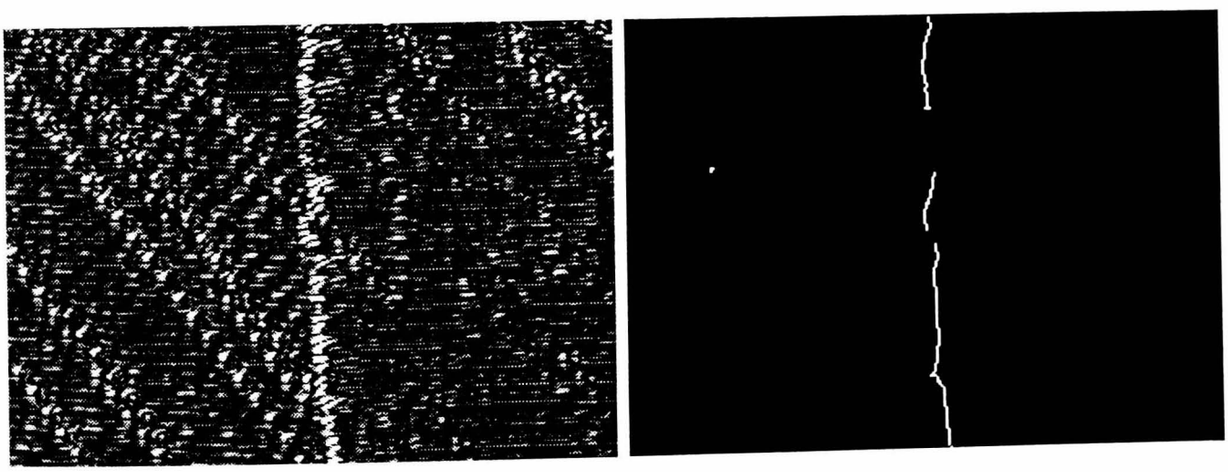
For an application such as this, the basis algorithm can be used in conjunction with three training regions to provide an improvement in performance. A mask can be optimised to enhance vertical boundaries (breaks in the weft) whilst minimising the background texture information (continuous weft) and also specifically minimising a region covering a specific trait of the texture (the diagonal glass fibre). The result of applying a mask trained in such a manner is shown in image **(c)** of **Figure (6.3.4)**.

As can be seen, the mask has not been wholly successful in eliminating the diagonal information, but the ratio of boundary to unwanted texture information is much improved.



(a) Image of carbon material after cutting.

(b) Result of applying vertical edge operator of **Figure (6.3.1)** to (a)



(c) Result of applying optimised edge operator to (a).

(d) Result of smoothing, thresholding, and thinning (c).

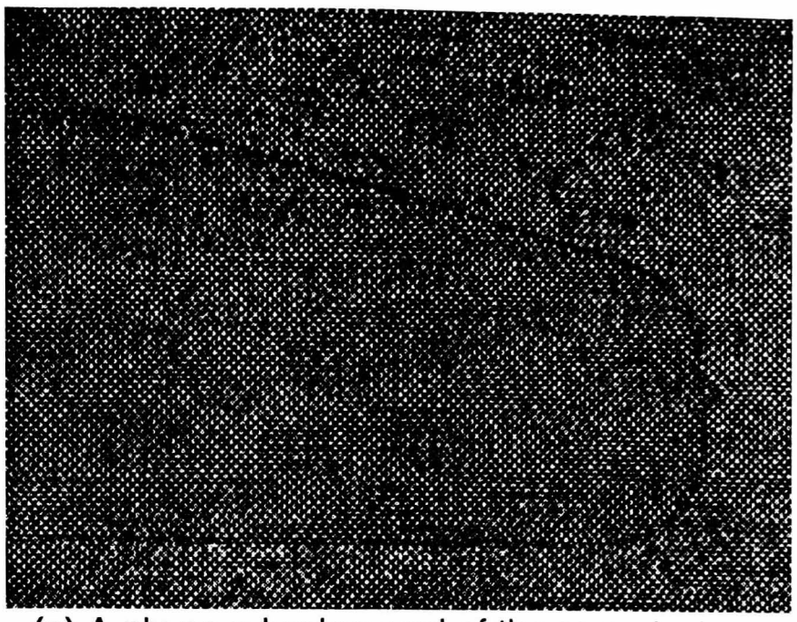
Figure (6.3.4). Detecting the boundary between pieces of carbon material after cutting.

After smoothing, thresholding, and thinning, the boundary is evident as is shown by image **(d)**. It is likely that a larger mask would provide a more robust solution at this image resolution, since such a mask would be better able to eliminate the diagonal information.

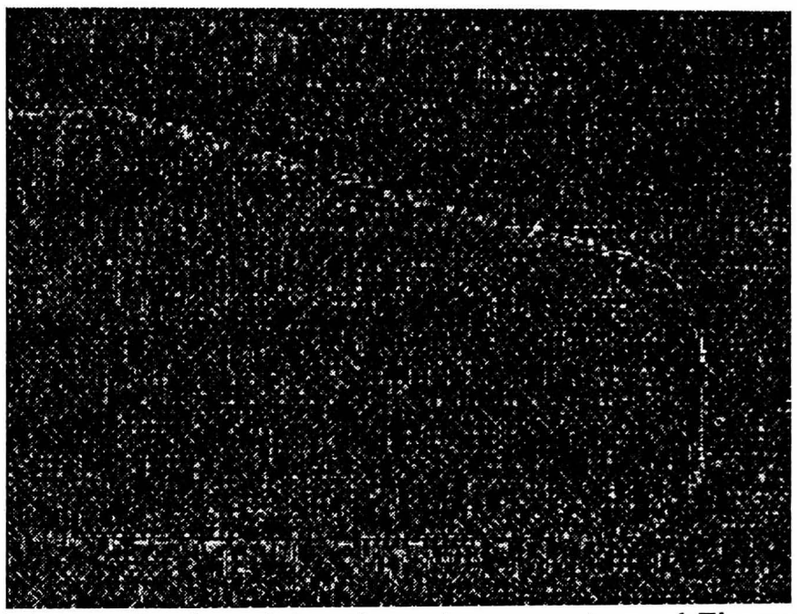
Application in the Robotic Assembly Cell.

Image **(a)** of **Figure (6.3.5)** shows an image of two pieces of carbon fibre. This image has been normalised before printing to improve the observed contrast. No such pre-processing was effected in the experiments. Both the background and the foreground plies in image **(a)** are at 0° orientation, with the result that the boundary is not clearly visible at all points in the image, even to the human eye. The magnitude image obtained using the conventional edge operators is shown in image **(b)** of **Figure (6.3.5)**. As can be seen, images such as this are particularly difficult to process using standard methods.

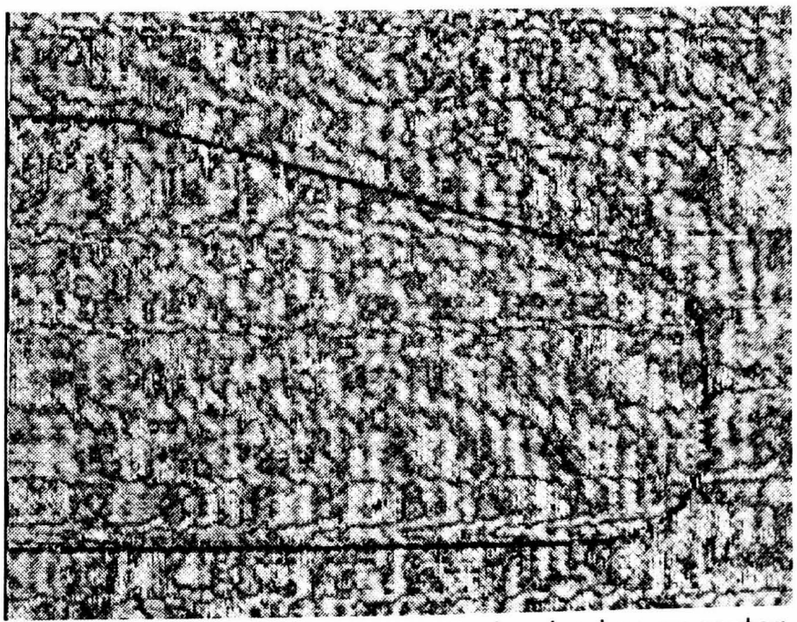
Image **(c)** shows the result of applying a single optimised mask to image **(a)**. This magnitude image is obviously different from any previously presented, in that the boundaries show up as dark areas against the bright areas produced by uniform texture. In all previous examples, masks have been optimised to enhance the boundary information and minimise background texture information. To detect the boundary between identically oriented pieces of carbon fibre, it is sometimes useful to produce masks which minimise the grey-level at region boundaries and maximise it in homogenously textured regions. This has increased the discrimination achieved, since the boundary areas commonly have a lower natural texture energy than the textured regions. The output of a mask optimised in this way can be inverted to produce an image where the edges are enhanced, or alternately any subsequent thresholding stage can be adapted to threshold out each pixel below the specified threshold rather than above. In the examples shown in this section the latter course has been followed.



(a) A ply on a background of the same texture.



(b) Result of applying the edge operators of **Figure (6.3.1)**.

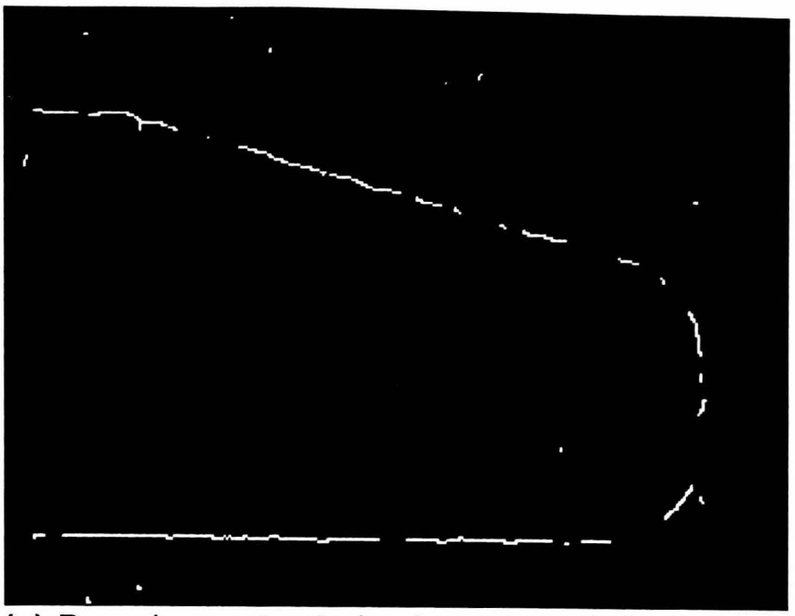


(c) Result of applying an optimised edge operator.

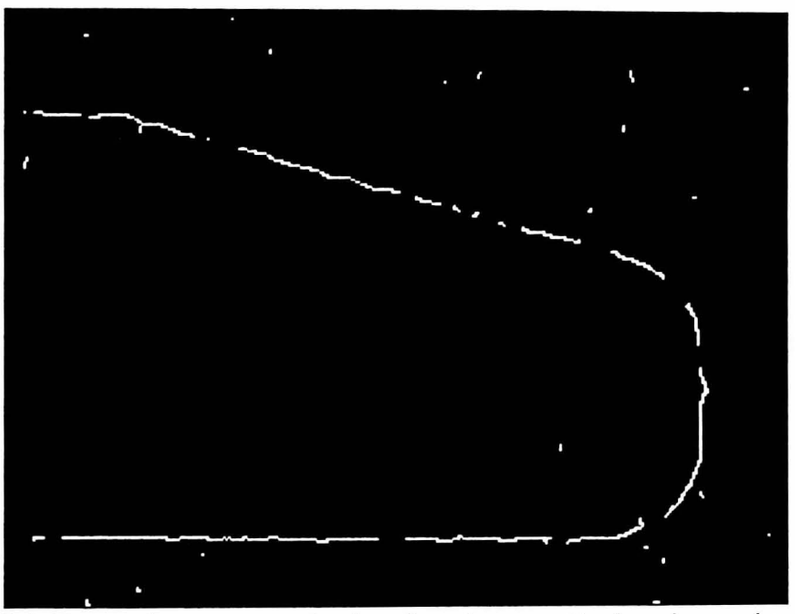
Figure (6.3.5). Comparison of edge operators for carbon fibre image.

Whilst the result shown in image **(c)** of **Figure (6.3.5)** is good, experiments with images taken from the robotic assembly cell have demonstrated that it is useful to optimise masks to be sensitive to more than one edge "type". What is meant by edge type can be explained by reference to image **(a)**. The top boundary shows up as being dark due to uneven lighting, whilst the bottom and vertical boundaries are more difficult to see, and are only really evident as a phase difference in the texture. In each lay-up, there will be some variation in these "phase edges", and so it is useful to train each mask using three training regions, two of which cover different edge types, and one of which covers an area of homogenous background texture. The masks produced have therefore been optimised to be sensitive to a range of edge types and orientations. Not every attempt to produce a mask is successful, since there is a limit to what one mask can be sensitive to, and so a process of trial and error in the positioning of training regions is necessary to produce good masks. Image **(c)** in **Figure (6.3.5)** was produced by a mask trained specifically to detect horizontal phase difference edges, but as the image shows, vertical and dark edges are also quite successfully detected. By combining two suitably optimised masks an even larger percentage of the relevant boundaries in each lay-up can be detected. The percentage of boundary detected can be increased by combining the results of more masks if the additional processing time incurred can be accepted.

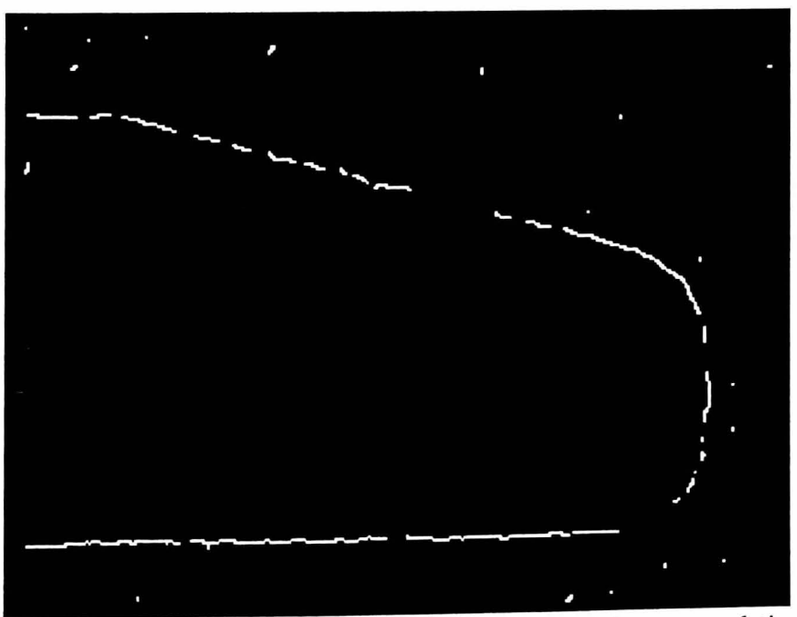
The above points are demonstrated by reference to the images shown in **Figure (6.3.6)**. Image **(a)** shows the boundary which can be extracted after applying one optimised mask to image **(a)** in **Figure (6.3.5)**, and smoothing, thresholding, and thinning the result. Image **(b)** shows the boundary which can be extracted by combining the output of two masks, where the second mask has been trained on vertical edges. In each case the threshold has been determined by thresholding out a predetermined area of the image. This simple technique is adequate for this particular application. It is in fact quite robust, since any slight overestimate of the boundary area results in thicker boundaries, rather than significantly higher noise levels. This effect is then eradicated by the thinning stage.



(a) Boundary extracted using one optimised mask.



(b) Boundary extracted using two optimised masks.



(c) Boundary extracted from a different image of the same ply, using the same two masks used in (b).

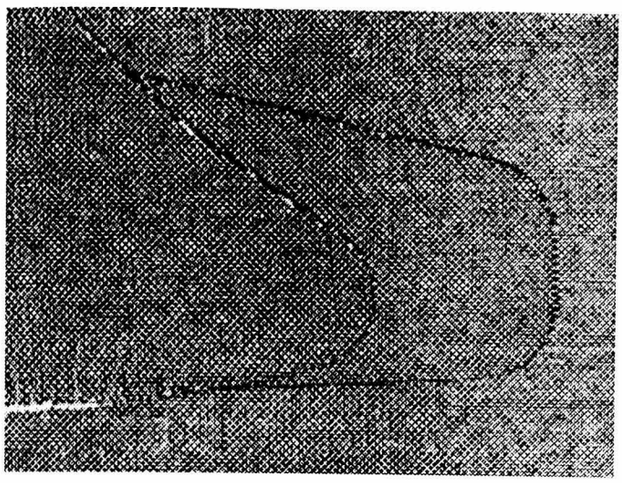
Figure (6.3.6). Boundaries extracted using optimised masks.

The last image, image **(c)** shows the boundary extracted from an image of the same ply after it has been re-laid. The ply has therefore moved slightly, with the result that the profile of the edges will have changed. As can be seen, the boundary is still detected, which shows that the edge operator is useful for real applications where there may be a significant variation from lay-up to lay-up. The edge operator may therefore be regarded as providing robust performance. The thresholding technique used is identical to that used in the other examples.

Finally, **Figure (6.3.7)** demonstrates that the masks developed perform equally well on other images of identically oriented carbon fibre. All the examples in images **(a)** to **(d)** of **Figure (6.3.7)** have been processed using the same parameter for boundary thresholding used in the previous examples. Despite the significant difference between the length of boundaries in these images and those in the previous examples, the thresholds determined by this method produce reasonable results. This is due to the relatively high edge/non-edge ratio achieved by the optimised edge operators.

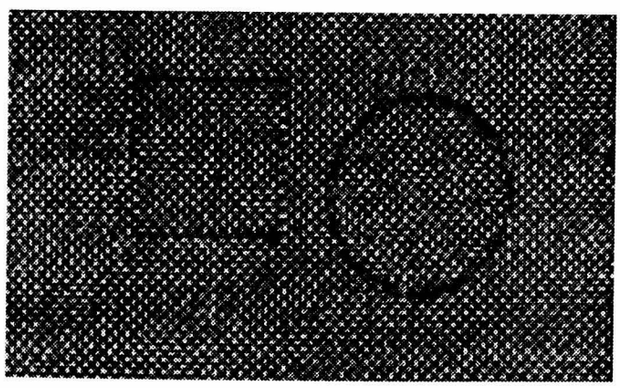
6.4 A Word About Lighting.

Carbon fibre is particularly susceptible to ambient lighting conditions, and in particular to any variation in the relative position of the dominant light source. Any such variation can result in a significant change in the observed texture of the materials. All the examples presented in this chapter have been produced from real images taken from the robotic cell, which does not have a controlled lighting environment. The masks have been tested on various images over a period of time, and therefore have been exposed to some variation in lighting conditions. The results presented are typical of the robust performance obtained. On occasion however, a mask proves particularly sensitive to lighting variation. The precise reason for this is unclear, but it may be due to a temporal lighting fluctuation having been present in the training image used to produce the mask. Since the masks are trained to be insensitive to a particular texture, then any significant change in the *observed* texture of a material will obviously affect the performance adversely.



(a) Two plies on a background of the same texture.

(b) Boundaries extracted using optimised masks.



(c) Two carbon shapes on a background of the same texture.

(d) Boundaries extracted using optimised masks.

Figure (6.3.7). Boundary extraction on other carbon fibre images. The masks and thresholding technique are the same as used to produce the results shown in **Figures (6.3.6)**.

Again it is worth stating that most masks produced are robust to a reasonable variety of lighting variations, and only some masks prove particularly sensitive. Even so, this highlights the importance of controlling the lighting environment in machine vision applications, especially for algorithms which rely on any sort of training.

6.5 Conclusions.

This chapter has demonstrated how masks can be optimised to perform as edge operators in textured images. The main points can be summarised as follows.

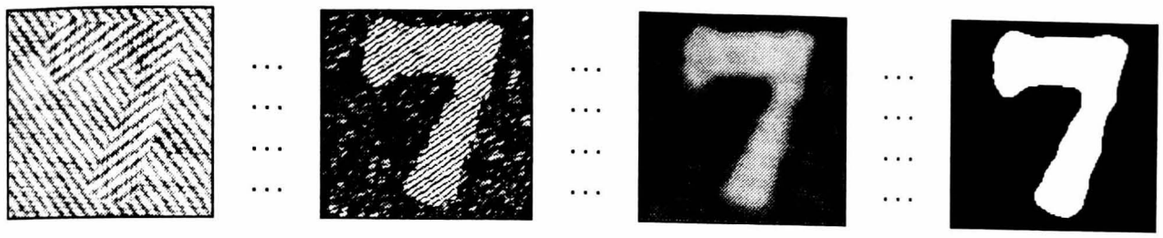
- Optimised masks can provide an improvement in performance over standard edge operators for certain applications.

- Optimised masks may allow simple fast boundary detection techniques, such as those demonstrated here, to be used where previously more sophisticated techniques were necessary.

- The basis algorithm is easily applicable to the task of optimising edge operators for use in textured images.

- The flexibility of the approach is increased by allowing the use of three training regions. For some applications, four or more training regions might be appropriate, but obviously there is a limit to the power of a single convolution mask.

- The lighting plays an important role in the performance of texture edge operators, as it does in any texture application.



CHAPTER 7

Inspection of Composite Preforms Using Texture Based Tools.

7.1 Introduction.

Previous chapters have detailed the development of texture analysis tools designed to be of use in the automated inspection of composite preforms. In this chapter an attempt is made to evaluate these tools for typical lay-up inspection tasks, and to gain some idea of the inspection error involved. This is necessary to establish confidence that these techniques are of potential use in inspection of composite lay-ups.

Inspection errors can be estimated in a theoretical way, or by empirical measurements. For this application a theoretical approach is not feasible since it would require accurate modelling of the textures presented by the carbon materials. This in itself represents a considerable research task. The alternative, empirical measurement of error, must therefore be adopted. For non-rigid materials such as carbon fibre plies however, there are also difficulties with this approach. The basic problem is that of obtaining a yardstick against which to compare the estimates obtained using texture analysis. For a rigid component this can be achieved using a device such as a co-ordinate measuring machine to obtain a very accurate estimate of component

dimensions. The difference between this estimate and the estimate obtained from a nachine vision system gives an accurate indication of the inspection error of the vision system **[Lyvers et al, 1989]**. For inspection of carbon fibre ply edges however, no eadily available mechanism exists for obtaining an accurate estimate of ply edge position. The approach adopted in this chapter therefore is, by necessity, ad hoc and mprecise. However, it is useful in providing an *indication* of the inspection error nherent in texture based inspection of composite preforms.

'.2 Review of the Inspection Requirements.

Composite materials (mainly carbon fibre based) are used to manufacture an ever increasing range of products in the aerospace industry. Composites exhibit ideal properties for aerospace, due to their superior strength-to-weight and stiffness-to-weight ratios in comparison with conventional alloy components. The first stage of composite component manufacture involves laying-up sheets of fibre, usually carbon, as detailed in **Chapter One**. The number and shape of these sheets, or plies, are letermined by the designer using mathematical modelling and CAD tools. The .ccuracy required in the lay-up process is defined in terms of the positional accuracy f ply edges in the lay-up, and is typically of the order of ±1mm (although it is very loubtful that this is actually achieved in manual lay-up). The task of automated nspection of composite lay-ups is therefore essentially one of checking the ply edges re where they are supposed to be. This thesis has detailed two methods for detection f ply edges. **Chapter Three** described how images showing plies of differing naterial/orientation could be segmented using texture analysis. **Chapter Six** showed ow edge operators could be trained to detect boundaries between plies of the same naterial and orientation. Both of these methods result in boundaries which deviate om the true edge by several pixels in places (see for example **Figure (3.7.2)** or **igure (6.3.5)**). Boundary errors of this type are common to any texture analysis nethod, since texture is a neighbourhood property by definition. In fact the egmentation errors produced by a convolution based texture analysis approach are ss than for any other method **[Du Buf et al, 1990]**. It is of course desirable to inimise any error in the inspection process. One way this can be addressed involves e use of boundary models. The remainder of the chapter will attempt to quantify the

nspection error intrinsic when using the texture analysis tools of **Chapters Three** and **Six**.

7.3 The Concept of Subpixel Edge Detection.

To consider how error in boundary estimates can be reduced, it is instructive o consider the concepts which underpin **subpixel edge detection**. Most subpixel dge operators depend on accurately modelling the imaging process. This involves nowledge of both the scene being imaged (edge profiles and/or boundary eometries), and the image formation mechanism itself (CCD sampling, analogue to igital conversion etc). As a result, subpixel edge detection is commonly performed s a two stage process.

In the first stage, knowledge of image formation is used to estimate the position f individual edges to sub-pixel accuracy. This may be accomplished by many different nethods. **[Lyvers et al, 1989]**, **[Koplowitz, Lee, 1991]** fit the local pixel information o an edge model and estimate edge position and orientation more accurately from this nodel. An alternative approach is to treat the pixel values as discrete samples of a ontinuous surface, and so this surface once reconstructed can either be re-sampled t a finer resolution **[Nawla, Binford, 1986]**, or edge estimation performed in the ontinuous domain **[Nomura et al, 1991]**. A third approach involves using second erivative operators, such as Laplacian of Gaussian (LOG). These operators produce zero crossing at an edge location which can be interpolated with a precision epending on the signal to noise ratio of the image **[Huertas, Medioni, 1986]**.

The second stage in many subpixel approaches involves utilising knowledge bout the objects being imaged. In practice this means fitting the edge points to a nodel of the boundary being inspected, usually a straight line or low degree olynomial **[Dorst, Smeulders, 1984]**, **[Klaasman, 1975]**, **[Nakamura, Aizawa, 1984]**. his fitting of edge points to a boundary model can in fact produce subpixel results ven if the edge points themselves have not been estimated to subpixel precision **Havelock, 1989]**. The resultant best fit line or curve is not constrained to interpolate hole pixel values, and so can be used to make measurements to subpixel level.

In this application no suitable method exists to perform the first stage of ubpixel edge detection, the location of individual edge points to subpixel accuracy.

However the second stage, fitting edge points to simple models, does seem to offer some potential. As already mentioned, plies are designed using CAD packages, and in fact are typically constructed of straight lines and arcs. The accuracy of this edge estimate can therefore be improved by fitting the detected boundary points to an appropriate model. Obviously, given the relatively large inaccuracies produced by texture segmentation, subpixel accuracy is not achievable. A moderate increase in overall accuracy should however be realisable.

It is appropriate here to mention image distortion, especially that caused by the lens. An edge which is physically straight (or circular or elliptic etc.) may not appear so when imaged. Such distortion is always present, but if significant can remove the justification for using simple analytic boundary models. An attempt was made to establish the distortion of the camera/lens system used in these experiments by measuring the dimensions of a small machined part when placed in different areas of the image (i.e. centre, top corner, bottom corner etc.). The images were examined in magnified form (using 8x magnification), and no variation in dimension was observed. This would seem to indicate that the distortion of the lens used in these experiments is slight i.e. less than one pixel across the field of view. Distortion will however still play a part in the results obtained. In the work presented here no attempt has been made to compensate for this. From the literature it would seem that ignoring the effect of distortion is common practice.

The remainder of this chapter will describe the boundary modelling approach adopted for both straight and curved ply edges, and present the results of experiments carried out in an effort to gauge the accuracy achievable from this method of inspection.

7.4 Inspection of Straight-Edged Plies.

This section will detail the method employed to estimate the error present when inspecting lay-ups consisting of straight edged plies. In fact many composite components in the aerospace industry are manufactured from plies exhibiting only straight edges. It is only more complex components, such as propeller blades, which require more complicated ply shapes. Inspection error for straight-edged plies is therefore of considerable interest. A common method for fitting boundary points to a

straight edge model is described in the following section.

7.4.1 Least Squares Fitting to a Straight Line.

Given a set of data points to be fitted to a straight line, the least squares method states that the line

$$y = a + bx \qquad (7.1)$$

should be fitted through the given points $(\mathbf{x}_1,\mathbf{y}_1)$, ..., $(\mathbf{x}_n,\mathbf{y}_n)$ so that the sum of the squares of the distances of those points from the straight line is minimum. The usual practice is to measure error in the vertical direction (the **y**-direction), as shown in **Figure (7.4.1.1)**. This assumes that the **x**-direction variable is independent i.e. there is no uncertainty in the **x**-direction. This assumption is generally not valid in image processing applications, where there is uncertainty in both **x** and **y** directions. In such a case the error should be measured as the perpendicular distance from a point to the line. This can be accomplished using a different form of the line equation, namely

$$ax + by + 1 = 0 \qquad (7.2)$$

If, however the data points define a line which is close to horizontal, then the representation of equation **(7.1)** in conjunction with the standard least squares formulation is appropriate i.e. the vertical distance measure is very close to the perpendicular distance. Similarly for vertical lines, then **y** can be treated as independent, and the horizontal distance from point to line taken as an error measure. For the sake of simplicity, attention is restricted to the inspection of horizontal edges, and so it is appropriate to make use of equation **(7.1)** and the standard least squares formulation. Such an approach is also justified by the fact that checking of horizontal edges is sufficient for some lay-up inspection tasks.

For a line defined as in equation **(7.1)**, a point on the line with **x**-coordinate $\mathbf{x}_j$ has the **y**-coordinate equal to $\boldsymbol{a + bx_j}$.

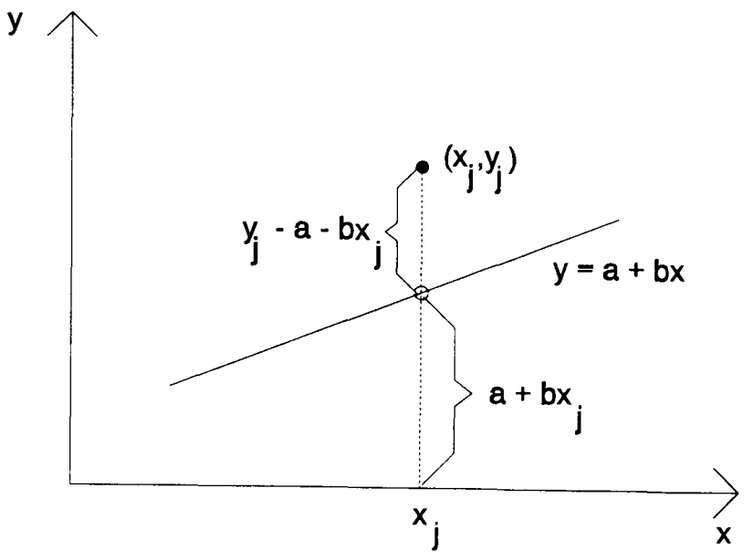


Figure (7.4.1.1). Vertical distance of a point (x_j,y_j) from a straight line $y = a + bx$.

Hence its distance from the data point $(\mathbf{x_j},\mathbf{y_j})$ as shown in **Figure (7.4.1.1)** is

$$|y_j - a - bx_j| \qquad (7.3)$$

ınd the sum of the squares in the vertical direction is

$$q = \sum_{j=1}^{n} (y_j - a - bx_j)^2 \qquad (7.4)$$

vhich will be a minimum when the line best fits the data in the least squares sense. 'he general approach taken to solving equation **(7.4)** is based on the realisation that vhen **q** is minimum, then the first derivative of **q** (i.e. rate of change of **q**) will equal :ero. Since **q** depends on both **a** and **b**, then the partial derivatives of **q** with respect ɔ both **a** and **b** must be considered. These must both equal zero for a minimum. i.e.

$$\frac{\partial q}{\partial a} = -2\sum (y_j - a - bx_j) = 0 \qquad (7.5)$$

$$\frac{\partial q}{\partial b} = -2\sum x_j(y_j - a - bx_j) = 0$$

umming over **j** from 1 to **n**. Equation **(7.5)** can be rearranged to form

$$an + b\sum x_j = \sum y_j \qquad (7.6)$$
$$a\sum x_j + b\sum x_j^2 = \sum x_j y_j$$

hese equations can be solved explicitly for **a** and **b** by Cramer's rule **[Kreysig, 1988]**. iven this approach to boundary modelling, what inspection error is present?

.4.2 A Framework to Estimate Inspection Error.

In order to estimate the error of ply inspection, it is necessary to have some ardstick to measure against, i.e. an accurate estimate of the "true" position of the ply. he determination of this is no trivial task. Initially it was thought that the lay-up robot ould be used to help estimate inspection error. The basic idea is that a ply would be aid-up on a white background (achieved by covering the lay-up table in paper) so nabling easy detection of ply edge. The robot would then remove the ply, wait for a ew seconds while a background ply was laid-up by hand, and then re-lay the ply in ne original position. The edge of the ply against the background ply could then be etermined using texture analysis or textured edge detection as appropriate, and the esult compared to the measurement made when the ply was laid-up against the white ackground. The difference in the two measures would be taken as the error in the exture analysis or textured edge operator estimate. However the basic assumption nderlying this scheme, that the robot could be relied upon to consistently lay-up a ply n (virtually) the same position, proved to be false. Experiments were conducted hereby a ply was continually laid-up on the white background and inspected by the ision system using the horizontal Sobel edge operator and least squares line fitting o obtain an accurate estimate of the position of one horizontal edge. The lay-up rocess was found to move the position of the ply by as much as 1mm per cycle. This s mainly due to the peeling action evident on ply release from the gripper device, and as verifiable by human observers. After 50 cycles the ply had moved by up to 40mm om its original position. Although this movement is consistent in its direction, there s a large variation in the magnitude of movement, and so it is not possible to use the

robot to provide a reliable yardstick of the true position of the ply. However these experiments have proven useful in that they have identified the intrinsic inaccuracy in the automated lay-up process. Investigation of the causes of this inaccuracy is the realm of other researchers.

Due to the problems in determining inspection error using the robot, a more reliable method has been devised. This is described as follows, and is illustrated in **Figure (7.4.2.1)**.

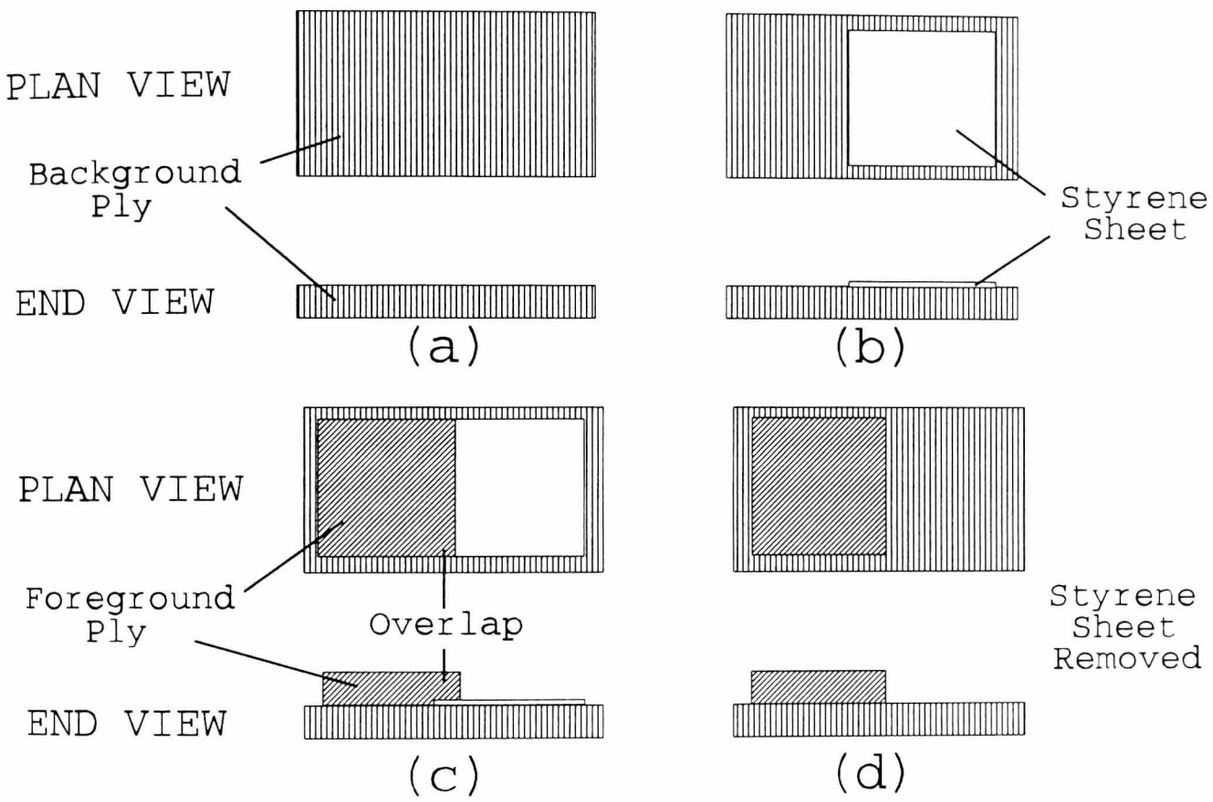


Figure (7.4.2.1). The right-most edge of the foreground ply can be detected by edge operators if a white sheet of styrene is used to give contrast. With the sheet removed, the results using texture can be obtained and compared.

A background ply is laid-up on the table ((a) in **Figure (7.4.2.1)**). A thin white sheet of styrene (chosen because it is flexible with a low coefficient of friction, but any similar material will suffice) is placed on top of the ply ((b) in **Figure (7.4.2.1)**). A foreground ply is then laid-up such that there is a very slight overlap (2 or 3mm) between the foreground ply and the styrene sheet ((c) in **Figure (7.4.2.1)**). Since there is a strong contrast between the black foreground ply and the white styrene background, the ply edge position can be estimated to subpixel precision by applying a conventional gradient edge operator (such as Sobel), thresholding at an appropriate level, and

performing a least squares fit to a straight line. Variations within threshold selection, within certain bounds, were found to have negligible effect on the fitted edge. This is due to the fact that the rectified output of the Sobel operator on either side of the ply/background edge is largely symmetric, and so increasing/decreasing the threshold will eliminate/add an approximately equal number of points on either side of the true edge position. The process of applying the edge operator and fitting the resulting points to a straight line is repeated over 50 images and the mean values of **a** and **b** taken to define the line representing the real ply boundary position. The required yardstick has now been obtained, i.e. an accurate estimate of the position of the ply edge in image space. The styrene sheet can now be removed ((d) in **Figure (7.4.2.1)**). Since the overlap is very slight and styrene has a very low coefficient of friction, the position of the plies is unaffected. The position of the foreground ply is then estimated using texture analysis or a textured edge operator as appropriate, and a least squares line fitting operation performed on the resultant edge points. The difference between the line obtained from texture analysis and the line obtained when the styrene sheet is present is taken as the error in the texture-based inspection process. The distance is measured between the two best fit lines in the vertical direction, which is again justified by the fact that both lines are very close to the horizontal. Distance is measured at two points (sufficient for a straight line), one chosen to be near the left extremity of the relevant ply edge, and the other near the right extremity, as shown in **Figure (7.4.2.2)**. For each point an **x**-coordinate is chosen, and this is then used to generate the corresponding **y**-coordinates for each respective line. The difference between the two **y**-coordinates is the distance from one line to the other (i.e. from the texture estimate to the estimated "true" position). A subpixel estimate of inspection error can be obtained using this process. This can be converted to absolute error if the pixel dimensions are known. A simple way to accomplish this is to image a steel ruler and so measure the number of millimetres the image covers in both the horizontal and vertical directions. The dimension of a pixel can then be calculated. This approach to calibration, although ad-hoc in character, is commonly used in industrial inspection applications.

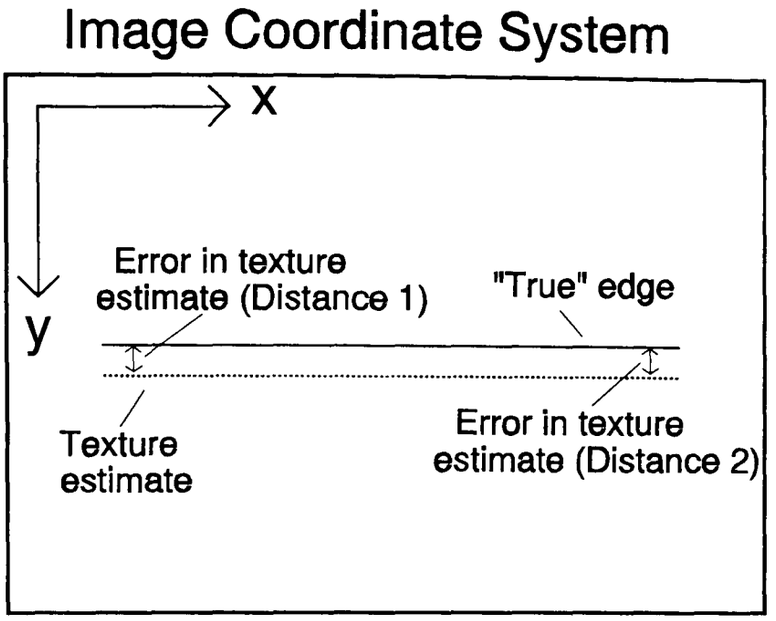


Figure (7.4.2.2). The difference between the texture estimate and the "true" edge is taken as the inspection error.

7.4.3 Parameters Affecting Inspection Error.

The inspection error is dependent on a number of variables which can be nvestigated. These are discussed as follows.

The Number of Textures to Analyse: There are three possible ways texture analysis :an be used to process the image shown in **Figure (7.4.3.1a)** in order to estimate)oundary position.

(1) Detect the texture of the foreground ply and determine the boundary from this (segmented image shown in **Figure (7.4.3.1b)**).

(2) Detect the texture of the background ply and determine the boundary from this (segmented image shown in **Figure (7.4.3.1c)**).

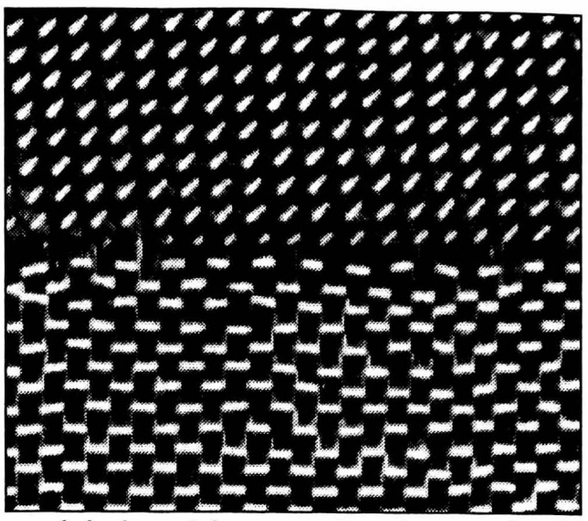
(3) Perform processes **(1)** and **(2)** (i.e. analyse both textures) and form a boundary by combining the two resultant boundaries.

This combining of boundaries would, using most methods of boundary epresentation, be a somewhat intricate and time consuming process. However since

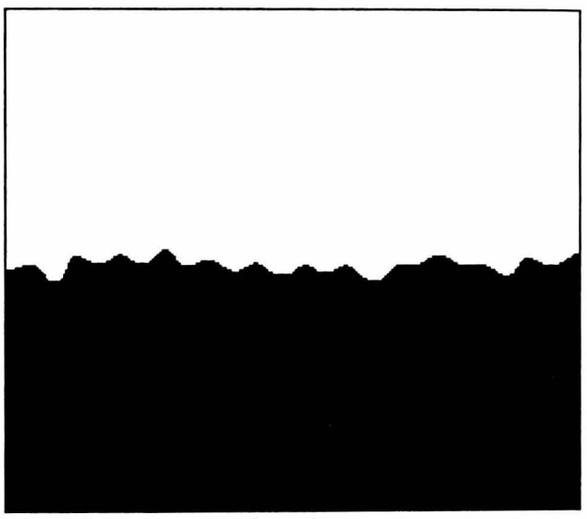
least squares fitting to boundary points is being performed, an elegant solution to the problem presents itself. The relevant boundary points from both texture boundaries are passed to the fitting algorithm and the best fit line over *all* the points is found. This implicitly performs the required combining of boundaries with almost no increase in computation time.

Image Resolution: An intuitive notion to reduce inspection error is to image the lay-up at higher resolution so that each pixel represents a smaller area. A one pixel error will therefore represent a smaller absolute error. That is, if each pixel represents an area of $1mm^2$ of the lay-up table, then a one pixel error represents an absolute error of 1mm. If magnification is increased so that each pixel represents an area of $0.5mm^2$ of the lay-up table, then a one pixel error represents an absolute error of 0.5mm. The reality however is not so simple. The most obvious drawback is that increasing the resolution means that more frames are required to inspect the same area, with a corresponding increase in processing resources. More importantly, image processing techniques which perform well at low resolution may well be unsuitable for higher resolution images. Indeed many techniques are implicitly designed for low resolution imaging. The most obvious example is convolution-based edge detection. The 3x3 Sobel mask is designed to detect edges in an image, the implicit understanding being that edges are of the order of 1 or 2 pixels wide. For high resolution images, an edge profile may be far wider. Considerable post-processing would be required to extract the desired information from such an image. In theory the answer is to use larger edge operators, but this imposes a huge computational load and is usually not feasible. Choosing a suitable image resolution can therefore be a trade-off between computational simplicity and acceptable inspection error. For this application this raises two main questions with respect to increasing image resolution: are the methods so far developed able to cope, and what is the effect on inspection error?

Post-Processing of Texture Boundaries: The effect on inspection error of four post-processing operations has also been investigated. The first, iterative fitting, applies to boundaries obtained by either texture analysis or by texture edge operator.



(a). Input image showing a ply boundary. Foreground ply at 90°, background at -45°.



(b). Segmentation based on texture analysis of foreground ply.

(c). Segmentation based on texture analysis of background ply.

Figure (7.4.3.1). Boundary detection by texture analysis.

The second and third, secondary smoothing and boundary sampling respectively, apply only to boundaries obtained using texture analysis. The fourth, boundary thinning, applies only to boundaries obtained using texture edge operators. These operations are described in more detail below.

(1) Performing an iterative least squares fitting process. This involves forming the best fit line in the normal way, and then calculating the distance from each point to the line. Points over a certain threshold are disregarded and a new line formed from the remaining points. This process is performed for a preset number of iterations.

(2) Secondary smoothing of texture boundaries before best line fitting. This process was described in **Section 3.8**, and was found there to reduce segmentation error on artificially constructed texture images which exhibited low boundary curvature.

(3) For texture boundaries, performing a sampling operation and forming the best fit line from the sampled points. This might be expected to reduce variance in the boundary points used in line fitting, and therefore produce a more accurate result.

(4) Thinning of edges resulting from texture edge operators. When texture edge operators are used, the resultant image is smoothed and thresholded to produce edges which are many pixels wide. These edge points can themselves be used to form the best fit line, or a thinning operation can first be performed.

7.4.4 Results Using Texture Analysis.

Tests were carried out using a variety of plies and material orientations (i.e. 0°/±45°/90°). For each lay-up configuration ten lay-ups were performed. This is a small sample size, but due to the expensive nature of the materials, and the time taken to perform the tests, this was deemed acceptable. Each lay-up was inspected ten times with each combination of parameters (i.e. 10x10 images processed for each combination of parameters) and the results

noted for each test. Results are presented in tabular form. **Table 7.1** gives the inspection results for 0°/90° ply lay-ups using texture analysis, with all results shown in pixels. The distance between the estimate of the "true" boundary and the boundary obtained by texture is measured at two points (Distance 1 and Distance 2 in **Table 7.1**) as described in **Section 7.4.2**. For each set of measures, the mean (μ) and the standard deviation (σ) were calculated, and the minimum and maximum values recorded. This process was repeated using iterative fitting, secondary smoothing (as described in **Section 3.8**), and boundary sampling, as well as combinations of these. Results for secondary smoothing with iterative fitting, and secondary smoothing with boundary sampling, are not included since the addition of secondary smoothing to these processes produced no significant change in the results.

Firstly the results obtained with no post-processing of texture boundaries are considered (i.e. no iterative fitting, secondary smoothing, or boundary sampling). These results appear in the first two rows of **Table 7.1**. The most obvious result is that the accuracy of boundary estimate is increased by analyzing both textures. This can be appreciated more easily by considering **Figure (7.4.4.2)** which illustrates the relative orientational and translational error more clearly than the raw data.

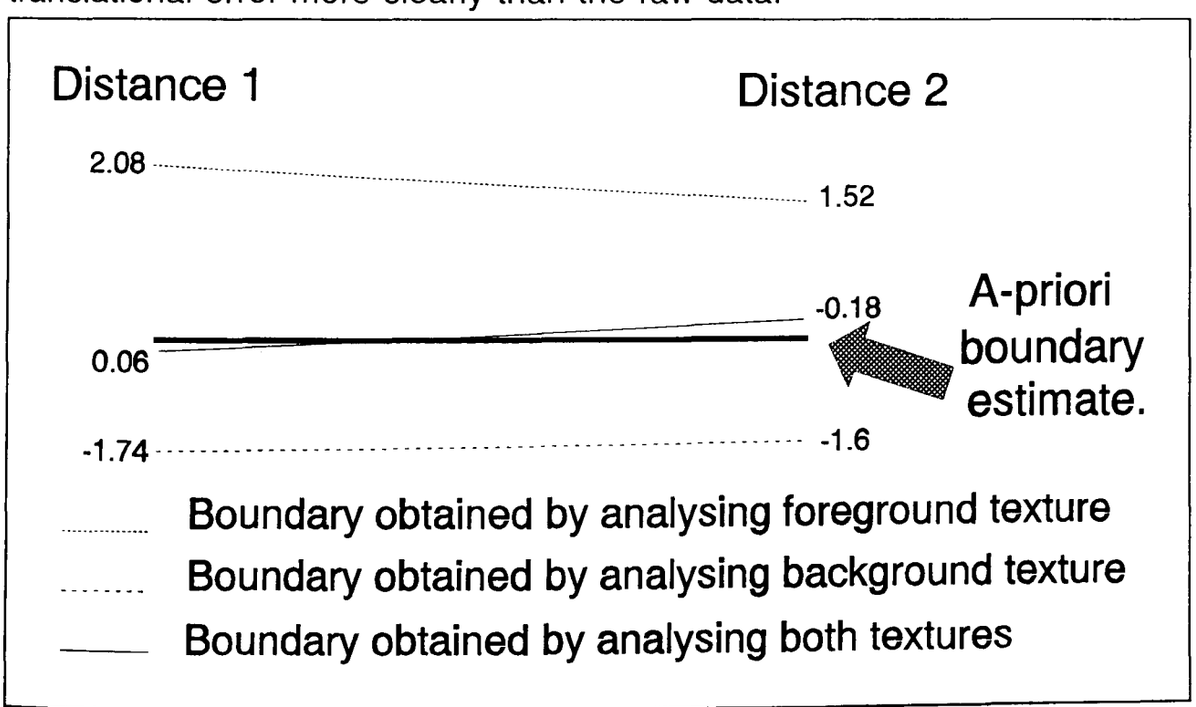


Figure (7.4.4.2). Illustration of the boundary estimates presented in the first two rows of **Table 7.1**. The figures denote the deviation in pixels from the expected ply edge.

Straight Edge 0°/90° 1 pixel is 0.61x0.59 mm^2	Options	Textures Analysed		
		Foreground	Background	Both
Distance 1		μ = 2.08 σ = 0.11 min = 1.88 max = 2.24	μ = -1.74 σ = 0.21 min = -1.37 max = -2.16	μ = -0.06 σ = 0.06 min = -0.02 max = -0.20
Distance 2		μ = 1.52 σ = 0.08 min = 1.37 max = 1.65	μ = -1.60 σ = 0.15 min = -1.36 max = -1.78	μ = -0.18 σ = 0.08 min = -0.10 max = -0.37
Distance 1	1	μ = 1.60 σ = 0.06 min = 1.47 max = 1.67	μ = -1.87 σ = 0.12 min = -1.68 max = -2.08	μ = -0.16 σ = 0.17 min = -0.09 max = -0.42
Distance 2	1	μ = 0.67 σ = 0.10 min = 0.43 max = 0.76	μ = -1.22 σ = 0.23 min = -0.70 max = -1.52	μ = -0.03 σ = 0.15 min = 0.01 max = 0.27
Distance 1	2	μ = 3.00 σ = 0.15 min = 2.65 max = 3.24	μ = -3.02 σ = 0.11 min = -2.77 max = -3.14	μ = -1.01 σ = 0.07 min = -0.89 max = -1.16
Distance 2	2	μ = 2.86 σ = 0.15 min = 2.59 max = 3.05	μ = -3.06 σ = 0.15 min = -2.86 max = -3.36	μ = -0.05 σ = 0.13 min = 0.01 max = -0.31
Distance 1	3	μ = 1.56 σ = 0.3 min = 0.83 max = 1.94	μ = -1.75 σ = 0.19 min = -1.46 max = -2.05	μ = -0.06 σ = 0.12 min = 0.01 max = -0.31
Distance 2	3	μ = 1.43 σ = 0.1 min = 1.32 max = 1.62	μ = -1.46 σ = 0.08 min = -1.37 max = -1.62	μ = 0.26 σ = 0.09 min = 0.14 max = 0.39
Distance 1	1+3	μ = 2.08 σ = 0.15 min = 1.9 max = 2.35	μ = -2.07 σ = 0.03 min = -2.03 max = -2.11	μ = -0.31 σ = 0.60 min = 0.05 max = -1.43
Distance 2	1+3	μ = 2.16 σ = 0.14 min = 1.97 max = 2.35	μ = -0.68 σ = 0.14 min = -0.57 max = -0.99	μ = 0.25 σ = 0.30 min = 0.10 max = 0.74
Distance 1	1+2+3	μ = 2.61 σ = 0.30 min = 2.43 max = 2.88	μ = -3.04 σ = 0.42 min = -2.48 max = -3.51	μ = -3.66 σ = 0.59 min = -2.71 max = -4.57
Distance 2	1+2+3	μ = 3.10 σ = 0.19 min = 2.70 max = 3.36	μ = -2.65 σ = 0.05 min = -1.52 max = -2.72	μ = 0.70 σ = 0.46 min = 0.26 max = 1.66

Table 7.1

Key: 1 - Iterative fitting, 2 - Secondary smoothing, 3 - Boundary sampling.

Note that this is merely an illustration and is not to scale. The actual length of the boundaries here is approximately 400 pixels, so the orientational deviation is grossly exaggerated.

It is clear from this diagram that the boundary from texture 1 lies above the real boundary, whilst that of texture 2 lies below. By forming the best fit over all points (i.e. combining the boundary information from both plies) then a more accurate result is obtained. This is generally true for all the lay-up tests performed. This seems to be due to the fact that ply edges present areas of texture different from both foreground and background plies. In the foreground ply this is due to the disturbance to the edge that the cutting operation produces. In the background ply it seems to be due to the slight shadowing effect near the edge of the foreground ply. As a result the estimates of foreground and background ply position tend to flank the area occupied by the ply boundary. Combining the foreground and background boundary estimates therefore results in a more accurate ply boundary estimate. For these lay-ups the mean deviation from the real edge is 0.06 pixels at the left edge, and -0.18 pixels at the right edge. More importantly, the maximum deviation measured (which really specifies the confidence interval of inspection accuracy) is 0.37 pixels. Unfortunately this result does not fully reflect the situation when inspecting lay-ups, as will be seen when the results for other lay-ups are considered.

The next two rows of **Table 7.1**, marked with a "1" in the "options" column give the results obtained by performing an iterative least squares fit to the boundary points. The only significant difference this has introduced to the results is that the variance of boundary estimates is increased (standard deviation = 0.17 for Distance 1 and 0.15 for Distance 2), and accordingly the maximum deviation is larger at 0.42 pixels.

The two rows marked with a "2" in the options column give the results obtained when secondary smoothing is applied to the texture boundaries before least squares fitting. From the table it can be seen that this process significantly increases the error of the boundary estimate. This is an interesting result, since the opposite effect was observed in **Section 3.8** in tests with artificially constructed texture images with low curvature boundaries. The reason for this discrepancy is not altogether clear, but possibly the real edges exhibit a higher degree of curvature than the artificial boundaries of the test

imagery. Whatever the reason this result illustrates the danger of limiting testing to a few artificial images. The data is much easier to measure and analyse for such images, but the results are not necessarily readily extendible.

Sampling of texture boundaries, marked "3" in the "options" column has little effect with these lay-up configurations other than to increase the standard deviations. The results shown were obtained by taking every fifth boundary point, which was derived empirically as providing good performance.

The last four rows of the table show results obtained by combining the various boundary post-processing procedures. The rows marked "1+3" in the options column are obtained by sampling the texture boundary (again every fifth point was taken), and then performing an iterative best fit to these points. The most notable effect is to reduce the reliability of the boundary estimate (σ = 0.60 and 0.30) and increase the maximum deviation (1.43 pixels). By combining all three operations (secondary smoothing, boundary sampling, and iterative fitting) this effect is increased, with σ = 0.59 and 0.46, and maximum deviation = 4.57.

The general conclusions to be drawn from **Table 7.1** are as follows. A more accurate estimate of boundary position is obtained if both foreground and background plies are analysed. Post processing of texture boundaries at this image resolution tends to produce less consistent results, and therefore reduces the reliability of boundary estimates. The best result was obtained using no post-processing, and gave a maximum deviation from the real edge of 0.37 pixels. Since the vertical pixel dimension is 0.59mm at this resolution, then this converts to a maximum error of 0.23mm. If this data were universally representative the conclusion would be that, at this image resolution, ply boundary position for straight-edged plies could be estimated to approximately +/-0.25mm. Unfortunately this is not the case.

Table 7.2 gives the inspection results for lay-ups where the foreground ply is +45° and the background ply 0°. The overall conclusions which can be drawn from this data are largely the same as from **Table 7.1** but the deviations from the real edge position are larger.

Straight Edge 45°/0° 1 pixel is 0.61x0.59 mm²	Options	Textures Analysed		
		Foreground	Background	Both
Distance 1		μ = 0.366 σ = 0.09 min = 0.27 max = 0.54	μ = 1.87 σ = 0.09 min = 1.74 max = 1.99	μ = 0.99 σ = 0.10 min = 0.82 max = 1.15
Distance 2		μ = -2.77 σ = 0.14 min = -2.49 max = -2.97	μ = -0.29 σ = 0.15 min = -0.02 max = -0.47	μ = -1.59 σ = 0.10 min = -1.38 max = -1.74
Distance 1	1	μ = 1.12 σ = 0.26 min = 0.70 max = 1.70	μ = 1.82 σ = 0.13 min = 1.53 max = 2.03	μ = 1.93 σ = 0.07 min = 1.81 max = 2.06
Distance 2	1	μ = -3.11 σ = 0.22 min = -2.72 max = -3.40	μ = 0.07 σ = 0.15 min = -0.05 max = 0.44	μ = -1.72 σ = 0.09 min = -1.57 max = -1.87
Distance 1	2	μ = -1.12 σ = 0.15 min = -0.85 max = -1.42	μ = 3.24 σ = 0.10 min = 3.05 max = 3.39	μ = 1.00 σ = 0.06 min = 0.93 max = 1.12
Distance 2	2	μ = -4.15 σ = 0.13 min = -3.96 max = -4.37	μ = 1.15 σ = 0.11 min = 0.98 max = 1.41	μ = -1.50 σ = 0.10 min = -1.30 max = -1.68
Distance 1	3	μ = 0.13 σ = 0.20 min = 0.03 max = 0.47	μ = 2.13 σ = 0.12 min = 1.99 max = 2.37	μ = 1.04 σ = 0.08 min = 0.87 max = 1.14
Distance 2	3	μ = -2.93 σ = 0.12 min = -2.76 max = -3.09	μ = -0.20 σ = 0.08 min = -0.04 max = -0.31	μ = -1.62 σ = 0.09 min = -1.45 max = -1.74
Distance 1	1+3	μ = -0.16 σ = 0.36 min = 0.11 max = -0.76	μ = 2.67 σ = 0.19 min = 2.21 max = 2.97	μ = 2.13 σ = 0.23 min = 1.56 max = 2.76
Distance 2	1+3	μ = -2.66 σ = 0.24 min = -2.41 max = -3.10	μ = 0.38 σ = 0.22 min = -0.02 max = 0.78	μ = -2.00 σ = 0.30 min = -1.46 max = -2.36

Table 7.2

Key: 1 - Iterative fitting, 2 - Secondary smoothing, 3 - Boundary sampling.

In fact, as can be seen from **Figure (7.4.4.3)**, the boundary estimate is skewed somewhat from the expected position. After investigation, this phenomena appears to occur as a result of edge disturbance in the foreground ply, which can transpire as a result of the cutting process. Such disturbance produces a texture variation which causes errors in the texture boundary. This effect is discussed further in **Section (7.4.6)**. The practical result of the edge disturbance however, is that the maximum error for inspection at this resolution has to be taken as 1.75 pixels, which translates to an absolute error

of +/- 1mm, too inaccurate for most composite lay-up applications. In an effort to improve upon this, more experiments were carried out at a higher image resolution, this time with a reduced pixel size measured at 0.26x0.25mm^2. This is approaching the limit of this form of texture based inspection for the carbon fibre materials used in this project. For higher resolution images the image area represented by the texture primitives (i.e. the weave or weft of the material) is such that 7x7 convolution kernels are not able to separate the textures.

The results of the experiments carried out at higher resolution for 45°/0° lay-ups are shown in **Table 7.3**. In general, the results are similar to those in the previous tables, but some differences are observable. Notably the mean deviation measured over the tests is reduced by applying all three post-processing operations, although the maximum deviation remains much the same. Therefore, although post-processing of texture boundaries obtained from high resolution images can have a positive effect, the extra processing overheads incurred, especially in iterative fitting, will usually not be worth the effort.

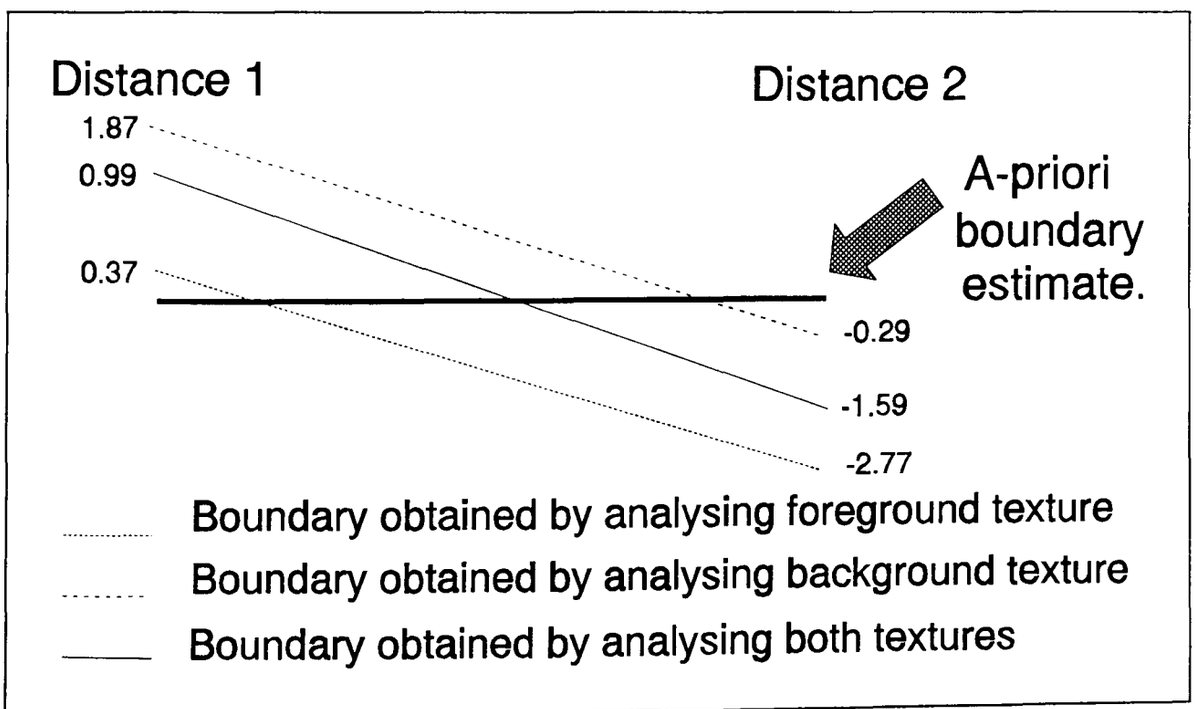


Figure (7.4.4.3). Illustration of the boundary estimates presented in the first two rows of **Table 7.2** (no post-processing).

Straight Edge 45°/0° 1 pixel is 0.26x0.25 mm^2	Options	Textures Analysed		
		Foreground	Background	Both
Distance 1		μ = -0.28 σ = 0.23 min = 0.05 max = -0.67	μ = -1.64 σ = 0.12 min = -1.46 max = -1.85	μ = -0.70 σ = 0.14 min = -0.46 max = -0.91
Distance 2		μ = -0.72 σ = 0.15 min = -0.54 max = -0.99	μ = -1.17 σ = 0.19 min = -0.87 max = -1.39	μ = -1.16 σ = 0.11 min = -0.99 max = -1.32
Distance 1	1	μ = 0.15 σ = 0.07 min = 0.03 max = 0.29	μ = -1.32 σ = 0.12 min = -1.17 max = -1.56	μ = -0.32 σ = 0.11 min = -0.13 max = -0.47
Distance 2	1	μ = -1.44 σ = 0.25 min = 0.89 max = -1.80	μ = -0.72 σ = 0.09 min = -0.59 max = -0.94	μ = -1.33 σ = 0.15 min = -1.18 max = -1.58
Distance 1	2	μ = 1.94 σ = 0.20 min = 1.66 max = 2.23	μ = -2.69 σ = 0.04 min = -2.65 max = -2.78	μ = -0.39 σ = 0.07 min = -0.25 max = -0.48
Distance 2	2	μ = -0.07 σ = 0.26 min = -0.03 max = 0.47	μ = -2.35 σ = 0.07 min = -2.27 max = -2.51	μ = -1.27 σ = 0.09 min = -1.15 max = -1.44
Distance 1	3	μ = 0.64 σ = 0.2 min = 0.34 max = 0.99	μ = -1.45 σ = 0.15 min = -1.20 max = -1.74	μ = -0.33 σ = 0.14 min = -0.13 max = -0.62
Distance 2	3	μ = -1.22 σ = 0.3 min = -0.71 max = -1.73	μ = -1.12 σ = 0.15 min = -0.86 max = -1.40	μ = -1.40 σ = 0.15 min = -1.19 max = -1.66
Distance 1	1+3	μ = 0.14 σ = 0.18 min = 0.03 max = 0.56	μ = -1.67 σ = 0.41 min = -0.65 max = -2.15	μ = -0.27 σ = 0.23 min = 0.05 max = -0.52
Distance 2	1+3	μ = -1.56 σ = 0.32 min = -0.79 max = -1.96	μ = -0.52 σ = 0.25 min = -0.09 max = -0.99	μ = -1.40 σ = 0.23 min = -1.06 max = -1.85
Distance 1	1+2+3	μ = 1.24 σ = 0.44 min = 0.52 max = 1.94	μ = -1.82 σ = 0.10 min = -1.62 max = -1.97	μ = -0.20 σ = 0.24 min = 0.07 max = -0.56
Distance 2	1+2+3	μ = 0.19 σ = 0.44 min = -0.08 max = 0.69	μ = -2.86 σ = 0.18 min = -2.69 max = -3.16	μ = -0.87 σ = 0.27 min = -0.53 max = -1.33

Table 7.3

Key: 1 - Iterative fitting, 2 - Secondary smoothing, 3 - Boundary sampling.

The most important fact to be gleaned from these experiments is that the error in pixels is similar at higher resolution as it is at lower resolution. The absolute error however, is less, since each pixel represents a smaller area.

Straight Edge 0°/90° 1 pixel is 0.61x0.59 mm^2	Options	Edge Operator
Distance 1		μ = -0.43 σ = 0.04 min = -0.33 max = -0.48
Distance 2		μ = 0.68 σ = 0.04 min = 0.59 max = 0.73
Distance 1	1	μ = -0.31 σ = 0.05 min = -0.20 max = -0.38
Distance 2	1	μ = 0.53 σ = 0.05 min = 0.47 max = 0.61
Distance 1	2	μ = 0.15 σ = 0.04 min = 0.08 max = 0.22
Distance 2	2	μ = 0.07 σ = 0.06 min = -0.03 max = 0.15
Distance 1	1+2	μ = 0.13 σ = 0.04 min = 0.07 max = 0.18
Distance 2	1+2	μ = -0.06 σ = 0.07 min = 0.06 max = -0.21

Table 7.4
Key: 1 - Iterative fitting, 2 - Thinning.

For the data in **Table 7.3**, taking the maximum deviation as 1.32 pixels as obtained with no post-processing, and the vertical pixel dimension as 0.25mm, then an absolute error of +/-0.33mm results.

7.4.5 Results Using Texture Edge Operators.

The same experiments that were performed using texture analysis were

performed using textured edge operators. The results for the first lay-up configuration (0°/90°, pixel resolution 0.61x0.59mm^2) are shown in **Table 7.4**. This time only two post-processing operations are applicable: iterative fitting, and thinning of edge points. From the data in **Table 7.4**, it can be seen that for textured edge operators, these post-processing operations do provide a decrease in error. With no post-processing the maximum deviation is 0.73 pixels (0.43mm), with iterative fitting 0.61 pixels (0.36mm), with thinning 0.22 pixels (0.13mm), with iterative fitting and thinning 0.21 pixels (0.12mm). This data compares favourably with that derived using texture analysis on this configuration at this resolution (+/-0.23mm). Textured edge operators were also used on the 45°/0° lay-up, again with a pixel resolution of 0.61x0.59mm^2, and the results are shown in **Table 7.5**. As with texture analysis, the disturbance of the edges in these lay-ups causes an increase in inspection error, but with textured edge operators the effect is not as pronounced. With no post-processing the maximum deviation is 0.92 pixels (0.54mm), with iterative fitting 1.11 pixels (0.66mm), with thinning 0.84 pixels (0.50mm), with thinning and iterative fitting 0.83 pixels (0.49mm). The boundaries obtained using textured edge operators can therefore be estimated to +/-0.5mm at this resolution, compared to +/-1mm with texture analysis. The best post-processing configuration would be to implement only the thinning process, since this produces almost identical results to thinning with iterative fitting, but with considerably less processing required. Another point to note is that textured edge operators produce more consistent boundary estimates than texture analysis, as shown by the lower standard deviation values.

A further application for textured edge operators is the detection of boundaries between plies at the same orientation. **Table 7.6** gives the results for boundary estimation with foreground and background plies both at 0°. The results for this ply configuration are consistent with those obtained when the ply orientations differ. From the data in **Table 7.6**, iterative fitting of points produced the best result for this ply configuration, with a maximum deviation of 0.37 pixels (0.22mm), whilst thinning resulted in a maximum deviation of 0.47 pixels (0.27mm).

Straight Edge 45°/0° 1 pixel is 0.61x0.59 mm^2	Options	Edge Operator
Distance 1		μ = 0.87 σ = 0.04 min = 0.80 max = 0.92
Distance 2		μ = -0.02 σ = 0.05 min = 0.01 max = 0.09
Distance 1	1	μ = 1.04 σ = 0.05 min = 0.96 max = 1.11
Distance 2	1	μ = -0.13 σ = 0.05 min = -0.02 max = -0.22
Distance 1	2	μ = 0.73 σ = 0.06 min = 0.61 max = 0.84
Distance 2	2	μ = 0.-0.37 σ = 0.08 min = -0.19 max = -0.48
Distance 1	1+2	μ = 0.58 σ = 0.10 min = 0.46 max = 0.83
Distance 2	1+2	μ = -0.19 σ = 0.11 min = -0.06 max = -0.38

Table 7.5
Key: 1 - Iterative fitting, 2 - Thinning.

The slight discrepancies in results for different lay-ups are probably due not only to different edge characteristics of the ply, but also to different characteristics of the edge operators themselves.

Straight Edge 0°/0° 1 pixel is 0.61x0.59mm^2	Options	Edge Operator
Distance 1		μ = -0.41 σ = 0.04 min = -0.35 max = -0.49
Distance 2		μ = 0.17 σ = 0.05 min = 0.09 max = 0.24
Distance 1	1	μ = -0.28 σ = 0.06 min = -0.19 max = -0.37
Distance 2	1	μ = 0.14 σ = 0.06 min = -0.01 max = 0.21
Distance 1	2	μ = -0.30 σ = 0.08 min = -0.21 max = -0.46
Distance 2	2	μ = -0.04 σ = 0.09 min = 0.01 max = -0.23
Distance 1	1+2	μ = -0.32 σ = 0.13 min = -0.07 max = -0.57
Distance 2	1+2	μ = 0.02 σ = 0.07 min = 0.01 max = 0.10

Table 7.6
Key: 1 - Iterative fitting, 2 - Thinning.

It is important to remember that these operators are trained, and so two masks trained for the same application but on different training areas can exhibit slightly different properties.

Unfortunately, the superior performance of texture edge operators at

low resolution does not carry over to higher resolution. In fact, when the foreground and background plies are of different orientations, it proved impossible to train a mask to enhance the boundary whilst suppressing the background textures. There is no reason to believe that this is caused by anything other than the fact that the maximum mask size (7x7, due to hardware limitations) is too small to cope. When both plies are at the same orientation however, it is possible to train a mask to detect boundaries whilst suppressing the texture, although the performance is not as good as at lower resolution. The results obtained from using such a mask are shown in **Table 7.7**. Again, fitting gives the best results with a maximum deviation of 2.26 pixels (0.57mm). This is similar to the results obtained at lower resolution (0.50mm).

7.4.6 Summary and Discussion.

A method of using texture based tools for inspection of straight edged plies has been devised. An attempt has been made to establish the error of this inspection method for different material configurations and image resolutions, and to obtain an understanding of the significant parameters affecting inspection.

The most important conclusion which must be drawn from these experiments is that the results are largely material dependant. The optimum parameters will vary for different materials and configurations, as will the resulting inspection error. To some extent this result could have been anticipated. In the texture analysis literature examples of segmentation are often given, and it is easily observable that classification results and segmentation accuracy vary according to the texture samples used as well as with the texture method employed (see for example **[du Buf et al, 1990]**). However the results given here are not obtained from synthetic imagery, but from images of physical objects (plies), and there is also a physical phenomenon which must be considered with regard to boundary detection: the effect of the cutting process on ply edges. All dry fibre composite materials have specific fibre directionality, and the relationship between this and the

direction of cut can affect the cleanness of cut at ply edges. The cleaner the cut, the more homogenous the texture around the ply edge (i.e. the fibres are not disturbed to form a slightly different texture effect which might be misclassified). The cleaner the cut therefore, the more accurate the segmentation produced by texture based inspection. From the results shown in **Tables 7.1** to **7.7**, it would seem that edges of material at ±45° produce the largest inspection error. This is in agreement with the experience gained in cutting dry-fibre materials: edge disturbance is least prevalent when the direction of cut is either perpendicular or parallel to the fibre direction. Other angles of cut with respect to fibre direction create more edge disturbance. Obviously this effect will depend on the material type and cutting method as well as the fibre orientation. It is difficult therefore to obtain definitive figures regarding inspection accuracy. Nevertheless, these experiments have been useful in giving a good indication of the magnitude of error i.e. ±0.33mm to ±0.5mm depending on material and inspection method.

As regards the significant parameters affecting inspection, only very general conclusions can be drawn. For texture analysis based inspection, the inspection error can be reduced by increasing the image resolution. There are two limitations of this. Firstly, at higher resolutions convolution kernels larger than 7x7 will be required to separate textures. Secondly, for higher resolution images there is a corresponding decrease in the field of view, which will have implications for the hardware required to inspect a component. For texture edge operators, increasing resolution is not guaranteed to reduce inspection error. This is probably due to the fact that at higher resolutions more detail is present in the background texture, and so the task of the edge operator in suppressing the background texture is that much more difficult. More "noise" is therefore present in the filtered image.

No method of post-processing the boundaries has been found which consistently and reliably reduces the inspection error.

Straight Edge 0°/0° 1 pixel is 0.26x0.25mm^2	Options	Edge Operator
Distance 1		μ = 2.21 σ = 0.17 min = 1.94 max = 2.57
Distance 2		μ = 1.43 σ = 0.07 min = 1.26 max = 1.53
Distance 1	1	μ = 2.19 σ = 0.05 min = 2.09 max = 2.26
Distance 2	1	μ = 1.27 σ = 0.06 min = 1.23 max = 1.44
Distance 1	2	μ = 2.18 σ = 0.56 min = 1.34 max = 3.24
Distance 2	2	μ = -0.04 σ = 0.34 min = 0.01 max = 0.48
Distance 1	1+2	μ = 0.63 σ = 0.47 min = -0.10 max = 1.16
Distance 2	1+2	μ = 1.72 σ = 0.45 min = 0.73 max = 2.33

Table 7.7
Key: 1 - Iterative fitting, 2 - Thinning.

7.5 Inspection of Curved Edges.

As well as straight edges, it is desirable to estimate edge location error for plies with curved edges. As already mentioned, the shape of plies in the aerospace industry is usually defined by straight lines and arcs. For a fully

implemented system it would be required to be able to model all the various conic arcs (circle, ellipse, parabola, hyperbola) as well as other low degree polynomials. For the sake of these experiments however, attention is restricted to circular arcs. This is justified by the fact that many ply shapes are constructed from straight lines and circular arcs. In addition the results gained from circular arcs should be representative of what might be obtained using any other conic sections.

7.5.1 Least Squares Fitting of Circular Arcs.

Least squares fitting of circular arcs is not as straightforward as for straight lines. Phrased in the literature as "nonintuitive behaviour" **[Ballard and Brown, 1982]**, results obtained using the explicit least squares solutions for circle fitting (based on the general conic equation shown as equation **(7.7)**) seem just plain wrong. That is, the resultant circle does not fit the input data at all well.

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0 \qquad (7.7)$$

Most workers, it would seem, have abandoned a closed form solution and adopted an iterative approach instead **[Ballard and Brown, 1982]**, **[Landau, 1986]**, **[Hooley, 1993]**. Two of these iterative algorithms were implemented and tested. One algorithm due to **[Landau, 1986]** was found to perform well only if the input data is spread evenly around the circumference (the iterative estimate of circle centre is based on the centre of gravity of data points). This algorithm is therefore not at all suited to fitting arcs. The other algorithm due to **[Hooley, 1993]** performed correctly. Both, however, require a large amount of computation per iteration and a large number of iterations. Given two hundred data points to fit to, the computation time was approximately 15 seconds for the **[Hooley, 1993]** algorithm. Attention therefore turned back to a closed form solution. **[Shunmugam, 1986]** has reported a least squares solution based on the parametric form of the circle shown in equation **(7.8)**.

$$x = r\cos\theta + x_c \qquad (7.8)$$
$$y = r\sin\theta + y_c$$

where $\mathbf{x_c,y_c}$ represents the coordinates of the centre of the circle of radius **r**. In this formulation the data points are represented in polar coordinates $(\mathbf{r_i},\theta_i)$. For a "well-centred" set of points (centre of the circle at or near the origin) the corresponding error measure for point **i** can be defined as

$$e_i = r_i - (r + x_c\cos\theta_i + y_c\sin\theta_i) \qquad (7.9)$$

which is the deviation of point **i** from the best fit circle centred at $(\mathbf{x_c,y_c})$ with radius **r**. The general least squares solution of equation **(7.9)** can be represented in matrix form as

$$\begin{bmatrix} \sum\cos^2\theta_i & \sum\sin\theta_i\cos\theta_i & \sum\cos\theta_i \\ \sum\sin\theta_i\cos\theta_i & \sum\sin^2\theta_i & \sum\sin\theta_i \\ \sum\cos\theta_i & \sum\sin\theta_i & \sum 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ r \end{bmatrix} = \begin{bmatrix} \sum r_i\cos\theta_i \\ \sum r_i\sin\theta_i \\ \sum r_i \end{bmatrix} \qquad (7.10)$$

The system of equations represented by **(7.10)** can be solved for $\mathbf{x_c}$, $\mathbf{y_c}$, and **r** by numerical methods **[Press et al, 1990]**. A simple and very robust approach is **Gauss-Jordan** elimination, and a version of this algorithm was implemented.

Experiments with this version of the least squares approach to arc fitting were instructive in that they showed that this method performed correctly for well-centred data points, but that if this condition was not met then "nonintuitive behaviour" once again resulted[1]. A necessary condition for using this algorithm in the field of image processing therefore, is that a reasonably

[1] Subsequent work has established that this is due to a flaw in the formulation of e error measure in the original paper.

accurate estimate of arc centre be obtained, either *a-priori* (i.e. from CAD data) or from the data points themselves. The data points can then be translated by an amount (x_0,y_0), where (x_0,y_0) is the estimate of arc centre. In effect, the data points are being translated to centre on the origin. The algorithm can then be used to calculate values for (x_c,y_c) and **r**. The final value for arc centre is then $(x_0+x_c\ ,\ y_0+y_c)$.

As long as the data points are pre-processed to centre at or near the origin, then the method of **[Shunmugam, 1986]** produces near identical results (to a few hundredths of a pixel for both centre and radius estimate) to the algorithm of **[Hooley, 1993]**. Processing time is however a mere 0.4 seconds for an un-optimised implementation of the Shunmugam algorithm, compared to the 15 seconds required by the iterative algorithm. In fact the 0.4 seconds could be significantly decreased by implementing the required trigonometric functions in look-up tables. In addition a more efficient equation solver could be implemented to take advantage of the symmetry of the matrix in equation **(7.10)**.

If the estimate of arc centre is badly inaccurate then the resulting performance can degrade. The sensitivity of the algorithm to inaccuracies in centre estimate was investigated as follows. An arc was created using a graphics routine, and it's centre and radius estimated to subpixel precision using the iterative algorithm. The performance of the closed form algorithm for the same arc was then evaluated. The initial estimate for arc centre was varied from the known centre (i.e. that deduced using the iterative algorithm), and the results (i.e. estimates of radius and centre) compared with those produced by the iterative algorithm. For variations in the initial estimate of arc centre up to ±10 pixels, the results were virtually identical to those produced by the iterative algorithm. The stability of the closed form algorithm is illustrated in **Figure (7.5.1.1)**. This shows that the sum of the residuals of the points fitted to the arc using the closed form algorithm vary very little as the estimate of arc centre decreases in accuracy (only the x ordinate was varied to produce this graph). The conclusion therefore, is that the closed form algorithm can easily withstand errors of up to ±10 pixels. For this application

a-priori knowledge of the expected centre position (obtained from ply design or training data) can be used, or alternately an estimate of the centre can be obtained by geometric means from the data points.

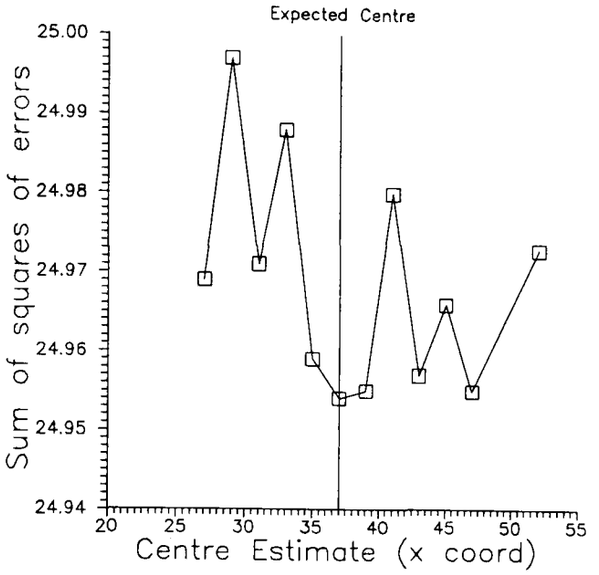


Figure (7.5.1.1). The best estimate of arc centre x-coord is 36.80. This graph shows how robust the least squares arc fitting is for increasingly inaccurate centre estimates. All figures in pixels.

7.5.2 A Framework to Estimate Inspection Error.

Inspection error of curved edges was estimated in a very similar way to straight edges. The plies cut this time were circular, and were cut using a specially machined metal template. The "real" boundary position in image space is determined in the same way as for straight edges in **Section 7.4.2**. A background ply is laid-up, and a sheet of styrene placed on top. In this instance the styrene sheet has been cut with a curved boundary which matches the profile of the ply foreground ply i.e. an arc. The foreground ply is laid-up, so that there is a slight overlap with the styrene sheet along the arc boundary. The position of this boundary can now be obtained by applying an edge operator. For these experiments only a portion of the circular boundary is processed subtending an angle of approximately 120°. This is consistent with the curved profiles of many plies in existing lay-up applications. The portion processed can be visualised as the arc swept out by the hand of a watch going from 5 minutes past to 25 minutes past. This being the case, the

vertical Sobel edge operator was used to detect edge points. Once a suitable threshold had been applied, the remaining points were fitted to an arc using the least squares formulation of **[Shunmugam, 1986]**. This process was repeated over fifty iterations and the mean result taken. (The initial estimate for the centre of this arc was established empirically at this point, and was thereafter used when performing least squares arc fitting of texture boundaries). The arc fitted to the Sobel edge points was then taken an accurate estimate of the boundary position. The styrene sheet is then carefully removed without disturbing the ply position. Texture-based estimates of boundary position can now be obtained.

For simplicity, the distance is measured between the estimated "true" edge and texture-based estimate in the horizontal direction. Distance is again measured at two points evenly distributed on the arc, as shown in **Figure (7.5.2.1)**. For each point a **y**-coordinate is chosen, and this is then used to generate the corresponding **x**-coordinates for each respective arc. The difference between the two **x**-coordinates is taken as the distance from one arc to the other (i.e. from the texture-based estimate to the estimated "true" position). This process enables us to obtain a subpixel estimate of inspection error, which can again be converted to absolute error by considering the dimensions of an individual pixel.

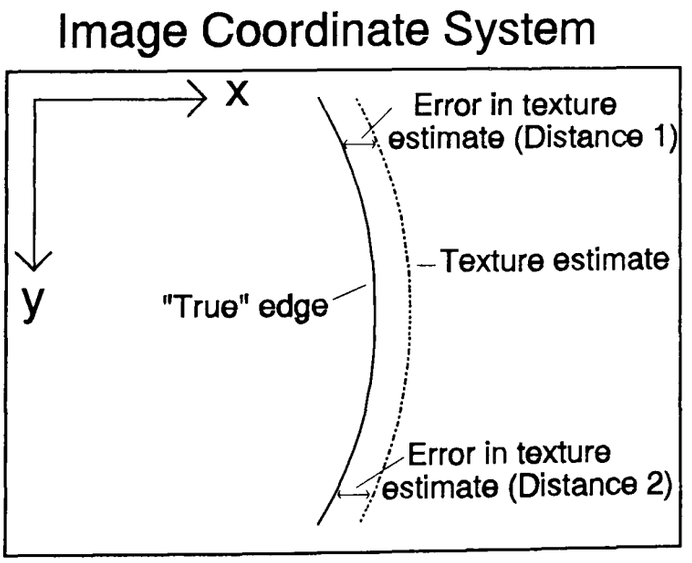


Figure (7.5.2.1). The difference between the texture estimate and the "true" edge is taken as the inspection error.

7.5.3 Results Using Texture Analysis.

Since with straight edges best results for texture analysis were obtained at higher resolution (1 pixel $\approx$ 0.25mm^2), inspection of curved edges using texture analysis was restricted to this resolution. **Table 7.8** shows the results obtained for the 0° foreground plies on 90° background ply at a resolution of 0.26x0.25mm^2. With no boundary post-processing the maximum deviation is 1.21 pixels (0.30mm). One point to note is that the results are less consistent than for straight edges (standard deviations are 0.24 and 0.33). This seems to be a feature of least squares arc fitting, which has an extra degree of freedom compared to line fitting, and seems as a result to be more sensitive to small changes in input data. Iterative fitting certainly appears to have a detrimental effect, increasing the maximum deviation to 2.51 pixels.

Secondary smoothing produces an increase in maximum deviation to 2.00 pixels, similar to the increase observed when secondary smoothing is applied to line fitting. Boundary sampling, taking every tenth point, made little difference. No combination of the above processes served to improve upon the original result obtained with no boundary post-processing.

The results for 45° foreground plies on a 0° background ply with a pixel resolution of 0.25mm^2 are shown in **Table 7.9**, and generally provide better results than those shown in **Table 7.8**. With no post-processing the maximum deviation is 0.54 pixels (0.14mm). The only anomalous result from this data comes from secondary smoothing, which for once produces a significant improvement in performance, giving a maximum deviation of 0.26 pixels (0.07mm). This result is inconsistent with other results presented here, and indeed with experience gained throughout many other tests. It does however indicate the difficulty in analyzing results taken from real images, where any number of factors can affect the results, and usually do.

From the data presented here then, the confidence of texture analysis boundary estimates for arc inspection is approximately ±0.30mm. This is a very similar result to that achieved in the inspection of straight edges using texture analysis (+/-0.33mm). This is somewhat surprising, in that the inspection of straight edges seems intuitively simpler.

Curved Edge 0°/90° 1 pixel is 0.26x0.25 mm²	Options	Textures Analysed		
		Foreground	Background	Both
Distance 1		μ = 0.44 σ = 0.08 min = 0.33 max = 0.61	μ = 0.33 σ = 0.68 min = 0.10 max = 1.30	μ = 0.48 σ = 0.24 min = 0.20 max = 0.94
Distance 2		μ = 0.78 σ = 0.09 min = 0.66 max = 0.95	μ = -1.17 σ = 0.19 min = -0.87 max = -1.39	μ = 0.19 σ = 0.33 min = 0.06 max = 1.21
Distance 1	1	μ = 0.99 σ = 0.18 min = 0.63 max = 1.23	μ = 2.39 σ = 0.14 min = 2.15 max = 2.61	μ = 2.02 σ = 0.32 min = 1.27 max = 2.51
Distance 2	1	μ = 1.31 σ = 0.15 min = 1.03 max = 1.53	μ = 1.02 σ = 0.27 min = 0.62 max = 1.44	μ = 1.78 σ = 0.28 min = 0.26 max = 0.66
Distance 1	2	μ = 0.41 σ = 0.15 min = 0.10 max = 0.64	μ = 2.58 σ = 0.37 min = 1.79 max = 2.97	μ = 1.56 σ = 0.25 min = 1.10 max = 2.00
Distance 2	2	μ = 0.51 σ = 0.11 min = 0.27 max = 0.66	μ = 1.43 σ = 0.42 min = 0.51 max = 1.84	μ = 1.01 σ = 0.29 min = 0.56 max = 1.56
Distance 1	3	μ = -0.03 σ = 0.23 min = 0.07 max = -0.39	μ = 0.48 σ = 0.48 min = -0.09 max = 1.01	μ = 0.36 σ = 0.39 min = -0.10 max = 0.97
Distance 2	3	μ = 0.48 σ = 0.19 min = 0.19 max = 0.73	μ = -0.80 σ = 0.36 min = -0.11 max = -1.42	μ = -0.11 σ = 0.41 min = -0.07 max = 0.88
Distance 1	1+3	μ = 0.55 σ = 0.40 min = -0.09 max = 1.21	μ = 2.04 σ = 0.79 min = 0.61 max = 3.27	μ = 1.47 σ = 0.31 min = 0.96 max = 2.01
Distance 2	1+3	μ = 0.95 σ = 0.33 min = 0.32 max = 1.54	μ = -0.06 σ = 0.69 min = -0.09 max = 1.25	μ = 1.38 σ = 0.32 min = 0.83 max = 2.04
Distance 1	1+2+3	μ = -0.94 σ = 0.36 min = -0.49 max = -1.56	μ = 5.14 σ = 0.58 min = 3.88 max = 6.19	μ = 2.41 σ = 0.30 min = 1.62 max = 2.73
Distance 2	1+2+3	μ = -0.66 σ = 0.35 min = -0.50 max = -1.56	μ = 3.41 σ = 0.51 min = 2.45 max = 4.46	μ = 2.05 σ = 0.28 min = 1.39 max = 2.42

Table 7.8

Key: 1 - Iterative fitting, 2 - Secondary smoothing, 3 - Boundary sampling.

Curved Edge 45°/0° 1 pixel is 0.26x0.25 mm²	Options	Textures Analysed		
		Foreground	Background	Both
Distance 1		μ = -0.70 σ = 0.12 min = -0.47 max = -0.87	μ = 1.29 σ = 0.19 min = 0.95 max = 1.55	μ = 0.36 σ = 0.12 min = 0.16 max = 0.54
Distance 2		μ = -1.15 σ = 0.12 min = -0.91 max = -1.30	μ = 1.20 σ = 0.17 min = 0.95 max = 1.55	μ = 0.10 σ = 0.10 min = -0.06 max = 0.26
Distance 1	1	μ = -0.21 σ = 0.13 min = 0.01 max = -0.52	μ = 1.10 σ = 0.13 min = 0.91 max = 1.35	μ = 0.69 σ = 0.16 min = 0.39 max = 0.91
Distance 2	1	μ = -0.61 σ = 0.11 min = -0.50 max = -0.90	μ = 0.98 σ = 0.16 min = 0.75 max = 1.24	μ = 0.46 σ = 0.16 min = 0.17 max = 0.70
Distance 1	2	μ = -2.02 σ = 0.10 min = -1.81 max = -2.16	μ = 2.25 σ = 0.06 min = 2.12 max = 2.38	μ = 0.13 σ = 0.07 min = 0.04 max = 0.26
Distance 2	2	μ = -2.49 σ = 0.09 min = -2.33 max = -2.62	μ = 2.26 σ = 0.07 min = 2.13 max = 2.42	μ = -0.1 σ = 0.06 min = 0.01 max = -0.18
Distance 1	3	μ = -0.95 σ = 0.12 min = -0.82 max = -1.18	μ = 1.31 σ = 0.21 min = 0.86 max = 1.66	μ = 0.21 σ = 0.08 min = 0.08 max = 0.35
Distance 2	3	μ = -1.59 σ = 0.11 min = -1.45 max = -1.80	μ = 1.26 σ = 0.18 min = 0.93 max = 1.54	μ = -0.04 σ = 0.10 min = 0.01 max = -0.17
Distance 1	1+3	μ = -0.54 σ = 0.36 min = 0.09 max = -1.17	μ = 1.31 σ = 0.38 min = 0.72 max = 1.97	μ = 0.80 σ = 0.32 min = 0.39 max = 1.46
Distance 2	1+3	μ = -1.16 σ = 0.21 min = -0.98 max = -1.64	μ = 1.13 σ = 0.50 min = 0.34 max = 2.21	μ = 0.50 σ = 0.25 min = 0.20 max = 1.05
Distance 1	1+2+3	μ = -2.59 σ = 0.19 min = -2.29 max = -2.93	μ = 2.40 σ = 0.30 min = 2.06 max = 2.89	μ = 0.94 σ = 0.23 min = 0.41 max = 1.24
Distance 2	1+2+3	μ = -2.75 σ = 0.15 min = -2.45 max = -3.03	μ = 2.54 σ = 0.30 min = 2.20 max = 3.15	μ = 0.62 σ = 0.30 min = 0.16 max = 1.19

Table 7.9

Key: 1 - Iterative fitting, 2 - Secondary smoothing, 3 - Boundary sampling.

Certainly the least squares fitting algorithm for straight lines appears to function in a more stable manner.

7.5.4 Results Using Texture Edge Operators.

Table 7.10 gives results for 0° foreground plies on a 90° background ply with a pixel resolution of 0.61x0.59mm. The best result is obtained with no post-processing of boundaries (maximum deviation 0.52 pixels, 0.32mm) in contrast to the result obtained when inspecting straight edges, where improved results were obtained by thinning and/or iterative fitting of boundary information.

Table 7.11, giving results for 45° foreground plies on a 0° background ply at the same resolution, shows best results when the edge points are thinned before arc fitting. The maximum deviation for these lay-ups is greater (1.48 pixels, 0.9mm) and indicates a relatively poor result for the edge operator. This is instructive in that it shows that the error of these operators is not really something which can be predicted. Rather, for a particular application the relevant masks, once trained would have to be tested to properly determine intrinsic edge estimation error. This is not a desirable process since it increases the work necessary to set up such a system, but for applications requiring guaranteed accuracy this seems advisable. For 0° foreground plies on a 0° background ply at this resolution the results are shown in **Table 7.12**. Best results are again achieved by thinning of boundaries before fitting giving a maximum deviation of 0.56 pixels (0.34mm), which is much more in-keeping with other results achieved using texture edge operators.

Finally, **Table 7.13** gives results for inspection of 0° foreground plies on a 0° background ply with a resolution of 0.26x0.25mm^2 per pixel. The best result is provided by thinning and fitting (maximum deviation = 1.69 pixels, 0.44mm) although the improvement over no post-processing is modest (maximum deviation = 1.74 pixels, 0.45mm), and so the latter method would probably be chosen. Again this shows that texture edge operators actually produce a more accurate estimate of boundary position with lower resolution

images.

7.6 Conclusions.

This chapter has detailed the implementation and testing of a method of using texture based tools for inspection of composite preforms. An attempt has been made to establish the error of this inspection method for different material configurations and image resolutions, and to obtain an understanding of the significant parameters affecting inspection. The process of determining inspection error has received negligible coverage in the literature, and certainly no results have been published regarding error inherent in texture based inspection in an industrial application. The process of estimating inspection error is difficult to formalise, and a rather ad-hoc scheme has been adopted here. It has not proven possible to obtain definitive figures for inspection error. The results will vary according to material type, orientation, and cutting method. In addition convolution kernels optimised on different training data may produce slightly different results. The experiments reported here do however indicate that an inspection error of ±0.33mm to ±0.5mm is intrinsic to this method of inspection, whether inspecting straight or curved edged plies. Reducing this inspection error to a more acceptable level is the topic of the next chapter.

It is worthwhile commenting on the variance of the results obtained. There are two possible causes of this: the texture-based tools, and the boundary fitting process. As regards the texture, it is certain that even when processing two images of the same lay-up captured within a fraction of a second of one another will produce slightly different results. This is a result of lighting variation (especially that caused by fluorescent tubes which oscillate at 50Hz), sampling effects, and possibly camera vibration. Again this indicates the limitations of comparative tests carried out on synthetic images. Another factor which will play a part in the variation observed in results is thresholding. Whilst the threshold algorithm used here performs adequately, slight variations in the texture analysis output can cause the chosen threshold to go up or down a point.

Curved Edge 0°/90° 1 pixel is 0.61x0.59mm^2	Options	Edge Operator
Distance 1		μ = 0.26 σ = 0.16 min = 0.03 max = 0.52
Distance 2		μ = 0.35 σ = 0.05 min = 0.20 max = 0.45
Distance 1	1	μ = 0.98 σ = 1.1 min = 0.43 max = 1.45
Distance 2	1	μ = 0.67 σ = 0.13 min = 0.39 max = 0.77
Distance 1	2	μ = 0.45 σ = 0.12 min = 0.32 max = 0.70
Distance 2	2	μ = 0.62 σ = 0.16 min = 0.29 max = 0.96
Distance 1	1+2	μ = 0.95 σ = 0.21 min = 0.63 max = 1.38
Distance 2	1+2	μ = 0.56 σ = 0.08 min = 0.40 max = 0.70

Table 7.10
Key: 1 - Iterative fitting, 2 - Thinning.

This slight change in threshold will affect hundreds, possibly thousands of pixels in the filtered image, *almost all of them on object boundaries*. This will obviously affect the boundary estimate. It is difficult to see how this effect could be avoided.

Curved Edge 45°/0° 1 pixel is 0.61x0.59mm^2	Options	Edge Operator
Distance 1		μ = -1.47 σ = 0.19 min = -1.06 max = -1.82
Distance 2		μ = 0.79 σ = 0.04 min = 0.74 max = 0.88
Distance 1	1	μ = -1.79 σ = 0.28 min = -1.25 max = -2.07
Distance 2	1	μ = 0.90 σ = 0.07 min = 0.84 max = 1.06
Distance 1	2	μ = -0.01 σ = 0.32 min = -0.07 max = -0.86
Distance 2	2	μ = 1.27 σ = 0.09 min = 1.14 max = 1.48
Distance 1	1+2	μ = 0.86 σ = 0.94 min = -0.21 max = 2.43
Distance 2	1+2	μ = 1.20 σ = 0.10 min = 0.99 max = 1.39

Table 7.11
Key: 1 - Iterative fitting, 2 - Thinning.

Concerning the boundary modelling stage, even given the slight variations in the texture boundaries, the variance in the best fit boundary does seem surprising (or even non-intuitive). Least squares fitting does seem rather sensitive to very small variations in the input data.

Curved Edge 0°/0° 1 pixel is 0.61x0.59 mm^2	Options	Edge Operator
Distance 1		$\mu = 0.36$ $\sigma = 0.14$ min = 0.13 max = 0.60
Distance 2		$\mu = 0.43$ $\sigma = 0.12$ min = 0.23 max = 0.59
Distance 1	1	$\mu = 0.70$ $\sigma = 0.8$ min = 0.50 max = 1.15
Distance 2	1	$\mu = 0.65$ $\sigma = 0.15$ min = 0.41 max = 0.80
Distance 1	2	$\mu = 0.33$ $\sigma = 0.08$ min = 0.27 max = 0.41
Distance 2	2	$\mu = 0.41$ $\sigma = 0.09$ min = 0.29 max = 0.56
Distance 1	1+2	$\mu = 0.85$ $\sigma = 0.23$ min = 0.73 max = 1.18
Distance 2	1+2	$\mu = 0.23$ $\sigma = 0.12$ min = 0.17 max = 0.53

Table 7.12
Key: 1 - Iterative fitting, 2 - Thinning.

Perhaps this is intrinsic to the least squares approach, which relies on determining the point where the error function is at a minimum. Small changes in input data may well have an exaggerated effect on the position of the minima of the function. Another possible source of error is the lens distortion alluded to in **Section 7.3**.

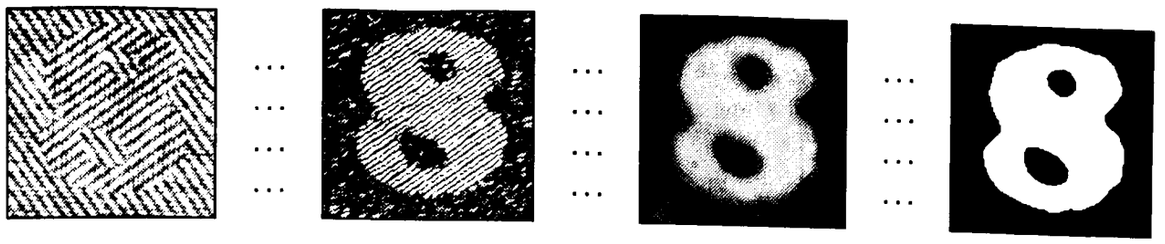
Curved Edge 0°/0° 1 pixel is 0.26x0.25mm^2	Options	Edge Operator
Distance 1		μ = 1.54 σ = 0.12 min = 1.35 max = 1.74
Distance 2		μ = 1.06 σ = 0.11 min = 0.86 max = 1.21
Distance 1	1	μ = 1.51 σ = 0.48 min = 0.67 max = 2.08
Distance 2	1	μ = 0.85 σ = 0.44 min = 0.12 max = 1.35
Distance 1	2	μ = 1.76 σ = 0.12 min = 1.56 max = 1.92
Distance 2	2	μ = 1.40 σ = 0.16 min = 1.10 max = 1.68
Distance 1	1+2	μ = 1.27 σ = 0.32 min = 0.73 max = 1.69
Distance 2	1+2	μ = 1.13 σ = 0.28 min = 0.67 max = 1.49

Table 7.13
Key: 1 - Iterative fitting, 2 - Thinning.

It is conceivable that this could contribute to the unexpected variations in model fitting, especially as regards arc fitting.

In summary, this chapter has detailed the implementation and testing of the texture-based tools described in previous chapters. Simple boundary models have been incorporated in the inspection process, and estimates of

inspection error have been obtained experimentally. In the next chapter a novel boundary refinement approach is investigated in an attempt to reduce inspection error.



CHAPTER 8

Boundary Refinement.

8.1 Introduction.

Chapter Seven demonstrated how texture analysis in conjunction with an appropriate boundary model could provide a fast and elegant method of inspecting carbon fibre lay-ups. The inspection error was of the order of ±0.33mm to ±0.5mm, which is acceptable for many current lay-up applications. This chapter will describe a novel boundary refinement approach developed in an attempt to reduce inspection error. This is in effect a post-processing operation, which takes the texture-based estimate as a starting point from which to perform a local search for "real" edge points.

8.2 General Approach.

The general approach to boundary refinement investigated in this chapter is presented in **Figure (8.2.1)**. It has already been mentioned in **Section (1.6)** that the problem with detecting edge points in a textured image is that the textures themselves produce many edge segments which confuse the situation. The scheme represented in **Figure (8.2.1)** relies on the assumption that since only a small area of the image is being searched which,

according to the texture information, contains the real boundary, then the probability of detecting "real" edge points is much higher.

---

for each boundary of interest
do
extract texture boundary
perform limited search perpendicular to boundary for edge points
form edge points into a new boundary

Figure (8.2.1). A general approach to texture boundary refinement.

The first stage, that of extracting the texture boundary, can be accomplished by texture analysis or by texture edge operators as appropriate. The boundary referred to is the best fit line or arc, rather than the raw texture boundary points themselves, since this provides a more accurate estimate of ply position. The more accurate the boundary estimate the smaller the area that should be searched for actual edge points, and so the lower the probability of choosing erroneous edge points.

The boundary refinement approach shown in **Figure (8.2.1)** is an intuitive one, and is similar to methods adopted by other workers for different applications. In **[Ballard, Brown, 1982]** a scheme is described based on adjusting a-priori boundaries. The scheme is attributed to **[Bolles, 1977]**, but in fact the paper contains no reference to this type of technique. Wherever it was first described, the procedure detailed in **[Ballard, Brown, 1982]** is as follows. Local searches are carried out at regular intervals along directions perpendicular to the approximate (a-priori) boundary. An edge operator is applied to each point along each perpendicular direction. For each such search, the edge with the greatest magnitude is selected from among those whose orientation is nearly parallel to the tangent to the approximate boundary. The set of edge points so chosen is fitted to an analytic curve and becomes the new boundary representation. No detail of performance is given in **[Ballard, Brown, 1982]**, and since the original reference is unclear, no other information is available.

A variation on the idea is described in **[Pavlidis, Liow, 1990]**, where initial segmentation of non-textured images is region-based, achieved using a split-and-merge algorithm. A second stage of processing, referred to as contour modification, is then performed to improve boundary accuracy by utilising local edge information. This process involves modifying the region-based contour to maximise a merit function. The function in question is composed of three terms representing edge magnitude, curvature, and phase change. The first term is directly proportional to the magnitude of an edge, so that strong edges are more likely to be chosen. The second term favours boundary smoothness and avoids sharp turns, especially in low contrast areas of an image. The third term corresponds to *changes* in direction of the boundary, which should be small. There is some degree of overlap between the functionality of the last two terms, although their mathematical formulation is quite different.

The scheme adopted in this work is a combination of the above methods, and is as follows. The texture boundary is obtained using the methods described in **Chapter Seven**, and this is taken as an initial estimate of boundary location. Local searches are carried out along this boundary in a perpendicular direction. At each point along this perpendicular search, an edge operator is applied and the result is evaluated using a ***merit function***, as defined in the following section. The point giving the highest output from this function is chosen as the new edge. The set of edges resulting from this localised boundary search is fitted to the relevant boundary model using the least squares method detailed in **Section 7.4.1**, and this is taken as the new boundary estimate. **Figure (8.2.2)** demonstrates the idea of this search.

The texture estimate is used to guide the process. For each local search, an initial search point is generated from a parametric representation of the texture estimate. The orientation of the perpendicular is calculated, and a parametric line representation used to generate the coordinates of each pixel to be evaluated as a potential edge point using the merit function. The spatial extent of the search on either side of the texture estimate is specified by a preset parameter, denoted as **k** in **Figure (8.2.2)**.

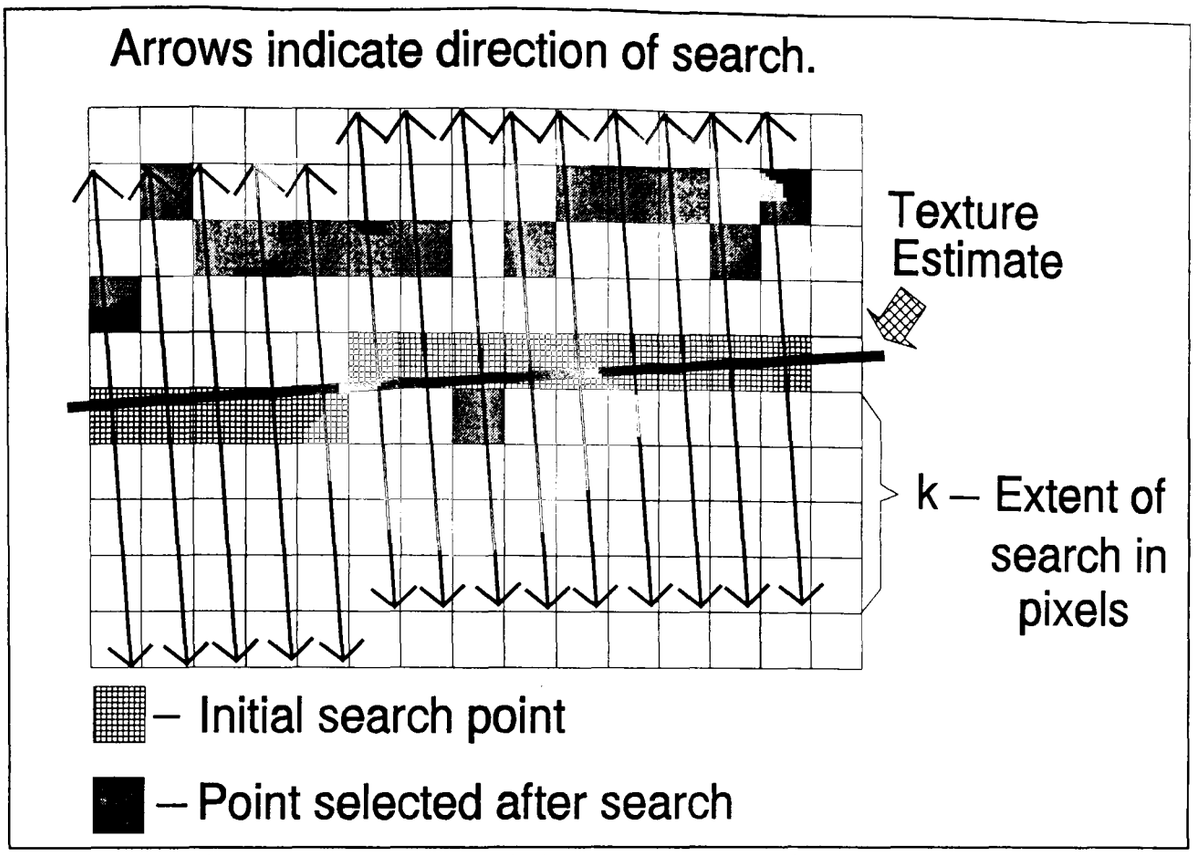


Figure (8.2.2). A localised search is carried out using the texture estimate as a guiding template. Each pixel encountered in the search is evaluated using the merit function described in **Section (8.3)**.

The calculations to generate pixel coordinates in each local search require repetitive floating point operations, and as a result the processing time required for each complete boundary search is approximately two seconds. If required, the process could be made more efficient.

8.3 The Merit Function.

The merit function developed for this application is composed of three terms, as shown in **equation (8.1)**.

$$merit(x,y) = mag(x,y) + \alpha * orient(x,y) + \beta * dist(x,y) \qquad (8.1)$$

where **x,y** are the coordinates of the pixel being evaluated. The function **merit(x,y)** is composed of three terms relating to the magnitude of the edge at pixel(x,y), the orientation associated with that edge, and the distance from pixel(x,y) to the boundary estimate obtained using texture information. The choice of edge magnitude and orientation as contributing terms is common to

most boundary search algorithms **[Ballard, Brown, 1982]**, **[Pavlidis, Liow, 1990]**. The choice of the third term which represents the distance from a candidate edge point to the estimate of boundary position obtained from texture is novel. Candidate edge points which minimise this term will be favoured by inclusion of this term. Conversely, candidate edges points deep within areas which have been classified as belonging to either the foreground texture (ply) or background texture (ply) will not be favoured.

Each of these terms is a function producing a result in the range 0..1. The precise implementation of these terms is described in the remainder of this section. The relative weighting which should be applied to the three terms is governed by the parameters α and β, and the determination of appropriate values for these parameters is covered in **Section 8.5**.

8.3.1 The Mag(x,y) Function.

At each point under evaluation, the corresponding edge magnitude is measured as shown in **equation (8.2)**.

$$magnitude = \frac{|gx| + |gy|}{z} \qquad (8.2)$$

where **gx** and **gy** are the respective responses of the edge operators used, and **z** is a normalising constant calculated from these edge operators as

$$z = \frac{\sum_i (|gx_i| + |gy_i|)}{4}$$

and **i** indexes each coefficient of the edge operators. The normalising constant is designed to ensure that the value obtained for edge magnitude is never greater than 255, the value that can be held in one image pixel, and assumes that the edge operators are symmetric and zero-sum. The choice of the denominator in computing **z** has been arrived at by consideration of the maximum possible output of **gx** and **gy** for any given edge pattern, although

this maximum can in fact only be generated when processing binary images. Nevertheless, the choice of 4 as the denominator in computing **z** provides adequate dynamic range for edge magnitude. The edge operators used in this scheme are conventional gradient edge operators which typically give a reasonably accurate indication of edge position and orientation. The actual edge operators for a particular application can be chosen to match the expected edge profile. The Sobel edge operators have been used throughout these experiments.

The larger the value obtained by application of **equation (8.2)** the stronger the edge at that location. In the boundary refinement procedure described in **[Pavlidis, Liow, 1990]** the edge magnitude was used directly in the merit function i.e. the greater the magnitude the stronger the case for that edge point to be chosen. For this application however, the situation is different. **Figure (8.3.1.1)** shows a magnified portion of a ply edge, and the magnitude image obtained by applying the Sobel edge operators.

From this figure it is evident that edge magnitudes on the ply boundary are less than the edge magnitudes produced as a result of the weft of the material (in fact it is this effect which prompted the investigation into texture analysis in the first place).

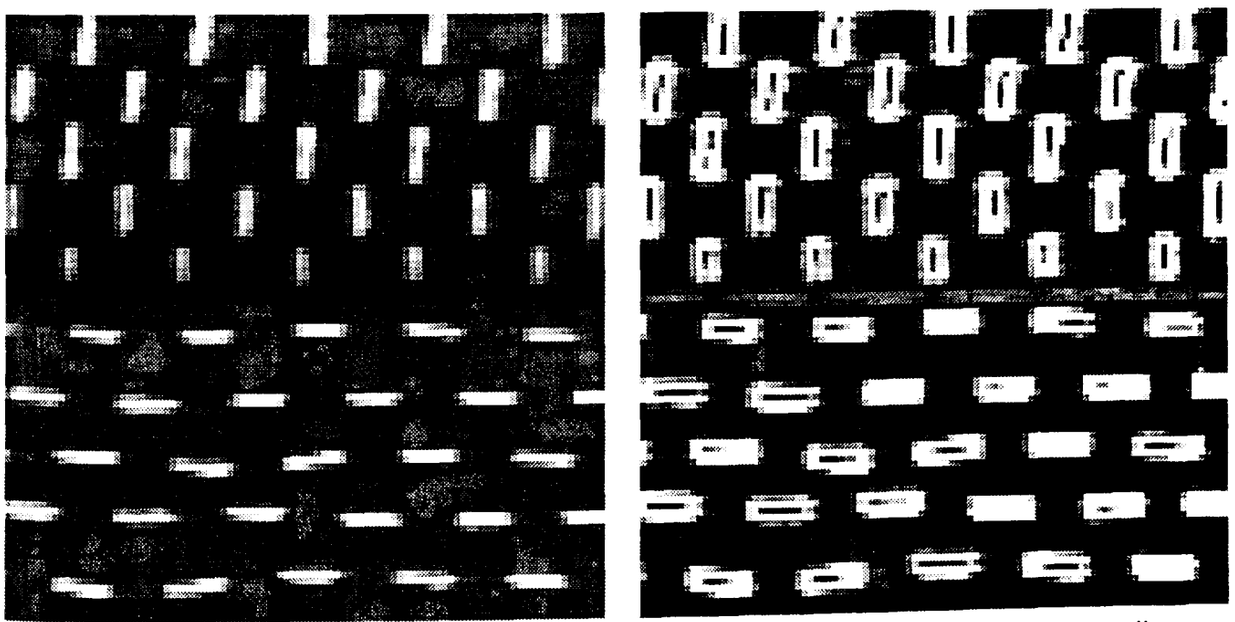


Figure (8.3.1.1). A magnified image of a ply edge and the corresponding magnitude image produced by application of Sobel edge operators.

It is therefore not a good strategy to choose simply the strongest edge, but rather to tailor the merit function to favour edges with a profile (magnitude) close to that commonly produced at ply edges. The function adopted for this is

$$mag(x,y) = 1 - \min\left(1, \frac{|e_{target} - e_{xy}|}{e_{target}}\right) \qquad (8.3)$$

where $\mathbf{e_{target}}$ is the expected edge magnitude of a point on a ply edge, $\mathbf{e_{xy}}$ is the measured edge magnitude of a point at pixel(x,y), and the function **min(a,b)** returns **a** if **a** < **b**, otherwise **b**.

The term $|\mathbf{e_{target}} - \mathbf{e_{xy}}|$ represents the absolute difference between the current edge magnitude and the magnitude expected of a point on a ply edge, regarded here as the edge "target". This difference measure is normalised by dividing by $\mathbf{e_{target}}$. The resultant value could be greater than 1 if $\mathbf{e_{xy}} > 2\mathbf{e_{target}}$, which is possible if $\mathbf{e_{xy}}$ represents a particularly strong edge, and so the **min(a,b)** function is used to ensure that the result does not exceed 1.

Finally this value is subtracted from 1, so that edge magnitudes which are close to the target value give a high function result (≈1), whilst edge magnitudes which are not close to the desired value give a low function result (≈0). A plot of the **mag(x,y)** function is shown in **Figure (8.3.1.2)**. This shows how the function will behave if the value for $\mathbf{e_{target}}$ is relatively low (≈0.25 in **Figure (8.3.1.2)**) which is in fact close to the situation for carbon fibre inspection. From this it can be seen that the function **mag(x,y)** appears "cropped" to produce a non-zero output only over a narrow range of edge magnitudes. As a result the function has a better response profile over the edge magnitudes that are most common, since very high edge magnitudes are rare and in this application provide no useful information as regards ply boundaries.

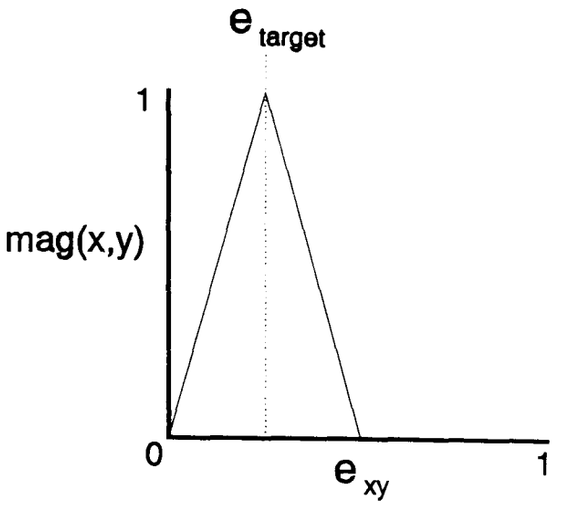


Figure (8.3.1.2). A plot of the function **mag(x,y)**, as defined in **equation (8.3)**.

8.3.2 The Orient(x,y) Function.

The second term of **equation (8.1)** is a function of edge orientation. This is simply required to give a high value (≈1) when the measured edge orientation is close to that of the texture boundary (and therefore hopefully to the orientation of the ply edge), and a low value (≈0) when it is not. The orientation of an edge is measured by **equation (8.4)**.

$$\theta_{xy} = \tan^{-1}\left(\frac{gy}{gx}\right) \qquad (8.4)$$

Similarly the orientation of the texture boundary can be calculated locally by

$$\theta_t = \tan^{-1}\left(\frac{dy}{dx}\right) \qquad (8.5)$$

where

$$dx = (tx_{i-n} - tx_{i+n}) \qquad (8.6)$$

$$dy = (ty_{i-n} - ty_{i+n})$$

and $(\mathbf{tx_i,ty_i})$ is the local texture boundary point, **n** is a small integer. In the

most straightforward case, that of straight edge inspection, θ_t need only be calculated once, since it will be constant for all points on the texture boundary.

Given θ_{xy} and θ_t, they represent similar orientations if they are numerically close, or if they are separated by almost 2π. The difference between θ_{xy} and θ_t is therefore

$$\theta_\Delta = \min(|\theta_t - \theta_{xy}|, ||\theta_t - \theta_{xy}| - 2\pi|) \qquad 8.7$$

θ_Δ ranges from 0 for identical θ_{xy} and θ_t, to π for opposite orientations of θ_{xy} and θ_t. To produce a function which will output 1 in the former case and 0 in the latter, θ_Δ must be normalised and subtracted from 1, thus

$$orient(x,y) = 1 - \frac{\theta_\Delta}{\pi} \qquad (8.8)$$

A graph of **orient(x,y)** versus θ_Δ is shown in **Figure (8.3.2.1)**.

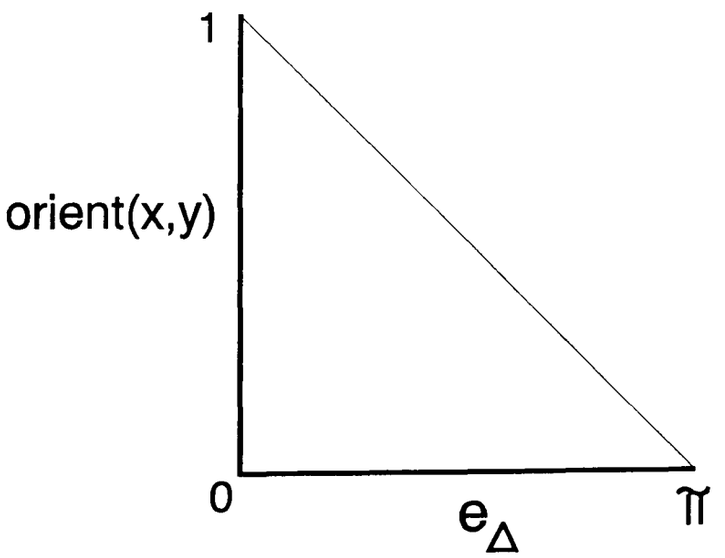


Figure (8.3.2.1). A plot of the function **orient(x,y)** as defined in **equation (8.8)**.

8.3.3 The Dist(x,y) Function.

The third term of **equation (8.1)** relates to the distance of each candidate point to the texture boundary estimate being used as a guiding template. Given that the texture boundary estimate is reasonably accurate,

then the probability of finding the real edge close to this estimate is high, and correspondingly the probability of an edge far from the texture estimate being on the ply boundary is low. The **dist(x,y)** function is designed to produce a high value (≈1) for candidate points on or near the texture boundary, and a low value (≈0) for candidate points further away from the texture boundary. The furthest point from the texture boundary which will be evaluated depends on the extent of the perpendicular search as determined by the parameter **k** in **Figure (8.2.2)**. The required function can therefore be achieved by **equation (8.9)**.

$$dist(x,y) = \frac{k - d_{xy}}{k} \qquad (8.9)$$

where **k** is the maximum spatial extent of the local search on either side of the texture boundary, and $\mathbf{d_{xy}}$ is the distance measured in pixels from pixel(x,y) to the texture boundary. If $\mathbf{d_{xy}}$ is 0, the output is 1. If $\mathbf{d_{xy}} = \mathbf{k}$, the output is 0. **Figure (8.3.3.1)** illustrates the output of this function when **k**=3. Note the step effect which results, since all distances are in whole pixel units.

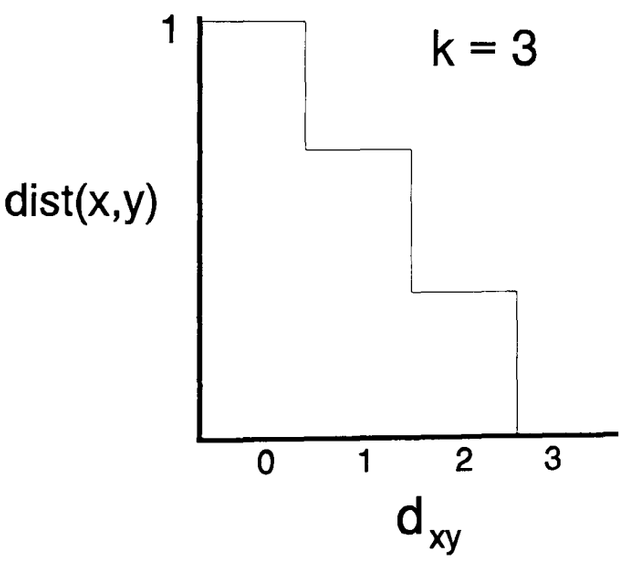


Figure (8.3.3.1). A plot of **dist(x,y)** as defined in **equation (8.9)**.

8.4 Determination of the Target Edge Magnitude.

In **Section 8.3.1** the idea was developed that points on a ply boundary

do not exhibit particularly strong edge magnitudes, and that points at or near the weft in fact produce stronger edge magnitudes. It was conjectured that an improvement in performance of the merit function could be achieved if edge points were assessed as to how well they matched the expected magnitudes of points on ply edges, the term $\mathbf{e}_{target}$ in **equation (8.3)**. Some method is therefore required to determine an appropriate value for $\mathbf{e}_{target}$.

Since the task is simply to maximise the function **mag(x,y)** for a single variable, then it is appropriate to employ an exhaustive search algorithm. In this, the value of $\mathbf{e}_{target}$ is varied linearly over its range and the performance for each value assessed. The way in which performance is assessed for a straight-edged ply is described as follows.

An estimate of the "true" position of a straight-edge ply boundary is obtained as described in **Section 7.4.2**. That is, it is laid-up overlapping slightly with a styrene sheet so that the ply boundary is easily obtainable using conventional edge operators. Edge points are fitted to a straight line using the least squares approach described in **Section (8.4.1)**. The process of applying edge operators and fitting the resulting points to a straight line is repeated 50 times and the mean values of **a** and **b** taken to define the line representing the real ply boundary position. The styrene sheet is then removed with no, or at least negligible, ply disturbance. The texture boundary is now obtained by texture analysis. Given the accurate estimate of boundary position and the texture-based estimate, the boundary refinement stage can be applied. By setting $\alpha = \beta = 0$, **merit(x,y)** as defined in **equation (8.1)** is made to rely solely on the performance of the **mag(x,y)** term. The value of $\mathbf{e}_{target}$ is set to an initial value, and the boundary refinement process performed. The resulting set of points are used to fit a least squares line. From this line, the deviation from the "real" boundary can be measured, as described in **Section 7.4.2**. It is this deviation which is used to assess the best value for $\mathbf{e}_{target}$. The boundary refinement process is repeated (using the same texture estimate) with increasing values of $\mathbf{e}_{target}$, and a graph of $\mathbf{e}_{target}$ versus deviation obtained. This process can be repeated for a number of different lay-ups in order to obtain an aggregate view of the appropriate value for $\mathbf{e}_{target}$.

Examples of the results obtained with this process are displayed graphically in **Figure (8.4.1)** for six lay-ups, all of which have a background ply at 0° orientation and a foreground ply at 90°. These graphs all follow a similar pattern. There is an initial peak (large error) corresponding to low magnitude edges. These "edges" are probably the result of noise an low contrast surface markings, and obviously offer no information on ply boundary position. There is then a well-defined minima centred at approximately 0.18 on the x-axis, which indicates that accurate boundary estimates are obtained with this value of $\mathbf{e}_{target}$. The shape of this minimum varies from graph to graph, but the effect is observable in all. For higher edge values the deviation increases before settling to a plateau which corresponds to the high edge magnitudes produced by the weft.

These graphs therefore validate the theory that pixels on the ply boundary exhibit a particular range of edge magnitudes. The conclusion from **Figure (8.4.1)** is that for 0°/90° lay-ups under the lighting conditions prevalent throughout the experiments, an edge target of approximately 0.18 is appropriate.

More generally, experiments with other lay-up configurations have shown that the particular range of edge magnitudes exhibited is dependant on material, orientation, and of course lighting. Whilst all produce graphs similar to those shown in **Figure (8.4.1)**, the optimum value for $\mathbf{e}_{target}$ can be anywhere between 0.17 and 0.28. This makes it impossible to select a single value for composite inspection, and so a value must be obtained for each edge configuration present in a component. This will not usually be too much of a drawback however, since the texture analysis or texture edge operators require similar training. In any case the number of possible configurations is limited, and many small components will exhibit only two or three different edge types. Another disadvantage is that this type of training to recognise object boundaries depends heavily on consistent lighting conditions, since any variation may possibly alter the edge profiles.

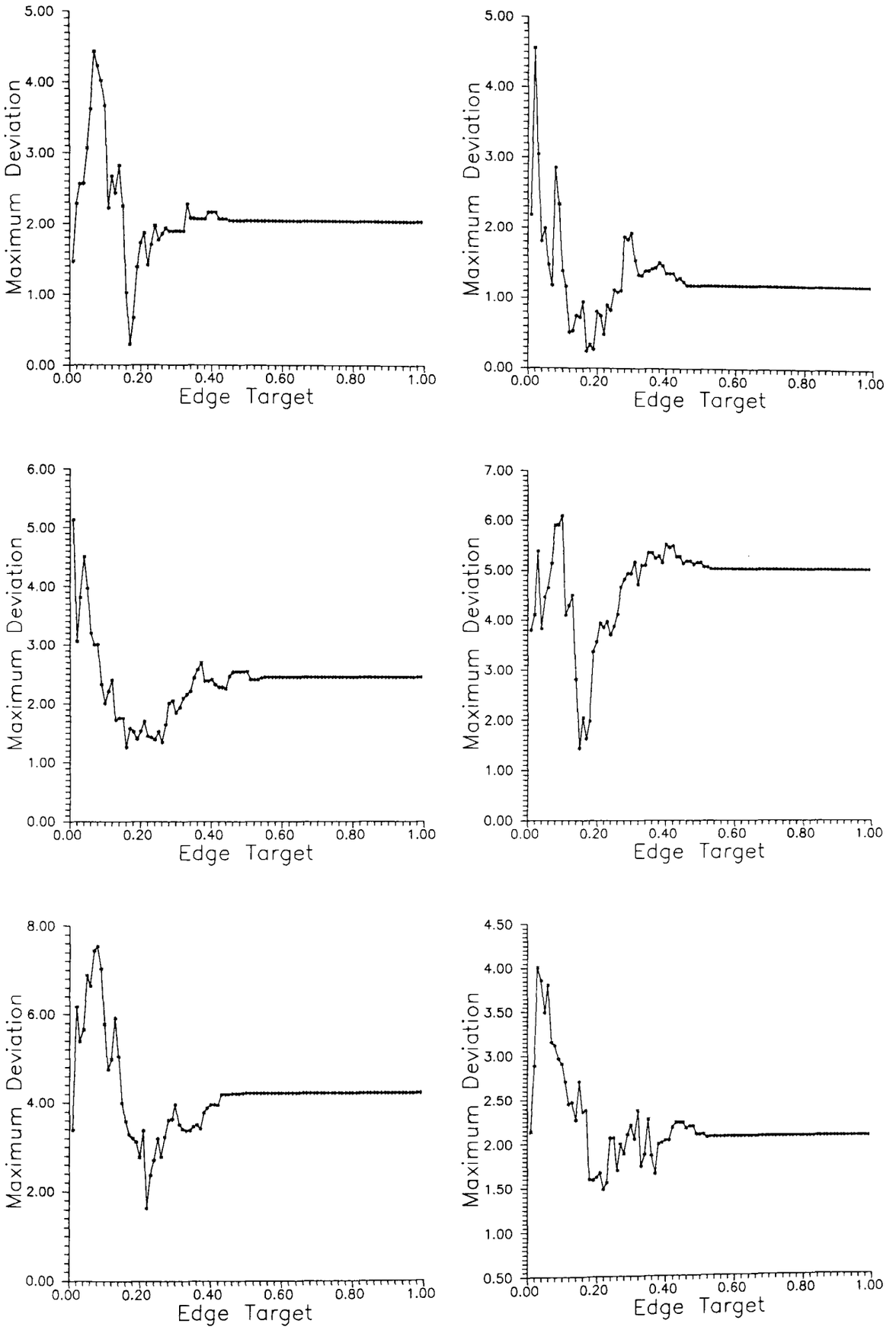


Figure (8.4.1). Graphs used to determine optimum value for $\mathbf{e}_{\mathbf{target}}$. The graphs record the maximum deviation from the "real" boundary position for different values of $\mathbf{e}_{\mathbf{target}}$ for six different lay-ups, all with plies at 0° and 90°.

For inspection of carbon fibre lay-ups however, it has been obvious from the beginning of this project that controlled lighting will be a pre-requisite of any industrial implementation. All things considered therefore, training an algorithm for boundary recognition imposes no extra limitations than were already in existence.

8.5 Determination of Merit Function Parameters.

For a merit function as defined in **equation (8.1)**, some method of deriving appropriate values for the parameters α and β is required. In **[Pavlidis, Liow, 1990]** parameter selection was performed by empirical means, selecting values which appear to produce good results. The method adopted here is based on training the algorithm to provide optimum performance for a particular application. The performance criterion is identical to that described in **Section 8.4** for determining the optimum edge target. That is, the parameters α and β are varied iteratively and a refined boundary estimate obtained for each iteration. The deviation from this estimate to the "real" edge is measured and provides an assessment of performance for that combination of parameters. The parameters giving the best result are chosen. Note that an appropriate value for edge target must already have been obtained using the process described in **Section 8.4**.

The search method used to determine α and β is different from that used to determine $\mathbf{e}_{target}$. Since two parameters are involved, then a more efficient direct search method has been implemented. This method is a variation on the widely used Hooke and Jeeves direct search algorithm **[Walsh, 1975]**. Given a function **f** (in this case the inspection error, parametrised by α and β) which is to be minimised the Hooke and Jeeves method will proceed as follows.

An initial base point (α, β) is chosen. For each variable a step length is chosen, giving $\mathbf{d}_\alpha$ and $\mathbf{d}_\beta$. Once the function has been evaluated at the base point (giving $\mathbf{f}(\alpha, \beta)$), the method progresses through a sequence of **exploratory** and **pattern** moves. The purpose of an exploratory move is to acquire information about the search domain in the neighbourhood of the

current base point i.e. at $\mathbf{f}(\alpha \pm \mathbf{d}_\alpha, \beta \pm \mathbf{d}_\beta)$. A pattern move attempts to speed up the search by using information already acquired about the search domain i.e. by moving in a "successful" direction in larger step sizes.

Since, in this case, the search domain is a function of only two variables, a modified version of the Hooke and Jeeves algorithm has been implemented which uses only exploratory moves. The details of this are as follows.

(1) Evaluate the function $\mathbf{f}(\alpha, \beta)$. $\alpha_{opt} := \alpha$, $\beta_{opt} := \beta$

(2) Iterations := 1

(3) Evaluate $\mathbf{f}(\alpha+\mathbf{d}_\alpha,\beta)$. If this is a success (the function value is smaller here) $\alpha_{opt} := \alpha+\mathbf{d}_\alpha$.

(4) Evaluate $\mathbf{f}(\alpha-\mathbf{d}_\alpha,\beta)$. If this is a success $\alpha_{opt} := \alpha-\mathbf{d}_\alpha$.

(5) Evaluate $\mathbf{f}(\alpha,\beta+\mathbf{d}_\beta)$. If this is a success $\beta_{opt} := \beta+\mathbf{d}_\beta$.

(6) Evaluate $\mathbf{f}(\alpha,\beta-\mathbf{d}_\beta)$. If this is a success $\beta_{opt} := \beta-\mathbf{d}_\beta$.

(7) $\alpha := \alpha_{opt}$, $\beta := \beta_{opt}$

(8) Iterations := iterations + 1.

(9) $\mathbf{d}_\alpha := \mathbf{d}_\alpha$/iterations, $\mathbf{d}_\beta := \mathbf{d}_\beta$/iterations. If some preset number of iterations has not been reached, then go to **(3)**.

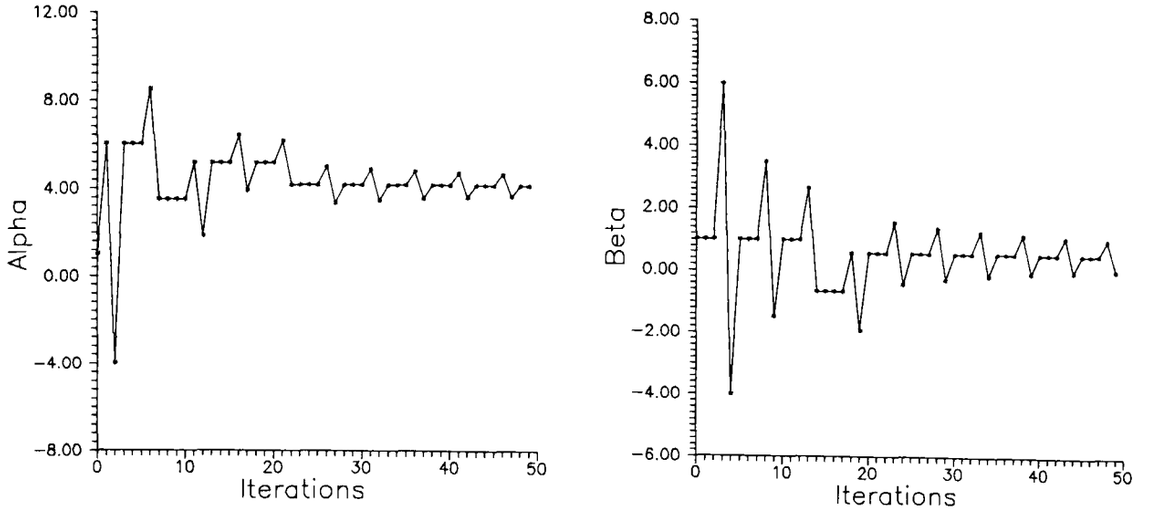
Again this process of parameter selection can be repeated over several lay-ups to obtain an aggregate view of the most suitable parameters for a particular lay-up configuration. **Figure (8.5.1)** illustrates how the algorithm converges to values of α and β which minimise the error function. These graphs are representative of results obtained using other lay-up configurations, and indicate that the task of obtaining consistent values for α and β is less straightforward than obtaining a value for $\mathbf{e}_{target}$. The general conclusions that can be drawn are as follows.

From equation **(8.1)** which defines the merit function, the weighting of the edge magnitude term is unity, and α and β represent the respective weighting of edge orientation and distance from the texture boundary. The most surprising aspect of **Figure (8.5.1)** is that for two of the three lay-ups the optimum value of β is negative, indicating that pixels close to the texture

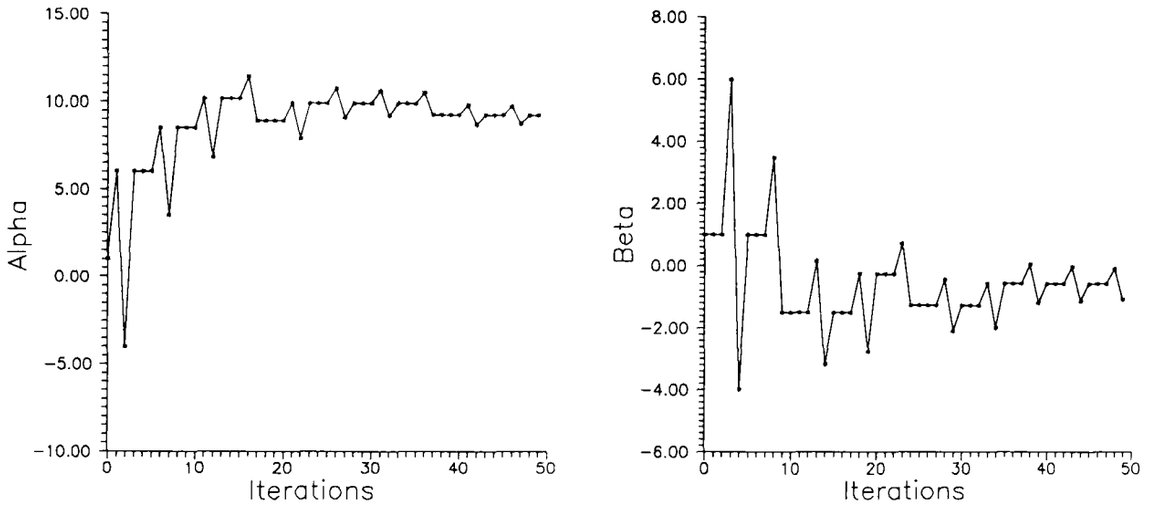
boundary are unlikely to constitute the ply boundary. In other words, the texture boundary in this case is not very accurate. The value obtained for β using other lay-up configurations tended to confirm this, which is surprising considering the results obtained in the previous chapter. This is probably due to the variation inherent in the texture estimate. It does not seem to provide a reliable enough feature for the boundary search algorithm. As a result, the distance from the texture boundary is not as useful a term in the merit function of equation **(8.1)** as might have been hoped. In many cases the weighting of the term, as signified by the value of β, is around zero.

Conversely, the other term in equation **(8.1)**, edge orientation, is obviously of considerable importance. Values for α in **Figure (8.5.1)** range from 4.17 to 9.29, which indicates that in the edge refinement scheme adopted here the orientation of the edge points gives more information about the likelihood of the point lying on a ply boundary than the edge magnitude.

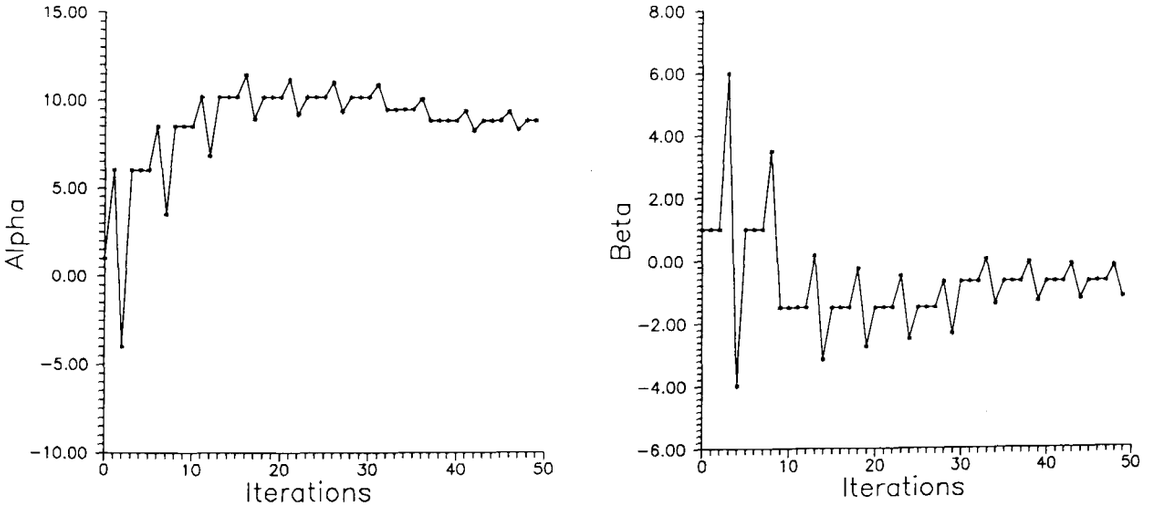
The graphs of **Figure (8.5.1)** also tend to raise questions regarding the stability of the search algorithm, i.e. does it converge? The search algorithm used has perhaps not been designed as carefully as it might have been, computational simplicity having taken precedence. However, from the experience obtained in performing these tests it is argued that further effort in this area is not warranted. The variation in ply edges is such that the most that can be obtained from the search algorithm is an indication of the relative importance for each parameter, not a precisely defined value for each parameter. The algorithm has been successful in achieving this, and provides a considerable improvement over the empirical choice of parameters favoured by other workers **[Pavlidis, Liow, 1990]**. An improved search algorithm could produce no more.



Final values: $\alpha = 4.17$, $\beta = 0.08$



Final values: $\alpha = 9.29$, $\beta = -1.04$



Final values: $\alpha = 8.83$, $\beta = -1.17$

Figure (8.5.1). Three pairs of graphs showing the convergence of α and β using the modified Hooke and Jeeves algorithm. Each pair of graphs were obtained from a different lay-up, all with plies at 0° and 90°.

8.6 Summary of Refinement Algorithm.

A boundary refinement algorithm has been developed to improve on the accuracy obtainable from texture-based estimates. The algorithm performs a localised search for ply edge points using the texture boundary as a guiding template. The points chosen are the ones which best maximise a merit function of edge magnitude, orientation, and distance from the texture boundary.

Successful operation of the algorithm depends on determining suitable values for three parameters of the merit function. The optimum choice of parameters may vary from one ply configuration to another i.e. the values which give optimum performance when inspecting 0°/90° lay-ups may not give optimum performance when inspecting other lay-ups.

The first parameter to be determined is an estimate of edge magnitude at ply boundaries. This is obtained using an exhaustive search algorithm over several training images. Once a suitable value is obtained it can be used in determining suitable values for the other two parameters. These parameters appertain to the relative weighting given to the other two terms in the merit function (edge orientation and distance from the texture boundary). A direct search algorithm is used to estimate the respective weighting that should be applied to the other terms i.e. α and β. Once these training processes have been completed, the boundary refinement process can be used in conjunction with the texture-based tools to provide more accurate inspection, as illustrated by the results in the next section.

8.7 Results.

Estimates of inspection error using texture based tools with boundary refinement have been obtained using the method described in **Section (7.4.2)**. That is, an accurate estimate of the "real" edge position is obtained, and the difference between this and the estimate obtained using texture analysis with boundary refinement taken as the inspection error in the inspection process. Experiments were carried out for four lay-up configurations: 0°/90°, 45°/0°, 45°/90°, and 0°/0°. These were chosen as being representative of the

configurations encountered in many practical applications. For each lay-up configuration, six lay-ups were performed (i.e. twelve plies cut and laid-up). For each lay-up the "real" boundary position was estimated as in **Section (7.4.2)**. The results using texture analysis and boundary refinement were then obtained from 25 images of the lay-up. For each image processed the maximum deviation from the "real" boundary position was recorded, both for the initial texture estimate and after boundary refinement. The data obtained from these experiments is presented in **Table (8.1)** for 0°/90° lay-ups, **Table (8.2)** for 45°/0° lay-ups, **Table (8.3)** for 45°/90° lay-ups, and **Table (8.4)** for 0°/0° lay-ups. The ply configuration, image resolution, and values used for $\mathbf{e}_{target}$, α, and β throughout the experiments is shown in each table. For each lay-up the table records the mean maximum deviation over the 25 images, and the maximum deviation measured in the 25 images. This information is given both for the initial texture estimate and after boundary refinement. The right-most column translates the maximum deviation measured after boundary refinement into millimetres.

Firstly, consider the boundary refinement parameters used for each configuration. For $\mathbf{e}_{target}$, values of 0.18, 0.22, 0.26, and 0.23 were found suitable for 0°/90°, 45°/0°, 45°/90°, and 0°/0° respectively. This indicates again that different material orientations exhibit different edge profiles. For α, values used were 5.0, 3.0, 5.0, and 2.5 respectively. For all ply configurations tested therefore, edge orientation is a good guide of boundary position. Values for β of 0.0, 0.0, 0.0, and 0.1 indicate that distance from the texture estimate provides little information.

The figures for inspection error from the texture based estimate cover a wide range. The figures for inspection error after boundary refinement show a very marked improvement. For 0°/90° (**Table (8.1)**), the mean error over the six lay-ups is better than 0.5 pixel. The maximum deviation over each set of 25 images is consistently under 1 pixel, and the overall maximum deviation is 0.8, which corresponds to an absolute error of 0.16mm. Results for the other configurations show a similar pattern with a maximum deviation in millimetres of 0.21, 0.21, and 0.18 for 45°/0° lay-ups, 45°/90° lay-ups, and 0°/0° lay-ups.

0°/90° 1 Pixel is $0.2mm^2$	$e_{target} = 0.18$ $\alpha = 5.0$ $\beta = 0.0$				
Lay-up #	Pixel Error in Texture estimate		Pixel Error After refinement		
	MEAN	MAX	MEAN	MAX	MAX(mm)
1	2.39	2.76	0.31	0.42	0.08
2	1.95	2.37	0.24	0.57	0.11
3	1.43	1.7	0.44	0.80	0.16
4	1.31	1.69	0.18	0.52	0.10
5	1.83	2.16	0.30	0.51	0.10
6	1.83	2.19	0.21	0.33	0.07

Table (8.1). Boundary errors before and after boundary refinement for 0°/90° lay-ups. Each lay-up was imaged and processed 25 times. The mean error and maximum error are reported above. All measurements in pixels, except where otherwise stated.

45°/0° 1 Pixel is $0.2mm^2$	$e_{target} = 0.22$ $\alpha = 3.0$ $\beta = 0.0$				
Lay-up #	Pixel Error in Texture estimate		Pixel Error After refinement		
	MEAN	MAX	MEAN	MAX	MAX(mm)
1	0.39	0.6	0.44	0.76	0.15
2	3.30	3.57	0.51	0.9	0.18
3	3.4	3.87	0.68	1.07	0.21
4	1.83	2.07	0.82	1.00	0.20
5	2.64	2.95	0.36	0.63	0.13
6	1.65	1.00	0.54	0.91	0.18

Table (8.2). Boundary errors before and after boundary refinement for 45°/0° lay-ups. Each lay-up was imaged and processed 25 times. The mean error and maximum error are reported above. All measurements in pixels, except where otherwise stated.

45°/90° 1 Pixel is $0.2mm^2$	$e_{target} = 0.26$ $\alpha = 5.0$ $\beta = 0.0$				
Lay-up #	Pixel Error in Texture estimate		Pixel Error After refinement		
	MEAN	MAX	MEAN	MAX	MAX(mm)
1	1.32	2.14	0.48	0.79	0.16
2	1.21	1.69	0.61	0.91	0.18
3	1.08	1.70	0.84	1.02	0.20
4	2.25	2.65	0.59	1.08	0.21
5	2.77	3.16	0.66	0.89	0.18
6	1.03	1.83	0.21	0.37	0.07

Table (8.3). Boundary errors before and after boundary refinement for 45°/90° lay-ups. Each lay-up was imaged and processed 25 times. The mean error and maximum error are reported above. All measurements in pixels, except where otherwise stated.

0°/0° 1 Pixel is $0.2mm^2$	$e_{target} = 0.23$ $\alpha = 2.5$ $\beta = 0.1$				
Lay-up #	Pixel Error in Texture estimate		Pixel Error After refinement		
	MEAN	MAX	MEAN	MAX	MAX(mm)
1	1.82	2.04	0.40	0.76	0.15
2	1.85	2.50	0.41	0.57	0.11
3	3.83	4.06	0.65	0.92	0.18
4	2.76	3.26	0.73	0.91	0.18
5	3.96	4.26	0.59	0.86	0.17
6	3.15	3.70	0.58	0.79	0.16

Table (8.4). Boundary errors before and after boundary refinement for 0°/0° lay-ups. Each lay-up was imaged and processed 25 times. The mean error and maximum error are reported above. All measurements in pixels, except where otherwise stated.

The raw data used to compile **Tables (8.1)** to **(8.4)** is shown in **Figure (8.7.1)**. This shows clearly the consistency of the results obtained. From all the tests performed, involving the processing of 600 images, on only 3 occasions was the estimated inspection error greater than 1 pixel: 1.07 for 45°/0°, 1.02 and 1.08 for 45°/90°. The improvements gained by boundary refinement can be appreciated by reference to **Figure (8.7.2)** which shows the texture estimate results and the boundary refinement results on the same graph. The reduction in inspection error is clear. It has to be concluded that boundary refinement can be used to provide a more accurate estimate of ply boundaries in inspection of carbon fibre workpieces. From the data presented here, the inspection error is of the order of ±0.2mm or less.

8.8 Conclusions.

This chapter has described a novel boundary refinement process developed in an attempt to reduce the inspection errors inherent in texture based boundary estimates. The algorithm takes the form of a local search, using the texture estimate as a guiding template. The points selected on each search are those which maximise a merit function of edge magnitude, edge orientation, and distance from the texture estimate. The merit function requires three parameters to be set to provide good performance: $\mathbf{e}_{target}$ specifies the probable edge magnitude of a point on the ply boundary; α weights the edge orientation term; β weights the distance from the texture estimate term. Values for these terms are obtained for each edge configuration using multiple training images in conjunction with simple function optimisation algorithms. The effect of boundary refinement has been examined over a representative range of ply configurations. The inspection error obtained was of the order of ±0.2mm or better.

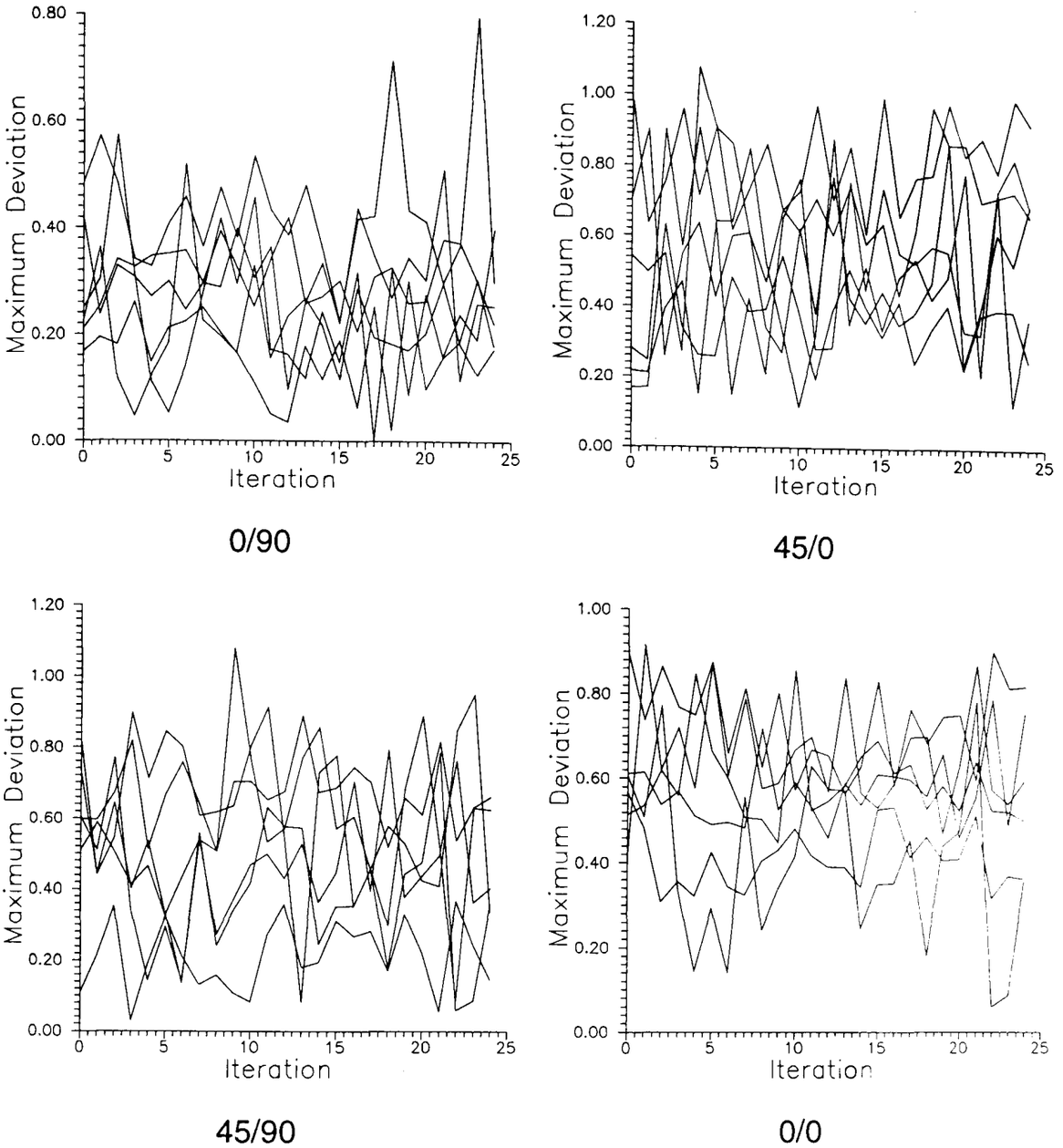


Figure (8.7.1). Graphs showing the results of accuracy testing for various ply configurations. Each graph gives the maximum deviation from the real edge measured over 25 iterations for six different lay-ups.

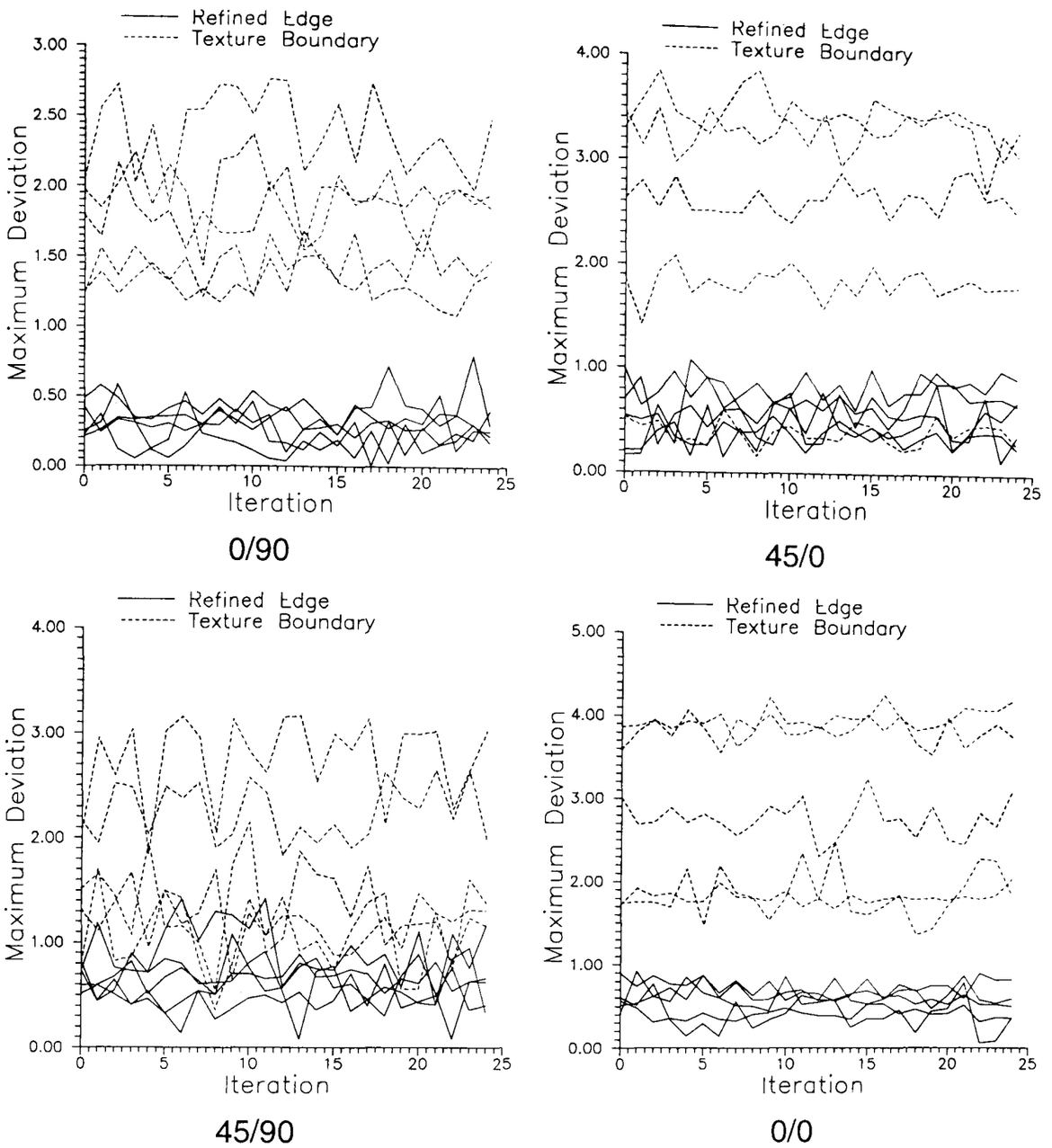
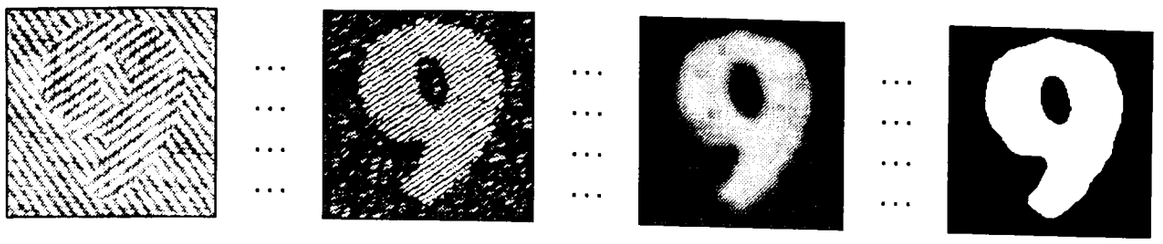


Figure (8.7.2). Graphs showing the relative improvement in inspection accuracy resulting from the edge refinement process. Each graph shows the maximum deviation from the real edge for both texture boundary and refined boundary measured over 25 iterations for five different lay-ups.



CHAPTER 9

Application of Techniques.

9.1 Introduction.

To date this thesis has concerned itself with the development and evaluation of texture based algorithms designed for industrial inspection of dry fibre composite lay-ups in the aerospace industry. The algorithms have been designed with industrial constraints in mind. They are computationally simple in comparison to many algorithms developed in academia, and would probably not stand direct comparison with more sophisticated methods. However they can be implemented in near-real-time on low cost commercially available hardware, and so offer an industrial readiness which more sophisticated methods do not. It is the aim of this chapter to put some flesh on this claim by using the algorithms developed to implement an inspection system as part of a prototype automated assembly cell. The performance of the cell and inspection system will be demonstrated using a sample lay-up application. The prototype assembly cell is described in detail in **Sections 9.2** through **9.5**. **Section 9.6** introduces the sample component, and **Section 9.7** details the inspection results obtained when laying-up the sample component. **Section**

9.8 considers the ability of the inspection process to detect lay-up errors, and **Section 9.9** presents the conclusions of the chapter.

9.2 Cell Hardware.

As detailed in **Chapter One**, the work described in this thesis has been carried out as part of a larger multidisciplinary project aimed at developing the enabling technologies required for automated lay-up of dry fibre composite components. Until recently the various research areas, including inspection, have to a large extent developed in isolation. A recently awarded EPSRC equipment grant facilitated the purchase of a computer controlled cutting table, and so removed the main obstacle to the development of a fully functioning prototype assembly cell encompassing all the research of the group. The hardware of the cell is shown in **Figure (9.2.1)**. The components of the cell are as follows: ABB IRB3000 articulated robot and controller; FANUC S10 articulated robot and controller; computer controlled cutting table and controller; electrostatic gripping device (EGD); vision system; tacking device; lay-up table. The main components are described in more detail in the following sections.

9.2.1 The Computer Controlled Cutting Table.

The cutting table's task is to cut the required ply shapes from a sheet of the appropriate material. The cutting table is basically a two-axis gantry robot, fitted with a CO_2 pulse laser which performs the cutting, as shown in **Figure (9.2.1.1)**. The cutting area of the table is 2.5 metres by 1.25 metres. The parameters of the system (i.e. cut speed, laser power etc.) can be adjusted depending on the material to be cut. The table is controlled by a software package called DNC3 running on a PC connected to the table via an RS232C link. The geometry of each ply in the component is designed using Autosketch, a PC based CAD package. DNC3 imports the data files generated by Autosketch. The position on the table where the plies are to be cut can be defined in Autosketch, or manually controlled in DNC3. The table is not at present capable of performing as part of a fully automated cell since it can

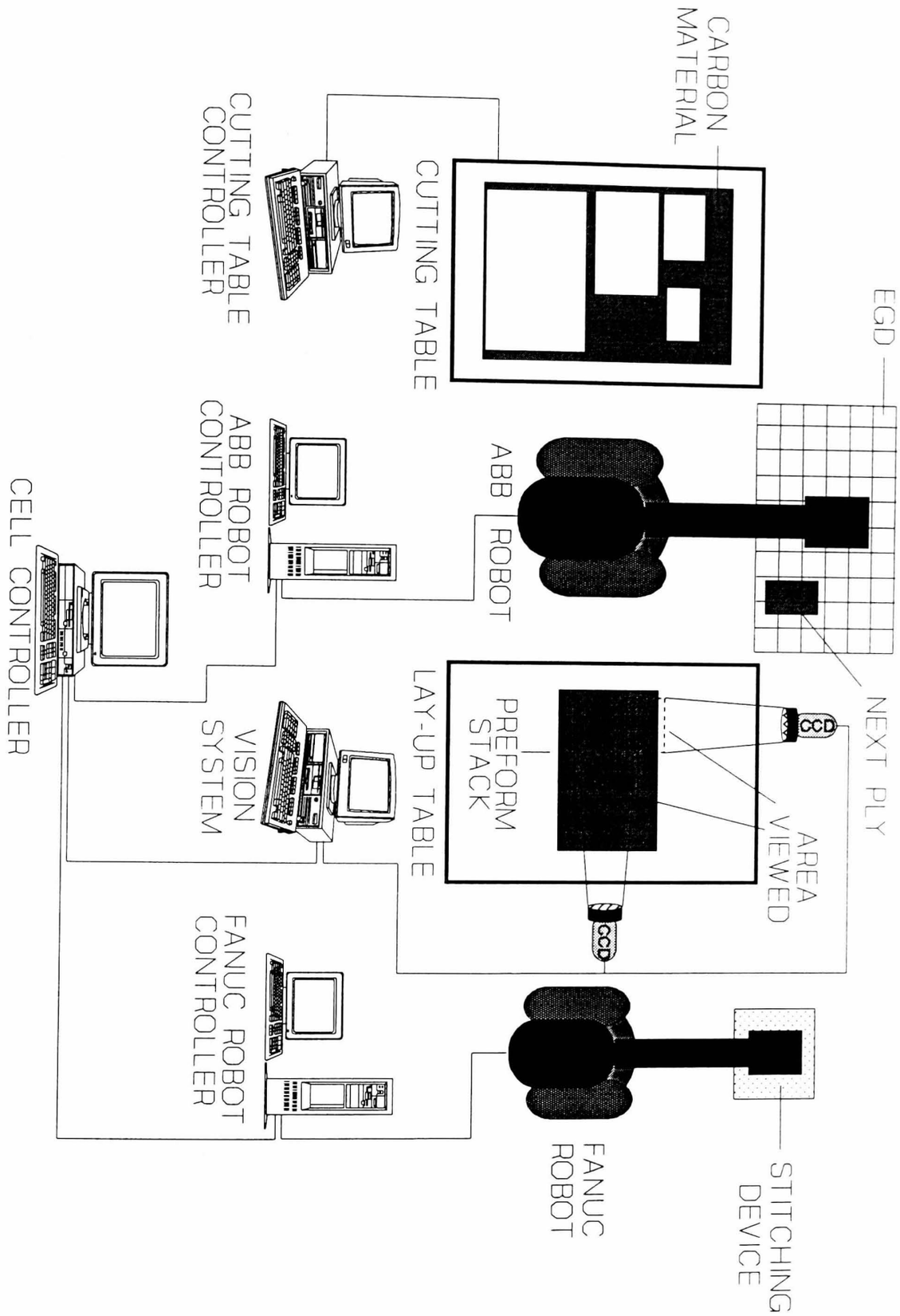


Figure (9.2.1). Hardware of the prototype assembly cell.

Figure (9.2.1.1). The computer controlled cutting table. The laser head is surrounded by a case of special plastic which acts as a filter to reduce the possible hazard of beam reflectance.

only be controlled interactively. That is, it cannot be controlled remotely by the cell control PC. It is hoped that this will be overcome by a revised version of the cutting table software, but at present human interaction is required to synchronise the cutting operation with the lay-up cycle.

Previously in this thesis, all plies inspected have been cut manually using a knife. Plies inspected within the cell however, will have been cut by laser. This process has the effect of charring the edges of plies, producing a small dark area on the edge. The effect of this charring is, if anything, beneficial to the vision system. Texture analysis and texture edge operators are not really affected, but the boundary refinement stage has a slightly easier task since the dark edges produce slightly more contrast at ply boundaries. In addition, laser cutting has the effect of "sealing" ply edges, which prevents fibres on the edge from fraying (the effect of which was noted to be detrimental in **Section 7.4.4**). From the point of view of the inspection task therefore, laser cutting of plies is very suitable. However, the cutting process

has yet to be properly optimised for carbon fibre materials. Plies are not always cleanly cut, and frequently require to be manually freed from the surrounding waste material before the robotic pick operation can proceed. The position of the ply may be disturbed slightly during this manual intervention, and so the figures for lay-up accuracy (given in **Section 9.7.2**) are not yet indicative of the potential of the automated lay-up cell.

9.2.2 The Lay-Up Table and Inspection Frame.

The plies are laid-up on a table with a flat aluminium top shown in **Figure (9.2.2.1)**. A frame has been constructed around the table on which cameras and/or lights can be mounted as required, and this is shown in **Figure (9.2.2.2)**. At present only a single camera is mounted on the frame. Lighting is provided by shining spotlights mounted on the table onto a white sheet suspended over the frame. In this way, the lay-up is illuminated by a very diffuse light. This is the best way to light carbon fibre since any direct light produces strong highlights in an image.

9.2.3 The Lay-Up Robot.

The plies are transported between the cutting table and the lay-up table by an electrostatic gripping device (EGD) attached to an IRB3000 articulated robot. The development of the EGD has been one of the fundamental research areas of the project **[Chen, Sarhadi, 1992]**. It is fully software configurable and so provides much greater flexibility than other gripping devices, such as vacuum grippers. The IRB3000 is controlled via RS232C link by the cell controller. The accuracy of the robot is quoted as ±0.15mm or better. The resolution of movement under PC control is 0.125mm on each of the x,y, and z axes. **Figure (9.2.3.1)** shows the lay-up robot with gripper attached. The cutting table and lay-up table are also visible.

Figure (9.2.2.1). The lay-up table.

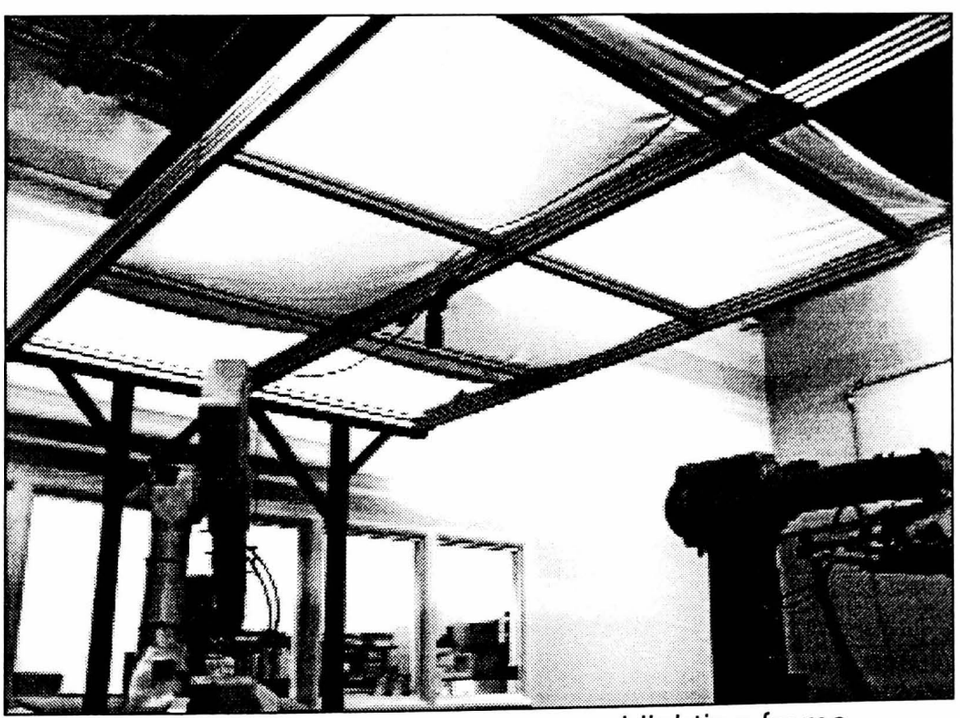


Figure (9.2.2.2). The camera and lighting frame.

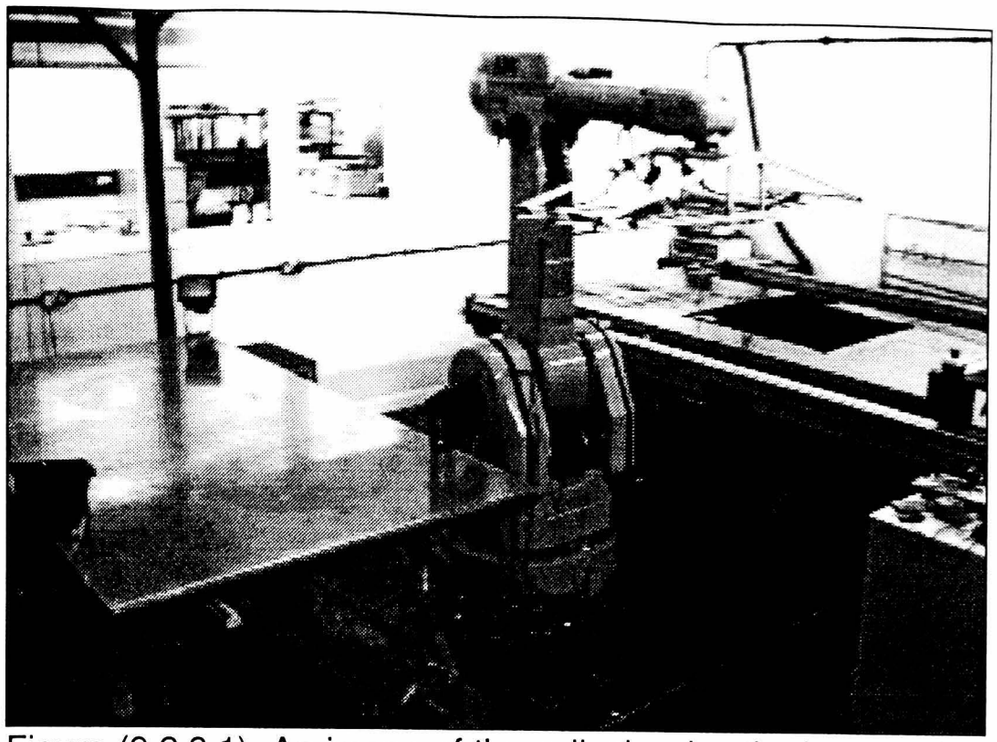


Figure (9.2.3.1). An image of the cell, showing the lay-up robot and EGD.

9.2.4 Vision System Hardware.

All the algorithms described in this thesis have been developed on a transputer based vision system comprising three distinct entities: a framegrabber card housing one T425 transputer, a transputer motherboard housing five T800 transputer modules (TRAMS), and a pipeline processing card housing one T425 transputer and several A110 processors configured to provide a fast convolution capability. A host PC running under DOS provides system I/O. It is this vision system which is currently used in the automated assembly cell. All software on the vision system has been written in OCCAM using the Transputer Development System (TDS). The hardware architecture of the system is shown in **Figure (9.2.4.1)**. The texture analysis process detailed in **Chapter Three**, and the mask optimisation processes detailed in **Chapter Five** have been implemented on the pipeline machine to maximise processing efficiency. An image can be passed to the pipeline card, segmented by texture, and returned to the framegrabber card in approximately one second, the bulk of this time being taken up by image transfer overheads **[King, 1994]**.

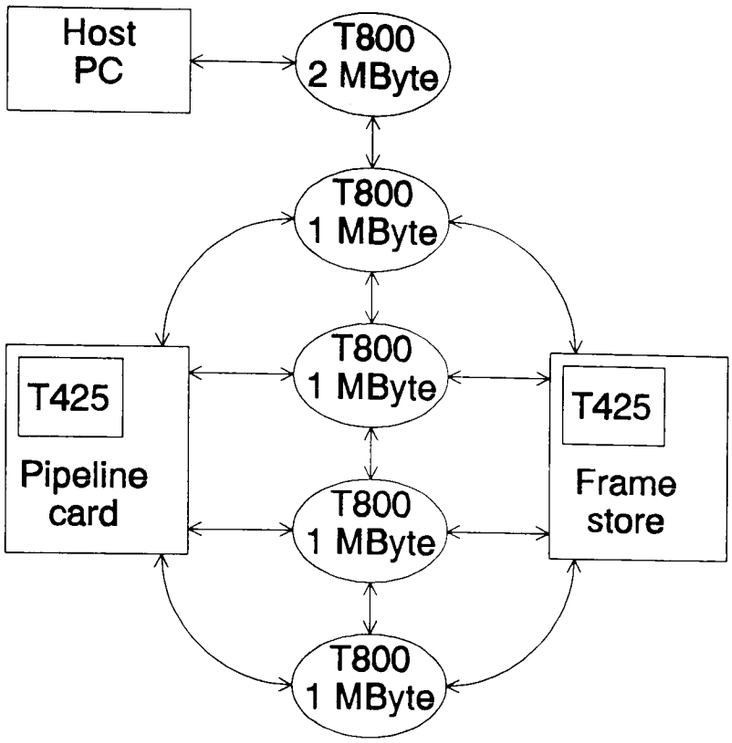


Figure (9.2.4.1). Diagram of the vision system hardware. The ellipses represent TRAMS, the arrows each represent a transputer link. From **[King, 1994]** with permission.

Currently available low cost framegrabbers and processors are capable of performing the required operations in substantially less time, due to the high bandwidth provided by dedicated video buses. At the moment the processing power of the transputer network is largely unused, save for the fast parallel calculation of the grey-level histogram used in some thresholding algorithms. The remaining processing is carried out on the framegrabber card. This card has video RAM which is a great advantage when developing algorithms, but does entail a longer memory cycle time. These processes are, however, parallel in nature, and so could be ported to run on the transputer network if increased speed using this system was required. Such an operation generally results in a speed increase of slightly more than 500%, due to the superior clock rate and memory cycle time of the TRAMS compared to that of the framegrabber card.

The host PC of the vision system is connected to the cell controller via an RS232C link, which enables the inspection process to be synchronised with

the lay-up process, and the result of the inspection related to the cell controller.

9.2.5 Tacking Device.

It is desirable to tack each newly laid-up ply to the stack of previously laid-up plies to prevent subsequent movement and enable manual transportation of the completed preform stack. Considerable research into the tacking process has been carried out over the years, and early work in this area within the research group led to a successful implementation and understanding of heat bonding **[Willcox, 1994]**. However this approach is not appropriate for all applications since the impurities introduced by the thermoplastic are unacceptable in some structurally critical component sections. As a result, a new tacking method based on stitching is currently under development. A stitching device mounted on a FANUC S10 articulated robot is used to tack each ply to the preform stack. As yet reliability is poor, and so for the purposes of the work described in this chapter the tacking operation will be omitted. This causes no problems, since the lay-up process using the EGD produces no movement of previously laid plies (a possible side effect of vacuum grippers). The stitching device and the Fanuc S10 robot are shown in **Figure (9.2.5.1)**. Also attached to the S10 is the laser inspection system described in **Section 10.4.1**.

9.2.6 Cell Controller.

The cell controller is a PC through which all components of the cell are linked. Cell control software is an invisible component of the cell, but a very important one. The main tasks performed include generation and control of data files for each component, robot control, calibration of cutting table with robot, calibration of vision system with robot (discussed in **Sections 9.3.1** and **9.3.2** respectively), configuration and control of gripper, communication with and synchronisation of each component in the cell. The way in which the cell is designed to operate is described in the following sections.

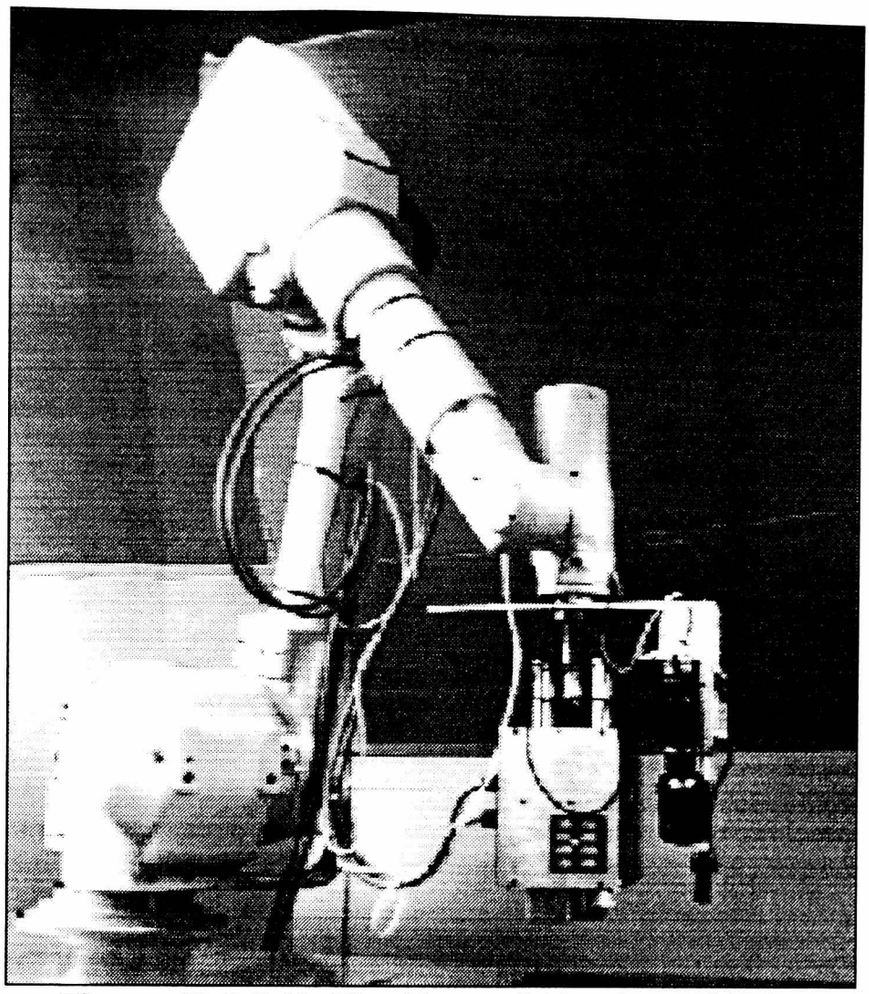


Figure (9.2.5.1). The Fanuc S10 robot, with the stitching device attached. Also shown is the laser triangulation device currently under investigation within the research group.

9.3 Cell Set-Up.

The objective of this section is to describe the various processes required to configure the cell for operation. The diagram of **Figure (9.3.1)** illustrates the requirements which must be fulfilled before the cell can operate. These can be divided into two different types of operation, summarised as follows.

System set up consists of three main processes: calibration of the lay-up robot against cutting and lay-up tables, calibration of the vision system against the lay-up robot, and creation of a database of inspection information for various materials used by the cell. Ideally the calibration processes would be carried out only once. In practice however, periodic calibration would be required because of camera movement due to vibration etc.

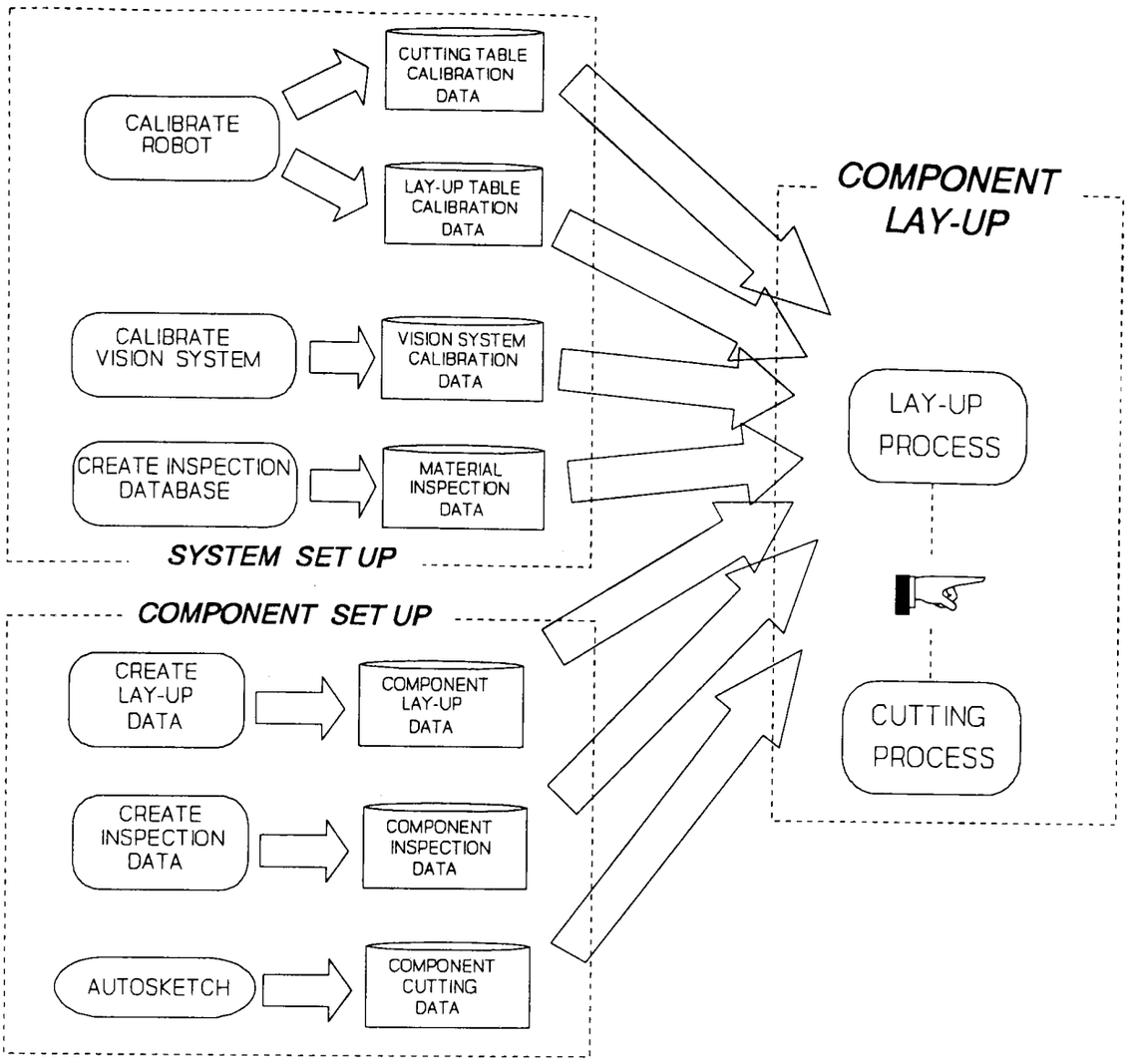


Figure (9.3.1). Operational structure of the cell.

The more likely problem is some movement of the camera/lens system, and with this in mind the vision system calibration process has been designed to be capable of automatic re-calibration before lay-up of each component.

The third operation required for system set-up is the creation of a file of information specifying how the various ply configurations should be inspected. For example, to inspect a ply of unidirectional material at 45° on a ply of unidirectional material at -45°, what mask should be used, how much smoothing is required, what threshold is to be used, etc? The creation of this file enables automatic generation of inspection parameters for a new component, providing the particular inspection configuration represented by each ply has previously been included in the material inspection data file. If this is not the case then the material inspection data file must be updated.

This approach to automatic generation of inspection parameters assumes that lighting conditions are consistent.

Component set up is required only once for each design of component to be manufactured. The aim of this process is to create the data files containing all the information the cell requires to perform lay-up. This includes a file containing information specific to the inspection task. This file is later used by the vision system.

Sections 9.3.1 and **9.3.2** describe the requirements for lay-up robot calibration and vision system calibration respectively, and provide an overview of the methodology adopted. **Section 9.3.3** describes the process for creating the inspection data for each material configuration. **Section 9.3.4** describes the creation of component specific data files.

9.3.1 Lay-Up Robot Calibration.

The lay-up robot must be calibrated with both the cutting table and the lay-up table. For the cutting table the objective is to be able to map the position of a newly cut ply in cutting table coordinates to the corresponding robot coordinates. To this end the robot is "taught" the position of various points on the cutting table using a specially designed tool. Once the coordinates of these points are known in both cutting table coordinates and robot coordinates, then a mapping between coordinate systems is possible. The details of this mapping are described in **Appendix B**.

Calibration of the robot and the lay-up table is, at this stage, merely concerned with obtaining the centre point of the table, a feature which is useful in determining component lay-up position. A more critical calibration with the lay-up table is carried out as part of the vision system calibration process described in the next section. The functionality of the lay-up robot calibration is illustrated in **Figure (9.3.1.1)**.

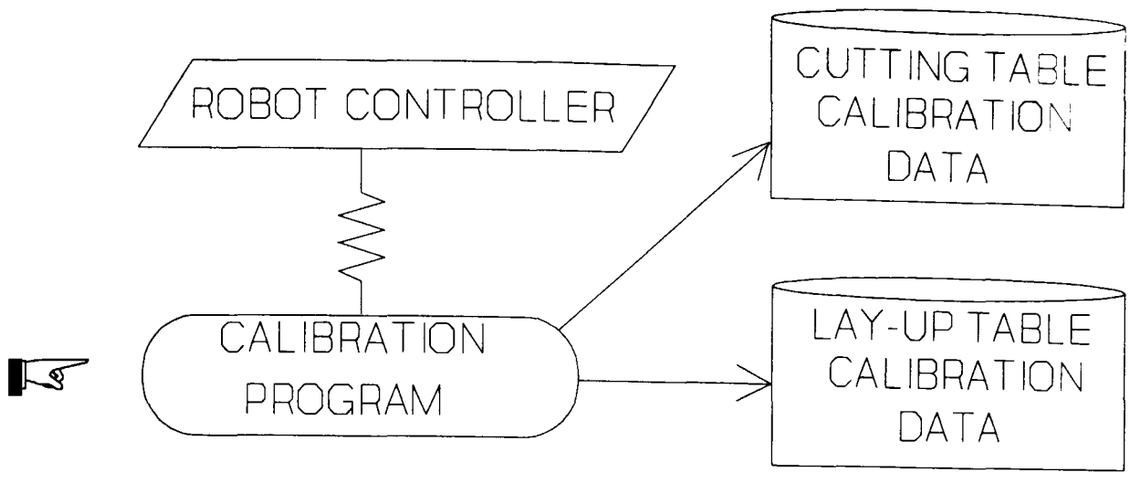


Figure (9.3.1.1). Calibration of the lay-up robot. The link between the robot controller and the calibration program is via an RS232C link. The "hand" icon indicates the requirement for human interaction.

9.3.2 Vision System Calibration.

In **Chapter One** the criteria for inspection of plies in the lay-up was described. Part of those criteria is that the ply boundaries detected by the vision system should be compared against the specification in the component design. Exactly how this is to be accomplished has not so far been detailed. Ideally, inspection should be based on the CAD data created when the component is designed. In order to achieve this a means of mapping the CAD coordinates to the vision system coordinates is required. The approach adopted here is to map both CAD coordinates and vision system coordinates to robot coordinates. The robot coordinate system is therefore used as a common coordinate system enabling direct comparison between CAD data and vision system results. The CAD data must anyway be mapped into robot workspace for ply lay-up. Mapping vision system coordinates into robot workspace therefore offers an elegant solution to the inspection problem, enabling easy comparison between CAD data and vision system data.

To map vision system coordinates into robot workspace, the robot and vision system must be calibrated together. This task has been investigated in some detail by a another researcher within our group involved in garment assembly **[Jones, 1994]**. The basic idea is that for a number of co-planar points in the camera field of view, the vision system coordinates and the

corresponding robot coordinates are obtained. Any image point can then be mapped to robot coordinates by reference to the nearest taught point, and an offset from that point which is a function mapping the offset in pixels to an offset in robot coordinates. The accuracy of the overall mapping is dependant on the number of taught points (since for a fine array of taught points the offset from any image point is small and so the potential error in offset small), and the function used to map the offset. For the garment assembly application, linear mapping functions were initially used with some success, but a significant increase in accuracy resulted from a more sophisticated spline based mapping function **[Jones, 1994]**. The degree of sophistication required for calibration in the composite assembly cell should be less for two reasons. Firstly the focal length of the lens in these experiments is 70mm, versus 16mm in garment assembly, and a long focal length reduces lens distortion. It is the lens distortion in the garment assembly cell which necessitated the use of a spline mapping function. Secondly, the field of view in the composite assembly cell is less, and so a one pixel error in the composite assembly cell represents a smaller absolute error than would be the case in the garment assembly cell. For these reasons a linear mapping approach has been adopted to calibrate robot and camera.

The area of the lay-up viewed by the camera is marked by a 3x3 array of targets. The coordinates for the centre of each target in both robot and vision system coordinates are obtained, and these are referred to as *taught points*. A mapping function uses this calibration data to enable mapping between robot and vision coordinates. The details of the calibration and mapping procedures are to be found in **Appendix B**.

The functionality of the vision system calibration in terms of system data creation is illustrated in **Figure (9.3.2.1)**.

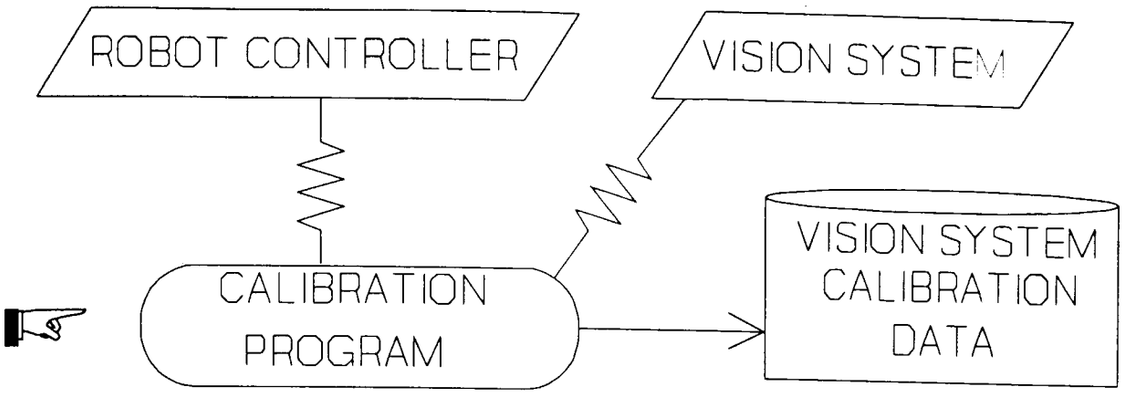


Figure (9.3.2.1). Robot and vision system calibration.

9.3.3 Material Inspection Data Creation.

Much of the effort in the development of the prototype assembly cell has been directed towards introducing as much flexibility as possible. The eventual goal, although some way off, is to provide a flexible manufacturing cell which can be configured to assemble a new component design directly from CAD. That is, all the cutting, pick and place, tacking and inspection parameters could be automatically generated without the requirement for any reprogramming. A scheme to evaluate the feasibility of this approach for the inspection task has been adopted here. This involves the creation of a database of information relating to the inspection of different material configurations.

For example, for unidirectional material, masks would be trained to perform boundary detection for ply configurations of 0°/0°, 45°/45°, etc, and texture analysis for ply configurations of 0°/±45°, 0°/90°, ±45°/90°. For each orientation the mask, smoothing performed, and threshold value used are noted. Boundary refinement parameters for each edge configuration are also obtained and noted, using the method described in **Section 8.5**. Once this information has been written to file, then for a component composed entirely of unidirectional material, the relevant parameters to inspect each edge can be automatically selected using the component CAD data.

Lighting conditions must be very stable for this approach to succeed. The lab area housing the prototype assembly cell is subject to considerable lighting fluctuations due to the influence of sunlight. The lay-up experiments

detailed later in this chapter were successfully carried out over an afternoon, and no problems were encountered. The performance of the masks themselves is generally robust, and the inspection process is not dependant on an optimum threshold selection, since the boundary refinement stage determines the final accuracy. However, a final decision regarding this approach must await a full evaluation of the cell at a future date.

The functionality of the process in terms of system data creation is illustrated in **Figure (9.3.3.1)**.

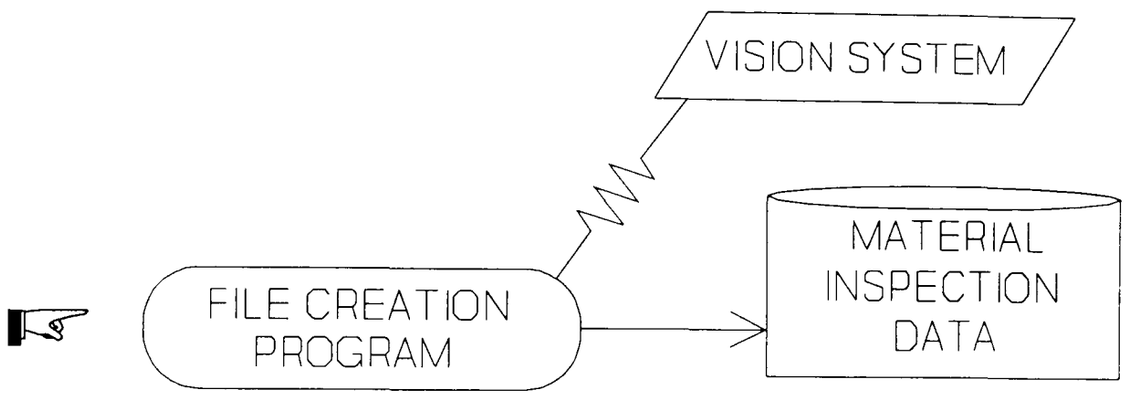


Figure (9.3.3.1). Creation of material inspection data..

9.3.4 Component Data Creation.

To set up the system to lay-up a particular component, the relevant data files must be created as shown in **Figure (9.3.4.1)**. At present there are three stages involved in creating the required data files. One for the cutting table using Autosketch, one for the lay-up robot using the *compfile* and *gripper* programs, and one for the vision system using the *visfile* program. This is obviously undesirable since inconsistencies due to operator error are possible. However at present no means of extracting the relevant data from the CAD file produced by Autosketch is available.

The lay-up data file for each component is created by the *compfile* program, which prompts the user for information such as the number of plies in the component, the dimensions of each ply, the position on the cutting table, the target position on the lay-up table, material type and orientation, etc. The *gripper* program enables the operator to define the area of the EGD used to

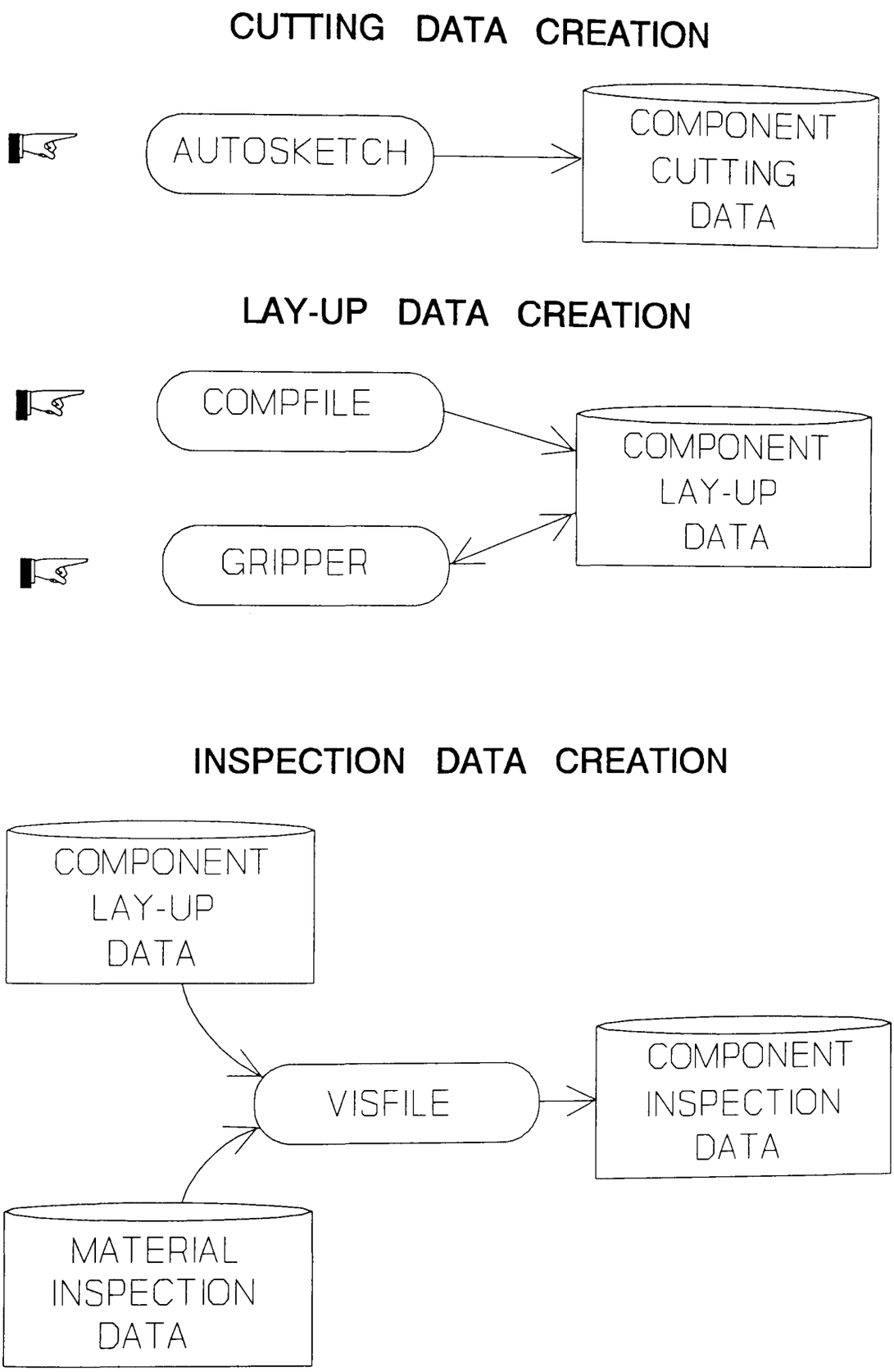


Figure (9.3.4.1). Creation of component specific data files.

pick each ply, and updates the data file with this information. This operation could in fact be performed automatically, but an appropriate method has yet to be devised.

In addition a component inspection file is created for use by the vision system. This file contains information regarding which masks, thresholds, etc. to use to process each image. This file is created by combining information from the component lay-up data file (material and orientation of foreground and background plies) with information from the material inspection data file (for a given combination of materials, which mask to use, what the threshold value is etc.). The result is a file specifying all the information necessary to process each image for each ply.

9.4 The Cell Lay-Up Cycle.

Once the various system and component set-up procedures have been completed, component lay-up can proceed. The lay-up cycle of the cell is depicted in **Figure (9.4.1).**

The first stage in lay-up is ply cutting, and this requires human interaction. DNC3 is used to control the table to cut the desired ply. The operator must then "tell" the cell control software which ply is to be picked. All other data required (cut position, lay-up position, gripper configuration etc.) is contained in the data files produced during set-up. The ply can then be picked and placed. The next stage is inspection of ply position by the vision system, the details of which are covered in **Section 9.5**. From the point of view of the cell controller, it need only send a "start" message over the RS232C link to the vision system, and await a message back indicating the result of the inspection.

9.5 Operation of the Vision System in the Cell.

This section will describe the operation of the vision system in the context of the cell, as well as detailing the actual image processing which takes place during inspection.

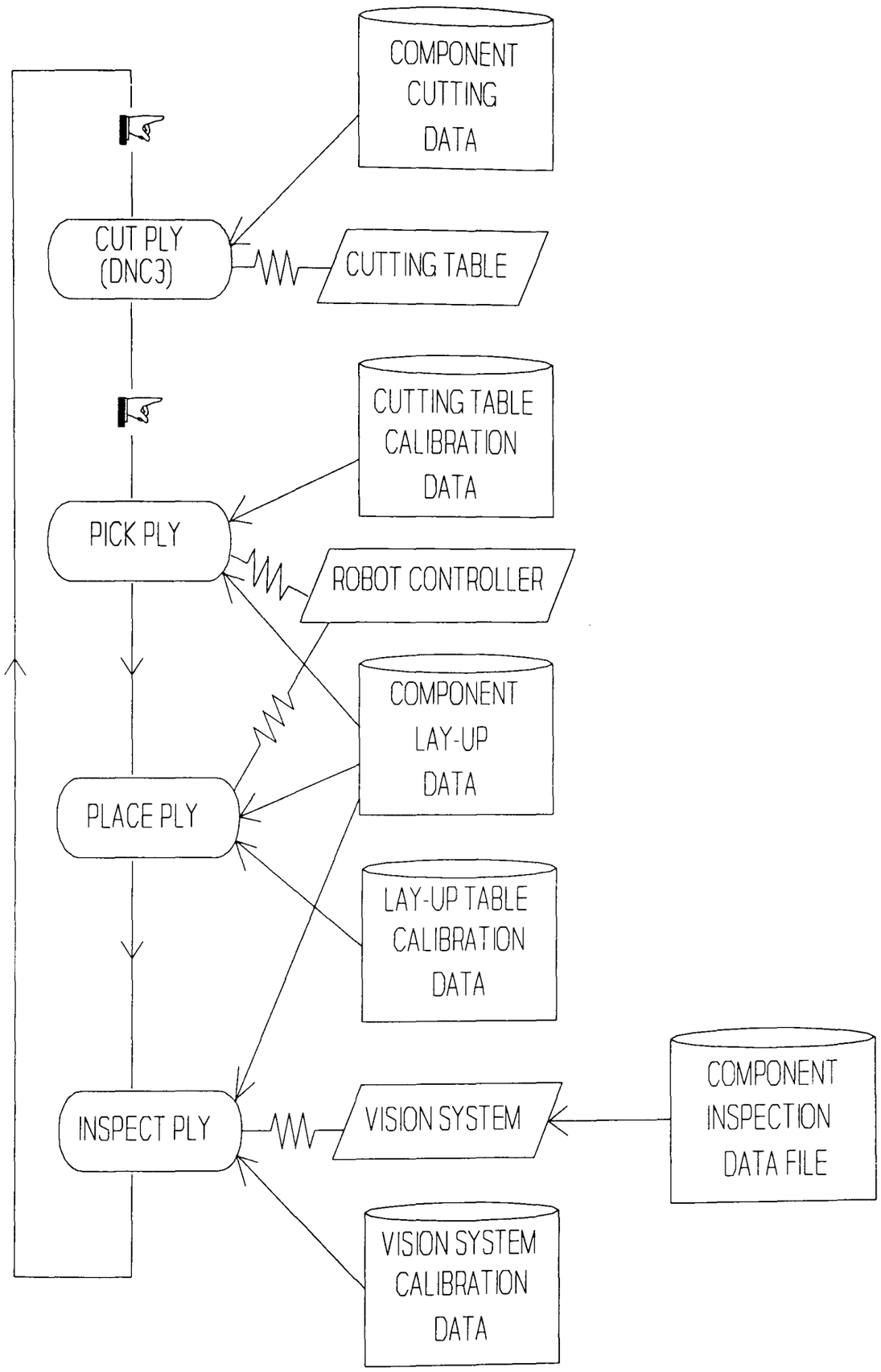


Figure (9.4.1). The lay-up cycle of the cell. Manual interaction is indicated by the "hand" icon.

9.5.1 Inspection Stage Overview.

Figure (9.5.1.1) illustrates the interaction between the cell controller and the vision system. Firstly the vision system reads the relevant component inspection file. The vision system then enters the component inspection loop. In this, the vision system awaits a message from the cell controller indicating which ply is to be inspected. A special *end-of-operation* message tag indicates that the lay-up process is at an end, and will cause the vision system to cease looping. If a ply number is passed, the ply number enables the relevant parameters (mask, threshold, etc.) to be selected from the data previously read from file. This approach provides a flexibility for the cell controller to request inspection of any ply in any order, a facility especially useful in development but also useful when problems in lay-up occur.

Once the relevant parameters have been selected, an image can be grabbed and processed as detailed in the following section. The last task for the vision system is to pass the position of the ply edge (or an appropriate error message if no edge was found) to the cell controller.

9.5.2 Image Processing Operations for Ply Inspection.

Figure (9.5.2.1) illustrates the processing carried out on each image. There are two main processing paths which can be followed, depending on whether the image is to be processed using texture analysis or a texture edge operator. In both cases a boundary refinement stage is performed. The only processes depicted in **Figure (9.5.2.1)** which may require some explanation are those relating to boundary extraction. For the texture analysis operation, object boundaries are extracted from the binary image as chain code. For texture edge operators the thresholded output (possibly after thinning) is considered to provide potential boundary points. In both cases the boundary points of interest are obtained by reference to the CAD data. From this the expected edge position in the image can be calculated (see **Appendix B**). The image area within a certain distance of this expected edge position (e.g. ±10 pixels) is considered to form an "area of interest". The only boundaries which will be processed are those which pass through this area of interest. If

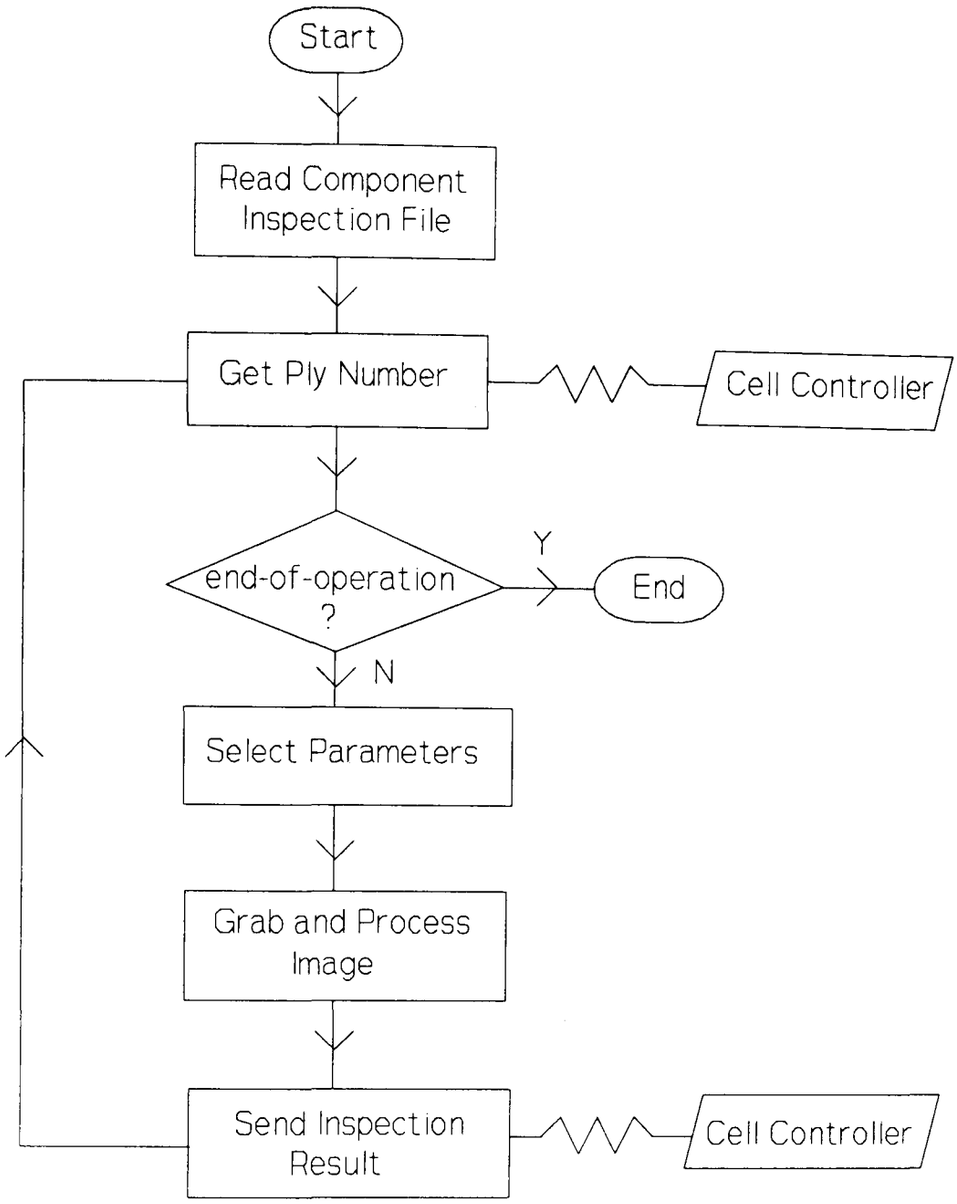


Figure (9.5.1.1). Interaction between vision system and cell controller.

insufficient boundary information is found in this area, then presumably the ply has not been laid-up or has been badly misplaced, and the vision system will report this with an appropriate error message.

All other processes depicted in **Figure (9.5.2.1)** have been described in previous chapters.

9.6 The Sample Component.

The sample component chosen to demonstrate the operation of the cell is illustrated in **Figure (9.6.1)**. It has a relatively simple structure and is

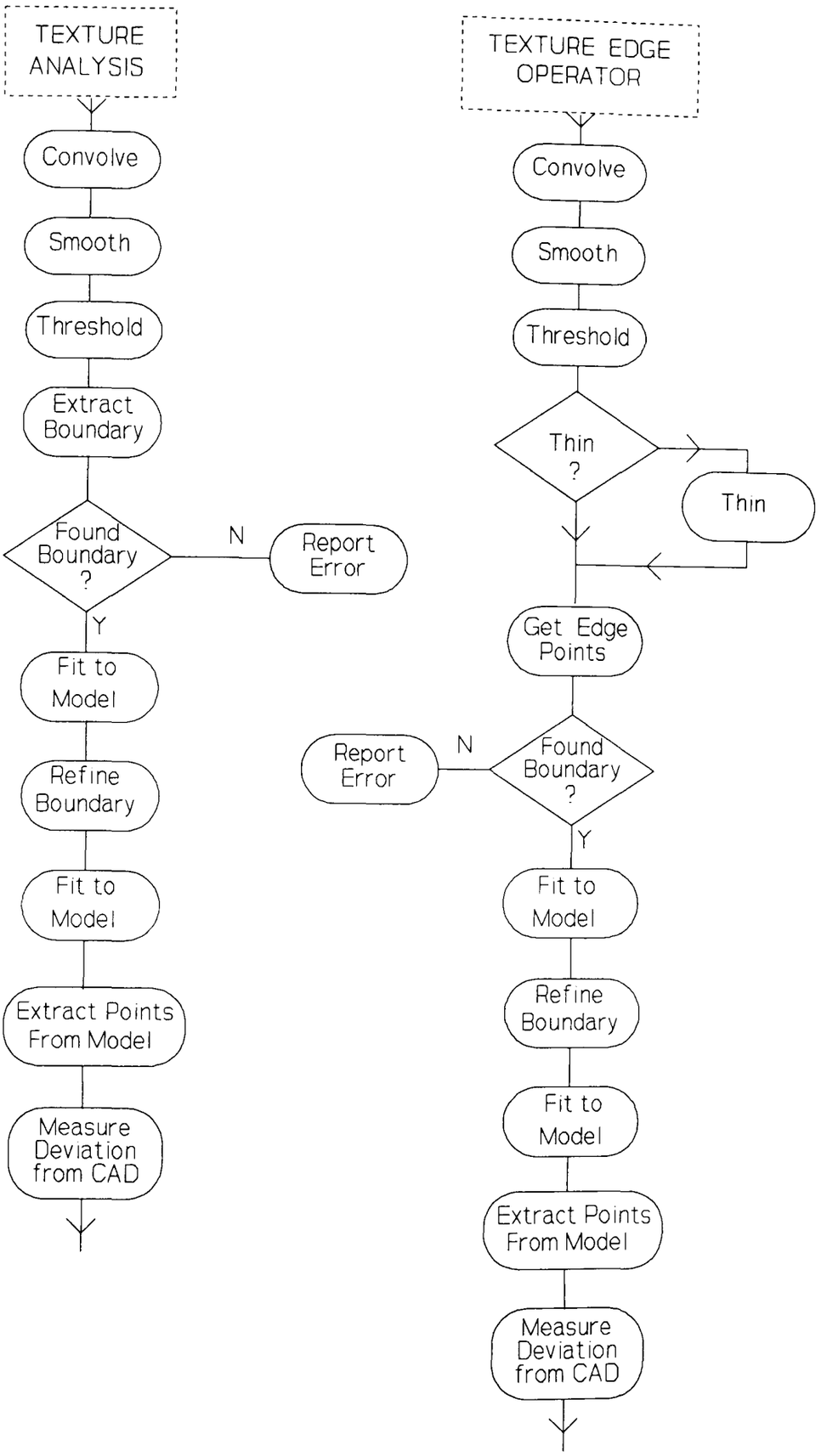


Figure (9.5.2.1). Image processing performed on each image inspected. Which of the above paths will be followed depends on whether the image is to be analyzed using texture analysis or a texture edge operator.

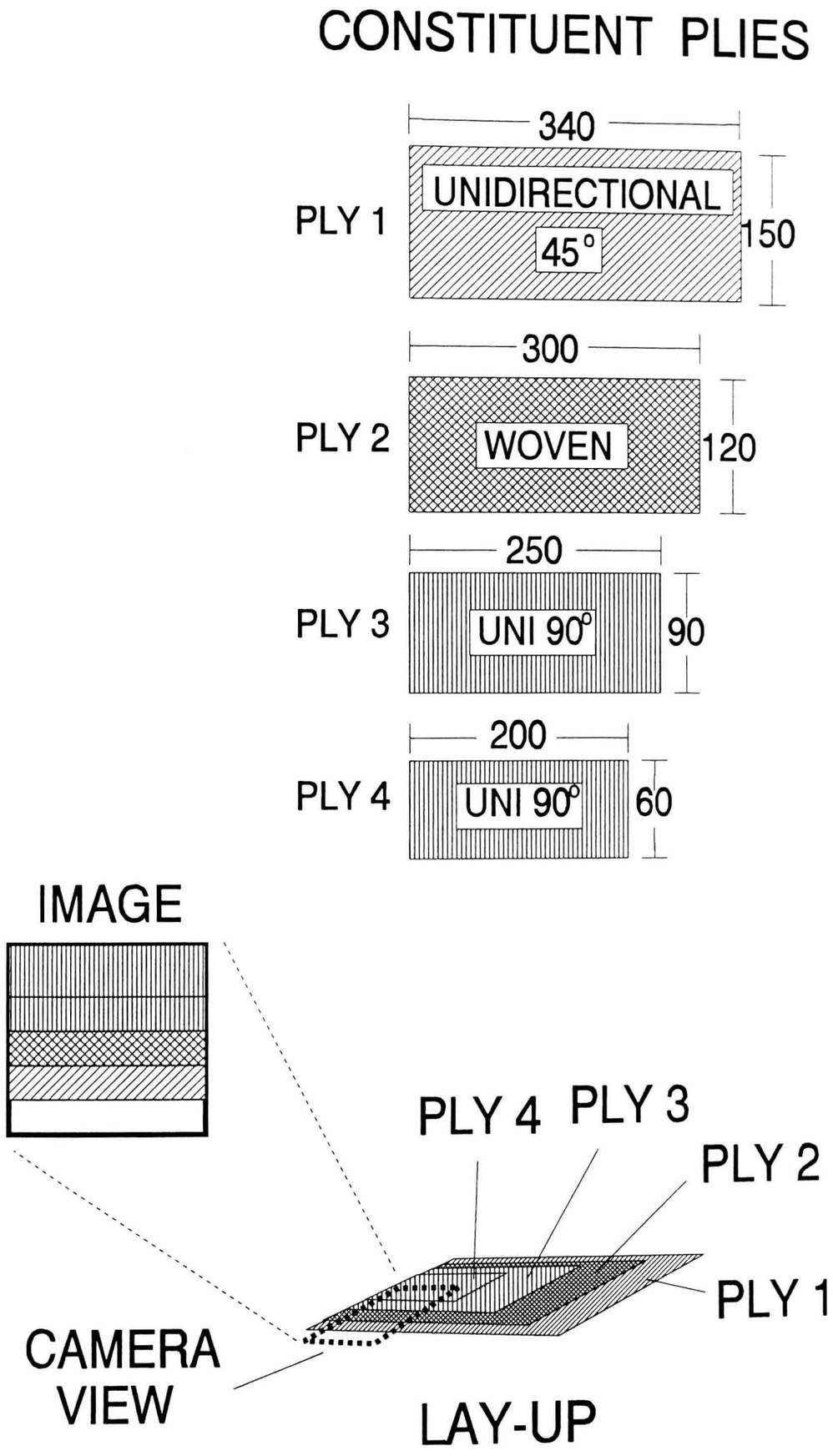


Figure (9.6.1). The sample component. All dimensions are in millimetres.

constructed only of rectangular shaped plies. This type of structure is commonly found in small aerospace components such as spars (used in structural reinforcement) and root wedges (used in propeller manufacture). The materials indicated are also typical of the application, with the majority of plies consisting of unidirectional carbon fibre at various orientations, interspersed with an occasional ply of woven carbon fibre. **Figure (9.6.1)** also indicates the approximate section of the component which will be inspected during lay-up. For production purposes multiple cameras could be employed to inspect other areas of the component as required. For the sample component one other camera viewing perpendicular ply edges would probably be sufficient. Using the techniques described in this chapter, the area viewed by each camera could be calibrated and processed independently with no need for registration between images. This approach is directly extendible to inspection of large components, although of course the hardware requirements will rise with the component size. The issue of inspecting larger components is considered in **Section 9.7.2**.

The tolerance for each ply in the lay-up is ±1 millimetre from the edge position specified in the component CAD data.

9.7 Cell Operation.

This section will detail the operation of the cell in laying-up the sample component. The sequence of operations is indicated and illustrated with images where appropriate, with particular attention paid to the role of the vision system. Interpretation of inspection data is considered, and the detection of lay-up errors investigated.

9.7.1 Lay-Up of the Sample Component.

The vision calibration process, as described in **Appendix B**, has been designed so that the vision system can be re-calibrated before each component lay-up. **Appendix B** provides an image sequence of this operation and the resulting calibration data.

The first step in the lay-up cycle is ply cutting. A sheet of the desired

material (in the case of ply one of the sample component, unidirectional carbon at -45°) is suitably positioned on the cutting table, and the cutting operation initiated. **Figure (9.7.1.1)** shows an image of the cutting table in operation. The very bright area in the image results from heat generated by the CO_2 laser which actually performs the cutting. After cutting, the gantry is withdrawn from the cutting area. The lay-up robot can then pick up the ply. **Figure (9.7.1.2)** shows an image sequence of the robotic picking operation, and an image of a ply on the gripper surface. The gripper has a modular design, with different areas of the gripper fitted with different dielectrics for testing purposes.

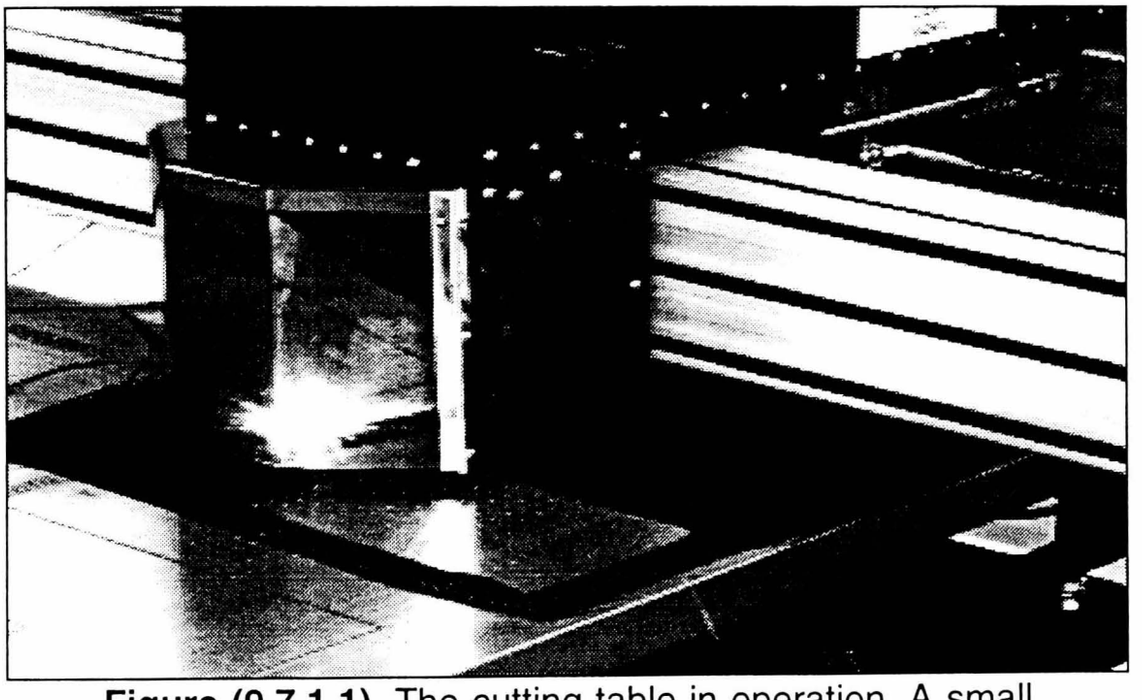


Figure (9.7.1.1). The cutting table in operation. A small rectangular ply is being cut. A rectangular hole is visible in the material where a previous test ply has been cut and picked.

The ply is then transported to the lay-up table and released. This is shown in the image sequence of **Figure (9.7.1.3)**. The cell controller then communicates with the vision system, indicating which ply is to be inspected. Vision system processing can then take place to obtain the position of the ply edge, and calculate deviation from the edge position defined in the CAD data. The mapping of CAD data into image space is achieved using the mapping function detailed in **Appendix B**.

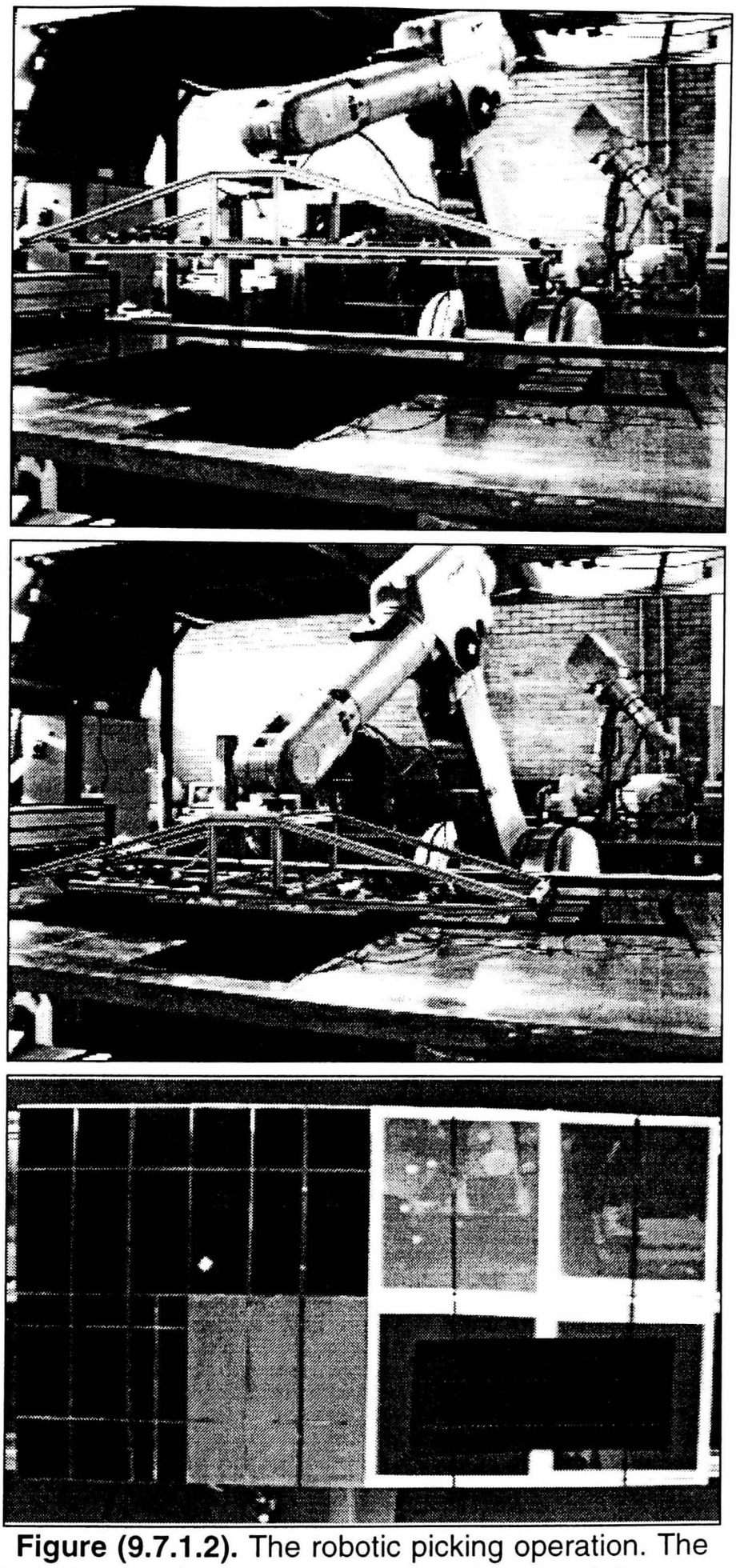


Figure (9.7.1.2). The robotic picking operation. The third image shows a ply on the underside of the gripper.

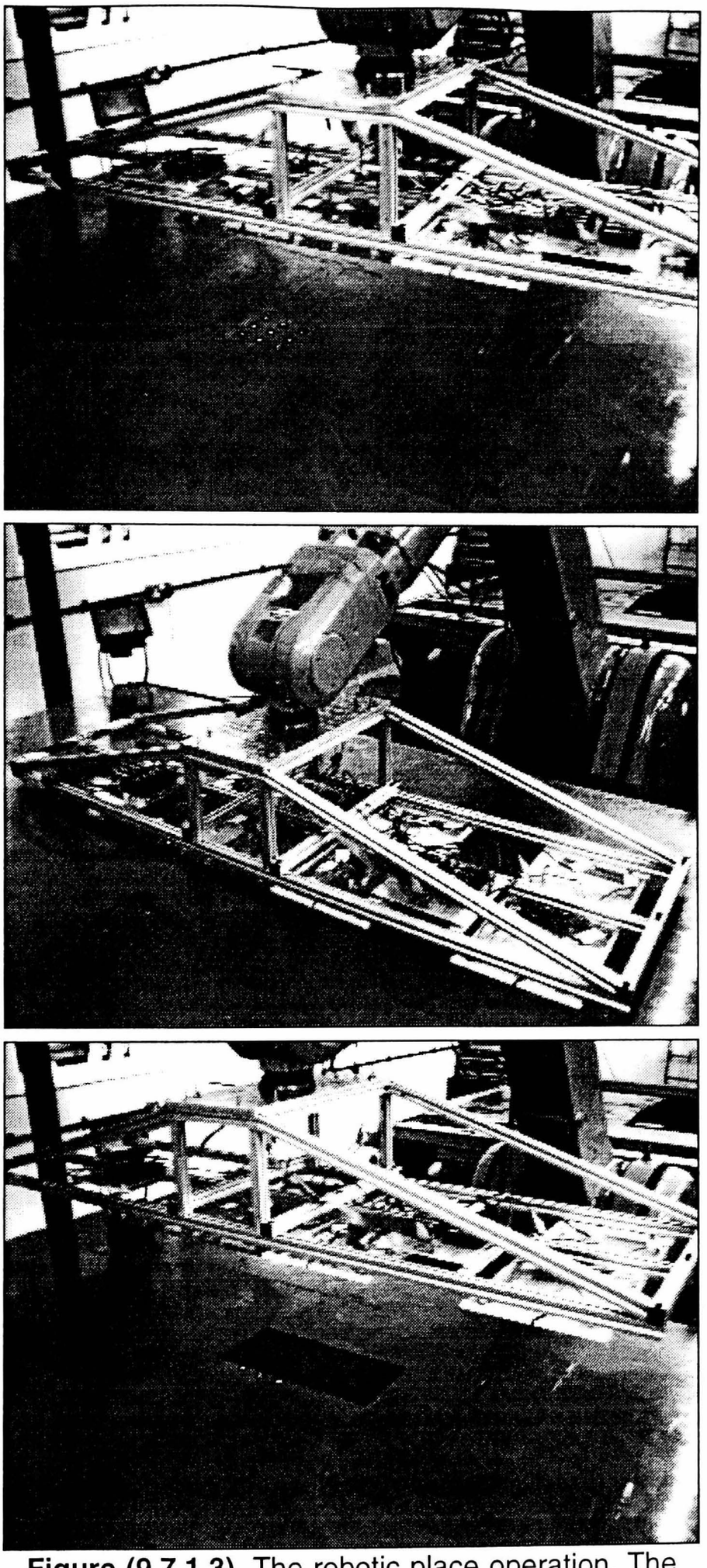


Figure (9.7.1.3). The robotic place operation. The array of targets on the table shows the area imaged during inspection.

For the purposes of these tests, the vision system does not provide a pass/fail result, but merely records the data relating to deviation from the CAD data. The lay-up cycle for the next ply can now begin. The waste material from the first cutting operation is removed, and a sheet of woven carbon material is placed in position. The cut, pick, and place operations for ply two can then proceed. The lay-up of plies three and four follow the same cycle. **Appendix C** contains all images captured by the vision system for these lay-up tests.

For the purposes of this chapter, four sample components were laid-up, shown together in **Figure (9.7.1.4)** This is a small sample, but the cell is not yet at a stage where it is appropriate to fully investigate performance evaluation. Both the cutting operation and the pick and place operations are still being optimised. In addition, the various calibration and mapping processes have yet to be fully evaluated. The purpose of this chapter is purely to demonstrate that the inspection techniques described in this thesis are suitable for implementation in an automated cell. The inspection results for the lay-ups are discussed in the next section.

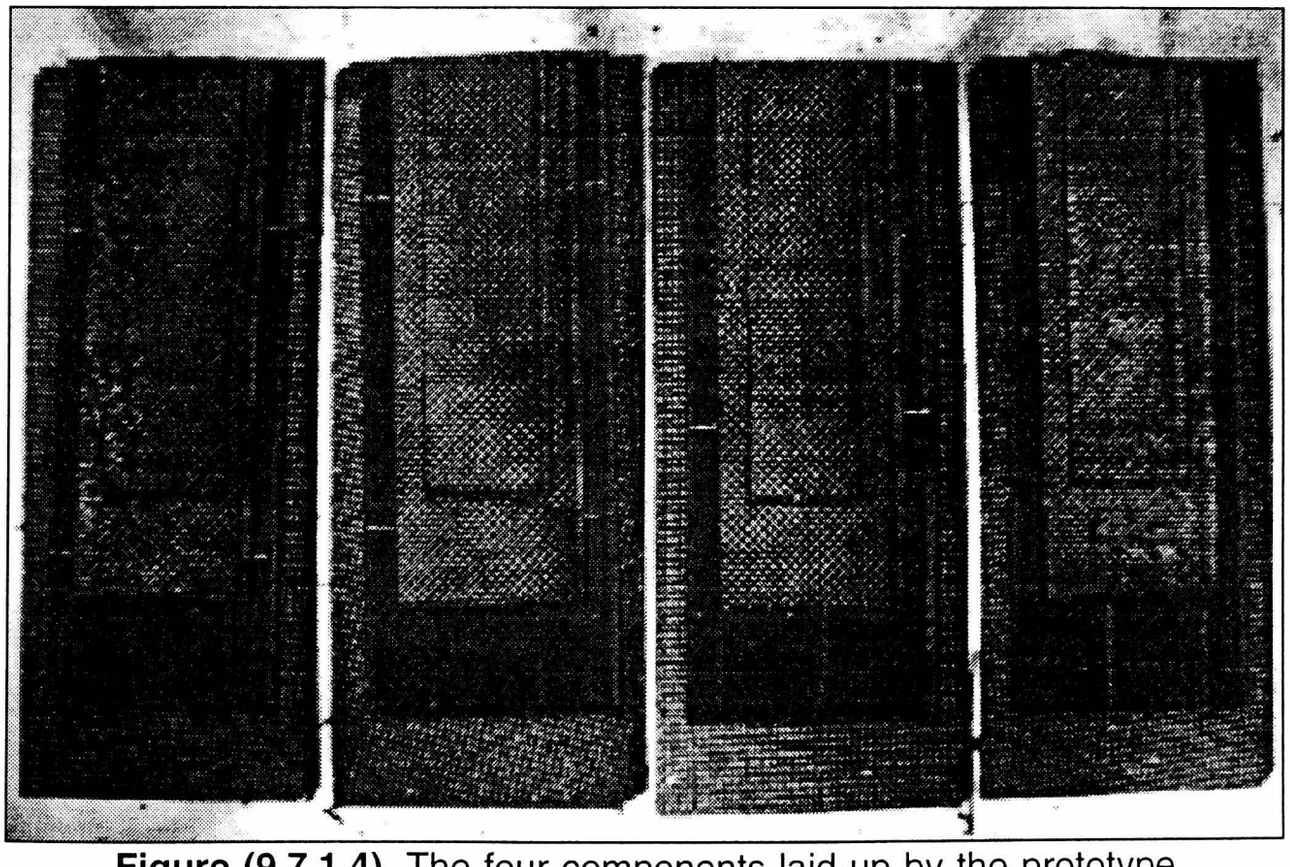


Figure (9.7.1.4). The four components laid-up by the prototype assembly cell.

9.7.2 Interpretation of Inspection Data.

Figure (9.7.2.1) presents the inspection results obtained for each ply of each component. The figure shows the edge of interest with the expected ply position (from CAD) drawn as a black line, and the estimated ply position from the vision system drawn as a white line. The left and right extremes of the inspected edge are also shown in a magnified window to enable a better evaluation of the result. In every case the vision system estimate of ply edge position is in good agreement with human visual judgement. This figure provides good evidence that the techniques developed in this thesis are appropriate to the task of composite lay-up inspection.

The deviation of the estimated ply position from the expected position is shown in both millimetres and pixels below the images of each edge. The resolution of the images is such that 1 pixel $\approx$ 0.2mm. This data is reproduced in tabular form for each component in **Tables 9.1** to **9.4**. From **Section 8.7**, the inspection error which must be taken into account is $\pm$0.2mm. Given that the tolerance on ply edge position is $\pm$1mm, how should these results be interpreted?

Figure (9.7.2.2) illustrates the data for the four components diagrammatically. The illustration is not to scale, and so the orientation of the detected edge is greatly exaggerated. The figure shows the ideal ply position (according to CAD), the $\pm$1mm limits and the inspection results for each ply of each lay-up. The figure also shows the "uncertainty zone" for each ply, caused by the $\pm$0.2mm inspection error allowance. If the result of the inspection falls within this zone then the vision system cannot say for certain whether the ply edge position is within specification or not.

From **Figure (9.7.2.2)**, only ply 1 in component #2 can definitely be said to be out of specification. Ply 3 in component #1, ply 2 in component #2, plies 2,3 and 4 in component #3, and plies 1 and 4 in component #4 lie within the uncertainty zone. All other plies (eight in total) can be said to be within specification. However this data only represents a small section (approximately 100mm) of ply boundary. From **Figure (9.7.2.2)** it is clear that interpolating this inspection result along the full length of the ply boundary would result in a

deviation greater than 1mm for all plies. There is obviously a consistent discrepancy between the orientation of the plies and of the boundaries extracted from CAD data. This source of this discrepancy (i.e. cutting, picking, placing, calibration process? etc) is not yet known, and further work in improving all aspects of cell operation is ongoing. However, inspection of the images shown in **Figure (9.7.2.1)** confirms that the discrepancy in orientation is not due to any vision system error.

In the general case (when the cell is functioning correctly), the validity of extrapolating the estimated edge position outside of the area inspected is doubtful. Extrapolation of straight lines must be approached with caution. Any slight error in the estimated orientation of ply edge position would, if extrapolated for any significant length, produce an estimate well outside the ±0.2mm inspection error expected within a single image. For this reason, extrapolation of the inspection result obtained from a single camera is not considered appropriate. The consequence of this is that the vision system can only check the position of the ply edge within the area viewed.

Taking this to its logical conclusion, to completely inspect a large component would require many cameras, each viewing only a small area of the ply edge. Such an approach, although not out of the question for aerospace components, is unlikely to be required. Most components have a "critical section" which *must* be laid-up within specification if the component is to be acceptable. If this section is within specification then the position of the other ply edges are relatively unimportant. Inspection would therefore concentrate on such critical sections of components. In addition, although carbon fibre is a non-rigid material, the flat handling approach adopted within the cell (i.e. pick and place to and from flat tables, using a flat gripper) means that there is no real opportunity for the material to distort so that some edges are significantly out of specification and others in.

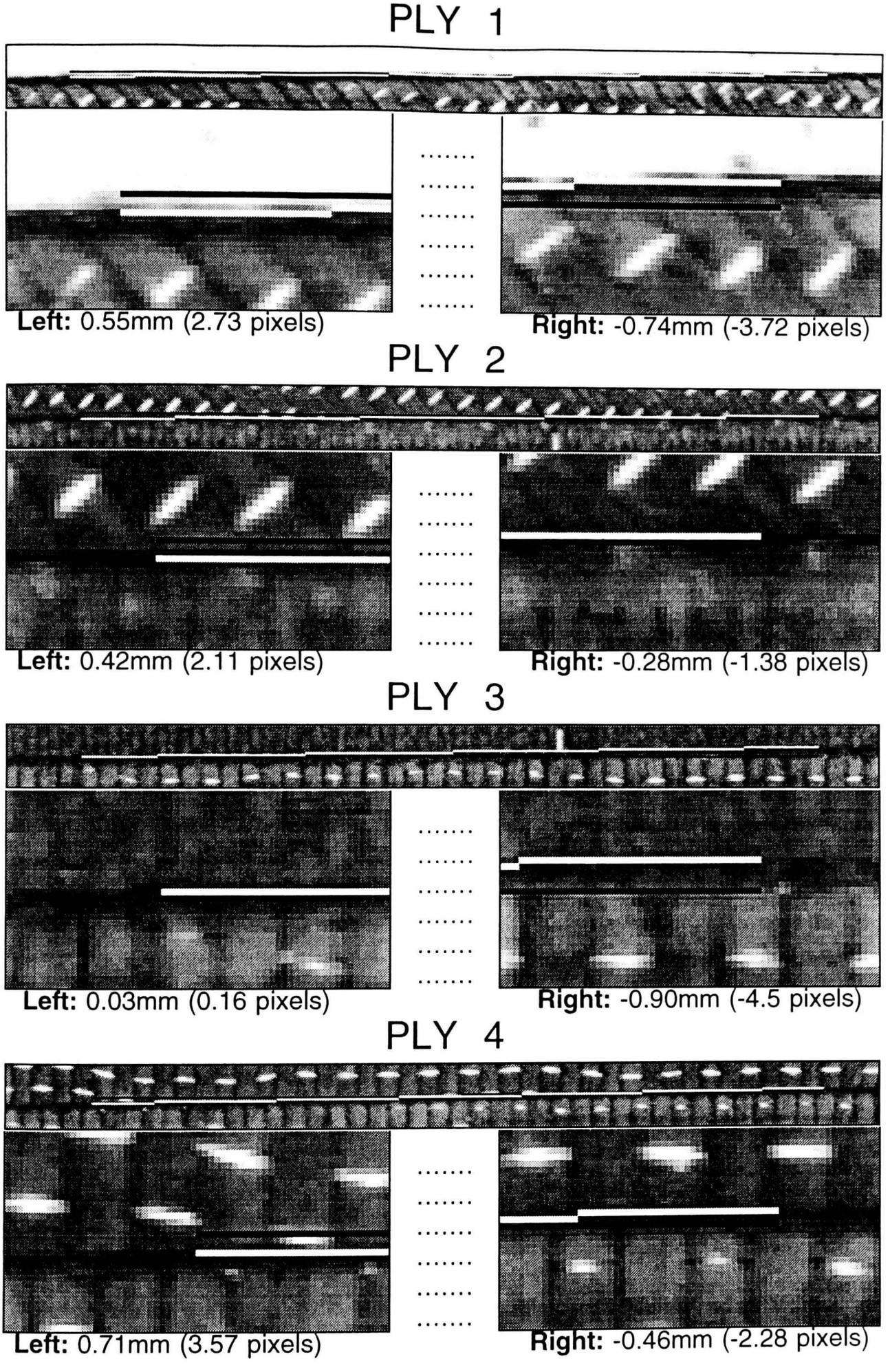


Figure (9.7.2.1a). Results for component #1. The black line is the expected boundary position, the white line the position estimated by the vision system.

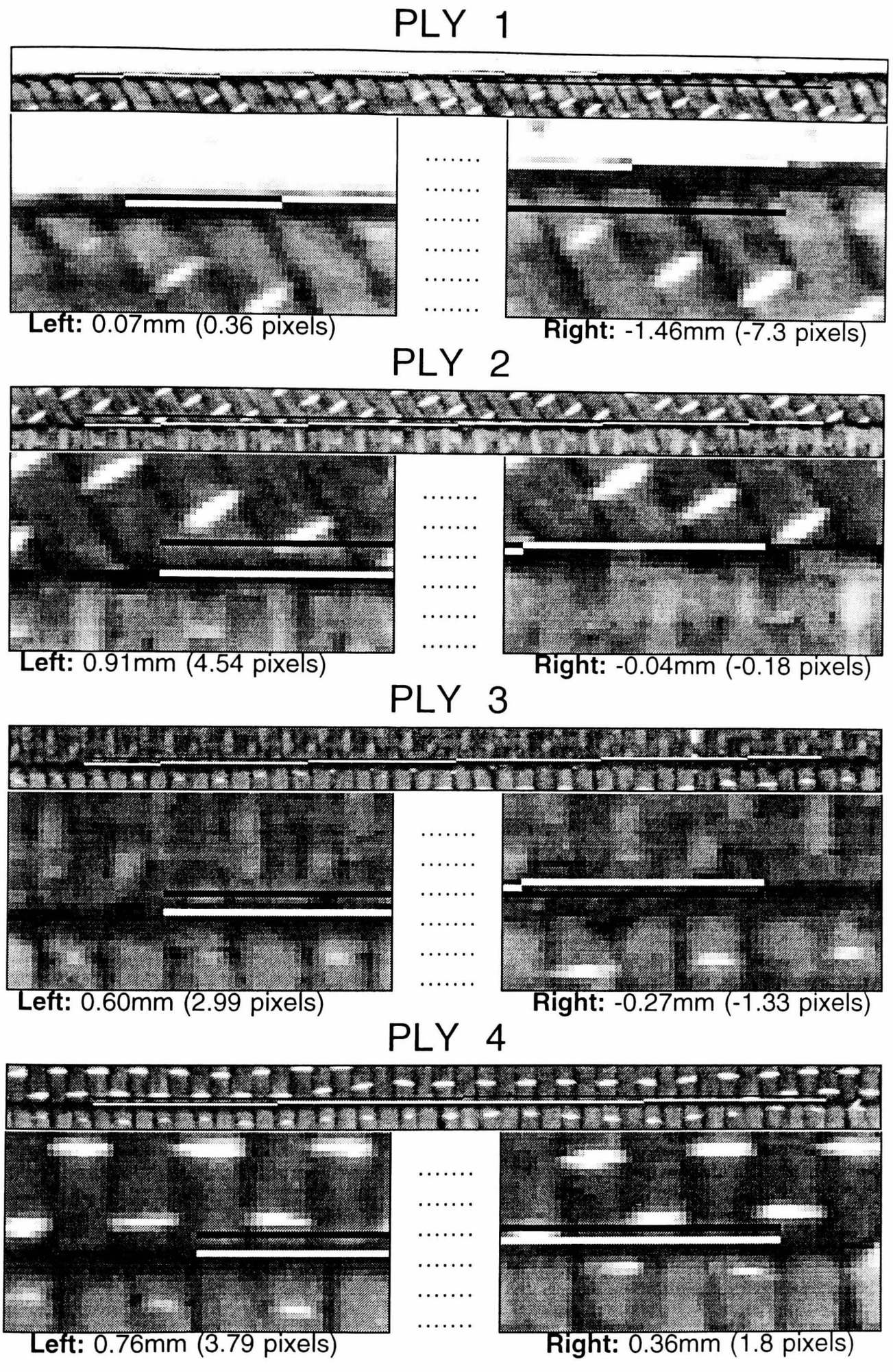


Figure (9.7.2.1b). Results for component #2. The black line is the expected boundary position, the white line the position estimated by the vision system.

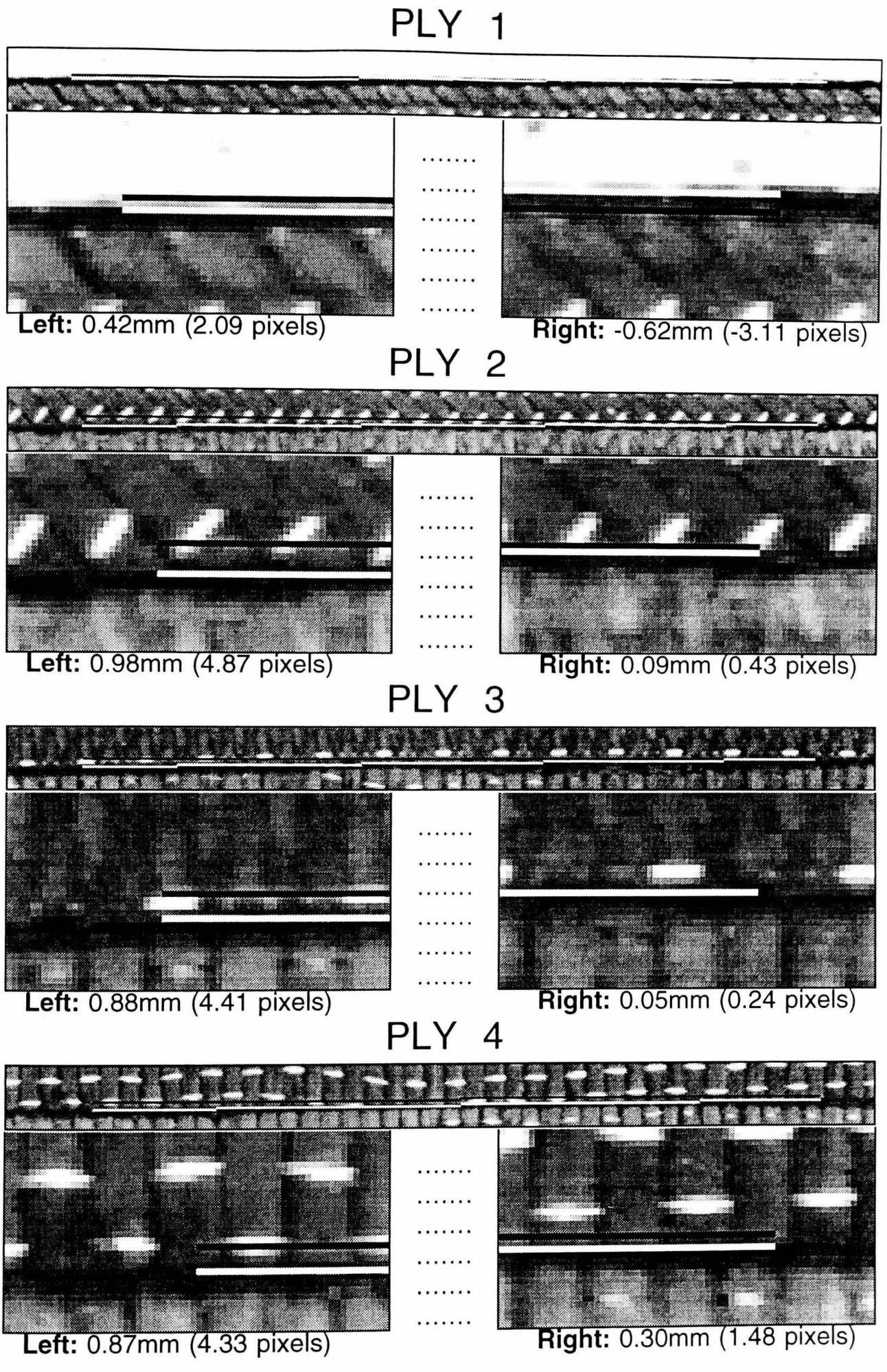


Figure (9.7.2.1c). Results for component #3. The black line is the expected boundary position, the white line the position estimated by the vision system.

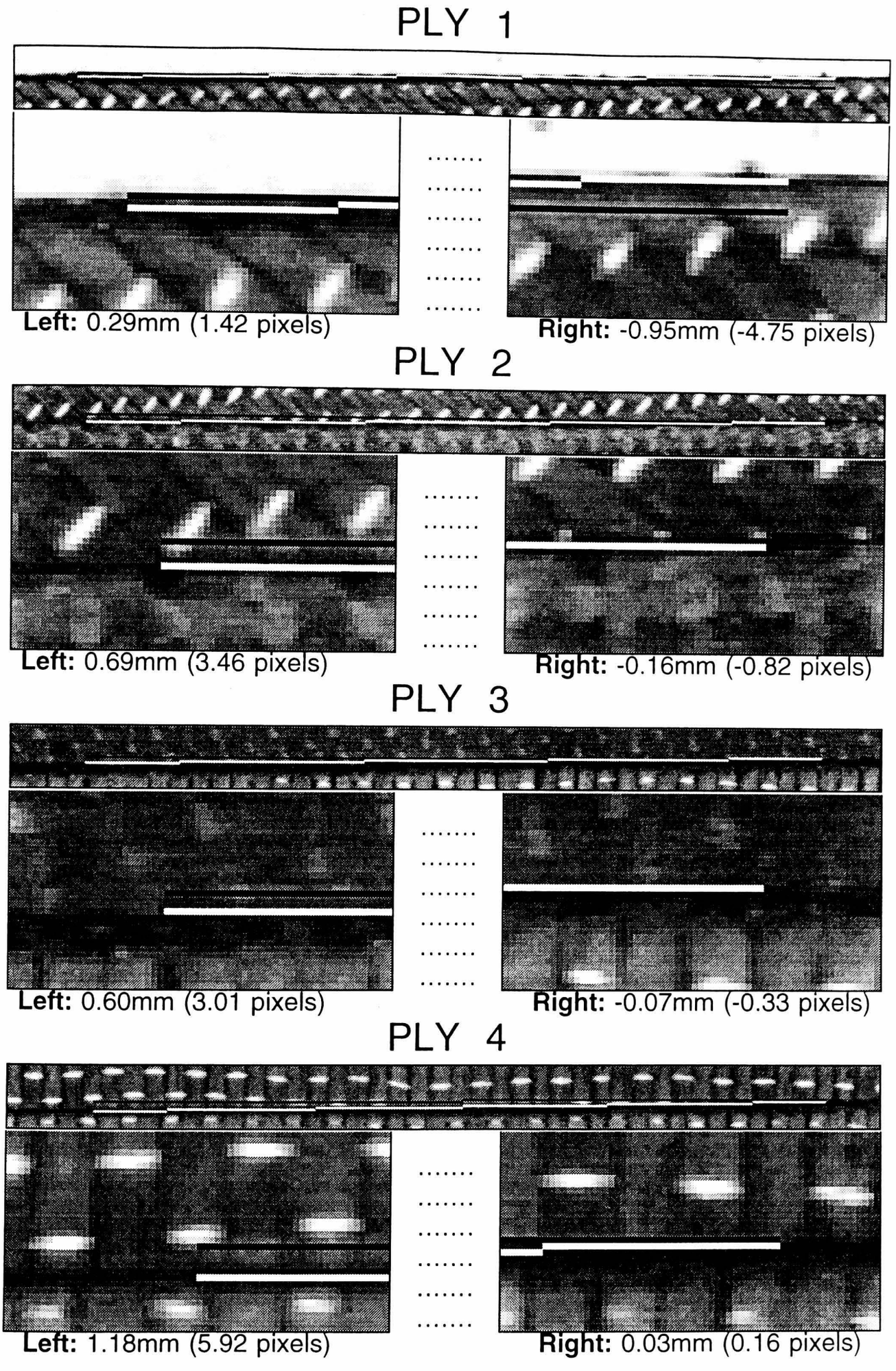


Figure (9.7.2.1d). Results for component #4. The black line is the expected boundary position, the white line the position estimated by the vision system.

The proposed approach therefore would be to inspect all *relevant* areas of lay-up, such that plies within specification within these areas are considered to be fully within specification. The observable accuracy of the results shown in **Figure (9.7.2.1)** give confidence that such an approach would be successful. However it will not be possible to fully evaluate the appropriateness of the approach until the cell has reached a stage where sample components can be laid-up, tacked together, and transported to BAe or Dowty for resin injection and subsequent testing.

In summary then, it can be said that the optimum manner of inspecting large components by viewing small areas of the boundary has yet to be established. However the results shown in **Figure (9.7.2.1)** prove that the inspection process within these small windows is sufficiently accurate and reliable.

9.7.3 Processing Time.

Table 9.5 shows the processing time taken to inspect each ply of each component. **Table 9.6** shows a breakdown of the time taken to inspect ply 2 of component #1, chosen as a typical example. The image passing overhead is the time taken to send and receive the image over the transputer links to and from the pipeline card. The texture segmentation time is the time taken to perform image convolution, smoothing, and thresholding on the pipeline card. Boundary processing time is the time taken to extract the boundary of interest from the segmented image. Boundary refinement time is the time taken to perform the boundary refinement process detailed in **Chapter Eight**. Display time is the time taken to draw the CAD data line and vision system estimate line into video RAM, and to display the inspection results on the VDU (transputer-host I/O is rather slow).

COMPONENT #1				
Ply #	Deviation in Pixels		Deviation in mm	
	Left Edge	Right Edge	Left Edge	Right Edge
1	2.73	-3.72	0.55	-0.74
2	2.11	-1.38	0.42	-0.28
3	0.16	-4.5	0.03	-0.9
4	3.57	-2.28	0.71	-0.46

Table 9.1 Deviation from CAD data for component #1.

COMPONENT #2				
Ply #	Deviation in Pixels		Deviation in mm	
	Left Edge	Right Edge	Left Edge	Right Edge
1	0.36	-7.3	0.07	-1.46
2	4.54	-0.18	0.91	-0.04
3	2.99	-1.33	0.60	-0.27
4	3.79	1.8	0.76	0.36

Table 9.2 Deviation from CAD data for component #2.

COMPONENT #3				
Ply #	Deviation in Pixels		Deviation in mm	
	Left Edge	Right Edge	Left Edge	Right Edge
1	2.09	-3.11	0.42	-0.62
2	4.87	0.43	0.98	0.09
3	4.41	0.24	0.88	0.05
4	4.33	1.48	0.87	0.30

Table 9.3 Deviation from CAD data for component #3.

COMPONENT #4				
Ply #	Deviation in Pixels		Deviation in mm	
	Left Edge	Right Edge	Left Edge	Right Edge
1	1.42	-4.75	0.29	-0.95
2	3.46	-0.82	0.69	-0.16
3	3.01	-0.33	0.60	-0.07
4	5.92	0.16	1.18	0.03

Table 9.4 Deviation from CAD data for component #4.

From **Table 9.6**, it is clear that the boundary refinement stage is the major contributor to ply inspection time. This process has not been optimised in any way. It requires many floating point calculations, but at present is running on a T425 transputer which has only software support for floating points operations. If ported to run on a T800 (which has floating point support in hardware), a significant speed increase should be achievable. In addition, it is essentially a parallel process, and so a further speed increase could be achieved by a parallel implementation on the network of four transputers. However, it is perhaps more appropriate to consider the processing time which would be achievable on commercially available image processing hardware.

The image passing overhead would virtually disappear, due to the high bandwidth of a dedicated video bus. The time for texture segmentation would be about the same as it is now (the 28 milliseconds per image convolution achieved by the pipeline card compares favourably with commercially available systems). The time taken for boundary processing and boundary refinement would both be reduced if, as is likely, a faster host processor and reduced memory access time were available (currently 20MHz and 6 cycle respectively), as well as hardware floating point support. In addition, boundary processing time could be more significantly reduced if hardware blob labelling were available. The final category of **Table 9.6**, display overheads, could really be eliminated completely.

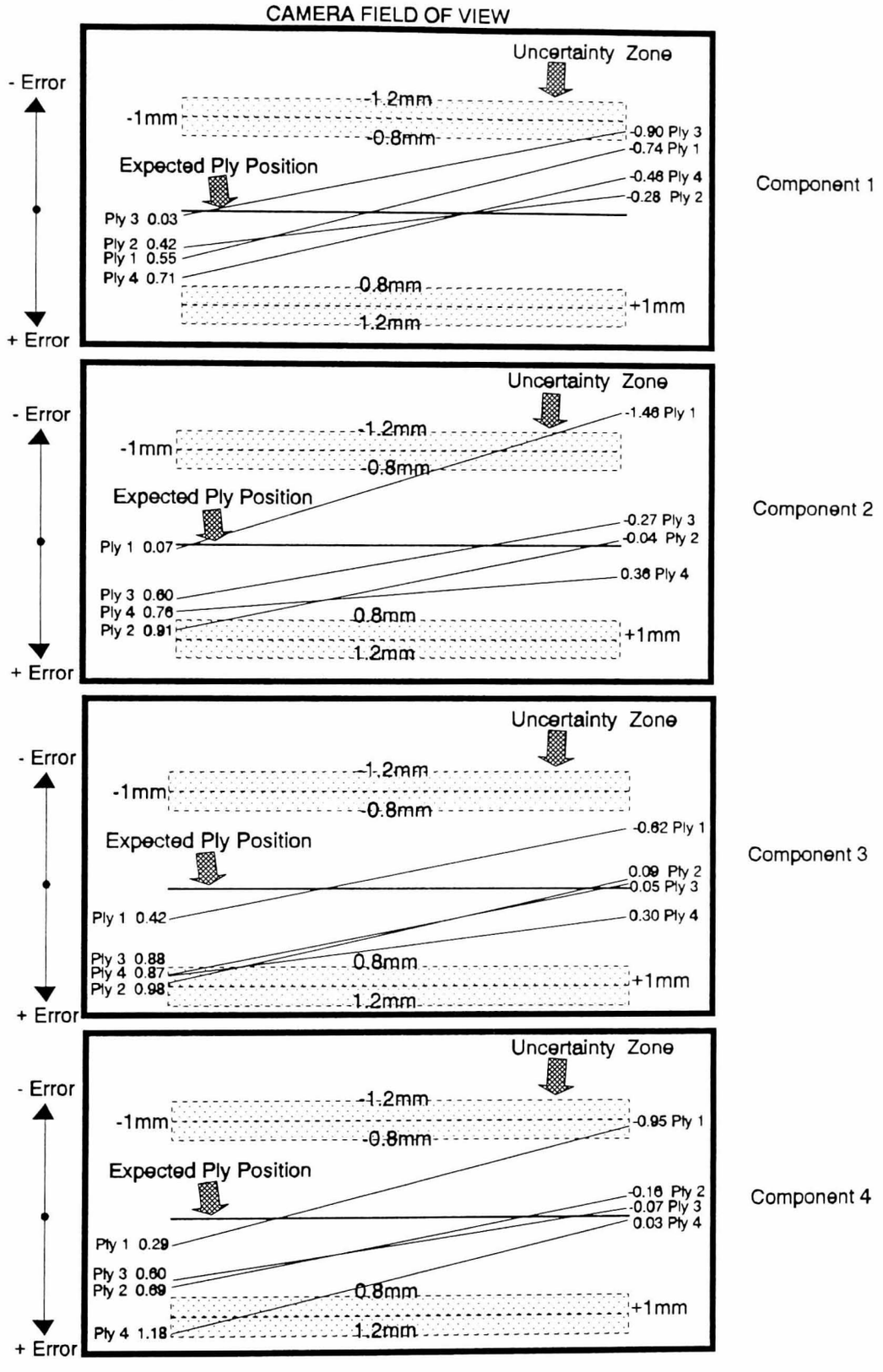


Figure (9.7.2.2). Illustration of inspection results for components 1-4. The ±1mm tolerance and the "uncertainty zone" caused by ±0.2mm inspection error are shown.

Component Number	Ply Number	Processing Time (seconds)
1	1	3.23
	2	3.35
	3	3.46
	4	3.39
2	1	3.31
	2	3.37
	3	3.44
	4	3.38
3	1	3.17
	2	3.37
	3	3.48
	4	3.43
4	1	3.26
	2	3.37
	3	3.49
	4	3.40

Table 9.5 Processing time for each ply.

OPERATION	TIME (seconds)
Image Passing Overheads	0.59
Texture Segmentation	0.22
Boundary Processing	0.15
Boundary Refinement	2.10
Display Overheads	0.29
Total	3.35

Table 9.6 A breakdown of the processing time for ply 2 of component #1.

To summarise, with an appropriate (but low cost) image processing system, inspection time should be approximately one second or better, per image. Most components would require multiple images to be processed, and so the number of images to be processed will determine the overall processing time. For small components requiring only two or three images per ply therefore, processing time will probably be approximately two or three seconds, whilst for larger components a proportionately longer time will be required. In either event, inspection time will not be the cycle bottleneck, since cutting, pick and place, and tacking each require significantly more time than inspection. If it were required to decrease inspection time, then the images could be grabbed in a very short space of time, and processed concurrently with the other operations in the cell.

9.8 Detecting Lay-Up Errors.

Previous sections have established the performance of the vision system within the prototype assembly cell for inspection of a sample component. The results of the inspection have been visually convincing, and figures for deviation from CAD data have been obtained. This penultimate section considers the appropriateness of the inspection technique for detecting lay-up errors, of which there are two possible types.

The most probable error is that the robot has failed to pick the ply from the cutting table, and so when it comes to inspection the ply is missing from the lay-up altogether. There is no specific test for this within the inspection process. If the ply is missing however, the vision system does not find a texture boundary in the expected area. The vision system is therefore able to report that the ply is either missing or significantly displaced. This has been verified by tests in which the vision system is asked to inspect a ply which, so to speak, is not there. In each case the vision system responded correctly.

The second type of error is that the ply is physically displaced from its intended position. This type of error is apparent in the images obtained during the lay-up of the sample component, and can readily be ascertained from the inspection data. As a final test however, a further set of experiments was

carried out to confirm that estimated ply position correlates with physical displacement of the ply. It was hoped that this could be accomplished within the cell by deliberately offsetting the robot lay-up coordinates by known displacements, and correlating the data with the vision system estimate of ply position. However, as in **Section 7.4.2**, the pick and place cycle of robotic lay-up proved too unreliable to ensure accurate displacements. An alternative scheme was therefore devised using a special lay-up table developed for a previous lay-up application. This table has a top surface mounted on two parallel rails. The table top is therefore moveable in one dimension as shown in **Figure (9.8.1)**.

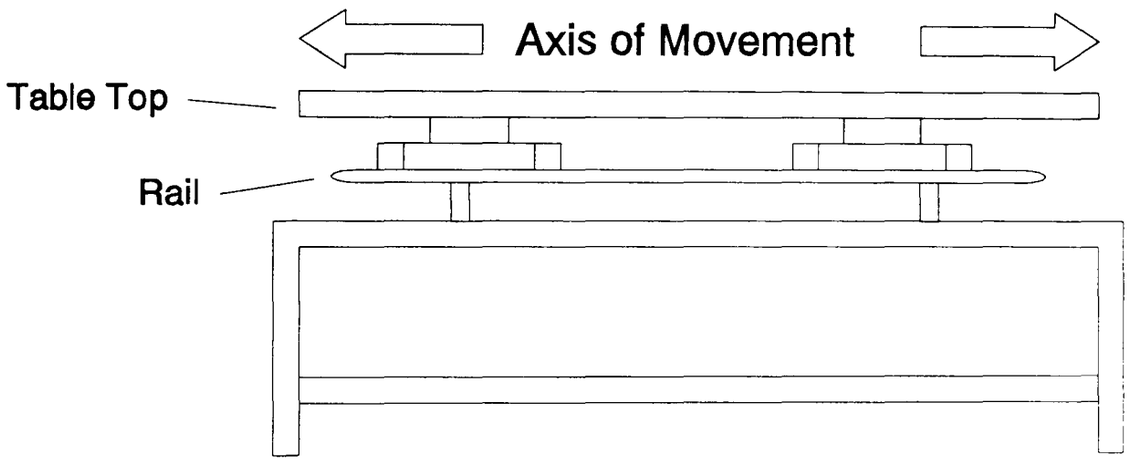


Figure (9.8.1). The stepper motor controlled lay-up table.

The position of the table top is controlled by a stepper motor. A lay-up of two plies was positioned on the table within the field of view of a camera connected to the vision system, and the edge position of the foreground ply obtained using a textured edge operator with boundary refinement. The lay-up was then displaced by moving the table top by 1.6mm, and again the position of the foreground ply determined by the vision system. The deviation in pixels from the original edge estimate was recorded, and the table then stepped back towards the original position by 0.2mm. The inspect-step-inspect cycle was repeated until the lay-up position had gone past the original position by 1.6mm. The process was repeated using three lay-up configurations from the sample component: woven on unidirectional at -45°, unidirectional at 90° on

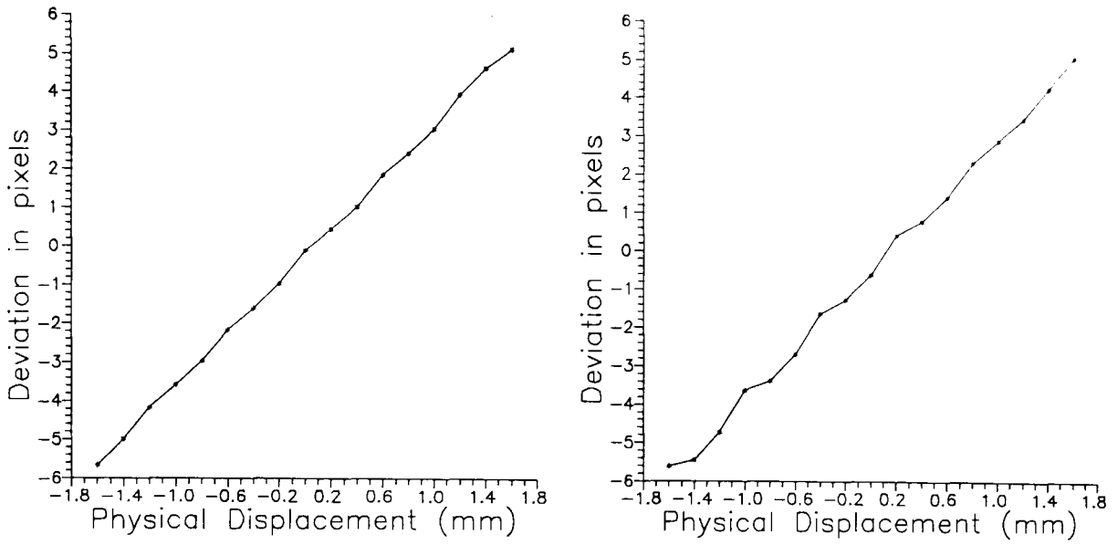
woven, and unidirectional at 90° on unidirectional at 90°. The case of unidirectional at -45° on the table background was not considered, since it represents a simpler inspection task, and so would not really provide results relating to texture based inspection. **Figure (9.8.2)** shows the results obtained in graphical form, with the left graph of each configuration showing the deviation measured at the left-most extremity of the foreground ply edge, and the right graph of showing the deviation measured at the right-most extremity.

As can be seen, overall the results are convincingly linear which indicates a good correlation between the physical displacement of a ply, and the estimate of its position by the vision system. A point of interest is that the left edge estimates are noticeably more linear than the right edge estimates. An investigation of this phenomenon revealed that it is not a feature of the inspection process (as had been feared), but rather appears to be a characteristic of table movement. The stepper motor gearing system is located on the left side of the table, rather than centrally. The effect of this is that the table top does not always move smoothly on the right-hand rail, and so some variation in movement can occur. This accounts for the less than uniform displacements detected by the vision system at the right edge of plies positioned on the table.

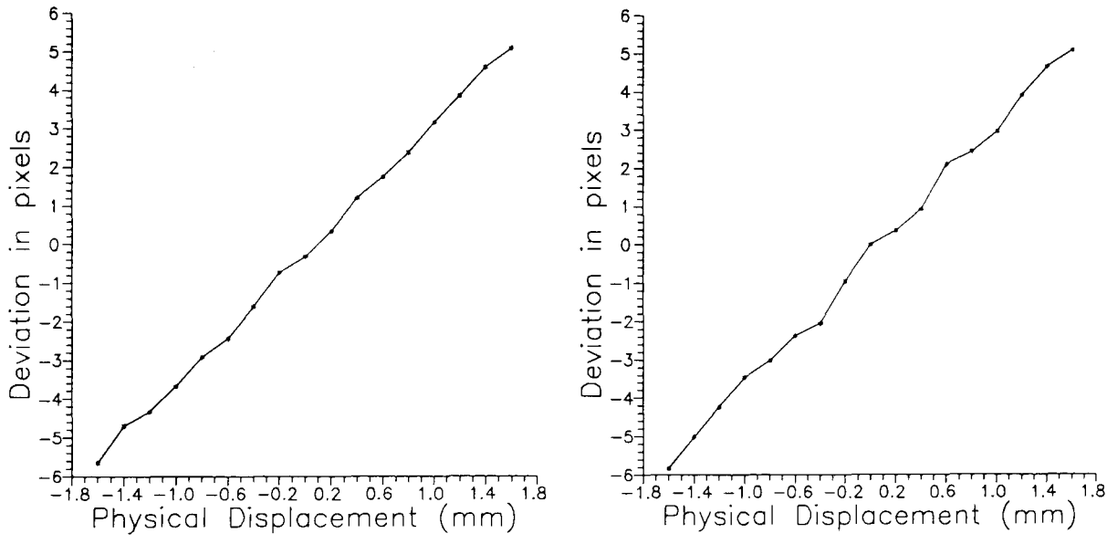
The conclusion from these experiments therefore, is that physical displacement of ply edges can be detected using the inspection techniques implemented in the prototype assembly cell.

9.9 Conclusions.

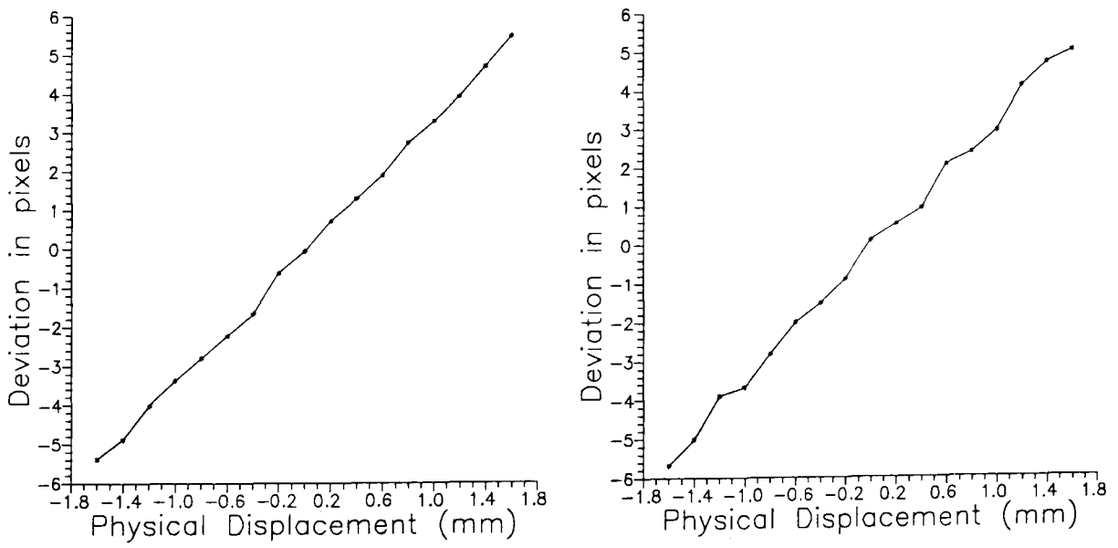
This chapter has described a prototype robotic assembly cell for composite lay-up. The aim of the chapter has been to demonstrate that the techniques described in this thesis can be used to implement an inspection system for an industrial application. Aspects addressed include calibration, coordinate mapping, CAD data integration, cell integration, interpretation of inspection results, and processing requirements. It is the contention of this chapter that the industrial readiness of the techniques described has been successfully demonstrated.



Woven on unidirectional at -45°.

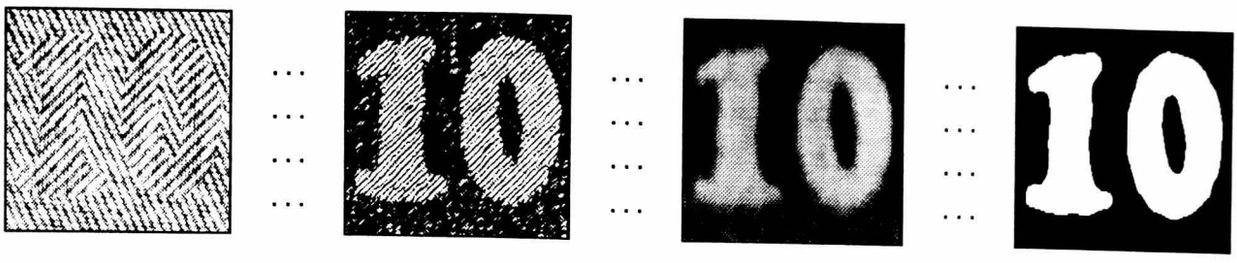


Unidirectional at 90° on woven.



Unidirectional at 90° on unidirectional at 90°.

Figure (9.8.2). Graphs showing the inspection results as the ply edge position is varied from -1.6mm to +1.6mm in steps of 0.2mm.



CHAPTER 10

Conclusions and Further Work.

10.1 Introduction.

Automation of dry fabric composite lay-up is an ongoing research topic. At the start of this project, the need for in-process inspection was recognised, but the feasibility of implementing a machine vision solution remained to be proven. No conventional techniques could meet the requirements of the application. The work presented in this thesis has shown that a machine vision solution to the inspection problems posed in dry fabric composite lay-up is feasible, and requires only low-cost hardware to operate. The main features of the thesis are summarised in the following section.

10.2 Summary of Thesis.

Advanced composite components have been introduced, and the current manufacturing process outlined. The particular problems faced in the machine vision application have been identified, and the criteria for automated inspection defined. A comprehensive survey of current statistical texture analysis methods has been carried out. A single channel texture analysis model has been presented, and examined in some detail. The main features

of the model are that it can successfully segment images containing only a few textures, and that it is elegant in the sense that it is computationally simple and easily realisable in hardware. The model has been shown to be appropriate for the task of automated inspection in composite lay-up. Gaussian smoothing has been demonstrated to provide more accurate boundary localisation than low-pass smoothing.

An existing convolution mask optimisation algorithm due to Benke and Skinner has been implemented and extensively tested. By dropping the symmetry constraint the classes of texture which can be optimally discriminated is increased. A new, more suitable, optimisation criterion based on grey level separation has been adopted. A new convolution mask optimisation algorithm, called the basis algorithm, has been presented and investigated. The masks produced by this new algorithm are weighted averages of a pre-determined mask set. The basis algorithm has been demonstrated to outperform the Benke and Skinner algorithm both in terms of optimisation rate and discrimination achieved. It has also been demonstrated that convolution masks can be optimised to perform as edge operators in textured images. Optimised masks may allow simple, fast boundary detection techniques to be used where previously more sophisticated techniques were required. The basis algorithm is easily applicable to the task of optimising edge operators for use in textured images.

The texture-based tools have been implemented and tested for typical composite inspection tasks. Simple boundary models have been incorporated in the inspection process, and estimates of inspection error have been obtained experimentally. The inspection error incurred is modest, approximately ±0.33mm. A novel boundary refinement algorithm has been developed in an attempt to reduce the inspection errors inherent in texture based boundary estimates. The algorithm takes the form of a local search, using the texture estimate as a guiding template. The points selected on each search are those which maximise a merit function of edge magnitude, edge orientation, and distance from the texture estimate. Optimum parameters for the merit function are obtained for each edge configuration using multiple

training images in conjunction with simple function optimisation algorithms. The effect of boundary refinement has been examined over a representative range of ply configurations. The inspection error obtained was of the order of ±0.2mm or better.

The industrial readiness of the algorithms developed has been shown by implementing an inspection system as part of a prototype automated assembly cell. The performance of the cell and inspection system has been demonstrated using a sample lay-up application.

10.3 Conclusions on the Progress Achieved.

The techniques described herein could be used to provide lay-up inspection for many of the dry-fibre preforms currently manufactured in the aerospace industry. Some of the more complex components (which exhibit more complex or irregular ply shapes) would require a more careful study of the most appropriate boundary models than has been carried out in this work. In addition, as discussed in **Section 9.7.2**, the best way in which to ensure adequate in-process inspection of larger components (i.e. how many images, which edge areas to inspect etc.) has not yet been established.

However these are largely component specific issues of the type routinely encountered in many machine vision areas. Such issues can be addressed, *provided* that a means of performing the fundamental inspection task is available. The work detailed in this thesis has shown that the fundamental problem for inspection of dry fabric composite components, that of ply boundary detection, can be successfully overcome. From the criteria set out in **Chapter One** therefore, the work must be judged successful. There are two good reasons why the approach taken to this inspection application has been effective.

Firstly, the approach taken to boundary detection is loosely based on an (apparently well founded) theory of human vision. This theory holds that there are two co-operative/competitive processes involved in human pre-attentive scene segmentation **[Grossberg, Mingolla, 1985]**. One is edge detection, and the other is feature filling, which may be considered as a kind

of region growing process. The inspection techniques described in this thesis are in essence a very simple sequential implementation of this model. The image is segmented according to texture, followed by a local boundary refinement process based on edge detection. The implementation of each of the sub-processes (texture analysis and boundary refinement) may in themselves be considered *ad-hoc*, but as the results show, the underlying approach is sound.

The second reason the techniques are successful, is that the parameters are not derived from an inflexible (either mathematical or empirical) model, but are in fact obtained by training. This is true of both sub-processes. As a result the inspection process is optimised for the task in hand. With the advent of artificial neural networks, such training based systems are becoming much more widespread. However, as the results of this thesis show, it is not only neural based algorithms which can benefit from training.

In the field of industrial inspection, every application is different. The majority of current techniques that have found their way into application are essentially crude, and more often than not are unable to adapt to new requirements without major reprogramming. Algorithms designed using the two criteria identified above (based on an appropriate model, and based on experiential learning) are likely to help us progress a step nearer those elusive generic vision algorithms which, at present, seem very far away.

10.4 Further Research.

Two areas suggest themselves as candidates for further research, and these are discussed in the following sections.

10.4.1 Reducing Inspection Error.

The tolerances on ply lay-up position are generally of the order of ±1mm for current applications. The techniques described in this thesis are able to provide confirmation to this resolution. However, if dry fabric lay-up is to present a viable manufacturing route for components with tighter tolerances, then the inspection system will need to be considerably more accurate.

Chapter Eight showed how the use of a boundary refinement stage in conjunction with boundary models could provide inspection with an error generally less than one pixel. One possible research area is in the development of a subpixel edge operator suitable for use with the low contrast images typically obtained from images of carbon fibre. This is a difficult application, significantly more demanding than the applications where subpixel operators are normally employed, such as dimensional measurement of machined parts by coordinate measuring machines (CMM's). However, the results achieved in **Chapters Eight** and **Nine** with the standard Sobel edge operators and simple boundary models are encouraging enough to provide some optimism for the possibilities of the technique.

A totally different technique which promises more accurate edge estimation than at present is laser triangulation, and this is now being investigated within the research group. In our implementation of this technique, a laser line is projected on the area where the edge is expected to be. A CCD camera views the laser line at an oblique angle. Any ply edge shows up on the image as a step in the laser line. Machine vision techniques can be used to obtain the step position and height in image coordinates. The small height difference produced by a ply edge (less than 1mm) means that the laser line must be imaged at very high resolution (approximately 40 pixels per millimetre). The advantage of this is that the edge position at a single point can be obtained very accurately. The disadvantage is that multiple images must be processed to inspect every edge of interest. The camera and laser system are mounted on an articulated robot which moves to each point to be inspected. The inspection time therefore depends on how many points are required and how long it takes to move the robot to each point. Initial results show that the error in estimating single edge points will be of the order of ±0.05mm. The limiting factor on inspection accuracy therefore, is likely to be the repeatability of the robot. Work on calibration, and matching inspected points with CAD data is continuing.

10.4.2 An Optimised Multi-Channel Texture Analysis System.

This thesis has been largely based on a very simple texture analysis method which uses a single convolution channel to achieve real-time segmentation of images containing a few textures. The technique relies on the fact that a convolution mask can be trained which is appropriate to the task. The extension of this idea to a full multi-channel training model has already formed the basis of other work within the research group. A sophisticated neural network based model has been developed, and shown to out-perform the standard multi-channel model using Laws or Gabor filters in terms of texture segmentation **[Zhang, 1995]**. This model does however require significant processing and memory resources, and so is not suitable for industrial application using existing commercially available hardware. Between these two extremes of a simple single-channel model using hardware convolution and thresholding, and a full multi-channel neural model using more sophisticated feature measures, there is an opportunity for a hybrid approach. Image processing hardware is providing more functionality all the time. Many systems now have the capability to perform two, four, or even eight image convolutions in parallel. Such systems could perform quite powerful texture analysis if a multi-channel training algorithm were developed which could be tailored to take advantage of the available hardware. For the sake of processing efficiency, the texture feature of each channel would be a texture-energy-like measure, as used in the single channel model. The efficiency of various classifier algorithms, and their suitability for implementation using standard hardware blocks would require investigation. The power of the resulting texture analysis system would depend on the number of channels available. This thesis has shown the considerable power of a single convolution channel for texture analysis. The successful development of the proposed algorithm would provide low-cost systems with a real-time texture analysis capability which would enable them to be used to tackle a wider range of industrial applications than is presently possible.

APPENDICES

APPENDIX A

The MAX-MIN Operator

Early work in texture analysis was aimed at identifying which visual characteristics a human observer might use to discriminate textures. Intuitively, one possibility is the number and pattern of **peaks** (points or lines of high intensity) and **troughs** (points or lines of low intensity) in the image. A "hill-climbing" operator (called the **MAX-MIN** operator) was developed which finds these peaks and troughs in a one dimensional scan direction of the image. Pseudo code for the operator is given in **Figure (A.1)**.

The operator can be applied to an image in both row scan and column scan fashion, so that extrema in all directions are detected. The algorithm alternates between searching for local maxima and searching for local minima, marking each as it finds them provided they satisfy a threshold requirement. This is a preset parameter. The other important parameter is *interval size*, which is measured in number of pixels. A small interval size means that all extrema which satisfy the threshold will be marked no matter how closely together they fall. For a larger interval size several extrema may be encompassed in one interval, and only the maxima (minima) of these will be marked, provided of course that the threshold is satisfied. If the image exhibits few peaks far apart, then a large interval size combined with a threshold will eliminate smaller extrema caused by noise, and also be computationally more efficient. If, however, the image exhibits many peaks close together, or a pattern of smaller peaks, then a small interval size with little or no threshold is necessary to highlight the detail. These points are more easily appreciated by reference to **Figure (A.2)**.

When the MAX-MIN operator is applied to an image containing cross and unidirectional ply, the transformed image looks encouraging. The different weaves produce different patterns, with the cross-ply generally 'busier' (more peaks and troughs) than the unidirectional. **Figure (A.3)** demonstrates the effect of the operator, with only the minima marked.

```
{ *** main procedures *** }

CLIMB(interval)
    IF maxima(interval) > maxima(last.interval)
        { not reached local maximum }
            CLIMB(next.interval) { continue search }
        ELSE { reached local maxima }
            IF (maxima - last.minima) > threshold
                mark.maxima(last.interval)
            END
            DIVE(interval)
            { look for next minima }
        END


DIVE(interval)
    IF minima(interval) < minima(last.interval)
        { not reached local minimum }
            DIVE(next.interval) { continue search }
        ELSE { reached local minima }
            IF (last.maxima - minima) > threshold
                mark.minima(last.interval)
            END
            CLIMB(interval)
            { look for next maxima }
        END

{ *** main loop *** }

initialise_variables
WHILE  dy < ymax
    WHILE  dx < xmax
        CLIMB(interval)
        DIVE(interval)
{ end main loop }
```

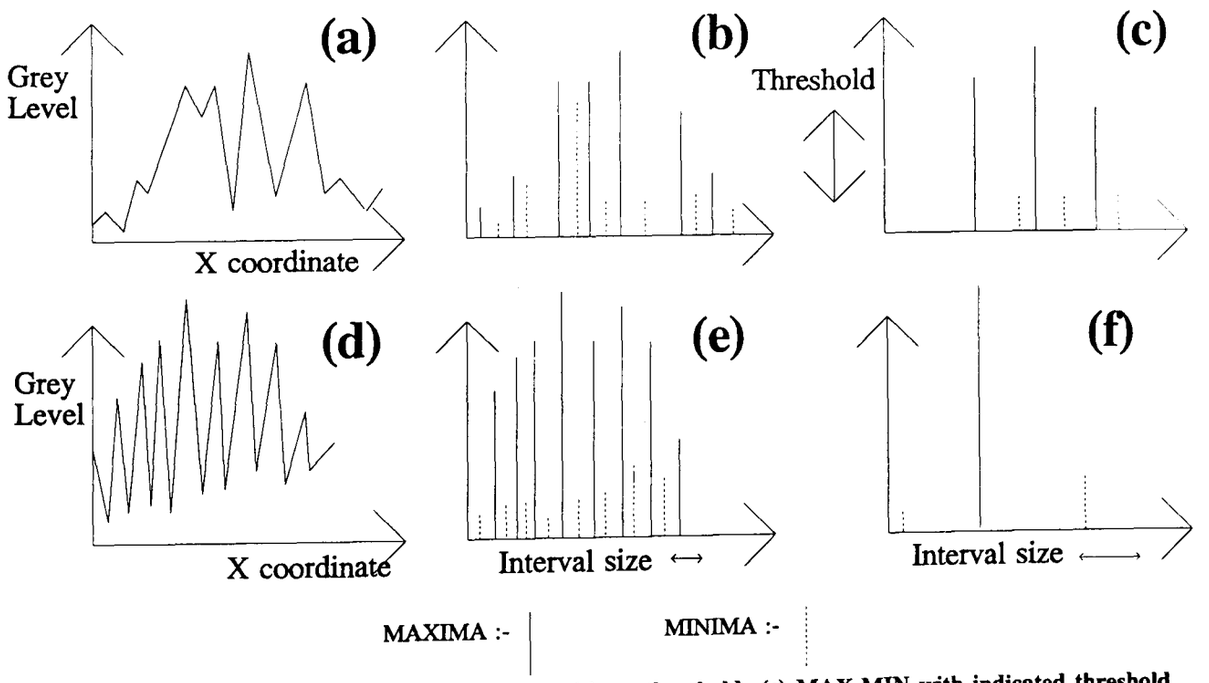
Figure (A.1). Pseudo-code for the MAX-MIN operator.

A prototype system was developed to test whether images of woven and unidirectional material could be effectively segmented using features measured after application of the MAX-MIN operator. After some experimentation, the features adopted were:

(a) Number of extrema per unit area.

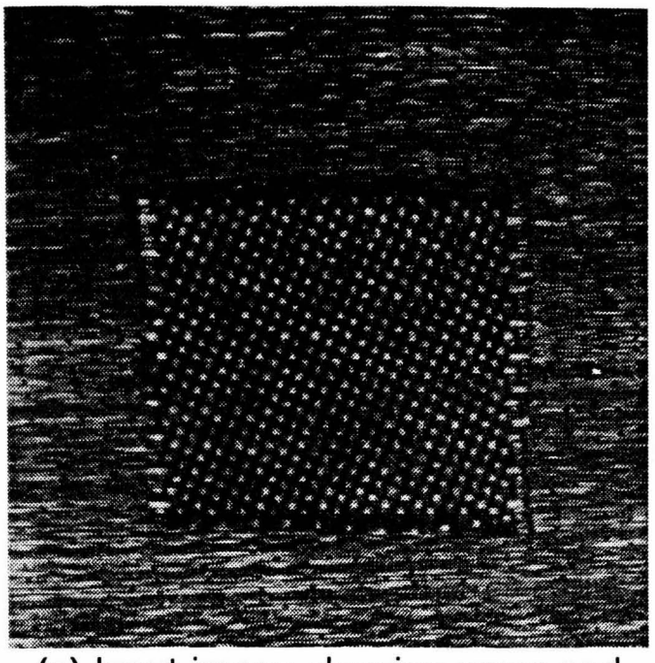
(b) Amount of diagonal space per unit area.

The 'unit area', or subimage, was chosen as 32x32 pixels, the diagonal space measured using a window of 2x8 pixels angled at minus forty-five degrees. To measure diagonal space, the following scheme was implemented. For every position within a subimage that no extrema are present anywhere in the 2x8 window, a counter was incremented. The reason for this is that the extrema in cross-ply seemed more regularly distributed, whereas in the unidirectional there seemed more small noise-like edge segments. The feature vector for each subimage is therefore two-dimensional, with one measure for number of extrema, and one for diagonal space. By application of a simple two-pass threshold, these features do provide a means of differentiating between cross and unidirectional ply. However, consistency and robustness are low, and it was concluded that better features are required than the extrema/space measures. This initial system had shown encouraging results however, and indicated that investigating texture analysis could be a fruitful approach to the problem.

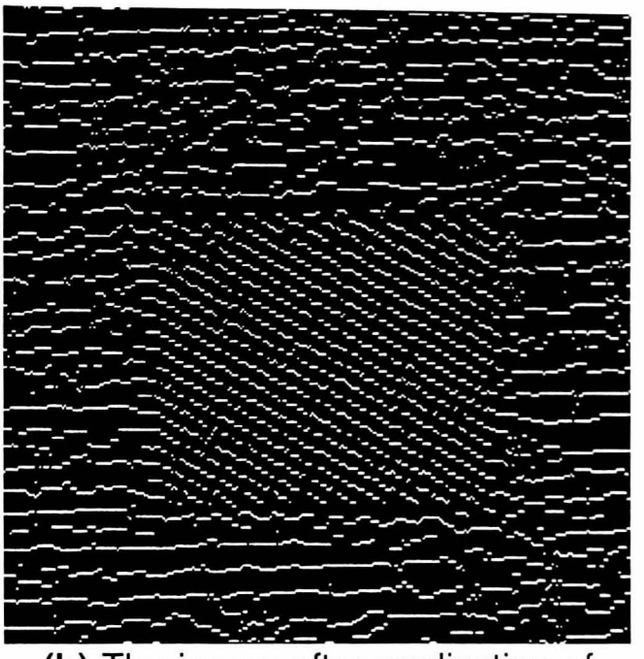


(a) Input image for (b) and (c). (b) MAX-MIN with no threshold (c) MAX-MIN with indicated threshold (d) Input image for (e) and (f) (e) Small interval size (f) Larger interval size.

Figure (A.2). Illustration of MAX-MIN operator.



(a) Input image showing cross and unidirectional material.



(b) The image after application of the MAX-MIN operator.

Figure (A.3). The effect of the MAX-MIN operator on an image of cross ply and unidirectional material. Only image minima have been marked.

APPENDIX B

Cell Calibration

B.1 Introduction.

Figure (B.1) shows a representation of the various coordinate frames present in the cell. These are: the lay-up robot coordinate frame (only the X-Y axis shown); the cutting table coordinate frame; the vision system coordinate frame(s) (an independent coordinate frame for each camera field of view).

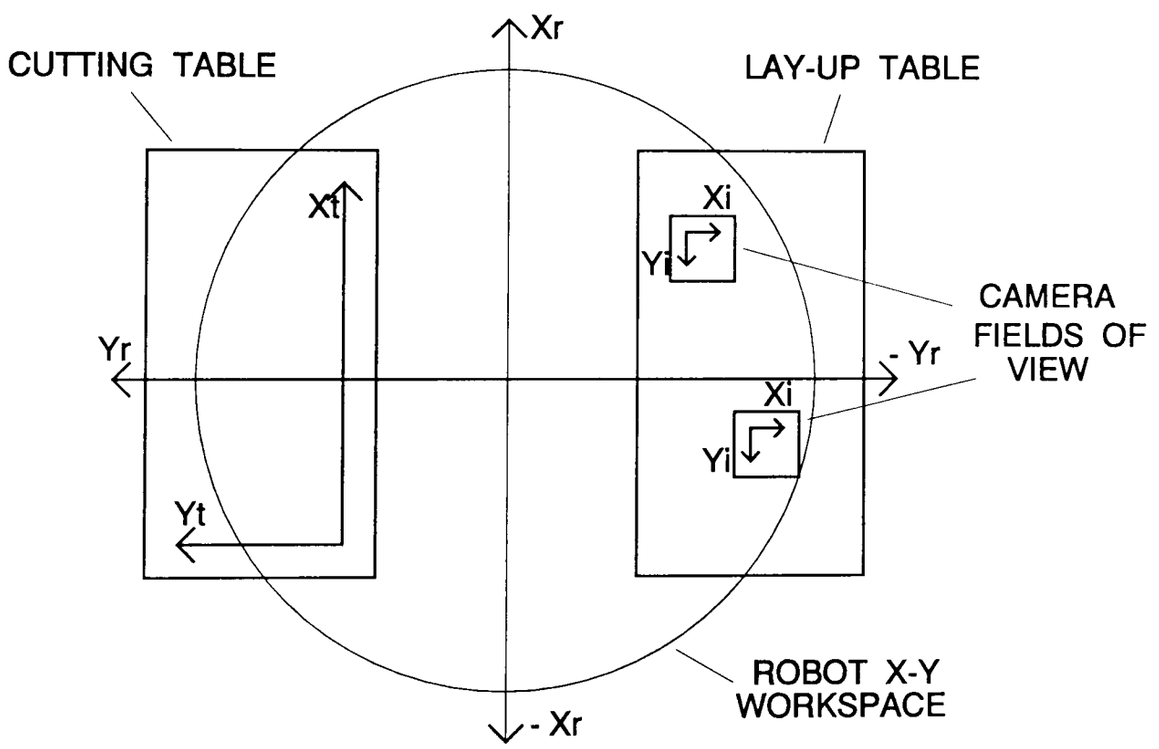


Figure (B.1). *Coordinate frames in the prototype assembly cell.*

There are two operations in the cell which require mapping of coordinates between these frames: the pick operation, and the inspection operation. For

the pick operation, the requirement is to map the centre point position of a newly cut ply in cutting table coordinates to the corresponding lay-up robot coordinates so that the ply can be picked. For the inspection process, the requirement is to map the expected position of ply edges on the preform stack (extracted from CAD data) into the field of view of each camera inspecting the lay-up. The approach adopted to fulfil these mapping requirements is detailed in the following two sections.

B.2 Lay-Up Robot Calibration.

The cutting table coordinate system involves X and Y coordinates only. These represent the offset in millimetres from an origin point located at one corner of the table. The lay-up robot has a real-world 3-D coordinate system. The resolution of the lay-up robot coordinate system is such that the smallest incremental movement is equal to 0.125mm in any axis. For this application the Z (height) coordinate will be constant over the cutting table. That is, the cutting table is considered flat. The task is therefore to map between two X-Y coordinate frames as shown in **Figure (B.2)**.

There are three operations involved in such a mapping: translation, rotation, and scaling. These operations can be formulated succinctly using homogenous coordinates. To map a point tx,ty in cutting table coordinates to the corresponding point rx,ry in lay-up robot coordinates, equation **(B.1)** can be used where xoff,yoff is the translation between the lay-up robot origin and the cutting table origin, θ is the angular displacement between the two coordinate frames, and Sx and Sy effects the scaling required on the X and Y table coordinates respectively.

$$\begin{bmatrix} rx \\ ry \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & xoff \\ 0 & 1 & yoff \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Cos\theta & -Sin\theta & 0 \\ Sin\theta & Cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} tx \\ ty \\ 1 \end{bmatrix} \qquad (B.1)$$

The parameters of the transformation (xoff,yoff, θ, and Sx,Sy) can be determined from three calibration points where both the cutting table

coordinates and the lay-up robot coordinates are known, as shown in **Figure (B.2)**.

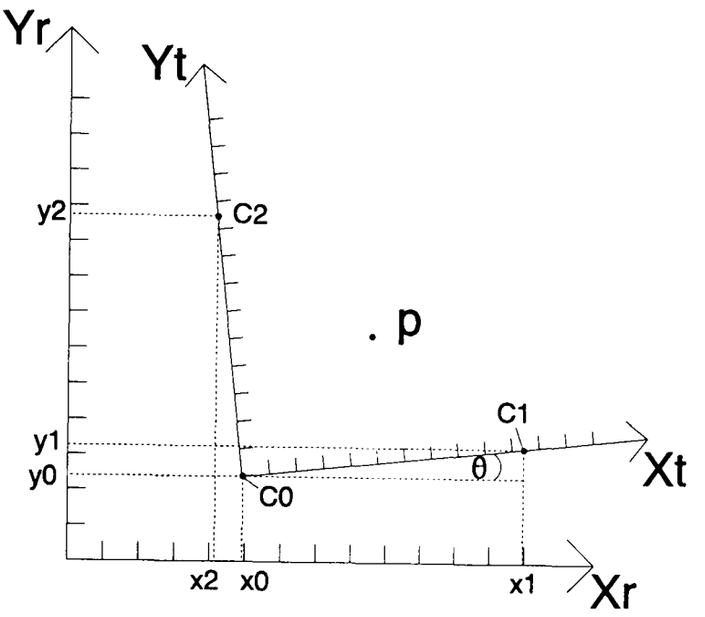


Figure (B.2). *Parameters required to map the coordinates of **P** in X_tY_t to X_rY_r are derived from three calibration points: C0,C1,C2.*

To this end, the lay-up robot is "taught" the position of three points on the cutting table. To achieve this the cutting table laser is set on minimum power, and used to mark the cutting table with cross-hairs at three calibration points (C0,C1,C2 in **Figure (B.2)**). These points are chosen as the origin, and a point on the X-axis and Y-axis respectively. The cutting table coordinates of these points are noted and stored to file. The lay-up robot coordinates for these points must now be obtained. This is achieved by use of a special positioning tool, shown in **Figure (B.3)**. The tool has been designed and machined so that the point of the tool occupies the same X-Y coordinate as the tool centre point of the lay-up robot. This is facilitated by use of an interference fit locating ring on the tool. The lay-up robot is now manoeuvred so that the point of the tool is coincident with the centre of the cross-hair marks on the cutting table. The cross-hair marks are useful in this process since they enable the position of the tool point to be assessed independently in the X and Y axes. This process can be achieved by eye with good accuracy, since the human vision system is very good at detecting discrepancies between adjoining edge positions. Once the lay-up robot position has been optimally matched to the cross-hairs,

the coordinates are noted and stored to file. The coordinates of the calibration points are now known in both cutting table and robot coordinates, and so the transformation parameters for equation **(B.1)** can be extracted.

$$\begin{aligned} xoff &= x0 \\ yoff &= y0 \end{aligned} \qquad (B.2)$$

$$\theta = \tan^{-1}\left(\frac{y1 - y0}{x1 - x0}\right) \qquad (B.3)$$

$$Sx = \frac{\sqrt{(x1 - x0)^2 + (y1 - y0)^2}}{|C1|} \qquad (B.4)$$

$$Sy = \frac{\sqrt{(x2 - x0)^2 + (y2 - y0)^2}}{|C2|} \qquad (B.5)$$

where |C1| and |C2| represent the magnitude in cutting table coordinates of the displacement from the cutting table origin of C1 and C2 respectively. For points on the Xt or Yt axes this will simply be the Xt or Yt ordinate.

The accuracy of the mapping process between cutting table and lay-up robot was tested, again using the positioning tool. New marks were created at arbitrary positions on the cutting table, and using the cutting table coordinates of these marks with the mapping function defined in equation **(B.1)**, the lay-up robot was moved to its estimate of that position. The results over a relatively small area (e.g. 40cm^2) could not be faulted. However, over a larger area (the cutting table surface is 2.5m x 1.25m) there was a noticeable error in the position of the lay-up robot, up to a magnitude of approximately 3mm. There are several possible reasons for this. Firstly, either or both of the motion systems may exhibit non-linearity. Secondly, an error may be introduced if the two axes frames are not exactly co-planar i.e. the cutting table is not flat and/or the lay-up robot base is not in exactly the same

plane as the cutting table. Thirdly, the estimate of θ obtained from the calibration points will contain a degree of error arising from positional inaccuracies in the cutting table and lay-up robot. This error will result in significant deviations when points far from the cutting table origin are mapped.

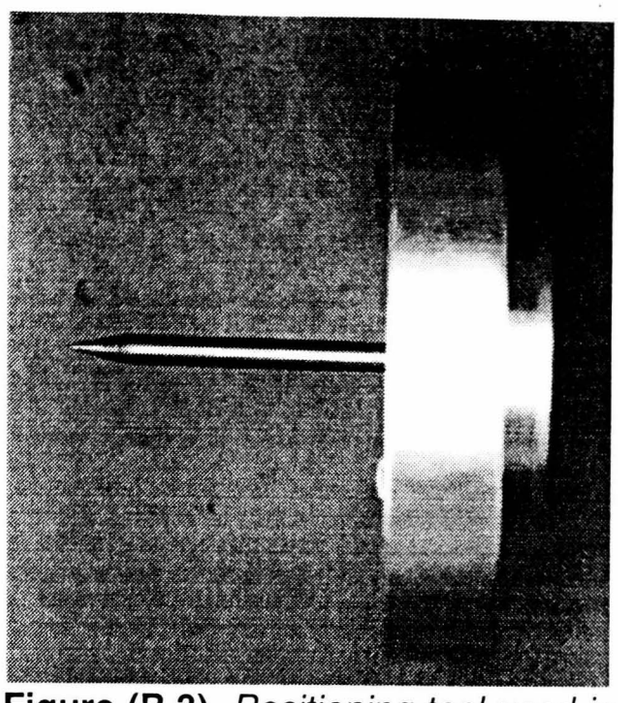


Figure (B.3). *Positioning tool used in calibration of the lay-up robot.*

In practice, all of the above errors will be present to some degree. To compensate for this, a more complex model would be required than the linear transformation adopted here. However, in this application there is a more pragmatic solution. It is not required to pick from any position on the cutting table. The cut position of the plies can be designated such that the plies are to be cut and picked from only a limited area of the cutting table. Therefore only this area need be calibrated. Serious errors arise only when the pick position is a significant distance from the point calibrated as the origin (C0 in **Figure (B.2)**). This error can be minimised by choosing the origin as being much closer to the actual pick positions, rather than at the actual cutting table origin. This is illustrated in **Figure (B.4)**. Using this approach, the error involved in ply picking has been minimised.

B.3 Vision System Calibration.

Part of the criteria for inspection of plies in the lay-up is that the ply boundaries detected by the vision system should be compared against the specification in the component design. Ideally, inspection should be based on the CAD data created when the component is designed. In order to achieve this a means of mapping the CAD coordinates to the vision system coordinates is required. The approach adopted here is to map both CAD coordinates and vision system coordinates to lay-up robot coordinates. The

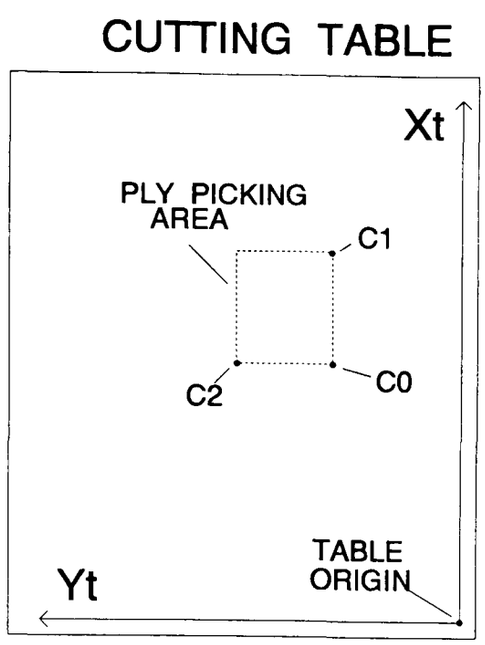


Figure (B.4). *Calibrated area of cutting table. Plies are cut such that their centre point lies within the area shown.*

lay-up robot coordinate system is therefore used as a common coordinate system enabling direct comparison between CAD data and vision system results. The CAD data must anyway be mapped into robot workspace for ply lay-up. Mapping vision system coordinates into robot workspace therefore offers an elegant solution to the inspection problem, enabling easy comparison between CAD data and vision system data. This task is analogous to that described in the previous section, in that the required mapping is between two X-Y coordinate frames. However in this case a linear transformation between coordinates will not suffice. The reason for this is that lens distortion will introduce non-linearities into the image. Experiments within the cell have shown that this distortion is present, but is modest with reasonable quality lenses (approximately one pixel across the field of view using a Tokina SZ-X 210 lens with the focal length set at 70mm). A slightly more complex mapping procedure has therefore been adopted to map between lay-up robot coordinates and vision system coordinates. The basic idea behind this is that for an array of co-planar points in the camera field of view, the vision system coordinates and the corresponding robot coordinates are obtained. Any image point can then be mapped to robot coordinates by reference to the nearest calibration point, and an offset from that point which is a function mapping the offset in pixels to an offset in robot coordinates. The accuracy of the overall mapping is dependant on the number of calibration points (since for a fine array of calibration points the offset from any calibration point is small and so the potential error in offset small), and the function used to map the offset. Since the estimated distortion over a full image frame of 720x512 pixels is only approximately one pixel, then a linear offset function has been adopted. If the

distortion of the camera lens is more severe, a more suitable spline-based mapping function can be used **[Jones, 1994]**.

Mapping between image and lay-up robot coordinates is illustrated in **Figure (B.5)**. For a point P to be mapped from lay-up robot coordinates to image coordinates, the nearest calibration point is determined, and designated as C0. C1 is chosen as the Xi axis neighbour of C0 nearest to P. C2 is chosen as the Yi axis neighbour of C0 nearest to P. The three points C0, C1, and C2 form a "mini-coordinate frame" within which a linear model is assumed to provide a good approximation for coordinate mapping. Equation **(B.1)** can therefore be used to provide the mapping, and the parameters for equation **(B.1)** can be extracted from equations **(B.2)**, **(B.3)**, **(B.4)**, and **(B.5)** as before.

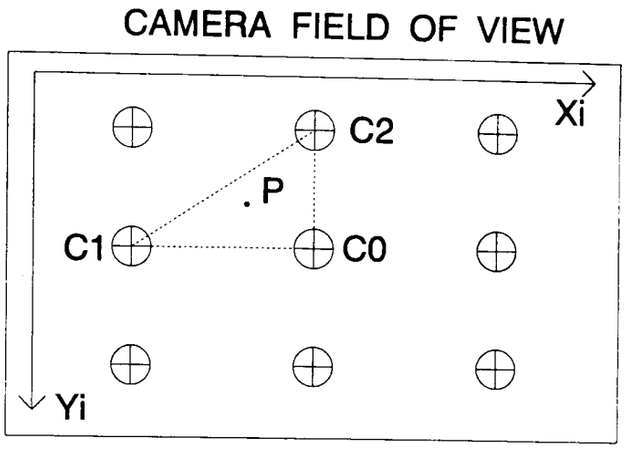


Figure (B.5). *The camera field of view showing the image coordinate system. The robot and image coordinates for the calibration array are known. The robot coordinates of point P are known.*

As with calibration of the cutting table, a means of obtaining calibration points in both coordinate frames is required. This has been accomplished as follows. An array of circular "targets" has been affixed to the surface of the lay-up table as shown in **Figure (B.6)**. Each target is a white circle printed on a black background, with a white cross overlaid in the centre of the circle. This cross is not visible in **Figure (B.6)**, due to a high level of ambient lighting. In fact this is desirable for the vision systems' view of the targets, since the method used to detect the circle centre does not use the cross, but rather uses the circularity of the target as described below. The crosses are, however, used to aid positioning of the lay-up robot over the centre of each target (again using the positioning tool shown in **Figure (B.3)**). Once obtained, the lay-up robot coordinates for the centre of each target are stored to file.

The lay-up robot has now, in effect, been calibrated against the lay-up

table. Now by calibrating the vision system against the lay-up table (again using the targets), the vision system and lay-up robot can be calibrated against each other indirectly. The advantage of this approach is that if the camera is moved for any reason (i.e. change of focus or aperture), or even if a new lens is to be used, then the camera can be re-calibrated with the lay-up table (and so with the lay-up robot) without the need for any robot involvement. In fact this approach means that the camera can be re-calibrated before lay-up of each component. This would eliminate any temporal variation, such as CCD drift.

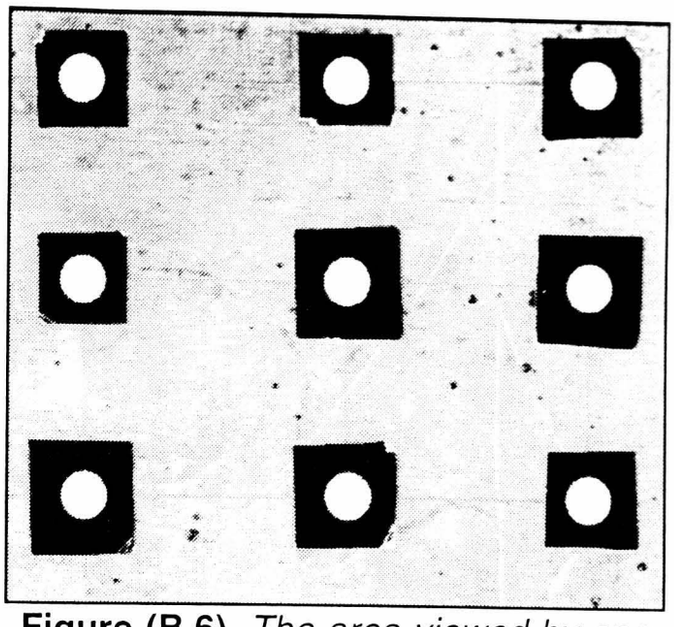


Figure (B.6). *The area viewed by one lay-up camera, showing the array of calibration targets on the lay-up table.*

The algorithm to determine vision system coordinates of the calibration points is shown in **Figure (B.7)**. An image containing the calibration points is obtained (e.g. **Figure (B.6)**), and a grey-level histogram is formed from this image. A threshold value is automatically selected by interrogating the histogram and selecting the value which will threshold out a pre-determined number of pixels, chosen as approximately the number of pixels represented by the targets. The image is thresholded, and the location of each target obtained by analysis of the "blob" positions in the thresholded image. The centroid of each target (blob) can now be determined to subpixel accuracy **[Zakaria et al, 1987]**. In this way an accurate estimate of the centre point of each target in image coordinates is obtained. In addition, the validity of the estimate of target centre can be checked by calculation of the second moment invariant **[HU, 1961]**. For a circle this value should be near it's lower bound, which is approximately 0.159. If the targets have been successfully detected and are found to be circular, then the centroid coordinates are written to file, and represent the calibration points in image coordinates. The calibration between camera and robot is now complete. The way in which the calibration

data is used is detailed in the following section.

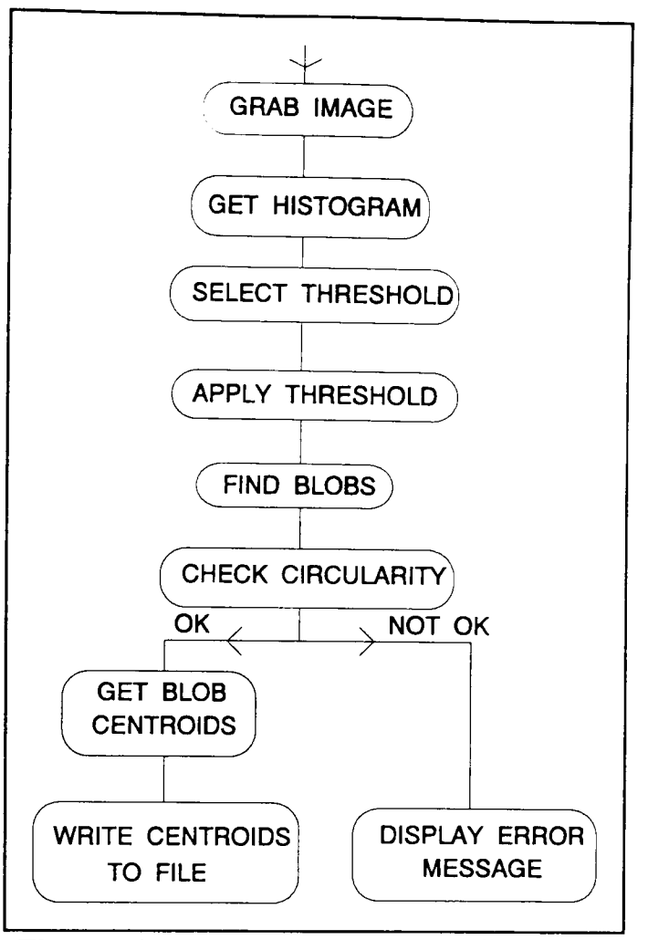


Figure (B.7). *Extracting coordinates of calibration points from image shown in* ***Figure (B.6)***.

B.4 Inspection from CAD.

This section will provide a brief overview of how inspection from CAD is achieved within the cell. At present this is only implemented for rectangular shaped plies, but work is ongoing to extend this functionality to include arbitrary ply shapes.

When a ply is laid-up on the pre-form stack, the centre point of the ply in lay-up robot coordinates, (Cx_r,Cy_r), can be estimated from knowledge of the ply cut position, lay-up robot pick position, and lay-up robot place position. Since the dimensions of the ply are known in millimetres, and there is a direct mapping from millimetres to robot coordinates (1mm = 8 increments in robot coordinates), then the vertices of a ply in robot coordinates can also be estimated. For example, the lay-up robot coordinates of V0 in **Figure (B.8)** are calculated as

$$V0 = \left(\left(Cx_r - 8\frac{dx}{2} \right), \left(Cy_r + 8\frac{dy}{2} \right) \right) \qquad (B.6)$$

Given that the coordinates of V0...V3 have been calculated and that the robot coordinates of the corners of the camera field of view are known (from the camera calibration process), then the line parameters (i.e. *m* and *c*) in lay-up robot space for each of the ply edges and for each of the field of view bounding lines can be calculated. Any points of intersection between ply edges

and camera field of view can then be calculated. These intersection points can then be mapped from robot coordinates into image space using the procedure described in the previous section. At present only ply edges which intercept both left and right (or top and bottom) edges of the field of view are mapped into image space to avoid the problem of detecting edge end-points near corners. **Figure (B.9)** shows an image taken from the prototype cell during lay-up of a sample component. Edges of four plies are visible in the image, and the expected ply edge position extracted from the CAD mapping procedure described above is shown overlaid. The field of view is considered to be only within the calibration array. As can be seen the agreement is good. From measurements taken using the vision system, the agreement between ply lay-up position and expected lay-up position extracted from CAD is better than one pixel. One pixel represents an area of approximately 0.2mm^2 of the lay-up table.

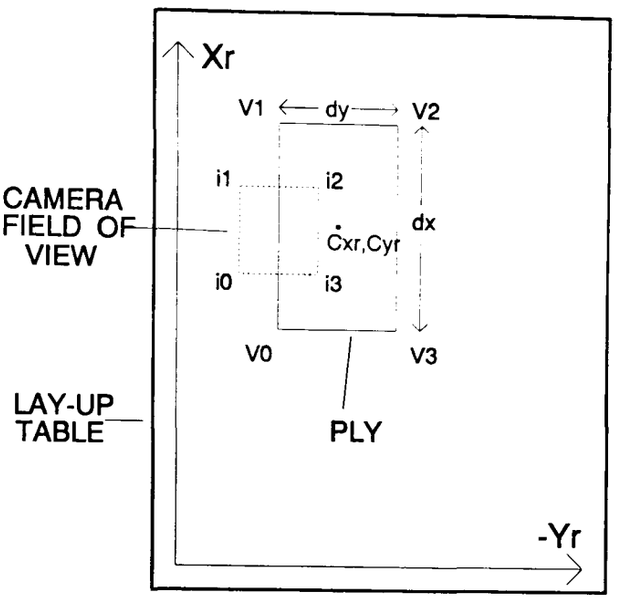


Figure (B.8). *The lay-up table shown in the lay-up robot coordinate frame. The intersection of a ply with a camera field of view is shown.*

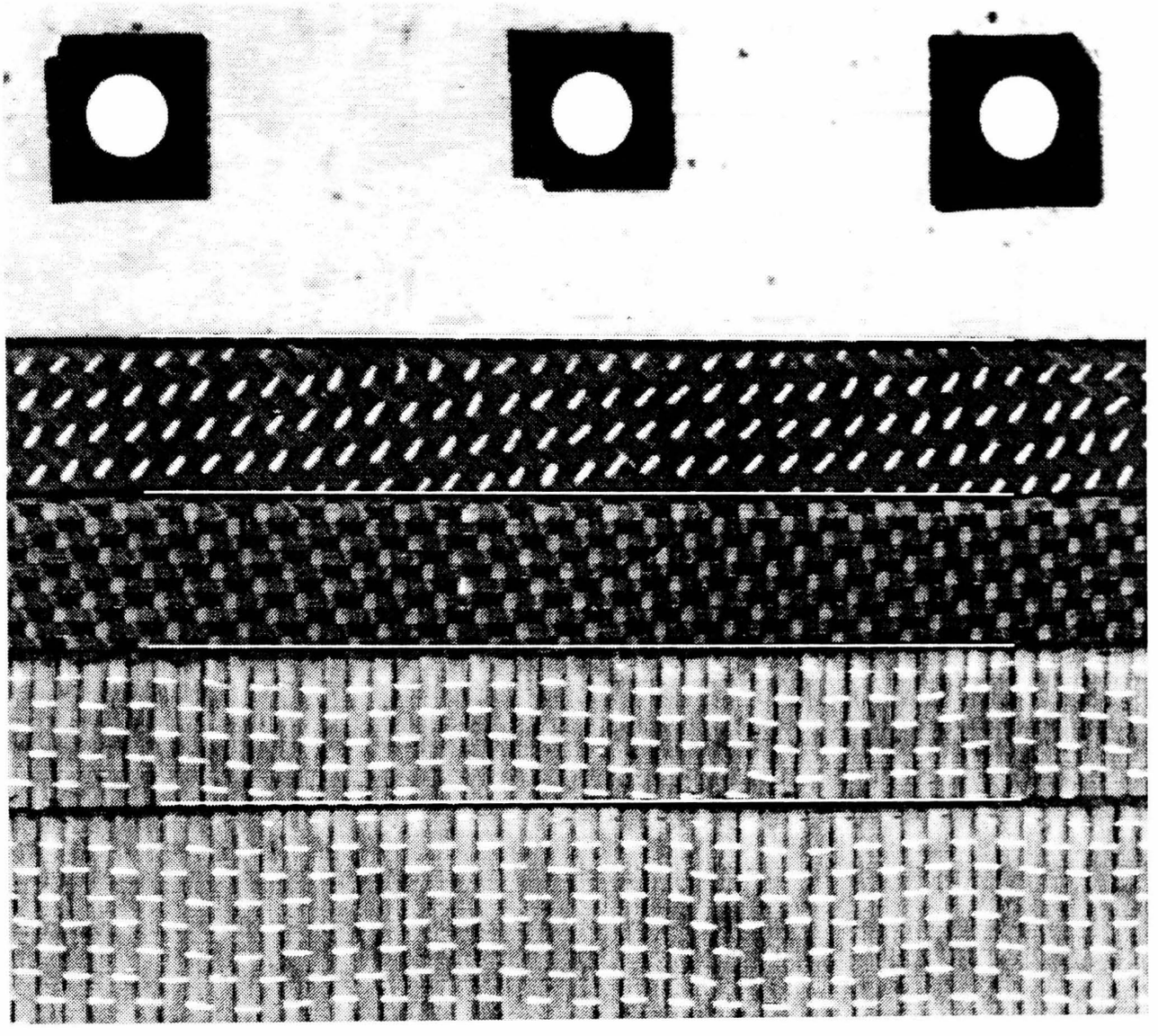


Figure (B.9). *A lay-up image taken showing four ply edges, and the expected edge positions extracted from CAD data.*

APPENDIX C

Lay-Up Images.

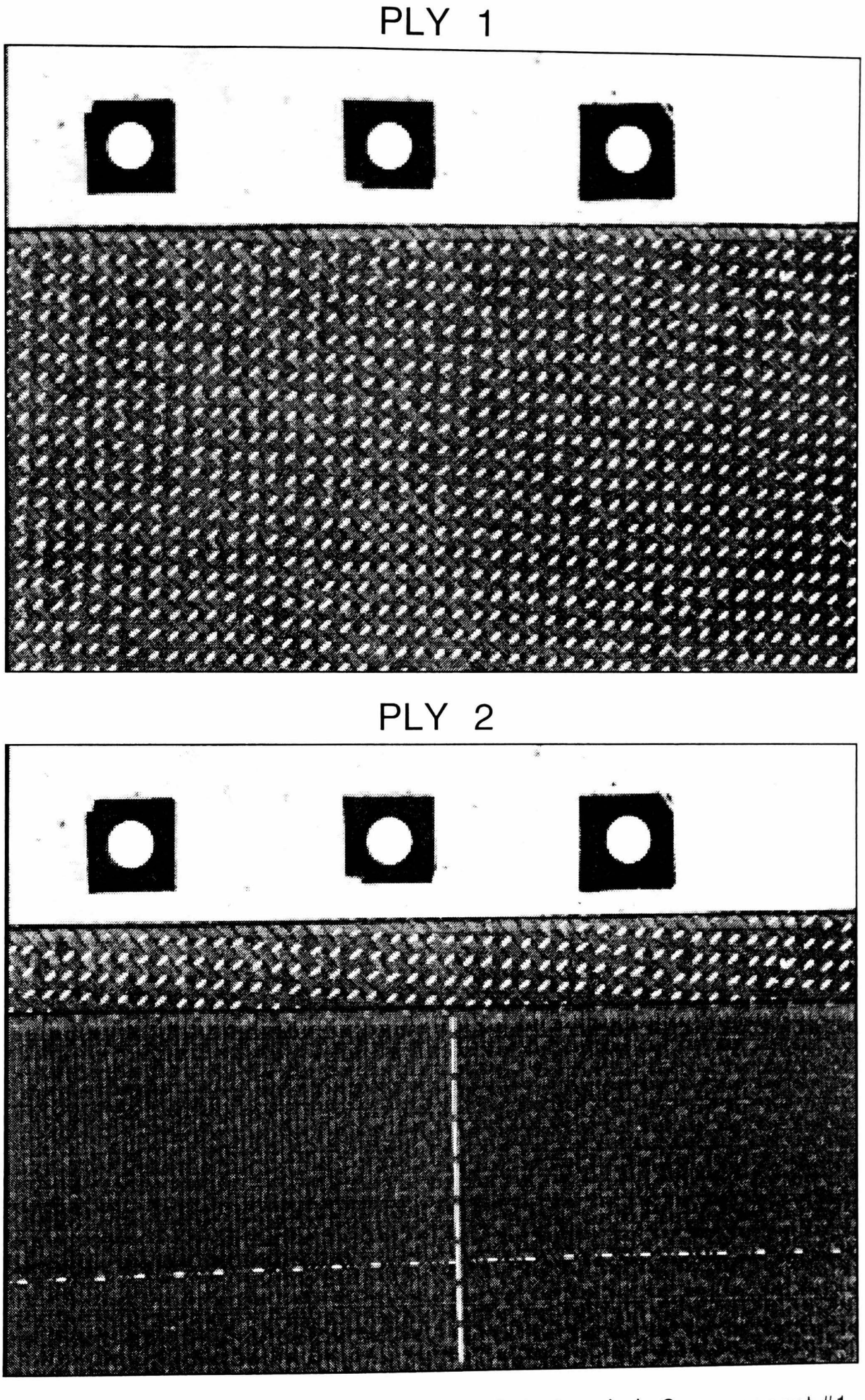


Figure (C.1). Input images for inspection of ply 1 and ply 2, component #1.

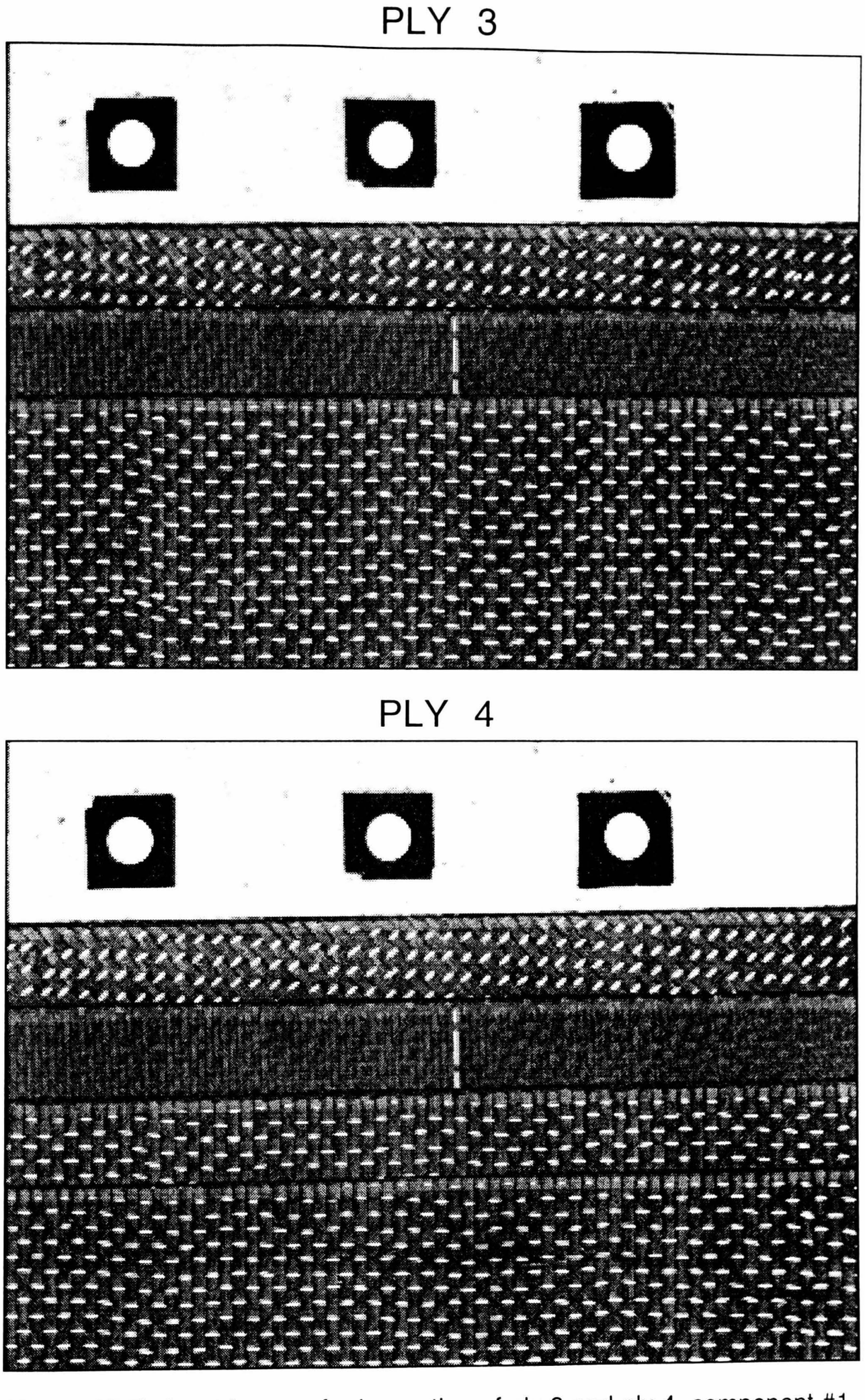


Figure (C.2). Input images for inspection of ply 3 and ply 4, component #1.

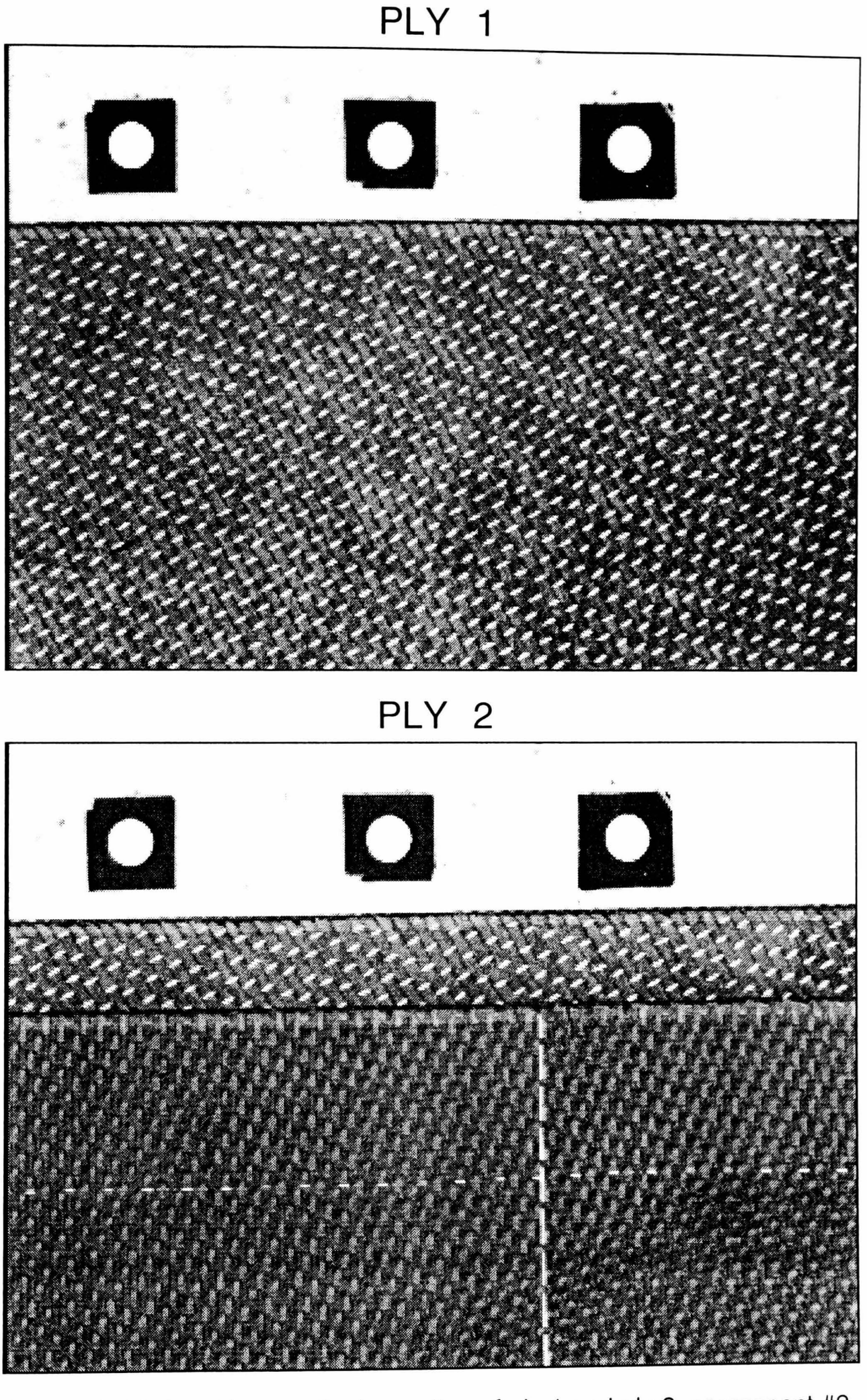


Figure (C.3). Input images for inspection of ply 1 and ply 2, component #2.

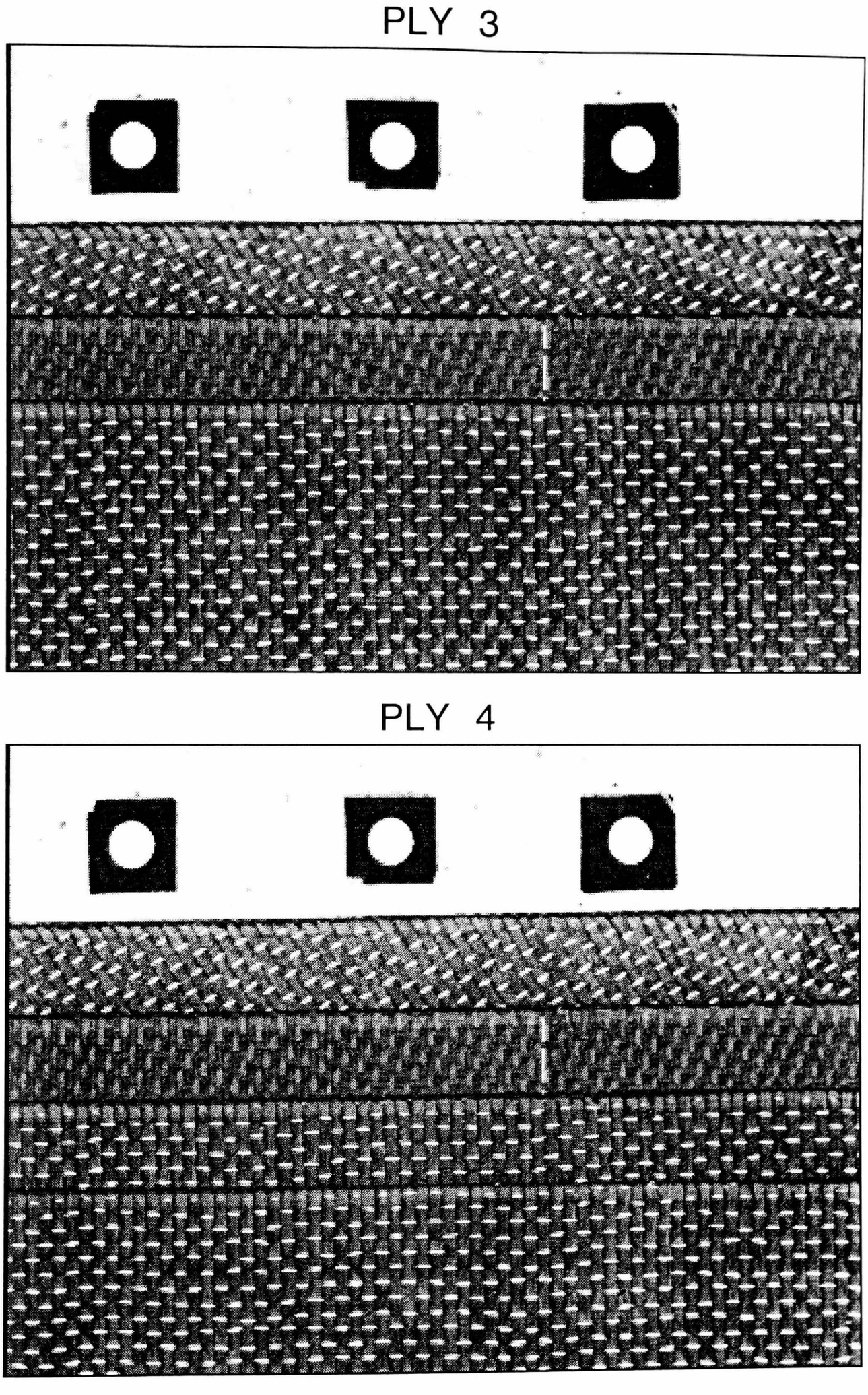


Figure (C.4). Input images for inspection of ply 3 and ply 4, component #2.

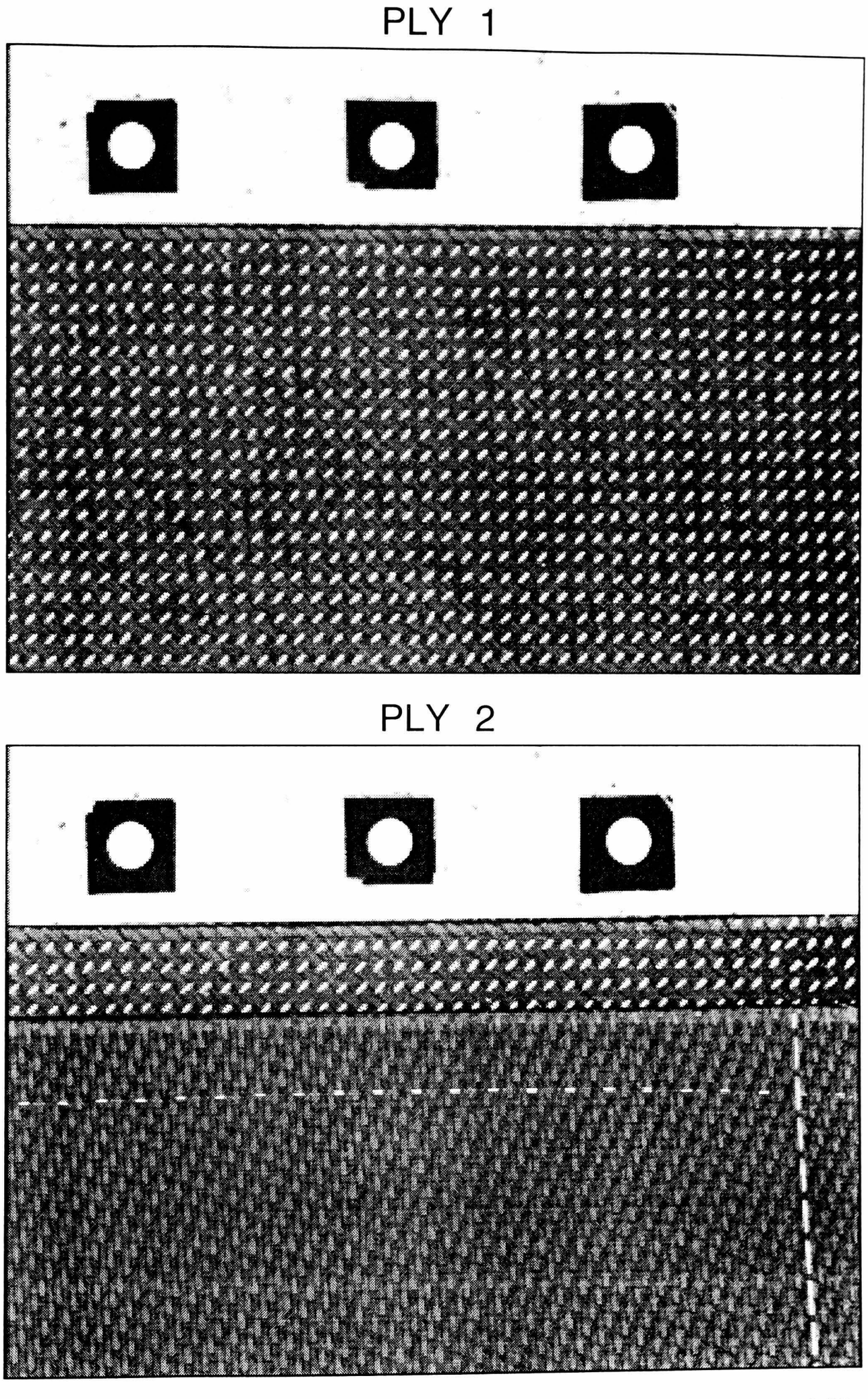


Figure (C.5). Input images for inspection of ply 1 and ply 2, component #3.

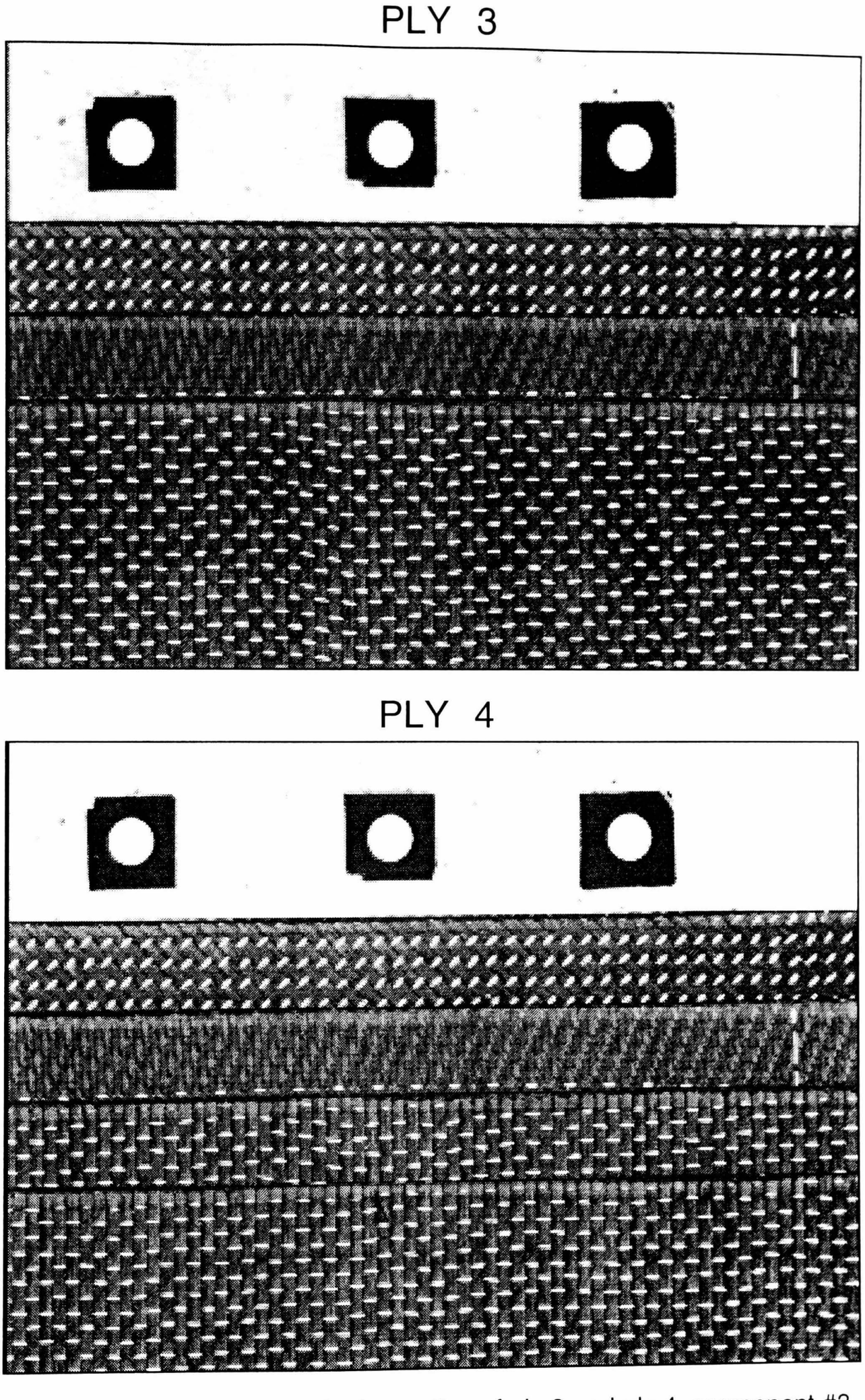


Figure (C.6). Input images for inspection of ply 3 and ply 4, component #3.

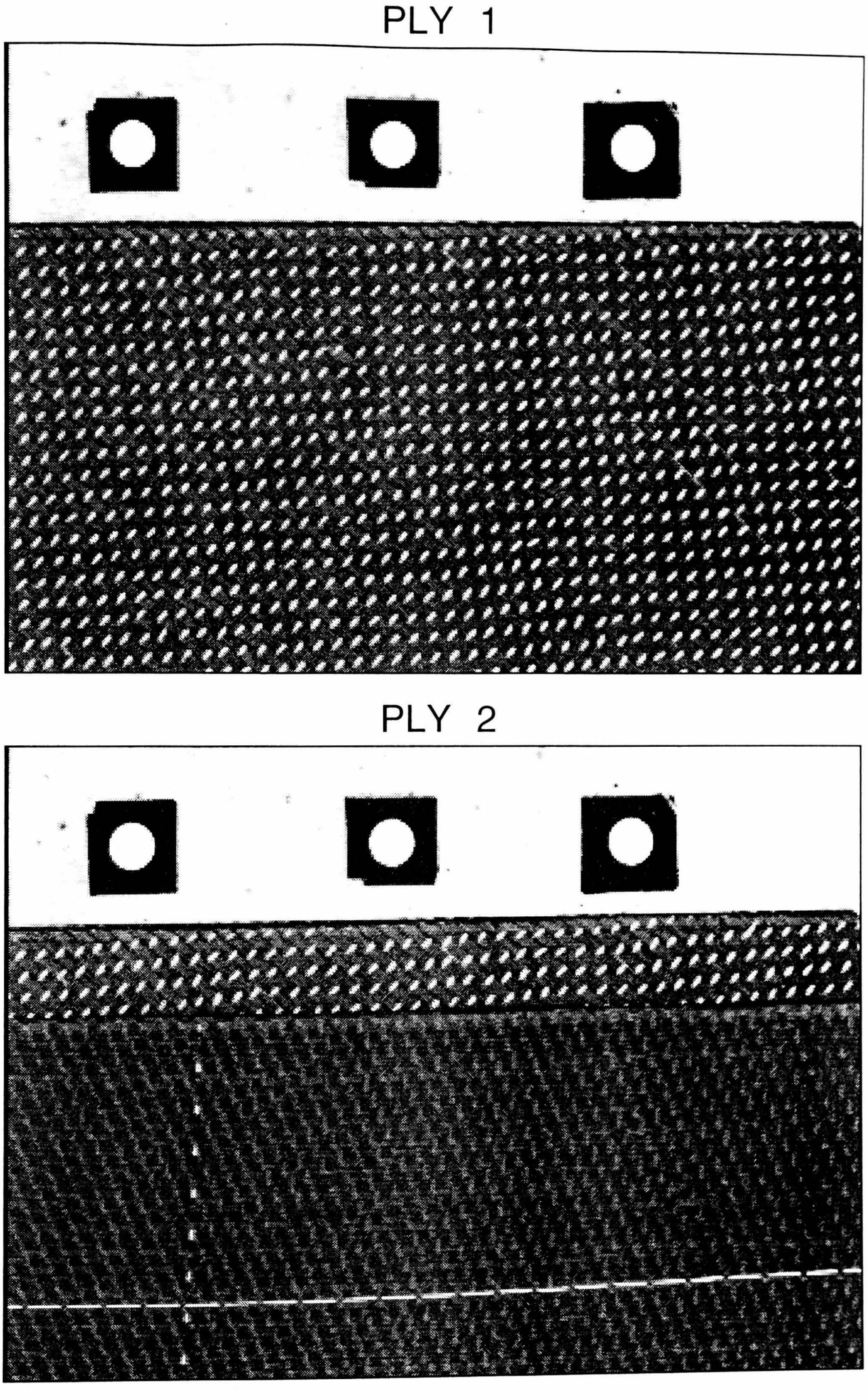


Figure (C.7). Input images for inspection of ply 1 and ply 2, component #4.

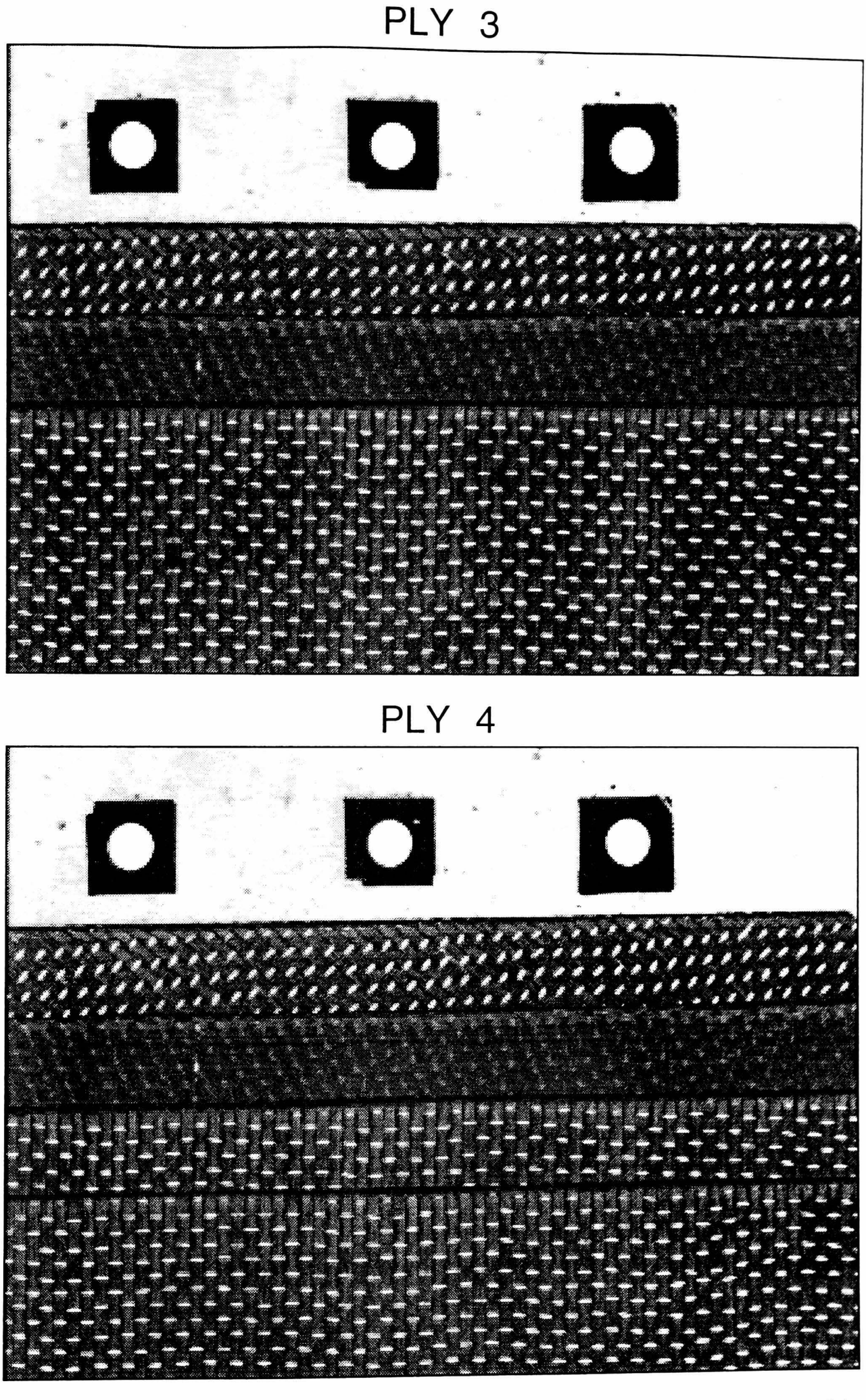


Figure (C.8). Input images for inspection of ply 3 and ply 4, component #4.

References

Every effort has been made to provide complete references, but this has not proven possible in all cases. Incomplete references have been marked with an asterisk in the following list. It has been considered necessary to include these references from an academic viewpoint.

*[Ade] Original Reference Lost.

[Ade,1983a] "Characterisation of Texture by Eigenfilter.", Signal Processing, Vol.5, Part 5, pp451-457, 1983.

[Ade,1983b] "Application of Principle Component Analysis to the Inspection of Industrial Goods.", Proc SPIE, Vol.397, pp216-223, 1983.

[Ait-Kheddache,1988] "Classification of Textures Using Higher Order Fractal Dimensions.", IEA/AIE-88, New York, ACM, 1988, pp679-688.

[Aleksander,Morton,1990] An introduction to neural computing. Chapman and Hall, 1990.

[Bajcsy,1973] "Computer Descripion of Textured Images.", Proc. 3rd Int. Joint Conf. on Artificial Intelligence, pp572-579, 1973.

[Ballard and Brown, 1982] Computer Vision, Prentice Hall, 1982.

[Bellman, Dreyfus,1962] "Applied Dynamic Programming."rinceton, NJ; Princeton University Press, 1962.

[Benke et al,1988] "Convolution Operators as a Basis for Objective Correlates of Texture Perception.", IEEE Trans. Systems Man Cybernetics, Vol.18, No.1, Jan/Feb 1988.

[Benke, Skinner,1987] "Segmentation of Visually Similar Textures by Convolution Filtering.", The Australian Computer Journal, Vol.19, No.3, August 1987.

[Bergen,Adelson, 1988] "Early Vision and Texture Perception.", Nature, Vol.333, 26 May, 1988.

[Besag,1972] "On the Statistical Analysis of Nearest Neighbour Systems", Colloquia Mathematica Societatis, Janos Bolyai, Ninth European Meeting of Statisticians, Budapest, pp 101-105, 1972.

[Bolles, 1977] "Verification Vision for Programmable Assembly", Proc. 5th Int. Conf. on Artificial Intelligence, August, 1977, pp569-575.

[Bolles,Cain,1982] "Recognising and Locating Partially Visible Objects: The local-Feature-Focus Method", Robotics Research, Vol.1, No.3, 1982.

*[Borghesi,Cantoni and Diani] "An Industrial Application of Texture Analysis.", Proceedings ???, 19??, (undated photocopy).

[Boubekraoui et al, 1984] Original refernce lost.

[Brodatz, 1966] Textures: A Photographic Album for Artists and Designers. New York:Dover, 1966.

[Brzakovic et al,1990] "An Approach to Defect Detection in Materials Characterized by Complex Textures.", Pattern Recognition Vol.23,No.1/2,pp99-107,1990.

[Caelli,1988] "An Adaptive Computational Model for Texture Segmentation.", IEEE Trans. Systems, Man, Cybernetics, Vol. 18,No.1, Jan/Feb, 1988.

[Caelli,Julesz,1978] "On Perceptual Analyzers Underlying Visual Texture Discrimination: Part 1", Biological Cybernetics, vol.28, pp167-175, 1978.

[Caelli,Moraglia,1985] "On the Detection of Gabor Signals and Discrimination of Gabor Textures." Vision Research, vol 25, no 5, pp671-684, 1985.

[Campbell,Robson,1968] "Application of Fourier Analysis to the Visibility of Gratings.", Journal Physiology, 197, pp551-566, 1968.

[Cano,Minh,1988] "Texture Synthesis Using Hierarchical Linear Transforms.", Signal Processing 15, pp131-148, 1988.
[Chen, Sarhadi, 1992] "Investigation of Electrostatic Force for Robotic Lay-Up of Composite Fabrics", Mechatronics, Vol.2, No.4, 1992.
[Chen and Pavlidis,1978] "Segmentaion by Texture Using a Co-Occurrence Matrix and a Split-and-Merge Algorithm.", CGIP 10, pp172-182, 1979.
[Chen,Dubes,1989] "Experiments in Fitting Discrete Markov Random Fields to Textures.", IEEE, 1989.
[Coggins,Jain,1985] "A Spatial Filtering Approach to Texture Analysis.", Pattern Recognition Letters, Vol.3, pp195-203, 1985.
[Cohen et al, 1989] "Classification of Natural Textures by Means of Two-Dimensional Orthogonal Masks.", IEEE Trans. Acoustics, Speech, Signal Processing, Vol.37, No.1, Jan 19898, pp125-128.
[Cohen,1986] "Markov Random Fields for Image Modelling and Analysis", in "Modelling and Application of Stochastic Processes", Edited by Uday B. Desai, Kluwer Academic Publishers, Boston,1986.
[Cohen,Cooper,1987] "Simple Parallel Hierarchical and Relaxation Algorihms for Segmenting Noncausal Markovian Random Fields.", IEEE PAMI, PAMI-9, No.2, March 1987.
[Cross,1980] "Markov Random Field Texture Models.", PhD dissertation, Dept. Computing SCience, Michigan State University, 1980.
[Cross,Jain,1983] "Markov Random Field Texture Models.", IEEE PAMI, Vol. PAMI-5, No.1, Jan. 1983.
[Daily,1989] "Colour Image Segmentation using Markov Random Fields.", IEEE, 1989.
[Daugman,1980] "Two-dimensional spectral analysis of cortical receptive field profiles." Vision Research, 20, 1980, pp.847-856.
[Davies,1987] "Design of Optimal Gaussian Operators in Small Neighbourhoods.", Image and Vision Computing, Vol.5, No.3, August 1987.
[Davies, 1990] Machine Vision: Theory, Algorithms, Practicalities. London, Academic, 1990.
[Davis,Clearman and Aggarwal,1981] "An Empirical Evaluation of Generalized Co-Occurrence Matrices.", IEEE PAMI, PAMI-3, NO.2, March 1981.
[Davis,Johns and Aggarwal,1979] Texture Analysis Using Generalised Co-Occurrence Matrices.", IEEE PAMI, PAMI-1, No.3, July 1979.
[De Valois et al,1982] "Spatial Frequency Selectivity of Cells in Macaque Visual Cortex.", Vision Research, 22, pp545-559, 1982.
[Derin,Elliot,1987] "Modeling and Segmentation of Noisy and Textured Images Using Gibbs Random Fields.", IEEE PAMI, PAMI-9, No.1, Jan 1987.
[Dewaele et al,1990] "Automatic Visual Inspection of Textured Textiles.", Proc. 5th Int. Conf. on Image Analysis and Processing, Italy 20-22 Sept. 1989.
*[Dinstein et al] "Fast Discrimination Between Homogenous and Textured Regions.", ?, 198?, (undated photocopy).
[Dorst, Smeulders, 1984] "Discrete representation of straight lines.", IEEE Trans. PAMI, Vol.6, 1984, pp450-463.
[Du Buf et al, 1990] "Texture Feature Performance for Image Segmentation.", Pattern Recognition, Vol.23, No.3/4, pp291-309, 1990.
[Duda,Hart,1973] Pattern Classification and Scene Analysis. Wiley, New York, 1973.
[Eichmann,Kasparis,1989] "Pattern classification using a linear associative memory.",Pattern Recognition, vol.22, No.6, pp.733-740, 1989.
[Fan,1989] "An Edge-Based Hierarchical Algorithm for Textured Image Segmentation.", 1989, IEEE.
[Faugeras,1978] "Texture Analysis and Classification Using a Human Visual Model.", Proc. 4th Int. Joint Conf. on Pattern Recognition, Kyoto, pp549-552, Nov. 1978.
[Fouques,Cohen,1989] "Partition Filters: A New Class of Morphological Operators for Segmenting Textured Images.", IEEE 1989.

[Gabor,1946] "Theory of Communication.", Journal of Instrumentation and Electrical Engineering, Vol.93, pp429-457, 1946.
[Gool et al, 1983] "Survey: Texture analysis anno 1983." CVGIP vol 29, pp336-357,1985.
[Gopal et al, 1990] "Multiple Channel Surface Orientation from Texture."Proc. SPIE Vol.1249, pp366-375, 1990.
[Gotlieb and Kreyszig,1990] "Texture Descriptors Based on Co-Occurrence Matrices.", CVGIP 51, pp70-86, 1990.
[Granland,1980] "Description of Texture Using the General Operator Approach." Proc. 5th Int. Conf. on Pattern Recognition, IEEE, Computer Soc., 1980, pp776-779.
[Grossberg, Mingolla, 1985] "Neural Dynamics of Perceptual Grouping: Textures,Boundaries, and Emergent Segmentation.", Perception and Phychophysics, Vol. 38, pp141-171,1985.
*[Haddon and Boyce,1989] "Simultaneous Image Segmentation and Edge Detection."Proceedings ???, pp411-415, 1989, (source unclear).
[Hall et al,1972] "A survey of preprocessing and feature extraction techniques for radiographic images", IEEE Trans. Comput., Vol.C-20, pp.1032-1044, Sept. 1971.
[Haralick et al,1973] "Textural Features for Image Classification." IEEE Systems Man Cybernetics, Vol SMC-3, No.6, November 1973.
[Haralick et al,1987] "Image analysis using mathematical morphology",IEEE Trans. PAMI, vol.9, No.4, July 1987.
[Haralick,1979] "Statistical and Structural Approaches to Texture.", Proc. IEEE, Vol.67, No.5, May 1979.
[Harmuth,1972] "Transmission of information by orthogonal functions",Springer-Verlag, Berlin and New York.
[Harwood et al,1983] "Texture Classification by Local Rank Correlation.", Place of publication unknown (ref: TR-1314 on paper), 1983.
[Havelock, 1989] "Geometric Precision in Noise-Free Digital Images.", IEEE PAMI, Vol.11,No.10,October 1989.
[He,Wang and Guibert,1987] "Texture Discrimination Based on an Optimal Utilization of Texture Features.", Pattern Recognition, Vol. 21, No.2, pp141-146, 1988.
[He,Wang,1990] "Texture Features Based on Texture Spectrum.", Pattern Recognition, Vol.24, No.5, pp391-399, 1990.
[He,Wang,1991] "Textural Filters Based on the Texture Spectrum.", Pattern Recognition, Vol.24, No.12, pp1187-1195, 1991.
[Hooley, 1993] "Design of 2-D Tactile Measurement System.", Final Year Project Report, Dept. M&ES, Brunel University, 1993.
[Horowitz and Pavlidis,1976] "Picture Segmentation by a Tree Traversal Algorithm.", ACM Vol.23, No.2, April 1976, pp368-388.
[Hoskin,Baker,1986] Composite Materials for Aircraft Structures, Published by American Institute of Aeronautics and Astronautics, New York, 1986.
[Hough,1962] Method and means for recognising complex patterns. US Patent 3069654.
[Hsiao,Sawchuk,1989] "Supervised Texture Segmentation USinf Feature Smoothing and Probabilistic Relaxation Techniques.", IEE PAMI Vol.11,No.12,December 1989, pp1279-1292.
[Huertas, Medioni, 1986] "Detection of Intensity Changes with Subpixel Accuracy Using Laplacian-Gaussain Masks.", IEEE PAMI,Vol.PAMI-8,No.5,September 1986.
*[Ikonomopoulos,Unser,1984] "A Directional Filtering Approach to Texture Discrimination.", Signal Processing, ?, 1984.
[Isaacson and Keller,1966] "Analysis of Numerical Methods", John Willey and Sons, New York, 1966.
[Ising,1925] "Zeitschrift Physik", vol.31, p.253, 1925.
[Iwama,Maida,1989] "Organizing and Integrating Edge Segments.", Journal of Experimental and Theoretical Artificial Intelligence, Vol.1, Part 2, pp113-132, 1989.
[Jain,1988] Algorithms for Clustering Data. Prentice Hall, 1988.
[Jain,Farrokhnia,1991] "Unsupervised Texture Segmentation Using Gabor Filters.", Pattern Recognition, Vol.24, No.12, pp1167-1186, 1991.

[Jardine,Whitworth,1990] "Fractal Based Synthesis of Visual Texture.", Application of Fractal Techniques in Image Processing, IEE Colloquium, London, DEcember 1990, pp8/1-4.
[Jarvis,1992] Automated Lay-Up of Composite Blades, MSc thesis, Durham University, 1992.
[Jeng,Woods,1989] "Texture Discrimination Using Doubly Stochastic Gaussian Random Fields.", IEEE, 1989.
[Jones, 1994] Vision-Guided Robotic Handling of Garment Sub Assemblies, PhD thesis, Dept. M&ES Brunel University.
[Julesz,1962] "Visual Pattern Discrimination." IRE Transactions Information Theory, vol 8, no 2, pp84-92, Feb 1962.
[Julesz,Bergen,1981] "Textons, The Fundamental Elements in Preattentive Vision and Perception of Textures.", The Bell System Technical Journal, Vol.62, No.6, July-August 1983.
[Kadar,Liebman,1988] "Robust Texture Extractors for Real-Time Pyramidal Architectures.", Proc. SPIE Vol.939, Hybrid Image and Signal Processing, pp48-57, 1988.
[Kaneko,1989] "A Generalized Fractal Dimension and its Application to Texture Analysis.", IEEE 1989.
[Kashyap et al,1982] "Texture Classification Using Features Derived from Random Field Models.", Patern Recognition Letters 1, pp43-50, 1982.
[Kasparis et al,1990] "Image Pattern Algorithms Using Neural Networks.", Proc. SPIE Vol.1297, Hybrid Image and Signal Processing II, pp298-306, 1990.
[Khotanzad,Chen,1989] "Unsupervised Segmentation of Texture Images by Edge Detection in Multidimensional Features.", IEEE Trans. PAMI, Vol.11,Part 4,pp414-421,1989.
[King, 1994] A Machine Vision System for Texture Segmentation. PhD thesis, Dept. M&ES Brunel University, 1994.
[Klaasman, 1975] "Some Aspects of the Accuracy of the Approximated Position of a Straight Line on a Square Grid.", CGIP, Vol.4, pp225-325, 1975.
[Koplowitz, Lee, 1991] "Edge Detection with Subpixel Accuracy.", Proc. SPIE Vol.1471 Automatic Object Recognition 1991.
[Kreysig, 1988] Advanced Engineering Mathematics, John Wiley & Sons, 1983.Kruger,Thompson and Turner,1974] "Computer Diagonals of Pneumoconiosis.", IEEE Transactions Systems Man Cybernetics, Vol SMC-4,o.1, Jan. 1974.
[Landau, 1986] "Estimation of a Circular Arc Center and its Radius.", CVGIP 38, 317-326, 1987.
[Laws,1980] Textured Image Segmentation, PhD dissertation, Dept. Engineering, University of Southern California, 1980.
[Lippman,1987] "An introduction to computing with neural nets." IEEE ASSP Magazine, April 87, pp.4-22.
[Lyvers et al, 1989] "Subpixel Measurements Using a Moment Based Edge Operator.", IEEE PAMI Vol.11,No.12,December 1989.
[Malik,Perona,1989] "A Computational Model of Texture Segmentation.", 22nd Asilomar Conf. on Signals, Systems and Computers, LA, 31Oct-2Nov 1989.pp490-494.
[Mandelbrot,1983] The Fractal Geometry of Nature, San Francisco CA: freeman 1983.
[Manjunath et al, 1990] "Stochastic and Deterministic Networks for Texture Segmentation.", IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol.38, No.6, pp1039-1049, 1990.
[Marr,1982] Vision, W.H. Freeman, New York, 1982.
[Marr,Hildreth,1980] "Theory of Edge Detection.", Proceedings Royal Society, London B, Vol.207,pp187-217,1980.
[McCarthy,1981] "Fifteen Years Experience with Composite Propellor Blades.", European Chapter SAMPE Int. Conf., Cannes, France 12-14, Jan 1981.
[Metropolis et al,1953] "Equations of State Calculations by Fast Computing Machines.", Journal of Chemical Physics, Vol.21, pp1087-1091, 1953.
[Michel et al,1989] "Unsupervised Segmentation of Texture Images.", Proc. SPIE, Vol. 1001, part 1, pp 582-590, 1989.

[Middleton, 1990] Composite Materials in Aircraft Structures, Harlow, Longman Scientific and Technical, 1990.
[Mitchell, 1990]"Machine Vision Techniques for Inspection of Composite Components", First Annual Report, Durham University, 1990.
[Mitchell et al,1977] "A Max-Min Measure for Image Texture Analysis.", IEEE Trans. Computers, April 1977. pp408-415.
[M. Sarhadi, X.Q. Chen, T.A. Mitchell, et al, 1991] Progress Towards AutomatedManufacturing of High Performance Components. ACME Research Conference, August 28-30, 1991, Leicester University, UK.
[T.A. Mitchell, M. Sarhadi, 1992] A Machine Vision System using Fast TextureAnalysis for Automated Visual Inspection. IEEE Int. Conf. on Systems Engineering, September 17-19, 1992, Kobe, Japan.
[T.A. Mitchell, M. Sarhadi, 1992] Optimising Edge Operators for Texture Boundary Detection. ICARV '92, 15-18 September 1992, Hyatt Regency, Singapore.
[X.Q. Chen, T.A. Mitchell, M. Sarhadi, 1992] A Vision-Integrated CAD Tool for Robotic Handling of Composite Materials. IMS '92, October 1-2, 1992, USA.
[M.Sarhadi, Z.Zhang, T.Mitchell, et al., 1993] Integrating Novel Tools and Techniques for Automated Manufacturing of Advanced 3-D Preform Structures. ACME Research Conference, Sheffield University, 7-10th September, 1993.
[T.A. Mitchell, Z. Zhang, M.Sarhadi, 1993] A Comparative Study of Texture Features for Automated Visual Inspection of Leather. Proceedings of the International Conference on Automotive Technology and Automation, Mechatronics Conference, Aachen, Germany, 13-17th September, 1993.
[M. Sarhadi, T.A. Mitchell, R.A.F. McCarthy, 1993] Robotic Lay-Up of High Quality Composite Preforms. Proceedings of the 14th International European Chapter Conference of the Society for the Advancement of Material and Process Engineering, Birmingham, England, October 1993.
[T. Mitchell, J. Chestney. M. Sarhadi, 1994] A Prototype Assembly Cell for Manufacture of Carbon Composite Components. Proceedings of EURISCON'94, Malaga, Spain, August 22-25, 1994.
[Monte et al,1988] "Texture Isolation by Adaptive Digital Filtering.", Image and Vision Computing, Vol.6, No.3, August 1988.
[Nakamura, Aizawa, 1984] "Digital Circles", CVGIP, Vol.26, pp242-255, 1984.Nawla, Binford, 1986] "On Detecting Edges.", IEEE Trans. PAMI, pp651-664, Sept. 1986.
[Niblack,1985] An Introduction to Digital Image Processing, Strandberg, Birkeroed, Denmark, 1985.
[Nomura et al, 1991] "Edge Location to Subpixel Precision and Analysis", Systems and Computers in Japan, Vol.22, No.9, 1991.
[Patel,Stonham, 1991] "A Single Layer Neural Network for Texture Discrimination.", IEEE Int. Symposium on Circuits and Systems, 1991.
[Patel,Stonham,1992] "Segmentation of Potash Mine Images Using Multi Layer Perceptron Networks.", IEEE Int. Conf on Automation, Robotics and Computer Vision, Singapore, 15-18 September 1992.
[Pavlidis, Liow, 1990] "Integrating Region Growing and Edge Detection.", IEEE Trans. PAMI,Vol.12,No.3,March 1990.
[Peleg et al,1983] Multiple resolution texture analysis and classification. CS-TR-1306, MCS-82-18408. Univ. of Maryland, July 1983.
[Pentland,1984] Fractal-based description of natural scenes. IEEE Trans. PAMI, PAMI-6, 1984, pp.661-674.
[Perkins,1978] "A Model-Based Vision System for Industrial Parts", IEEE Transactions Computers, Vol.C-27, No.2, February 1978.
[Perry,Lowe,1989] "Segmentation of Non-Random Textures Using Zero Crossings." Proc. IEEE In. Conf. Systems, Man, Cybernetics, November 14-17 1989, Cambridge MA, pp1051-1054.
[Pietikainen et al, 1982] "Texture Classification using Averages of Local Pattern Matches.", TR-1098, Computer Vision Lab., University of MAryland, 1982.

[Pollen,Ronner,1982] "Spatial computation performed by simple and complex cells in the visual cortex of the cat." Vision Research, 22, pp 101-118.
[Prager, 1980] "Extractin and Labelling Boundary Segments in Natural Scenes."IEEE PAMI,PAMI-2,Jan.1980,pp16-27.
[Press et al, 1990] Numerical Recipes in C, Cambridge University Press, 1990.
[Ramponi,Sicuranza,1989] "Texture Discrimination Via Higher-Order Statistics.", Workshop on Higher-Order Spectral Analysis, Vail, USA, 28-30 June, 1989, pp100-105.
[Rosenfeld, Kak, 1982] Digital Picture Processing, Volumes 1 and 2, New York, Academic Press, 1982.
[Rossol,1981] "Vision and Adaptive Robots in General Motors", Proc. 1st Int. Conf. on Robot Vision and Sensory Controls, Stratford-upon-Avon, UK, April 1981.
[Samet, 1980] "Region Representation: Quadtrees from Boundary Codes.", Communications ACM 23, March 1980, pp163-170.
[Serra, 1982] Image analysis and mathematical morphology. Academic Press, 1982.
[Shunmugam, 1986] "On Assessment of Geometric Errors.", Int. Journal of Production Research, No.2, pp413-425.
[Skinner et al,1990] "Application of Adaptive Convolution Masking to the Automation of Visual Inspection.", IEEE Trans. Robotics and Automation, Vol. 6, No.1, Feb. 1990.
[Sklansky,1978] "Image Segmentation and Feature Extraction.", IEEE Trans. Syst. Man Cybern, 8, 237-247, 1978.
[Spann,Wilson,1985] "A Quad-Tree Approach to Image Segmentation Which Combines Statistical and Spatial Information.", Pattern Recognition Vol.18,No.3/4,pp257-269,1985.
[Stone,1990] "Shape from Texture: Textural Invariance and the Problem of Scale in Perspective Images of Textured Surfaces.", British Machine Vision Conference, Oxford 24-27 Sept. 1990. pp181-186.
[Tan,Constantinides,1989] "Texture Feature Extraction Based on Primitive Analysis.", IEEE, 1989.
[Tanimoto,Pavlidis,1975] "A hierarchical data structure for picture processing." CGIP 4,2, June 1975, pp.104-119.
[Triendl,1972] "Automatic Terrain Mapping by Texture Analysis.", Proc. 8th Int. Symposium on Remote Sensing of Environment, Michigan, 1972.
[Udny et al,1968] An Introduction to the Theory of Statistics, 1968, 11.11-11.27, Hafner Publishing Company, New York.
[Unser, 1983] "A fast texture classifier based on cross entropy minimisation.", Proc EUSIPCO-83, in Signal Processing II: Theories and Applications, H.W. Schussler Ed. Amsterdam:Elsevier, 1983, pp261-264.
[Unser,1986] "Sum and Difference Histograms for Texture Classification.", IEEE PAMI, Vol.PAMI-8,No.1,Jan 1986.
[Unser,Eden,1989] "Multiresoluion Feature Extraction and Selection for Texture Segmentation.", IEEE PAMI, Vol.11, No.7, July 1989.
[Visa,1990] "A texture classifier based on neural network principles." Second International Joint Conference on Neural Networks, 1990.
[Vonnegut, 1970] Slaughterhouse 5, or The Children's Crusade, A Duty-dance with Death. Jonathan Cape, 1970.
[Walsh, 1975] Methods of Optimisation, G.R. Walsh, John Wiley and Sons, 1975.
[Ward et al,1979] "CONSIGHT: An Adaptive Robot with Vision", Robotics Today, pp26-32,Summer 1979.
[Wermser,Liedtke,1982] "Texture Analysis Using a Model of the Visual System." 6th Int. Conf. on Pattern Recognition, Munich October 1982.
[Weszka et al, 1976] "A Comparative Study of Texture Masures for Terrain Classification", IEE Transactions Systems Man Cybernetics, Vol. SMC-6, no.4, pp269-285, April 1976.
[Weszka,Rosenfeld,1975] "A comparative study of texture measures for terrain classification, in Proceedings of the Conference on Computer Graphics, Pattern Recognition and Data Structure, UCLA, May 1975, pp.62-64.

[Weszka, Rosenfeld, 1976] "An Application of Texure Analysis to Materials Inspection.", Pattern Recognition, Vol.8, pp195-199, 1976.

[Willcox, 1994] Automated Manufacture of Dry Carbon Fibre Preforms, an MPhil Dissertation, Dept M&ES, Brunel University, 1994.

[Xiaohan et al,1991] "A New Algorithm for Texture Segmentation Based on Edge Detection.", paper presented on a visit to Brunel University, 1991.

[Zhang, 1995] Adaptive Texture Analysis, PhD thesis, Dept. M&ES, BrunelUniversity, 1995.

[Zhang,Sarhadi,1992] "A neural network approach for the segmentation of textured images." Second Int. Conf. on Automation, Robotics and Computer Vision, 15-18 September, 1992, Singapore. ICARV '92.