Fi ɔer hou

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Surface Scanning with Uncoded Structured Light Sources

Alan Robinson

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

January 2005

Abstract

Structured Light Scanners measure the surface of a target object, producing a set of vertices which can be used to construct a three-dimensional model of the surface. The techniques are particularly appropriate for measuring the smoothly undulating, featureless forms which Stereo Vision methods find difficult, and the structured light pattern explicitly gives a dense graph of connected vertices, thus obviating the need for vertex triangulation prior to surface reconstruction. In addition, the technique provides the measurements almost instantaneously, and so is suitable for scanning moving and non-rigid objects. Because of these advantages there is an imperative to extend the range of scannable surfaces to those including occlusions, which often reduce or prevent successful measurement.

This thesis investigates ways of improving both the accuracy and the range of surface types which can be scanned using structured light techniques, extending current research by examining the role of occlusions and geometric constraints, and introducing novel algorithms to solve the Indexing Problem. The Indexing Problem demands that for every pattern element in the projected image, its counterpart, reflected from the surface of the target object, must be found in the recorded image, and most researchers have declared this problem to be intractable without resorting to coding schemes which uniquely identify each pattern element. The use of *uncoded* projection patterns, where the pattern elements are projected without any unique identification, has two advantages: firstly it provides the densest possible set of measured vertices within a single video timeframe, and secondly it allows the investigation of the fundamental problems without the distraction of dealing with coding schemes. These advantages educe the general strategy adopted in this thesis, of attempting to solve the Indexing Problem using uncoded patterns, and then adding some coding where difficulties still remain.

In order to carry out these investigations it is necessary to precisely measure the system and its outputs, and to achieve this requirement two scanners have been built, a Single Stripe Scanner and a Multiple Stripe Scanner. The Single Stripe Scanner introduces the geometric measurement methods and provides a reference output which matches the industry standard; the Multiple Stripe Scanner then tests the results of the investigations and evaluates the success of the new algorithms and constraints. In addition, some of the investigations are tested theoretically, by using synthetic data and by the solution of geometric diagrams.

These evaluations of success show that, if occlusions are not present in the recorded data, the Indexing Problem can often be completely solved if the new indexing algorithms and geometric constraints are included. Furthermore, while there are some cases where the Indexing Problem cannot be solved without recourse to a coding scheme, the addition of occlusion detection in the algorithms greatly improves the indexing accuracy and therefore the successful measurement of the target surface.

Publications

A. Robinson, L. Alboul and M.A. Rodrigues, *Methods for Indexing Stripes in Uncoded Structured Light Scanning Systems*, Journal of WSCG, 12(3), pp. 371-378, 2004.

M.A. Rodrigues and Alan Robinson, *Image Processing Method and Apparatus, Rotational Position Estimator and 3D Model Builder (Tripods),* UK Patent Application 0402420.4, February 4th, 2004.

M.A. Rodrigues, Alan Robinson and Lyuba Alboul, *Apparatus and Methods for Three Dimensional Scanning, Multiple Stripe Scanning Method*, UK Patent Application No. 0402565.6, February 5th, 2004.

Acknowledgments

The author would like to thank the Geometric Modelling and Pattern Recognition group of

Marcos Rodrigues,
Lyuba Alboul,
Georgios Chliveros,
Jacques Penders,
Gilberto Echevarria,
David Cooper,
Charilaos Alexakis and
Alexander Chamski,

at Sheffield Hallam University for their unstinting help in producing this thesis. Many thanks must also go to Sanaz Jabbari for help in formulating the system geometry.

Dedicated to Mat Tatlow, an inspiration throughout this work.

Contents

List of Figures

List of Tables

Chapter 1: Introduction

Two distinct visual approaches towards the detailed and accurate representation of real objects are computer graphics and computer vision. In this context "vision" refers to the automatic recording of some properties of an object: photography records reflected light, and motion capture records the position of the joints in a moving body. On the other hand "graphics" has no such direct link with the real world; the output, such as a painting or a CAD drawing, is rendered by hand or machine and may be continually modified to approach a representation of reality. Many applications combine the two approaches, and this combination of graphics with some form of automated vision can be seen in Canaletto`s *camera obscura* paintings, Warhol`s screen prints, and in movies such as Disney`s "Snow White" where cine film of actors (i.e. "vision") is retraced and combined with traditional animation drawings ("graphics"). Computer vision requires a considerable hardware investment in video capturing or motion sensing, but the benefits over computer graphics rendering are significant. As [Watt, 2000] points out:

> *"Real world detail, whose richness and complexity eludes even the most elaborate photo-realistic renderers, is easily captured by conventional photographic means".*

This "richness and complexity" is important for both measurement and visualisation uses. In the former category are such applications as reverse engineering, inspection and archiving [Levoy *et al.,* 2000] [GOM, 2004] [Yang *et al.,* 1995], all of which require accuracy in the measurements. In visualisation applications [Marek and Stanek, 2002] the viewer is less critical of exact measurement, but will find a recording of a real object more intuitively lifelike than a computer generated one due to unquantifiable nuances in the captured model.

The entertainment industry is particularly interested in acquiring the three-dimensional model in real time, but at present some form of compromise is made, using both vision and graphics. A standard method of producing an animated face is to map photographic images onto a preformed computer model of a speaking head [Liu *et al.,* 2000]. This gives an impression of true 3D vision, but will miss much of the appealing detail, and is a time-consuming process. Recent research, especially in Structured Light Scanning and Stereo Vision, suggests the possibility of producing a model of a complete surface area such as a human face within the timebase of one video frame [Hall-Holt and Rusinkiewicz, 2001]. It is easy to imagine how much more lifelike the animated character would be with this method, as the automatic real time

process provides synchronisation of face and voice, a conspicuous absence in photo-realistic renderings.

1.1. Methods of Recording 3D Surfaces

The general objectives of this research area can be usefully separated into two processes: firstly to measure the three-dimensional position of points on the surface of a target object, and secondly to order these points and thereby reconstruct a model of the surface as a mesh of connected polygons. The former process, which we call "point acquisition", is geometric, defined by the spatial position of the system in Euclidean space; the latter process, which we call "surface reconstruction", is a graph topology issue whereby the acquired points are connected in a graph of vertices and edges [Bondy and Murty, 1976]. The resulting 3D model has an increasing number of industrial, medical and multimedia applications, which require continual improvement in the speed of acquisition, accuracy of measurement and verity of the reconstructed surface [Valkenburg and McIvor, 1998]. In the field of Machine Vision using video images, two main methods, Stereo Vision and Structured Light Scanning, compete to fulfil these requirements.

Stereo vision, the system used for human sight, simultaneously records two spatially related images and recognizes corresponding features in each scene [Marr and Poggio, 1979]. Knowledge of the geometric relation between the two images allows the three-dimensional position of the features to be computed. The key to this system is the recognition of corresponding features, and this task is known as the **Correspondence Problem** [Zhang, 1996]. If there is a dense map of these features then the target objects can be reconstructed accurately and comprehensively, but if the object is a smoothly undulating form then features are difficult to find and the surface is difficult to measure. A further problem is that the acquired points do not explicitly provide a way of reconstructing the surface; forming the polygonal mesh from the point set is itself the subject of considerable research effort [Curless and Levoy, 1996].

The Structured Light method projects a pattern of light onto the target surface, and a sensor such as a video camera records the deformed light pattern reflected from the surface. Here the correspondence problem is different: rather than finding the same feature in two stereo images we need to find the correspondence between a projected pattern element and its reflection from the target surface onto the sensor. If the spatial relationship between the camera, projector and their lenses is known, then the three-dimensional position of the illuminated

points on the surface can be measured. This method is ideally suited to measuring the featureless forms which Stereo Vision methods find difficult, and the projected pattern provides a definable and controllable density which is not possible using feature recognition. However, if the surface is not smooth, or is highly reflective, the coherence of the projected pattern may be lost in the recorded image, and the surface will be difficult both to measure and to reconstruct. A benefit of the Structured Light system is that the light pattern gives an order to the acquired points, so that if the points are successfully derived then their connection sequence implicitly follows. For example, if we project a grid pattern of dots, the derived points will be connected in the same way as the projected dots, and a polygon mesh can be easily formed.

Therefore we see that the usefulness of these two methods is almost complementary: Stereo Vision is applicable for densely featured surfaces, and Structured Light Scanning for smoother surfaces. Unfortunately it is not necessarily possible to know in advance what type of surface will be encountered, and which method will be preferable. Research is currently attempting to extend the usefulness of each method into the domain of the other. For Stereo Vision this means increasing the thoroughness of the search for common features and thereby increasing the density of the point set; for Structured Light Scanning this means making sense of the recorded image when the light pattern is severely deformed by the shape of the target surface.

1.2. Structured Light Scanning

Early Structured Light systems used a single stripe or spot of laser light to measure a small part of the object in each scan [Rioux and Blais, 1989]. Now the availability of controlled light output from LCD projectors allows the projection of a more complex pattern of light to increase the area measured in a single scan. The classic Single Stripe scanning system provides the profile of a "slice" through the target object. In order to build a model of the complete surface a number of spatially related profiles must be scanned. To achieve this a sequence of scans is captured. For each scan, the target object is moved in relation to the scanner, or the projected stripe moves in relation to the object, the movement being controlled to the same resolution as required by the scanning system. A system may require an accuracy of 1:20000 [Beraldin *et al.*, 2001].

To avoid the need for accurate mechanisms and in order to speed up the acquisition process, a number of stripes (or other pattern elements) can be projected at the same time and

captured as a sequence of stripes in a single frame [Valkenburg and McIvor, 1998]. However, it may be difficult to determine which captured stripe corresponds to which projected stripe, when we attempt to index the captured sequence in the same order as the projected sequence. We name this the **Indexing Problem**, and its solution will form the main subject of this thesis.

1.3. Defining the Principal Research Questions

1.3.1. The Indexing Problem

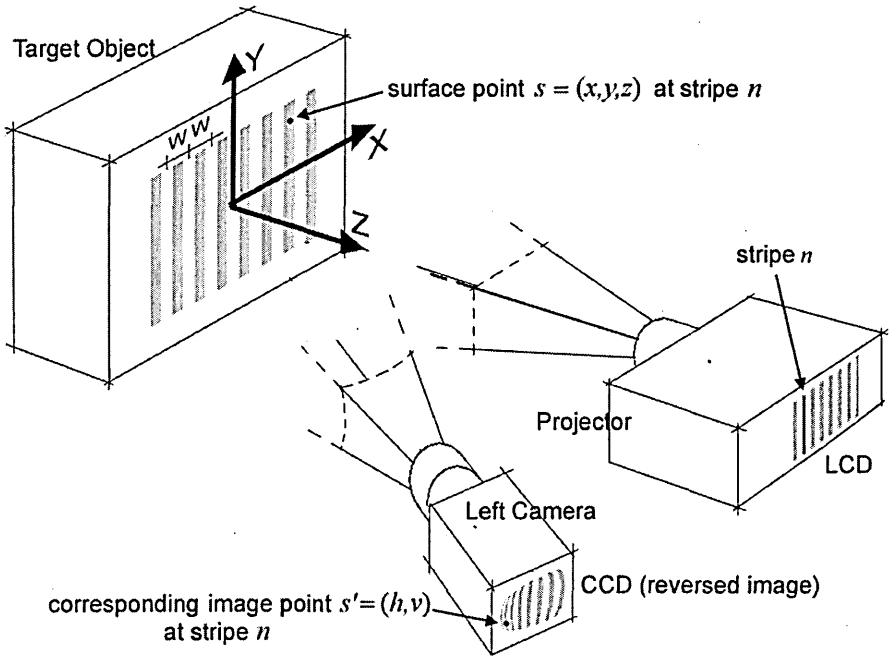

Figure 1.3.1: The structured light scanning system.

Figure 1.3.1 shows the basic Structured Light Scanning system, here using a sequence of vertical, equally spaced stripes, shown grey for convenience. The vertical stripe sequence is indexed, and the stripe with index n is shown as the dark stripe in the projector LCD. A point s = (x,y,z) which is on the surface of the target object is illuminated by stripe n, and is recorded on the camera CCD as point s' on the camera image plane. The position of s = (x,y,z) is given by the scanning function $scan(h,v,n) = (x,y,z)$, where (h,v) is the position of the point s' as seen in the recorded image, and n is the index number of the stripe. Calculating the position of s is

therefore dependent upon correctly identifying the index n in the recorded image, which may be a difficult task.

1.3.2. Coding Schemes

Because of these indexing difficulties, methods have been devised to encode the stripes, by colour, stripe width, or by a combination of both. Coding schemes are discussed in detail in Section 2.4.3, and reviewed by [Salvi *et al.*, 2004]. However, coding does bring disadvantages: with colour indexing there may be weak or ambiguous reflections from surfaces of a particular colour, and with stripe width variations the resolution is less than for a uniform narrow stripe. This last problem can be addressed by projecting and capturing a succession of overlapping patterns of differing width but this means that it is not possible to measure the surface in a single frame. Single frame or "one-shot" capture is desirable because it speeds up the acquisition process, and leads to the possibility of capturing moving surfaces. Moreover, most of the coding schemes use a sequence which repeats, typically, every eight times, in which case ambiguities are still possible. These coding options are illustrated in Figure 1.3.2 where at (1) we see an uncoded pattern, at (2) a 3-colour coding, at (3) a pattern with varying stripe width, and at (4) we see four binary patterns which are successively projected to give a Gray coding scheme.

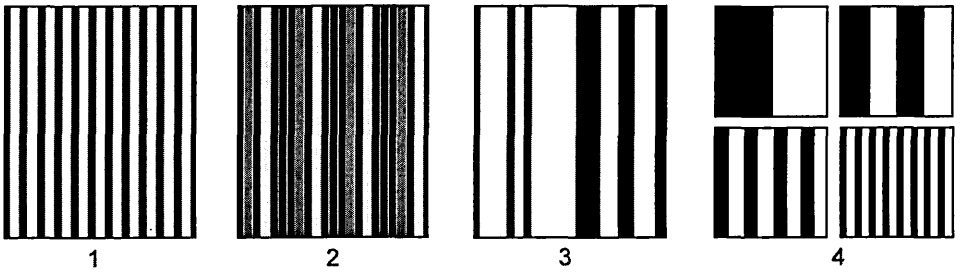

Figure 1.3.2: Stripe coding schemes.

The general strategy of this thesis is to determine how far the Indexing Problem can be solved with uncoded stripes, where the correct order of stripes in the captured image is determined by original algorithmic methods. It may then be appropriate to add a limited amount of coding to solve some of the more intractable issues. A comparison will also be made with dot patterns, which require different designs for the indexing algorithms.

1.3.3. The Principal Research Questions

From the two preceding sections it is now possible to pose the two main questions which will be addressed in this thesis:

1. How can we correctly identify in the sensor a specific projected pattern element, expressed as an index? This is the Indexing Problem.

2. How far can the Indexing Problem be solved using an uncoded projection pattern?

In order to answer these questions a number of investigation topics will be proposed, and we begin by discussing some of the issues which will be encountered in this research.

1.4. Choosing the Projection Pattern

Other researchers have proposed many types of pattern for Structured Light Scanning, such as dots [Proesmans and Van Gool, 1998], grids, horizontal stripes and vertical stripes [Daley and Hassebrook, 1998] [Guhring, 2001]. The patterns can be useful categorized into two types: two-dimensionally discrete patterns such as dots or grids, and one-dimensionally discrete patterns such as horizontal stripes or vertical stripes. The one-dimensionally discrete patterns are continuous in their second dimension, so that vertical stripes are discrete from left to right, but continuous from top to bottom. An advantage of this is that vertical stripes can be sampled vertically at any point - in other words a portion of the vertical stripe will be present on every pixel row. To sample discrete elements such as dots would require compliance with the Nyquist-Shannon Rule [Nyquist, 1928] which states that the sampling frequency must be greater than twice the highest frequency of the input signal. In spatial terms this means that in the recorded image the spacing between dots would have to be greater than two pixels.

Given this clear advantage of stripes, it raises the question of whether there is any reason for using two-dimensionally discrete patterns such as dots. Chapter 6 will show that knowing the horizontal displacement of a stripe in the recorded image is sufficient to calculate the surface

vertex for that point, but that knowing the horizontal and vertical displacement of a dot can provide confirmation of the index for that particular dot.

1.5. Geometric and Topological Considerations

Both the geometry and the graph topology of the 3D model must be considered. The final goal is a geometrically congruent model, i.e. an isometric representation in Euclidean space of the target surface. However, because the scanned model will be discrete whereas the target surface is continuous, a discrete approximation must be made, of a connected graph of vertices (see Figure 1.6.1, right) with each vertex having a precise location in E^3 space. If the system parameters are correctly measured in the calibration process, and the graph of vertices is correctly connected, then the resulting discrete model will be an *ideal approximation.*

If the system parameters are incorrectly measured, then this ideal approximation will be deformed; the vertices will change their location and the distances between them will change; therefore the model will lose its isometry and geometric congruence. This deformed shape will still be topologically equivalent, because the transformation between the two shapes is *homeomorphic*, i.e. bicontinuous and one-to-one.

Apart from by incorrect measurement of the system parameters, another way in which the shape may change is by connecting the graph of vertices in a different way, thus changing the *combinatorial structure* [Huang and Menq, 2002] of the graph. For instance, if a vertex is connected to a different *stripe path* (see Section 7.6), the index will change and the 3D location of that vertex will change (see Section 1.3.1). Finding the correct connection between vertices, i.e. the combinatorial structure, is another way of expressing the task of the Indexing Problem.

In fact it may be useful to consider two separate tasks: the topological one of correctly connecting the vertices, and then the geometric one of finding the E^3 location of each vertex.

1.6. Vertex Connections

In Figure 1.6.1 we see a detail of a recorded stripe pattern (left), the array of peaks at the centre of each stripe for each row (centre), and the derived mesh of 3D vertices representing the target surface (right). The peak array (centre) can be considered as a graph of vertices, as can the derived mesh, and these two graphs are identical in the sense that they have the same number of

vertices, and that the connections between their respective vertices are the same. Therefore once the connections are established in the peak array, it becomes a simple task to connect the vertices in the surface mesh. When thinking about how the Indexing Problem would be solved for this example, two related tasks can be considered: to find the "connected path" for each stripe, and to index the stripes in the correct sequence. Visually it is easy to see that there are three stripes each with a connected path from top to bottom, and that they could be indexed from left to right as stripes 1, 2 and 3.

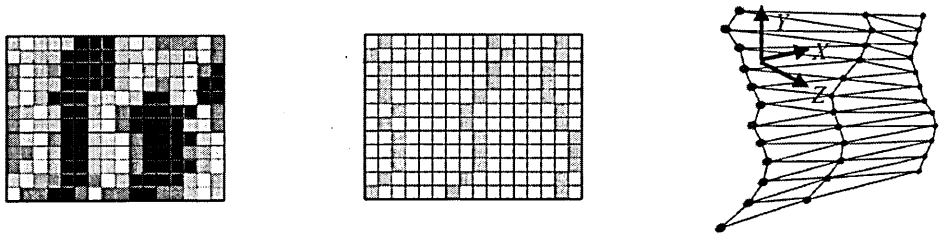

Figure 1.6.1: Visualisation of the stripe pattern.

We will investigate the ways in which the peak connections can be defined and implemented. In defining these connections care is taken to comply with the common terms used in graph theory such as "connectedness", "connectivity", "adjacency" and "paths".

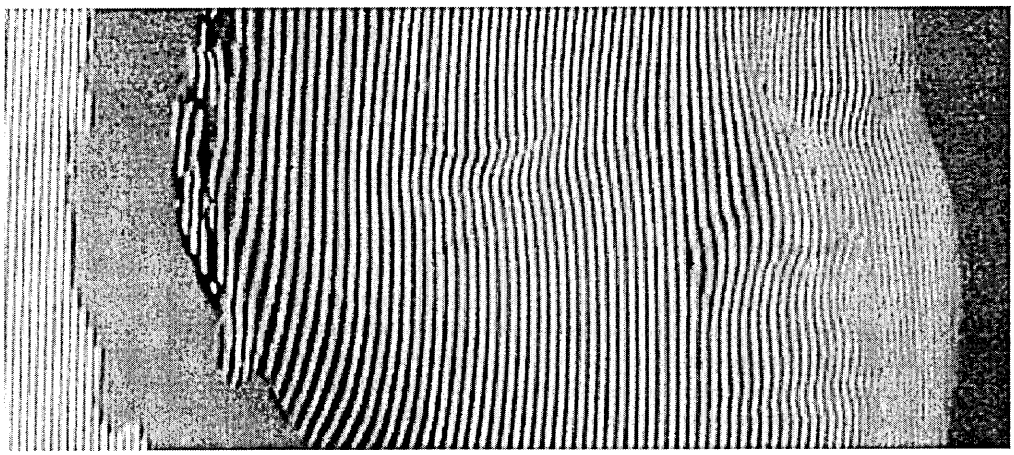

Figure 1.6.2: Recorded image from structured light scanner (part).

Figure 1.6.2 shows part of a typical image recorded by a Structured Light Scanner. The projected pattern of evenly spaced, vertical white stripes appears to be deformed by the shape of the target surface. Looking at the pattern, we can try to interpret or recognize the paths and

sequence of stripes by eye. Intuitively it is possible to trace the route taken by many of the stripes, but there are some areas where either the stripe is missing (around the ear) or there is ambiguity over which direction it should take (to the right of the nose). There may also be situations where we trace an apparently continuous stripe where there is in fact a disconnection (possibly under the nose). In this situation the nose is *occluded* from the camera viewpoint, and the shadow to the left of the ear is occluded from the projector viewpoint.

The investigations into connections will use this intuitive visual approach as a starting point for some of the automatic algorithms, and will also look at methods of overcoming the problems set by occlusions.

1.7. Ambiguous Interpretation

Figure 1.7.1 shows at (a) a hand-drawn, hypothetical image from a scanner. Two attempts to index the stripes (using colour as a guide) are shown at (b) and (c), with a visualisation of their resulting surfaces at (d) and (e), where the coloured stripes are superimposed onto the surfaces for demonstration purposes. In (b) we assume that a complete stripe is present at S.

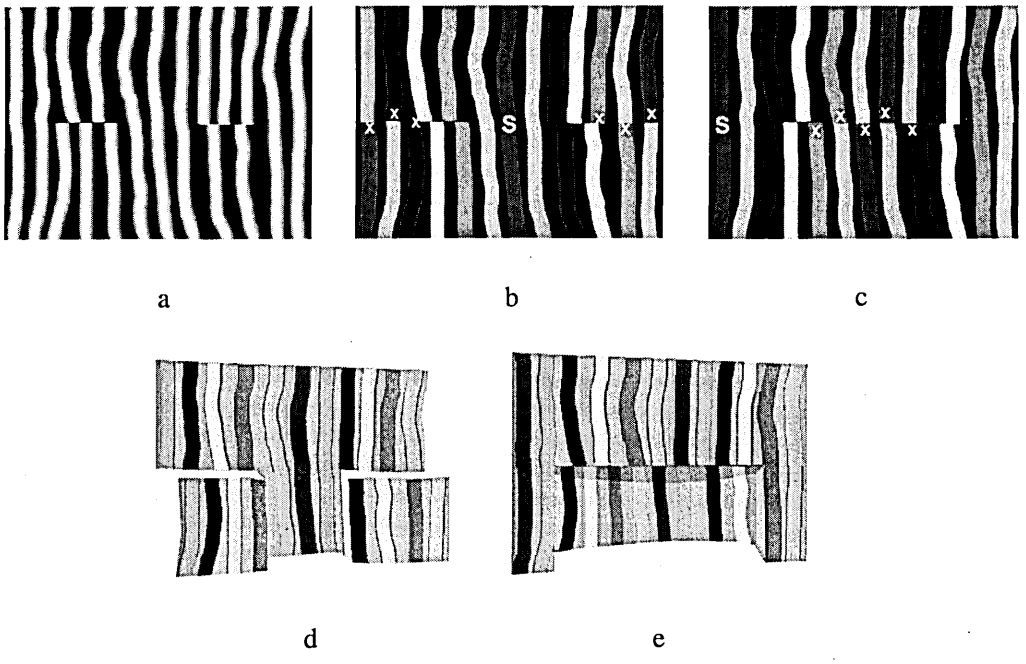

Figure 1.7.1: Synthetic pattern of stripes showing two possible interpretations.

The indexing of the rest of the image follows from there and, in order to make sense of the stripe pattern, two horizontal steps are assumed in the areas marked with "x". A similar process occurs for version (c), this time with the assumed complete stripe S being at the extreme left. Once again a horizontal step is assumed, this time at the centre. The visualisations at (d) and (e) show the horizontal steps which are assumed in order to solve the Indexing Problem. This hypothetical example suggests that there will be cases where apparently "connected" stripe elements are in fact "disconnected", and also that if this area is the full extent of the image it may be impossible to decide which interpretation is correct.

1.8. Definitions and Identification of Occlusions

There are a number of potential reasons why there may appear to be a disconnection, an ambiguity or an incorrect connection in the recorded stripe patterns. One likely source is occlusions [Park *et al.*, 2001], where part of the surface is hidden from either the projector (the shadow on the wall in Figure 1.6.2) or the camera (the hidden side of the nose). It may be possible to improve the success of the indexing algorithms by better understanding the contribution of occlusions. In Figure 1.6.2 we know, because the object is a face, that there is an occlusion at the nose which will then give us clues as to what course the stripes are likely to take. However, an automatic algorithm will not have this prior knowledge, unless it is included within the program; but it may be possible to categorize occlusions and predict their likely affect. One major investigation topic will look closely at how the relationship between projected stripes and their recorded image is affected by the presence of occlusions, and provide definitions and categorization of occlusions which will be useful for this and other work.

1.9. Geometric Constraints

Some questions arise which relate vertex connections with the geometric setup of the system. The recorded image, such as in Figures 1.6.2 and 1.7.1, includes topological and geometric information, both of which are required to calculate the parameters of the scanning function *scan(h,v,n)*. Here "topological" is used in the sense of "graph connectivity in 2D", and this connectivity is used to find the index n for all stripes; accurate geometric measurement of the deformation of the stripes will give the values of h and v. Effectively the topological and

geometric information is combined in the same data set, the video image. It may therefore be possible to use this duality to answer questions in the topological graph by referring to the Euclidean space, and *vice versa.*

For instance, in Figure 1.7.1 (c) the three centre stripes are assumed to be "disconnected", split by an invisible horizontal step; and yet it may be possible to calculate geometrically that any such step would require the stripes above and below the step to be differently spaced. This is shown in Figure 1.9.1 (left) where the spacing between the stripes above the step is w' units and the spacing below the step is w'' units. This would then prove that the assumed topology, of a horizontal step, is impossible if there is no change in spacing.

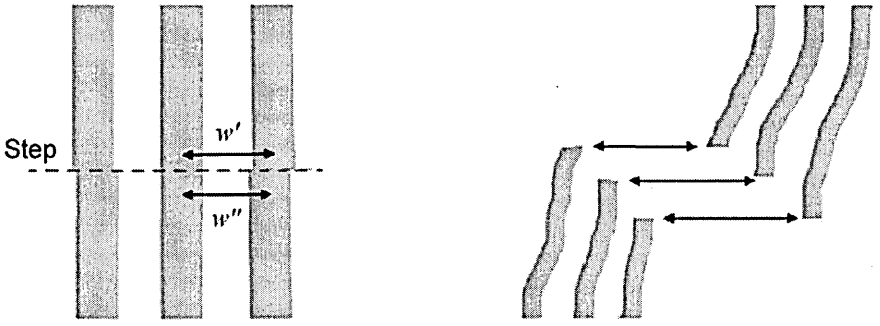

Figure 1.9.1: Horizontal step (left), and lateral translation (right).

Also, we notice in Figure 1.6.2 that the stripes on the ear lobe are continued on the back wall, but much lower down in the image. There is a shadow occlusion between the head and the wall. Can we constrain the system so that when a stripe stops due to an occlusion it will always be picked up again in the same horizontal line? (as in Figure 1.9.1 right). It may be that we can set up the apparatus in such a way that we can answer these questions and thereby improve the reliability of the connection algorithms and other indexing methods.

A concept which is commonly used in Stereo Vision is that of Epipolar Geometry, whereby a spatial constraint is imposed so that corresponding pairs of features must be somewhere along known epipolar lines, thus reducing the Correspondence Problem to a one-dimensional search along that line [Faugeras *et al.,* 1993]. Epipolar Geometry (see Chapter 6, Section 6.3) will be used to investigate the condition described in Figure 1.9.1 (right), and also to look at the comparative benefits of using dot and stripe projection patterns.

1.10. Definitions of Success

While conducting these investigations we will be mindful of how best to measure the success of the outcomes. In some applications, such as industrial testing, it may be preferable to have a smaller surface area with a high degree of confidence in the accuracy of the model; in other areas, such as computer animation, it may be acceptable to have a number of errors but with as large a patch of surface as possible. A further problem will be how to test for accuracy. It will be necessary to have some kind of template object, with independent knowledge of its dimensions, to test against; but if we are to test a complex surface, as we must, how will we independently measure that surface?

In some cases, such as the hypothetical pattern in Figure 1.7.1, it may be impossible by our primary methods to determine which surface representation is true, or which is more likely. At this point it will be necessary to ask whether our initial strategy, of using an uncoded stripe pattern, is feasible. Further strategies may be considered, such as a categorization of the surface by curvature or Fourier analysis, implementation of a limited colour coding scheme, or assuming that the system is scanning a specific shape, say a face or a production line item, which would have the advantage of including prior knowledge in the algorithms.

Definitions and descriptions of the methods used in this thesis will be given in Chapter 5.

1.11. Summary of Investigation Topics

From the above discussion five primary investigation topics are educed:

Inv 1. **The use of geometric constraints, and in particular Epipolar Geometry, to alleviate the Indexing Problem.**

Inv 2. **A comparison between the benefits of two dimensionally discrete patterns such as dots, and patterns such as stripes which are discrete in one dimension and continuous in the other.**

Inv 3. Algorithms for recognizing and indexing the pattern elements, and defining the boundaries of the target surface.

Inv 4. Understanding the relevance of occlusions to the Indexing Problem.

Inv 5. Testing the Success of the Indexing Methods.

The research methods and strategy employed for these investigations are discussed in Chapter 3.

1.12. Original Contributions

This thesis makes original contributions in regard to geometric constraints, connectivity, measures of success, and occlusions. The contributions made are described at the start of each relevant chapter, and summarised in the Conclusions.

We take the popular methods using Epipolar Geometry and Rectification and adapt them to investigate the use of Dot and Stripe Patterns in Structured Light Scanning. In particular we introduce the ***Unique Horizontal Constraint*** and determine how far this can be successfully used to solve the Stripe Indexing Problem. As far as is known, this method has not been used by previous researchers.

Issues related to the connectivity of the vertex graph include definitions of ***N-, S-, E- and W-Neighbours***, leading to definitions of ***Stripe Paths*** and ***Neighbourhoods***. These types form the basis of descriptions of ***Patches***, and are used as ***Patching Conditions*** in the novel algorithms which have been designed to successfully index the stripes. These novel indexing algorithms comprise ***Scanline***, ***Scan and Search*** and ***Flood Fill*** methods, which adapt and extend existing search methods used in Pattern Recognition and Segmentation applications. Here we also pose the ***Size v. Accuracy Question***: "For a given scanning application, what is the relative importance of indexed patch size compared with indexing accuracy", and this question must be addressed by the application designer.

In order to measure the success of these algorithms, all the indexing methods are executed using a *test object*, which is compared to a hand-produced *template* result. Three novel measures: ***total patch size***, ***accurate patch size*** and ***indexing accuracy*** are used to evaluate the new methods. These measures overcome the difficulty of assessing the success of scanning a surface with occlusions.

To overcome the problems caused by occlusions, we make specific definitions which relate occluded areas on the target surface to the resulting recorded image: ***Near Occlusion Points***, ***Far Occlusion Points***, ***Projector Occlusion Points*** and ***Camera Occlusion Points***. From these definitions four ***boundary types*** are categorized: the ***Concave Pair***, the ***Contour Single***, the ***Concave Tuple*** and the ***Contour Tuple***. The investigation then looks specifically at how Concave Pairs can be identified in the recorded image, and thence designs a novel algorithm to ***detect and mark the occlusion boundary***. This is added to the Patching Conditions, giving improved levels of indexing accuracy. These investigations relate to occlusions which are not horizontal. In addition, a theoretical instance has been designed which shows that it is possible to have ***exactly the same spacing between stripes, above and below a horizontal occlusion (or step)***. This means that spacing between stripes cannot be readily used as an indicator of the presence of a horizontal occlusion.

1.13. Overview of the Thesis

The objective of the thesis is to investigate the two principal research questions:

- how can we correctly identify in the sensor a specific projected pattern element, expressed as an index (this is the Indexing Problem), and

- how far can the Indexing Problem be solved using an uncoded projection pattern?

In Chapter 2 we look at prior work in the field of surface measurement, particularly those areas which use triangulation rangefinding. This covers techniques such as laser scanning and coded structured light scanning. Particular problem areas are reviewed, such as occlusion handling, connection algorithms and calibration methods, and work in the associated discipline of Pattern Recognition is described. Applications such as heritage replication, reverse engineering, medical uses and multimedia productions are listed.

The strategy, methods and investigations undertaken in this thesis are described in Chapter 3. A brief outline is given of the five investigation topics: geometric constraints, the comparison between dots and stripes, the connection algorithms, occlusions, and testing methods. The strategy of black box design *versus* white box design of the algorithms is

discussed, as are the merits of design by imitation of human methods. In addition, the research process is shown as a flow diagram incorporating the investigation topics.

Chapter 4 gives a detailed account of the Single Stripe Scanner which has been built as part of the investigations. Many of its processes are used and adapted by the Multiple Stripe Scanner, and the results obtained provide a useful comparison and measure of the success of the whole project. Partly using the data types and methods from the Single Stripe Scanner, Chapter 5 describes in detail the data structures which are used in the thesis, the solving of the scanning functions, and the problems associated with data acquisition and processing. The techniques used for system calibration are also described.

The three following chapters concentrate on the main investigation topics: geometric constraints, indexing algorithms, and occlusion issues. In Chapter 6 the principles of Epipolar Geometry are explained, how they are implemented in relation to Stereo Vision, and methods for adapting the techniques in Structured Light Scanners. There follows a comparison between the benefits of using dot and stripe patterns using epipolar constraints. In Chapter 7 definitions of connections between stripe elements are given, some of which are analogous to work in Pattern Recognition. Initial algorithms are designed to search through the graphs and find correct indices, resulting in patches of indexed surface. A particular problem is identified: that of incorrect indexing due to occlusions. Chapter 8 gives definitions of camera and projector occlusions, and the ways in which occluded areas are formed. From these definitions, methods are designed to mark occluded areas in the recorded image, and thence the indexing algorithms are revised.

Chapter 9 tests the processes, and gives comparative results of the success achieved with different algorithms, which are modified in ways suggested by the investigations. In Chapter 10 conclusions are drawn from the investigations, and suggestions are made for further work based on the results and conclusions.

Chapter 2: Prior Work on Surface Measurement

Surface measurement can be broadly split into two categories: contact and non-contact methods. Here we are concerned with non-contact techniques which are generally known as "rangefinding" [Jarvis, 1983][Karara, 1989][Blais, 2004], and within that domain we make a further separation between image-based methods, interferometry, time-of-flight, and triangulation methods.

2.1.1. Image-based methods

Image-based methods take a collection of recorded images of the target object and by various processes calculate a 3D rendition of the target object. A technique of [Wood *et al.*, 2000] and [Wilburn *et al.*, 2002] uses "surface light fields" which invert the usual ray-tracing methods to produce a rendition of the specular surface of the object.

It is also possible to estimate spatial position from the optical flow [Vedula *et al.*, for publication 2005] [Verri and Trucco, 1998] of a target object through a sequence of images [Zhang and Faugeras, 1992], [Koch *et al.*, 1998]. The optical flow of the scene can be used to separate the target object from the background [Iketani *et al.*, 1998] [Ngai *et al.*, 1999] [Bregler *et al.*, 2000], to recognize hand gestures [Jeong *et al.*, 2002], and facial motion [Reveret and Essa, 2001]. The supersampling which the recording of a moving object provides also allows for the development of super-resolved surface reconstruction [Cheeseman *et al.*, 1994] [Cheeseman *et al.*, 1996].

2.1.2. Interferometry and other wave analysis techniques

Interferometry measures differences between two phase-separated beams of light, as devised in the classic Michelson-Morley experiment [Michelson and Morley, 1887]; moiré and fringe projection can be used to measure very small surface depths [GOM, 2004][Maruyama and Abe, 1993]. [Clarke *et al.*, 1997] have used the polarisation of laser light to estimate depth.

2.1.3. Time-of-flight systems

Time-of-flight systems transmit a signal onto the target surface, and a sensor positioned close to the transmitter records the time taken between transmission and sensing. Because of the short times involved, time-of-flight systems generally work over a large distance, although techniques such as single photon counting [Pellegrini *et al.*, 2000] [Wallace *et al.*, 2002] can bring the distances down to less than one metre.

2.2. Triangulation Rangefinding

The term "triangulation" as used here relates to the concept of trigonometric measurement, where two angles and two vertices of the triangle are known, and thence the position of the third vertex can be calculated. We will look at two topics which comprise the bulk of the research activity, Stereo Vision and Structured Light Scanning. In addition, calculation of depth using focus as a parameter [Subbarao and Liu, 1998] is a well researched area, especially in low-resolution applications such as robotics [Nayar *et al.*, 1995].

2.3. Stereo Vision Techniques

As described in Section 6.3, the Stereo Vision process records two spatially related images of the same scene, finds corresponding features in the two scenes, and thence calculates the position of that feature in the system space. Where the same techniques are used for both Stereo Vision and Structured Light methods, they will be described below, but early work in this field was done by [Koenderink and van Doorn, 1976] and [Marr and Poggio, 1979]. Research has also investigated the use of multiple cameras [Chen and Medioni, 1992] [Faugeras *et al.*, 2002], and specifically three-camera systems [Blake *et al.*, 1995].

The finding of corresponding features in the two scenes, known as the Correspondence Problem, is considered to be a difficult task [Zhang, 1996], and the similar problem for Structured Light Scanners we call the Indexing Problem.

2.4. Structured Light Scanners

2.4.1. Single Stripe Scanners

The earliest successful light projection rangefinders used laser beams, usually in a single stripe of light. Lasers provide a focusable, high level of illumination in a small package; in comparison slide projectors or other white light systems are cumbersome and difficult to control. The stripe of light is cast onto the target object and the recorded image is used to calculate the depth of the profile where the light stripe intersects the target surface. To produce a sequence of profiles which can be used to model a complete surface, either the stripe has to be moved in relation to the target object, or the target object moved in relation to the projected stripe. Either way, this must be done in a spatially controlled manner.

At the National Research Council of Canada (NRC) [Rioux and Blais, 1989] modified a standard camera lens by inserting a mask between two lens elements which produced a double image on the CCD sensor. The laser was projected *via* a rotating mirror controlled by a high precision galvanometer. An RMS deviation in the range axis was 40μm for a 300 mm volume was reported. [Beraldin *et al.*, 1992] developed a method to calibrate the synchronised scanner and compare experimental results with theoretical predictions. The calibration method used a planar wedge with fiducial markings to provide data for a look-up table array, which would then map to collected data for the target object. Estimates were then made of the range errors, and it was found that the limiting factor was the laser speckle impinging on the CCD, which noise far outweighed that contributed by electronic and quantization noise.

The NRC scanners produced a high accuracy, reported to be in the order of 40μm; however, they required precisely engineered moving parts to scan through the range plane. Many alternatives have been developed which require less complex mechanisms.

The work described above by the National Research Council of Canada led to an important and prestigious direction for projected light scanners: heritage applications. The Digital Michaelangelo Project [Levoy *et al.*, 2000] used single stripe scanners to produce surface models of many of Michaelangelo`s sculptures, including the 5m tall statue of *David*, to an accuracy of 1mm. From this model a copy of the sculpture can then be made [Beraldin *et al.*, 2001] using computer controlled milling machines, to replace the original statue in difficult environments (susceptibility to pollution or vandalism, for instance). In the Digital Michaelangelo Project a colour image was recorded alongside each scan, so that the modelled surface could be accurately coloured.

The scanning of sculptures, pottery and other artefacts is a classic use of the single stripe scanner, where the target object is rigid and often has a diffuse surface. Single stripe scanning technology has now produced a range of stable, popular products for these rigid surface applications. Research has moved to structured light scanning, which captures more than one profile, by projecting a more complex pattern of light.

2.4.2. Solutions using fixed camera/source systems

At the Machine Vision Unit at the University of Edinburgh, [Trucco and Fisher, 1994] used a laser stripe projection in a fixed relationship with two CCD cameras, the target object being moved laterally using a microstepper motor to scan along the X-axis. Using two cameras reduced spurious results caused by specular reflection and occlusions. A look-up table would be created using a stepped ramp as a calibration object, which would then map to the target object data. Standard deviation results using the calibration object were approximately 0.1 mm, depending on the specularity of the surface. The advantage of using a calibration matrix is that non-linearity, especially that caused by the spherical geometry inherent in the system, does not have to be overcome using mathematical calculations. However, calibration can be a time consuming process which is not suitable where more immediate results are required.

[McCallum *et al.*, 1996] developed a hand-held scanner, also using a fixed relationship between the laser and the camera to project and receive the light stripe. In this case the stripe moves freely across the object as the apparatus is manually operated. The position of the apparatus in world co-ordinates is given by an electromagnetic spatial locator. Inaccuracies in the spatial locator reduce the resolution of this system to 1.5 mm at a range of 300 mm, and are a major drawback to its widespread use. This scanner has been built as a commercial product [McCallum *et al,* 1998].

When multiple stripes or other sequenced patterns are used, to increase the area scanned in a single video frame, the Indexing Problem occurs, and the following research has been conducted to solve the problem.

2.4.3. Coding schemes

As described in Chapter 1, Structured Light Scanners [DePiero and Trivedi, 1996] project a pattern of light, such as dots, grids or stripes, in order to increase the area of surface captured in one recorded image. To solve the Indexing Problem, akin to the Correspondence Problem in the Stereo Vision field, many codification strategies have been proposed and implemented, and are reviewed by [Salvi *et al.,* 2002]. These include binary coding, n-ary coding, De Bruijn sequences and colour coding.

Binary coding schemes [Daley and Hassebrook, 1998] [Valkenberg and McIvor, 1998], project a pattern of different width stripes so that the pattern can be successfully indexed. Because the stripes have different widths, the highest possible resolution will not be achieved at the widest stripes, and therefore further patterns are projected which overlap each other and result in a successfully indexed pattern with a uniform resolution. The disadvantage of this is that a number of frames, fourteen for the Daley-Hassebrook method, nine for the Valkenberg-McIvor method, are required for each complete scan. Valkenberg and McIvor report the importance of using substripe estimation and incorporating lens distortion into the projector model.

[Rocchini *et al.,* 2001] extended their binary coding by incorporating a colour scheme of red, green and blue stripes, although this still requires nine frames to complete the scan. [Hall-Holt and Rusinkiewicz, 2001] have proposed and implemented a coding scheme using coloured stripes which produces a surface model from one single recorded image. The choice of pattern has been optimized for moving scenes, and the results produce scans without the need to calibrate object motion. However, the target object must move fairly slowly in order to avoid erroneous temporal correlations. [Zhang *et al.,* 2002] also project a pattern of coloured stripes, and thence detect an edge with a triple colour value (r,g,b). The correct indexing is sought using a multiple hypothesis technique, which admits the possibility of more than one solution. A score is then given for each possible solution, based on a match between the colour triple in both the projector scanline and the camera scanline. Importantly, this assumes that the system is constrained such that the displacement of each point must be along a scanline, according to the constraint suggested by [Faugeras, 1993]. Results show that this method is able to correctly model disconnected components, where no surface connectivity exists, although artefacts still occur due to false edges at the boundaries. Another one-shot colour coding scheme has been developed by [Sinlapeecheewa and Takamasu, 2002] using six colours in an eleven stripe sequence, giving a unique 66-stripe pattern. The conclusions give only general indications of the

success of the method. [Sa *et al.*, 2002] use the boundary edges of a colour coding scheme, but here again the results are given only visually, and state the future objective of processing slowly moving objects.

A method employing two successive patterns of uncoded stripes, one vertical and one horizontal, has been proposed by [Winkelbach and Wahl, 2002]. Here the spacing between stripes is used, along with the comparison between the horizontal and vertical patterns, to create, at each node of the combined grid pattern, a surface normal. These normals are then used to create the surface model. Results with standard commercial cameras and projectors report a standard deviation of the surface normals of 1.22 degrees.

2.4.4. Combining Stereo Vision with Structured Light

In order to model more complex 3D scenes, with a number of unconnected objects, [Scharstein and Szeliski, 2003] have combined the techniques of stereo vision and binary stripe projection. Using two cameras and one projector, they report disparities of between 10 and 20 percent on these difficult scenes. [Devernay *et al.*, 2002] have produced a one-shot scanner using a camera and projector; a white noise pattern is projected, and a single video image captured. The correspondence problem is solved using standard correlation-based stereo algorithms. Reported limitations in this system are dealing with occlusions, highly-textured surfaces, and surfaces which are almost normal to the camera view.

2.4.5. Estimating the centre of the stripe in the recorded image

A comparison of eight algorithms to find the centre of the light stripe has been made by [Fisher and Naidu,1996]. It is assumed that the light incident upon the target object follows a Gaussian distribution, although it is noted that the sensing system of the CCD will affect the fidelity with which this distribution is recorded. Of particular interest are the "Centre of Mass" algorithms, which calculate the arithmetic mean of the brightest pixel and the two (COM3), four (COM5) and six (COM7) adjacent pixels. The COM3 algorithm performs poorly, but the other seven produce empirical results in the same range. [Curless and Levoy, 1995] devised a spacetime method which analysed successive adjacent scans in order to correct reflectance, shape and laser speckle distortions. As the algorithms processed the data after it had been collected, no revisions

to the existing hardware were required. This subject area of comparing and interpreting data in adjacent frames is an important topic for research. An alternative to using the Gaussian profile is to use the transitions between stripes [McIvor, 1997] which may be a more reliable method for multiple stripe systems. [McIvor and Valkenburg, 1997] proposed three substripe estimation methods: polynomial fitting using stripe indices, boundary interpolation using an edge detector, and sinusoidal fitting with six harmonics. Tests were executed using a planar surface, which gave the best results (the lowest standard deviation) to the sinusoidal fitting method.

2.4.6. Laser speckle phenomena

Laser speckle can be considered as either a source of unwanted noise or as a useful measuring entity. Laser speckle is caused by coherent light of a specific wavelength continually hitting the same microscopic particle (and a discrete number of its neighbours) and producing an interference pattern along the line of sight. With incoherent light the wavelengths are constantly changing so that an averaging effect is produced at the sensor, which sees a comparatively uniform reflectance. Laser speckle reduces the predictability of the beam profile and limits the accuracy of range measurements [Beraldin *et al.*, 1992]. The light stripe data can be operated on by filtering convolutions [Becker, 1995] which blur the stripe in a Gaussian form, although the wider stripe can cause problems for multi-stripe applications. [Hilton, 1997] has used laser speckle in a constructive way to measure the roughness of a surface.

A series of experiments conducted by [Clarke, 1994] pointed a light beam along the axis of a CCD camera, and placed a sheet of white paper at measured intervals between the light source and the camera lens. Results using laser light and white light sources showed maximum errors for white light of 0.199 pixels, and for a series of lasers of between 0.389 and 1.292 pixels. Clarke concludes that the coherence of lasers has a highly significant effect in degrading the location of the laser target.

2.5. Calibration

Calibration of a triangulation rangefinder requires the determining of two sets of parameters: intrinsic and extrinsic. Early work on the geometric measurements was undertaken by [Brown, 1971]. The intrinsic parameters are those which determine how the light pattern is projected

through the projector lens, and how the incoming light is sensed through the camera lens and onto the sensor. The extrinsic parameters are those which set the spatial position of the projector, cameras and any other elements of the apparatus. Both automatic and manual methods have been devised, some using calibration charts or objects, others deriving the parameters from the target object themselves. [Trucco and Fisher, 1994] calibrated a single stripe scanner by sliding a ramp of known dimensions across the stripe and thence deriving a look-up table of depth values for all parts of the recorded image. This provides both intrinsic and extrinsic parameters, but requires considerable preparation before each scanning session. [Shen and Menq, 2002] used a calibration plane as a target object, and projected a grid pattern onto the plane which was positioned precisely with respect to the scanner. System parameters were then derived from the resulting image transformations. [Chen and Li, 2003] proposed a self-calibration of six degrees of freedom of the scanning system, measured *via* a projected light pattern. Reported results are very good, but this method does not determine the absolute values of parameters. Researchers in stereo vision [Dornaika and Chung, 2001] have been able to provide automatic calibration based on the epipolar geometry of the two-camera system.

The Tsai camera model [Tsai, 1987] provides a widely used equation for modelling the distortion applied to the projected image by the lens of the camera (or projector), and [Fryer *et al.,* 1994] have provided a model for simple C-mount lenses.

2.6. Occlusion Handling

Little work has been done in the structured light scanning domain to deal with occlusions, other than by coding methods. An exception is [Park *et al.,* 2001] who use a two-projector system to reduce the occlusion problems, and give a brief description of occlusion categories. [Castellani *et al.,* 2002] have proposed methods to detect and deal with occlusions when one object, in this case a primitive shape such as a sphere or box, is hidden from the camera view. The visible images are segmented, boundaries are delineated, and then continued into the occluded region in order to reconstruct the hidden surfaces. This work is at an early stage, and concludes that reconstruction can be hard to resolve and incorrect constructions may arise. [Senior *et al.,* 2001] have devised "appearance models" to visually track objects through occlusions in a video sequence, and they report accurate results for the sequences on which they have "concentrated their efforts". [Dell'Acqua and Fisher, 2002] have successfully reconstructed simple geometric scenes which contain occluded elements.

Stereo Vision research deals with occlusions as a matter of course, as mismatches in the feature correspondences are caused by an occlusion between the two viewpoints. Some work has actively used discontinuities and occlusions to improve the resulting surface model. [Ishikawa and Geiger, 1998] have modelled occlusions, discontinuities and epipolar-line interactions, in order to compute the disparity map (of mismatched features). Of particular interest are the "monotonicity constraint" which requires neighbouring points to match, and definitions of "joints" and "edges" on the proposed occlusion boundary. These definitions contribute towards the success of a maximum-flow algorithm to globally optimize the problem of solving the disparity map. [Lei and Hendricks, 2001] have made simple definitions of a "left occlusion", "middle occlusion" and "right occlusion" each of which points to a different algorithm for interpolating the stereo view and realigning the pose of a speaker in a telepresence system. This shows that conditions based on occlusion definitions can be used to synthesise subjectively satisfactory results.

Occlusions are related to "apparent contours" - the silhouette of an object when viewed from a particular point. [Cipolla and Blake, 1992] used apparent contours to derive 3D shape, and a general investigation of this issue has been undertaken by [Koenderink, 1984].

2.7. Pattern Recognition

The machine vision community has developed many methods for identifying shapes in bitmapped video images, typically by using Image Segmentation algorithms [Horowitz and Pavlidis, 1976] [Huang and Menq, 2001]. The design of these methods is often heuristic, but researchers have also used neural network models to preprocess the images and to apply feature extraction. An overview of this work is given by [Egmont-Petersen *et al.*, 2001].

In general, two common techniques are used: edge detection and region segmentation. Scanline edge detection [Jiang and Bunke, 1998] firstly scans each row, column and diagonal in the image to obtain a set of quadratic curves with a midpoint and two endpoints. The endpoints are then classified and given "strength" criteria in order to give an optimal set of edges. The paper reports a much higher correct segmentation and speed compared with five region segmentation programs, in particular the classic work by [Besl, 1988]. A comparison between edge- and region- based segmentation states that edge detection tends to produce nonconnected boundaries, whereas region segmentation requires more complex control structures, and there is a tendency to produce oversegmentation.

2.8. Connection methods

Connection methods use Graph Theory [Bondy and Murty, 1976], and are useful for both edge and region detection. The methods are usually recursive, moving through all possible positions until a stopping condition occurs, such as that we are no longer on the edge, or no longer in the region. The well known "flood-fill" algorithm [Hill, 1990] is used in painting programs, where a region is filled with colour, using a recursive process which stops either at a boundary or at a previously coloured pixel. [Pavlidis, 1982] has developed a series of connection definitions for pixels in a two-dimensional bitmap array. From a particular pixel the adjacent pixels are described as "direct neighbours" if they share a side, and "indirect neighbours" if they share only a corner. These definitions are then used as conditions to control the tracing of a connected line through the bitmap image. [Braga-Neto and Goutsias, 2003] have developed measures of "multiscale connectivity", whereby the measure varies depending on the scale of the object, and use these and other definitions to segment greyscale images. This work is theoretical and does not provide experimental results.

2.9. Using Geometric Constraints

[Hu and Stockman, 1989] used a grid projection, and applied constraints to algorithms in order to extract the scene. The constraints included definitions of the neighbours of grid intersections to determine intersection points, and the epipolar constraint to then project the ray into the projector plane. This combination of topological and geometric constraints was further developed by [Proesmans and Van Gool, 1997], who used the projected grid and defined North, East, South and West neighbours at each intersection, and designed an algorithm to "snake" along the gridlines. Unfortunately, no conclusive test results are reported for either method. [Fofi *et al.,* 2000] have devised a number of geometric constraints relating to a colour-coded grid projection, in order to self-calibrate the system. Firstly, the "parallelogram constraint" states that in the system a projected square results in a parallelogram on the sensor plane. Secondly, an orthogonal constraint determines the depth of the line at the intersection of two orthogonal planes. These constraints are applied to determine the system geometry when a planar surface is scanned.

2.10. Surface Reconstruction and Registration

Surface reconstruction is straightforward for Structured Light Scanners as the light pattern gives an implicit order to the outputted point cloud. However, this order is not necessarily the optimal method of reconstructing the surface, and [Huang and Menq, 2002] have described methods of reconstruction using optimization methods. [Alboul *et al.,* 2004] have used a metric of "Total Absolute Curvature" to optimize surface reconstruction. An interesting development is that [Segall *et al.,* 2001] and [Baker and Kanade, 2001] have used Bayesian methods to reconstruct surfaces at higher sampling rates than in the original recording.

If more than one scan is made of a particular rigid object, it will be necessary to "join" the two surface models together. This requires "registration" whereby the relative orientation of the two coordinate systems is found so that the two patches can be placed in the same coordinate system, and "fusion", whereby the two sets of vertices are combined in the same reconstructed surface. [Besl and McKay, 1992], [Chen and Medioni, 1992] and [Liu and Rodrigues, 2002] have devised methods based on the classic Iterative Closest Point (ICP) algorithm to find the same (or a very close) surface location in each patch. [Rodrigues *et al.,* 2004] have further developed this concept by reducing the point clouds to a set of vertex "tripods", thus reducing the level of iteration required.

2.11. Conclusions

This review of other work related to 3D surface measurement shows that topics such as Stereo Vision and Coded Structured Light methods have been the subject of considerable investigation, whereas only a few recent works have looked at the possibility of using Uncoded Structured Light techniques. Techniques using Epipolar Geometry and Rectification have proved successful for Stereo Vision methods, and there is evidence that the same techniques may be usable in Structured Light Scanning. There is also a considerable body of literature in the Pattern Recognition and Feature Segmentation domain, which will be useful in solving the Indexing Problem. Some researchers give measured results which will be a useful gauge of the success of the investigations.

Chapter 3: Research Methods and Strategy

3.1. Principal Research Questions

The principal questions addressed by this thesis, as outlined in Chapter 1, are:

Q1. How can we correctly identify in the sensor a specific projected pattern element, expressed as an index? This is the Indexing Problem.

In other words, every dot or stripe in the projected pattern must have a unique index, and every dot or stripe recognized in the recorded image must be associated with its correct index number.

Q2. How far can the Indexing Problem be solved without using coding schemes?

This thesis employs the strategy of solving the Indexing Problem as far as possible without using coding schemes such as colour or stripe width, and then identifying cases where some form of coding is necessary. The design of indexing algorithms and occlusion detection methods is taken further than other researchers who make an early assumption that the Indexing Problem is intractable without coding schemes [Hall-Holt and Rusinkiewicz, 2001] [Rocchini *et al.*, 2001].

3.2. Investigation Topics

In order to address these two questions the following topics will be investigated:

1. The use of geometric constraints, and in particular Epipolar Geometry, to alleviate the Indexing Problem.

2. A comparison between the benefits of two dimensionally discrete patterns such as dots, and patterns such as stripes which are discrete in one dimension and continuous in the other.

3. Algorithms for recognizing and indexing the pattern elements, and defining the boundaries of the target surface.

4. Understanding the relevance of occlusions to the Indexing Problem.

5. Testing the Success of the Indexing Methods.

3.3. Investigation Strategy

In order to evaluate the success of these investigations two routes are taken: the "black box" approach of evaluating a method based on its test output without necessarily understanding why one method produces better outputs than another, and the "white box" approach of finding logical reasons for a decision and building that logic into the design of the method.

For the black box approach a change is made, for whatever reason, in the process design, and this new process is then run in the testing program. The new test output will be noted, but the disadvantage of this black box approach is that we may not know the reason why a particular design change, say a newly devised stopping condition, has resulted in a change in the test results. It lends itself to a heuristic trial-and-error strategy using intuitive ideas about what might be the best answer to a particular question. However there are times when the logical relationships required by the white box approach are so difficult or complex to determine, that this heuristic, intuitive approach is advisable.

The white box approach looks for a logical reason for leading the investigation down a particular path. For example, the study of Epipolar Constraints gives a method of "horizontal searching" for features in the recorded image which follows directly from the geometry of the epipolar method. Hence a relationship is established between the spatial geometry of the system and the topology of the recorded patterns, and this is typical of the kind of logical relationship which the white box method is trying to find.

A noticeable difference between the white box and black box methods is that the former tends to study the particular subject more widely in order to establish fundamental principles,

whereas the latter tends to home-in very quickly on a particular detail. Some instances of the wider studies are the general definitions which are made of occlusions, most of which are not used in this work; but it is hoped that these will be useful for future research by ourselves and others.

3.4. Imitation of Human Methods

Another important strategy used in pattern recognition algorithms is to note the way a human attempts to tackle the problem, and then replicate that method in an automatic algorithm in the computer. For instance, we can trace the path of a seemingly connected stripe by eye, mark that stripe with a certain colour, and then continue to identify further stripes. This is a laborious but possible method by hand, and is an instructive exercise for forming some initial ideas about how and why the algorithms will work. The advantage of incorporating these human methods into a computer program is that the results can be tested in number-crunching analyses which are beyond the brainpower of humankind.

The key issue which arises in interpreting human methods is that of *prior knowledge*. The test object used throughout the investigations is a sculpture of a head, which has well-known features such as the mouth, eyes and nose. A human will be tempted to use prior knowledge of the shape to influence their decisions when attempting to index the data; they may for instance assume that a stripe cannot cross an occlusion at the nose. It is important when replicating human methods, in the computer, that all prior knowledge influences are discounted in the designs.

3.5. Testing Methods

For both white box and black box approaches there will need to be some way of evaluating the outputs. Simple measurable shapes such as planes, cubes and spheres can be used to evaluate the linearity and accuracy of the system and this is particularly useful for calibration, i.e. finding and checking the intrinsic and extrinsic parameters.

However, these simple shapes are unlikely to thoroughly test the indexing algorithms, which are a key part of this thesis. A surface is required which has camera and projector occlusions, and this in turn can be problematic to measure. The method used to test the indexing

algorithms employs a "template" object which is processed by hand so that, with prior knowledge, an accurate set of indices is created. This set of template indices is compared to the indices produce by the algorithms which are under test.

These two testing methods are very different in concept: the first is a geometric method of spatial measurement, and the second is a topological method of ordering the pattern elements. Both concepts will be met continually in our investigations.

3.6. Geometric and Topological Approaches

In this thesis "geometry" is used in the sense of exact measurement of the location of a point in two- or three-dimensional Euclidean space. The 3D model of the target surface has a rigid shape composed of vertices located in E^3 space, and the values of these vertices will change if the surface is, say, stretched.

Here the sense of "topology" relates to the graph of vertices which is derived by the preprocessing methods in the 2D image plane. The research methods will, at different times, be studying either the topology or the geometry of the model, and in many cases the thesis will be investigating the relationship between these two views, especially in the recorded image, which combines graph topology and geometry within its data structure.

3.7. The Investigation Process

Figure 3.7.1 shows the steps by which the research questions are investigated. These steps are now described in detail.

Investigation 1. Geometric Constraints

The motivation for this investigation is that in the Stereo Vision field, epipolar geometry and rectification have been used to help solve the Correspondence Problem of finding corresponding features in each image. This leads to the possibility that some of the same geometric constraints may be used, or extended, in this thesis. Also, the calibration of both intrinsic and extrinsic parameters is an onerous task, and this task can be minimised by setting up the system so that

the number of parameters is reduced. This is achieved by particular spatial alignment of the apparatus.

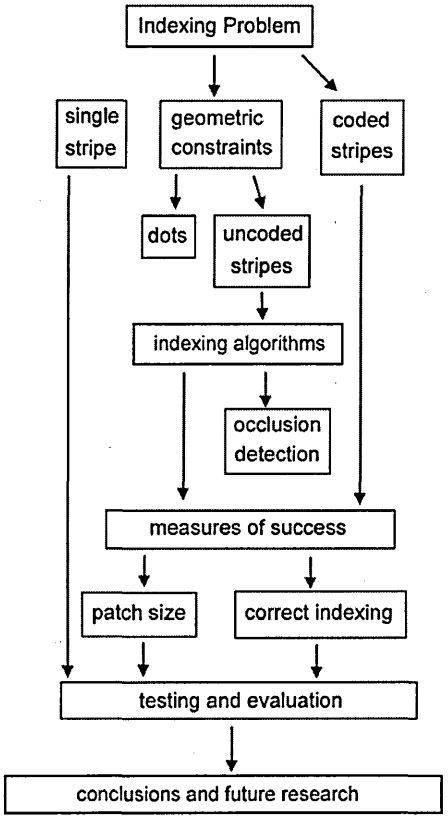

Figure 3.7.1: Steps in the Investigation Process

Investigations will also be undertaken to examine whether exact spatial measurement, such as the spacing between stripes, can be used to inform connection constraints in the indexing algorithms.

Investigation 2. Dots *v.* Stripes

Dots are discrete in both spatial dimensions, say horizontal and vertical, as opposed to stripes which are continuous in one dimension and discrete in the other. This means that in theory, using epipolar geometry and the geometric constraints, each dot has a unique range of positions in the image plane. We will examine whether this uniqueness can be usefully applied to solve the Indexing Problem, or whether continuous stripes are more appropriate.

Investigation 3. Indexing Algorithms

Some methods must be devised to search through the recorded image data, pick out pattern elements, and index them. Intuitively the process seems similar to that used in pattern recognition processes to search around a field of pixels. In this thesis the search methods will be adapted to deal with the chosen pattern elements - stripes. Definitions will be introduced to categorize stripe elements and how they connect through the data, and a number of problem areas and ambiguities will be found.

Investigation 4. Occlusion Detection

This and other work identifies occlusions as one of the principal sources of indexing errors. Here a major effort will be made to define and categorize occlusions, and thus modify the indexing algorithms based on logical relationships between the surface shape and the projected patterns, rather than using a heuristic approach.

Investigation 5. Testing for Success

Firstly, the scanning system will be tested to ensure that the system parameters are accurately measured. Secondly, the accuracy and linearity of the outputs will be compared to the industry standards, and to the outputs obtained from our Single Stripe Scanner. Thirdly, the indexing methods will be tested against the results obtained by the template data. Two measures of success will be considered: the size of the "patch" of indexed surface, and the indexing accuracy within this patch. These two measures are chosen because different applications are likely to require different proportions of patch size *versus* accuracy. For instance it may be that a large inaccurate patch is preferred if other means can then be found to remove the inaccuracies, or it may be that a small patch can be registered alongside another overlapping small patch to create a larger surface.

Finally, conclusions will be drawn from these results, and some initial ideas will be suggested as to the direction of future research.

Chapter 4: The Single Stripe Scanner

The Single Stripe Scanner projects only one pattern element *per* recorded image, and therefore does not incur the Indexing Problem. Nevertheless, an analysis of the results obtained from such a system is useful as a comparison with the Structured Light Scanners which project a sequence of pattern elements in each image. In addition, many of the methods which have been used in the Single Stripe Scanner can be readily adapted for Structured Light systems.

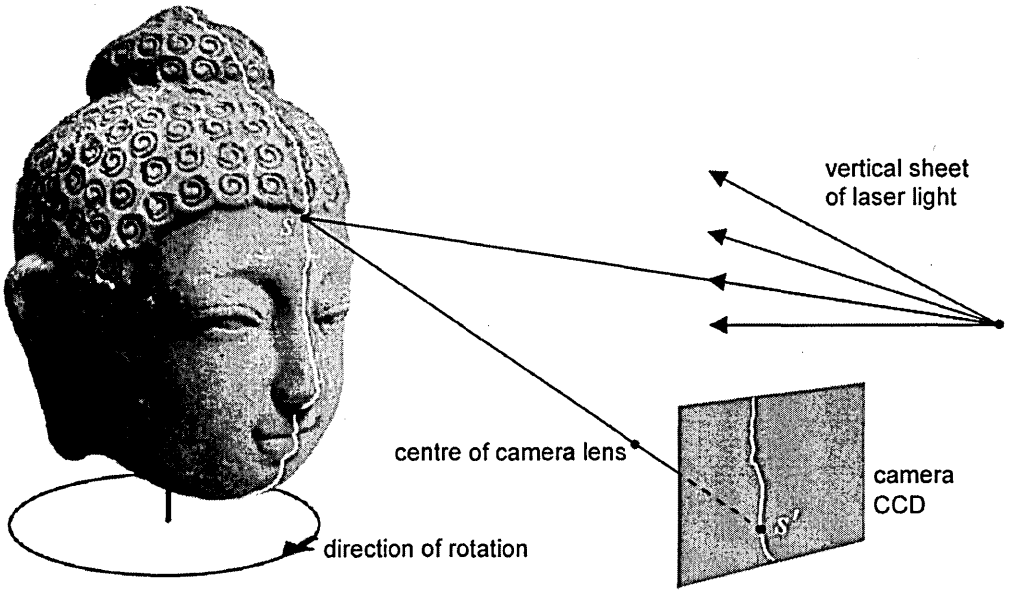

Figure 4.1.1: The Single Stripe Scanner.

Figure 4.1.1 shows the arrangement of the Single Stripe Scanner built as part of the investigations. The apparatus consists of a CCD video camera (768x576 pixels, interpolated by the video card to 800x600 pixels) and a laser light source (635 nm wavelength) which projects a vertical sheet of light towards the target object. The camera and laser source are in a fixed geometric relationship, and the target object rotates on a calibrated turntable. A light stripe is cast onto the object, and a single video frame is recorded for each successive rotation of the object, which is typically less than one degree. The distance from the lens to the centre of the turntable is 300 mm, and the object in this case is 200 mm high.

The profile of the reflected stripe is recorded in the video camera, and this represents a projection of point s on the surface of the target object to a point s' in the image plane of the camera CCD. The objective of the scanner is to measure the location of surface points in E^3 space, which requires a precise description of the projection. The description is categorised by two sets of system variables: the *intrinsic* parameters concerning the way light is emitted from the laser and recorded by the camera, and the *extrinsic* parameters measuring the spatial position of the system elements.

4.1. Measurement of the Intrinsic and Extrinsic Parameters

Figure 4.1.2a shows a plan view in a Cartesian coordinate system composed of orthogonal X-, Y- and Z-axes, looking down the Y-axis, of the apparatus.

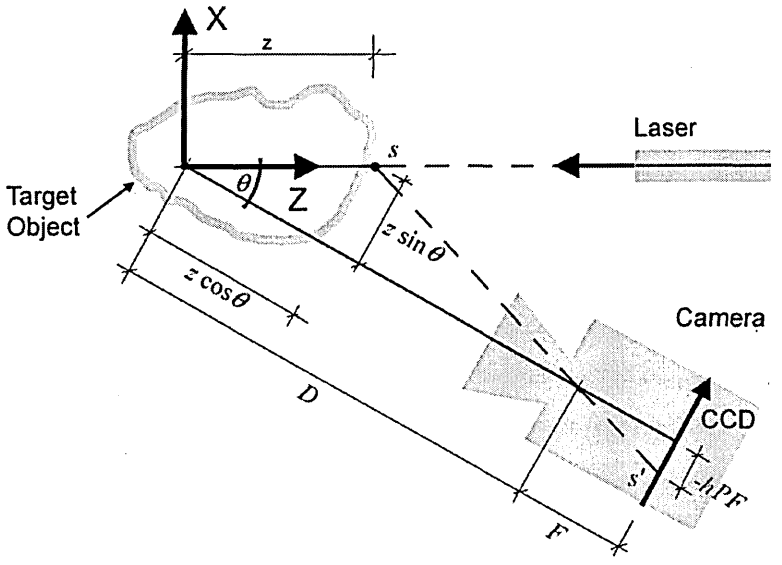

Figure 4.1.2a: The apparatus viewed "from above".

Figure 4.1.2b shows the same apparatus viewed "sideways", i.e. along the X-axis.

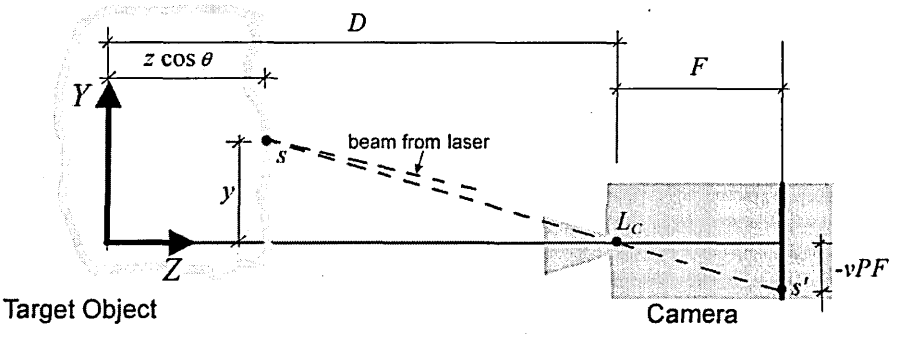

Figure 4.1.2b: The apparatus viewed "from the side".

The apparatus is positioned so that:

- the sheet of laser light is parallel to the *Y*-axis,
- the CCD plane is parallel to the *Y*-axis,
- the straight line from the centre of the CCD to the centre of the lens (the "camera axis"), passes through the origin (0,0,0) of the *XYZ* axes,
- the distance between the centre of the lens and the origin is given as *D*, and
- the camera axis subtends the laser sheet at angle θ.

Here *D* and θ represent the extrinsic parameters of the system, which must be calculated prior to the scanning process. The intrinsic parameters are:

- the focal length *F*, which is the distance from the centre of the lens to the CCD sensing plane, and
- the size *PF* of each square pixel. Note that the camera axis must intersect the centre of one pixel.

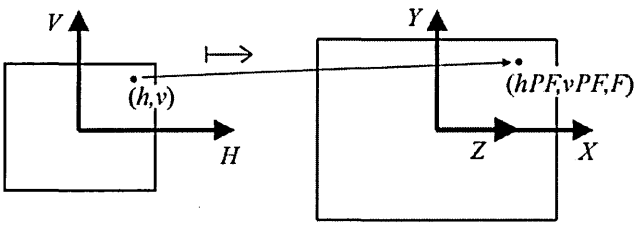

Figure 4.1.3: Transformation of the image into the system space.

The pixel size is expressed as *PF*, so that the focal length *F* can be cancelled in the calculations. At this stage it is assumed that the lens projection is linear, modelled as a "perfect" pinhole camera, and that the CCD is a continuous 2D space capable of finding a location for any point *s'*. This CCD space is shown in Figure 4.1.3 (left) with a sensed point (h,v) which maps to a point (hPF,vPF,F) in the system space (right).

The parameters of surface point $s = (x,y,z)$ can then be found as follows:

by similar triangles in Figure 4.1.2a

$$\frac{-hPF}{F} = \frac{z\sin\theta}{D - z\cos\theta}$$

(note that by construction $F > 0$ and $D > z\cos\theta$),

therefore

$$z = \frac{DhP}{hP\cos\theta - \sin\theta}.$$

Eq. 4.1.1

By similar triangles in Figure 4.1.2b,

$$\frac{-vPF}{F} = \frac{y}{D - z\cos\theta}$$

(again, note that by construction $F > 0$ and $D > z\cos\theta$),
therefore

$$y = vP(z\cos\theta - D).$$

Eq. 4.1.2

From Figure 4.1.2a

$$x = 0.$$

Eq. 4.1.3

A complete model of the surface is then produced by rotating the target object on the turntable by a given angle.

4.2. Intrinsic Distortion

The accuracy of the extrinsic parameters depends on the initial spatial measurement of the apparatus, and the amount of movement away from those measurements which occurs during scanning. The intrinsic parameters are much more difficult to measure, mainly because the camera lens does not project the image in a "pinhole" fashion [Fryer *et al.,* 1994]. The lens distorts the recorded image as seen on the sensor plane, and although this is discussed in detail in Section 5.6.1, an approximation is given here. Radial distortion considers the distance of an image point from the centre of the image; the greater the distance the less the distortion. In this simplified version, using the Tsai Camera Model [Tsai, 1987], an undistorted point at u is distorted to position d, such that

$$x_u = x_d(1+\kappa r^2),$$

Eq. 4.2.1

$$y_u = y_d(1+\kappa r^2),$$

Eq. 4.2.2

where

$$r^2 = {x_d}^2 + {y_d}^2.$$

Eq. 4.2.3

The coefficient κ is determined by calibration.

4.3. Processing the Data

The input data to the system is the recorded image which is sensed on the camera CCD. The data is stored as a bitmap array $C \times R$ of pixels arranged in C columns and R rows with luminance $l(c,r) \in [0,255] \subseteq \mathbb{N}$.

The position of the vertical stripe in the recorded image is found for each row r of pixels by finding the brightest pixel m in that row, such that $m = \arg\max_{c \text{ in } [1, C]} l(c,r)$. The stripe is then expressed as a sequence of maximum brightness values $M = \{m_i \mid m_i \in [1, C], i = 1, 2 \ldots R\}$.

4.3.1. Subpixel Estimation

As stated above, the sensor space is considered to be continuous, and any location (h,v) of a sensed point on the CCD should be found to as high an accuracy as possible. So far the stripe has been expressed as a sequence M of integers representing discrete pixels, but methods exist to estimate the centre of the stripe to subpixel accuracy [Fisher and Naidu, 1996]. In Figure 4.3.1 the luminance values $l(c,r)$ along one pixel row, row r_i, are shown, with the brightest pixel being at $c = m_i$. Without subpixel estimation, the location of the centre of the stripe in that row is then given by $(h,v) = (m_i, r_i)$. The subpixel estimator takes account of the neighbours of the brightest pixel to adjust the estimation of the centre of the stripe by δ_i pixels, giving a revised location as $(h,v) = (m_i + \delta_i, r_i)$.

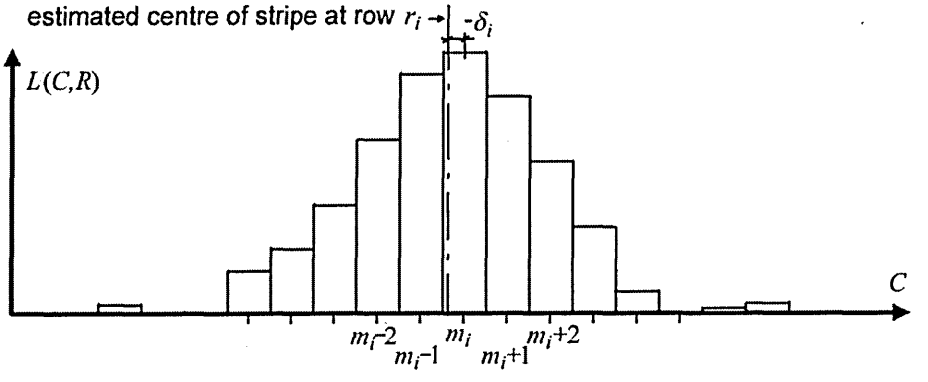

Figure 4.3.1: Luminance across one row in the pixel array.

Here the estimation uses a *centre of mass* formula [Fisher and Naidu, 1996], giving

$$\delta_i = \frac{2l(m_i-2,r_i)+l(m_i-1,r_i)-l(m_i+1,r_i)-2l(m_i+2,r_i)}{l(m_i-2,r_i)+l(m_i-1,r_i)+l(m_i,r_i)+l(m_i+1,r_i)+l(m_i+2,r_i)},$$

Eq. 4.3.1

where $l(c,r) \in [0,255]$ gives the luminance of the pixel at column c, row r, and the brightest pixel on each row r_i is (m_i,r_i), i = 1, 2, ...R. The formula uses the value of the brightest pixel (m_i,r_i), the value of the two pixels to the right, $(m_i + 1,r_i)$ and $(m_i + 2,r_i)$, and the two pixels to the left, $(m_i - 1,r_i)$ and $(m_i - 2,r_i)$. Therefore we refer to this as the *5-pixel estimator.*

From Eqs. 4.2.1, 4.2.2, 4.2.3 and 4.3.1 it is now possible to calculate the location of surface point $s_i = (x_i,y_i,z_i)$ from its corresponding sensed point $s_i' = (h_i,v_i)$. Note, however, that the set of surface points S can only have at most R members, corresponding to the sensing of the vertical stripe at each pixel row in the CCD.

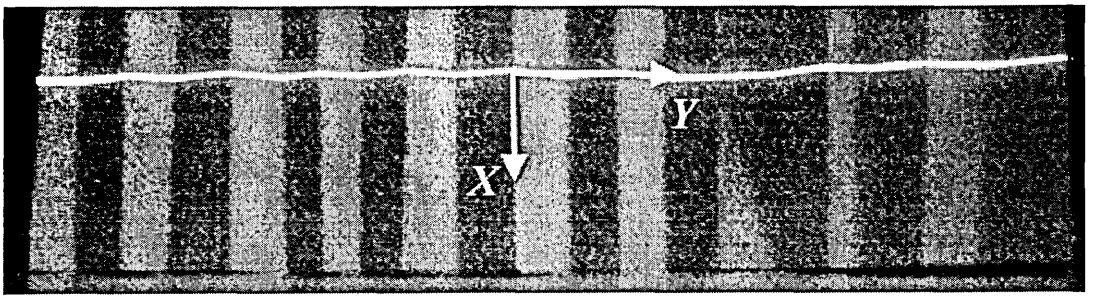

Figure 4.3.2a: Laser stripe projected onto folded paper (Z-axis points towards the viewer).

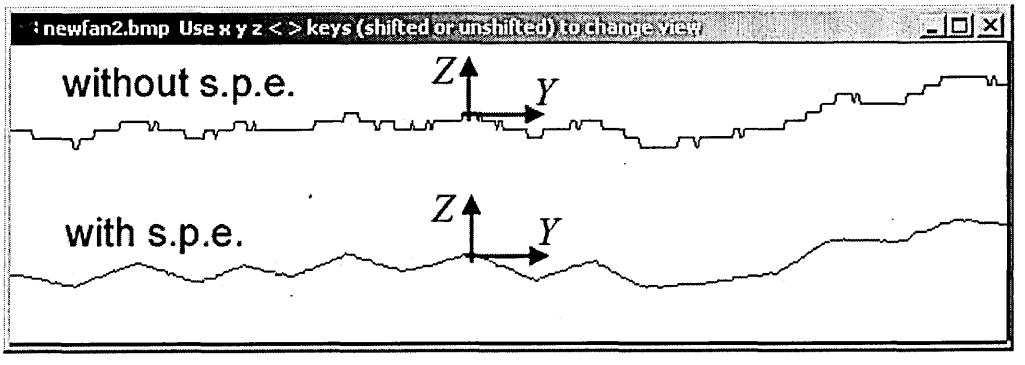

Figure 4.3.2b. Depth profile from Figure 4.3.2a (slightly magnified).

Figures 4.3.2a and 4.3.2b show the subpixel estimator in action. A laser stripe is projected onto a lightly folded sheet of paper (Figure 4.3.2a). The resulting depth calculations (Figure 4.3.2b) show the profile of the folded paper with and without subpixel estimation (s.p.e.), and the increased resolution of the subpixel estimator can be clearly seen. Recalling the 5-pixel estimator of Eq. 4.3.1

$$\delta_i = \frac{2l(m_i-2,r_i)+l(m_i-1,r_i)-l(m_i+1,r_i)-2l(m_i+2,r_i)}{l(m_i-2,r_i)+l(m_i-1,r_i)+l(m_i,r_i)+l(m_i+1,r_i)+l(m_i+2,r_i)},$$

and the luminance function $l(c,r)\in[0,255]\subseteq\mathbb{N}$ which has 2^8 possible values, the total possible values v for δ_i is $(2^5)^8$ if each of the five pixels can have any value from 0 to 255. However, Section 4.3 states that (m_i,r_i), is the brightest (or equal brightest) in that row, and therefore must be greater than or equal to the other four pixel values. This reduces the possible permutations of v to

$$v=\sum_{n=1}^{256} n^4$$

Eq. 4.3.2

possible events, which is a very large number in the order of 10^{11}. Although many of these events give the same value, such as zero, the s.p.e. is clearly capable of giving a large range of values.

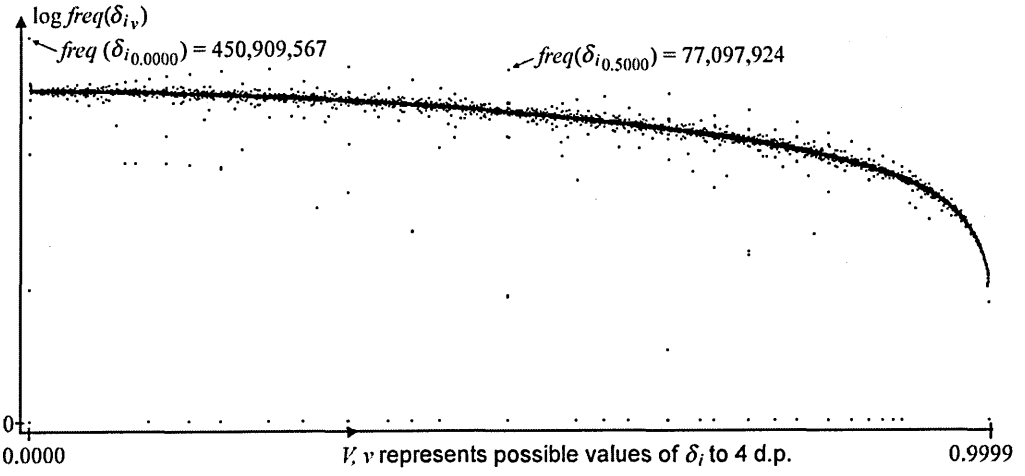

Figure 4.3.3: Possible values of δ_1 constrained to 4 decimal places.

Figure 4.3.3 plots all the possible positive values of δ_i when constrained to 4 decimal places, against the frequency of those 10,000 possible values; a plot of negative values would be similar. Although most values are evenly distributed, there are spikes of high and low frequencies around the simple fractions such as 0.5 and 0.25. For example, although there are 450,909,567 combinations of parameters giving a value of zero for δ_i, the next highest value is 0.00007849, and this value is only met once, when the numerator of Eq. 4.3.1 is 1 and the denominator is 1274. The frequency of values then gradually increases as more combinations give the same result, and this can be seen in the extreme left part of the plot. It should also be noted that although there are a large number of permutations of Eq. 4.3.1, the actual range of values will be much smaller, and initial estimates put the range as <1,360,689.

At this stage these results are noted, but their significance in terms of the accuracy of the subpixel estimator are not pursued. It appears that the estimator will tend to "lock onto" zero and simple fractions such as 0.5; it may also be that because the distribution is periodic it will tend to produce aliasing artefacts.

However, a visual estimation of the subpixel accuracy from data such as Figure 4.3.2 suggests a much lower range of possible values, in the order of eight intervals per pixel. The disparity between the theoretical and apparent resolutions will not be investigated at this stage, except to cite the usual suspects: noise from various sources, and inaccuracies and artefacts introduced by the use of discrete number types in the computer programs.

4.4. Testing the System

The Single Stripe Scanner was built in order to establish the basic design and implementation principles of triangulation scanning, and to compare the work produced by these investigations against the work produced by others in the field of what is now a very stable technology. A review of the literature (Section 2.4.1 to 2.4.2) shows a typical accuracy for depth measurement of 0.1 mm for a target object volume of 100 mm x 100 mm x 100 mm.

The scanning time is set at the video frame rate of 25 Hz. for each profile, and the time taken for a single rotation is therefore mainly dependent upon the mechanical method of rotating the turntable. The system is tested by scanning a planar surface at a known position in the system space, employing the subpixel estimator.

4.4.1. Results from Scanning a Planar Surface

Referring to Figure 4.1.3, the test piece is a sheet of white melamine board placed in the X-Y plane of the scanner, and rotated by -10 degrees about the X-axis. Angle θ is estimated at 10 degrees, distance D is 320 mm, and the pixel size ratio P is 0.000675. The CCD and video card record the image as an 800 high by 600 wide array of pixels with monochrome luminance. The test object is scanned 21 times, the turntable being rotated by 0.25 degrees between each scan.

The viewing volume, in other words the maximum size of object which can be scanned, is calculated from Eqs. 4.1.2 and 4.1.3. At the centre of the turntable, where $z = 0$, the maximum height viewed y_{MAX} is where $v = -400$, and the minmum height y_{MIN} is where $v = 400$, so that

$$y_{MIN} = -vPD = -89.1 \text{ mm},$$

$$y_{MAX} = 89.1 \text{ mm}.$$

The depth ranges between z_{MIN} where $h = 300$, and z_{MAX} where $h = -300$, so that

$$z_{MIN} = -2673.0 \text{ mm},$$

$$z_{MAX} = 179.2 \text{ mm}.$$

If the object rotates around the turntable to give a circular motion, the range of x values will be the same as the range of z values.

The viewing volume is therefore an irregular shape, and in practice it is sensible to express the maximum target object size as a cube with a height of

$$y_{MAX} - y_{MIN} = 178.2 \text{ mm}.$$

Table 4.4.1 shows the results of scanning the test piece, for scan 10 out of 21. The projected stripe is recorded on the CCD, and thence a set $\{z_1, z_2, \ldots z_R\}$ of 800 scanned depth values is calculated. The test ramp is measured by hand, and a linear interpolation then gives the

predicted value of each of the 800 surface depths. These are subtracted from the scanned values to give a set $\{e_1, e_2, ...e_R\}$ of error values. In each of samples 1 to 8, 10 consecutive error values $\{e_{START}, e_{START+1}, ...e_{END}\}$ are chosen, and the arithmetic sample mean of those 10 depths is then calculated ("mean error" column). The next column shows the standard deviation of those samples.

Samples 9-11 use a wider population sample to analyse the first half of the stripe (sample 10), the last half of the stripe (sample 11), and the whole stripe (sample 9).

sample	START	END	mean error (mm)	standard deviation
1	50	60	-0.756	0.122
2	150	160	-0.064	0.078
3	250	260	0.244	0.063
4	350	360	0.373	0.068
5	450	460	0.417	0.111
6	550	560	0.110	0.115
7	650	660	-0.221	0.230
8	750	760	-1.118	0.244
9	10	790	-0.153	0.729
10	10	400	-0.208	0.679
11	400	790	-0.099	0.773

Table 4.4.1: Mean error and standard deviation for Scan 10 out of 21.

4.4.2. Comments on the Test Results

In sampled data there is often a regular swing positively and negatively of the error values, caused by the periodic sampling steps, or *quantization noise* (as seen in Figure 4.4.2 for "no s.p.e."). The mean error value will tend to cancel out these positive and negative swings, whereas the standard deviation value will maintain them. Therefore if the mean error value is much lower than the standard deviation value, it may suggest that the errors are due to quantization noise.

The results in samples 1 to 8 show a standard deviation of the error of between 0.063 and 0.244 mm, in the 10-value samples. The smallest errors are around pixel row 300, and errors increase above and below this row. This effect is confirmed in the results for the other 20 scans,

although no explanation is offered at this stage. The wider samples 9 to 11 show a standard deviation of the error of around 0.7 mm. A visual inspection of the results suggests that this may be due to effects introduced by the video card, which interpolates the 768x576 camera output to 800x600 pixels, by doubling up every 32nd row. There is no obvious pattern of differences between the mean error and standard deviation values, which suggests that quantization noise is not a factor.

Figure 4.4.2 which shows a visualisation of the scanned ramp, magnified to show one twentieth of the surface. The diagram shows a clear reduction in the quantization noise when using the subpixel estimator, although there is still noticeable noise. Some possible causes of this noise may be

- laser speckle (see below),
- data processing in the video card,
- unevenness of the test surface, and
- residual quantization noise after subpixel estimation.

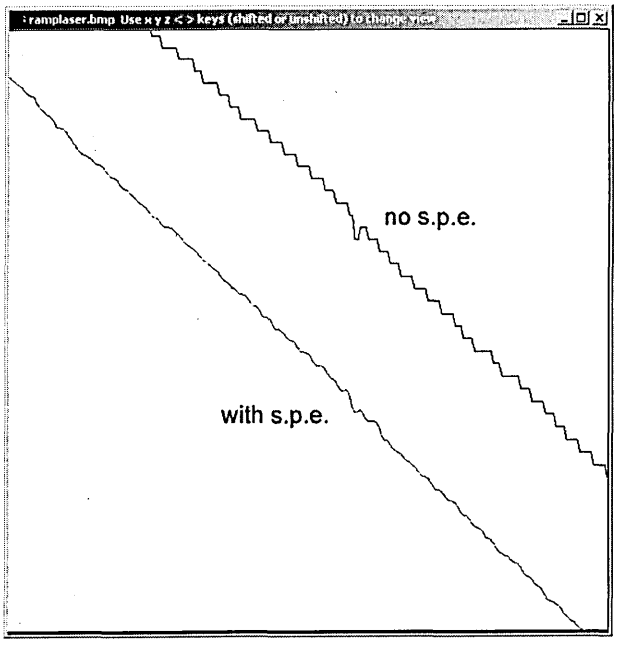

Figure 4.4.2: Profile of scanned ramp with and without subpixel estimation.

4.4.3. Laser Speckle

As well as testing the system using a planar ramp, several objects were scanned. The terracotta Buddha head, as seen in Figure 4.1.1, exhibited the effect known as "laser speckle" [Becker, 1995] when scanned. The microscopic facets on the terracotta surface reflect the coherent light in a constant (because of the single wavelength) manner, and their reflection produces an interference pattern which confuses the viewer or camera into thinking that the surface is actually in a slightly different position. This effect is less noticeable on the smoother melamine surface of the test plane.

The speckle is unavoidable when using laser light, but it does not occur using incoherent light, because the wavelengths are continually varying, and an averaging effect takes place. Therefore a revision to the apparatus was made whereby a stripe of white light was projected from a slide projector. A comparison was made between the scans of the Buddha using laser light, and using white light. Figure 4.4.3 shows the visualisation of a detail of the Buddha using laser light (left) and white light (right); the noisiness caused by laser speckle is clearly visible.

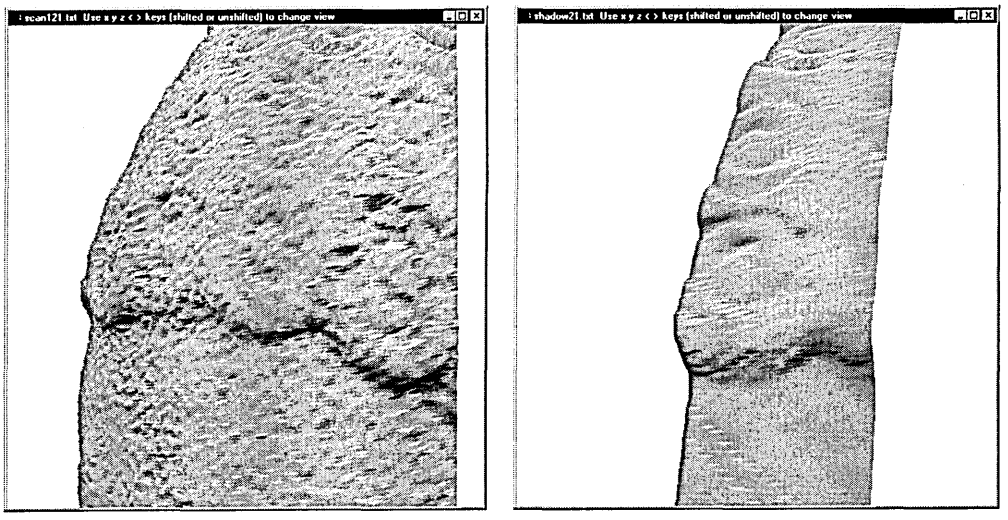

Figure 4.4.3.Detail of scanned terracotta head, using laser light (left) and white light (right).

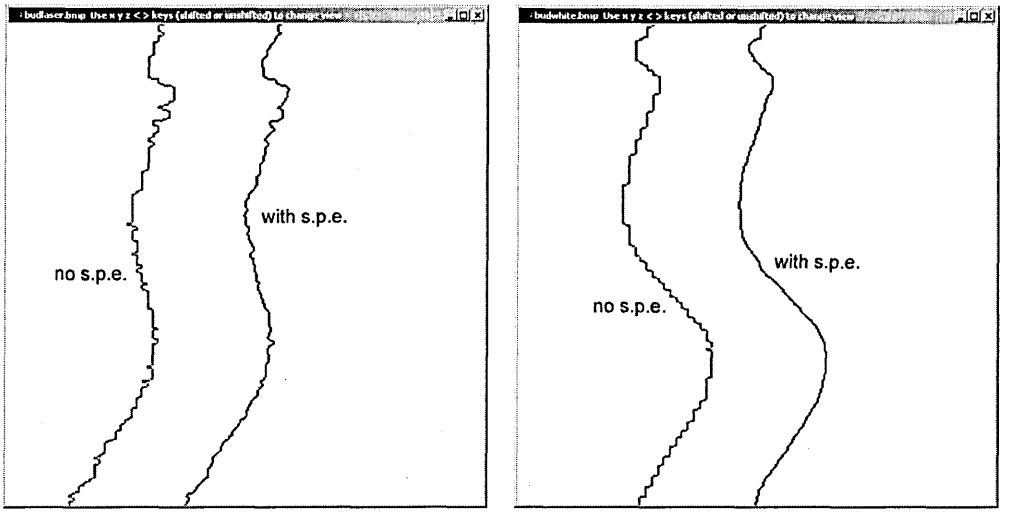

Figure 4.4.4: Depth profiles of Figure 4.4.3.

Figure 4.4.4 shows the plot of a single stripe from the Buddha scans, with and without subpixel estimation (s.p.e.), again using laser light (left) and white light (right). Note that the slightly different shapes of the stripe profile are due to the two scans being taken from different positions.

Both these visualisations, in Figures 4.4.3 and 4.4.4, show a greater accuracy of representation using white light compared with using laser light. The laser light exhibits a high degree of speckle which degrades the output data (for the terracotta surface of the target object). Further tests, not included in this thesis, show that defocusing the camera reduces the speckle effect, but with a loss of resolution.

4.5. Conclusions

The Single Stripe Scanner has a resolution, which can be expressed as the standard deviation of the error between the expected and actual measurements, of 0.063 mm to 0.773 mm. The maximum volume of the target object can be expressed as a cube of sides 178.1 mm. The stated accuracy is dependent upon successfully using the subpixel estimator with a stripe width of at least five pixels, and an inverse radial distortion function as shown in Eqs. 4.2.1, 4.2.2 and 4.2.3. The noise level is reduced by projecting incoherent light rather than laser light, to

eliminate the laser speckle effect. Therefore in this thesis laser speckle will be considered a disadvantage, although it can also be used positively as a measure of roughness [Hilton, 1997].

Further improvements can be expected by using a video capture card which does not interpolate the output from the camera.

Chapter 5: Data Structures and Processes

The input data for the scanner is the stored luminance values of the recorded camera image. The data must be processed and presented in the most useful way for the investigations, and as we have seen in the Introduction, these investigations will require both accurate spatial measurement, and ways of expressing the connections between pattern elements. This chapter begins with some general observations about the Multiple Stripe Scanner, gives formulae for the spatial measurements, and then describes the data structures and processes in detail.

5.1. The Multiple Stripe Scanner

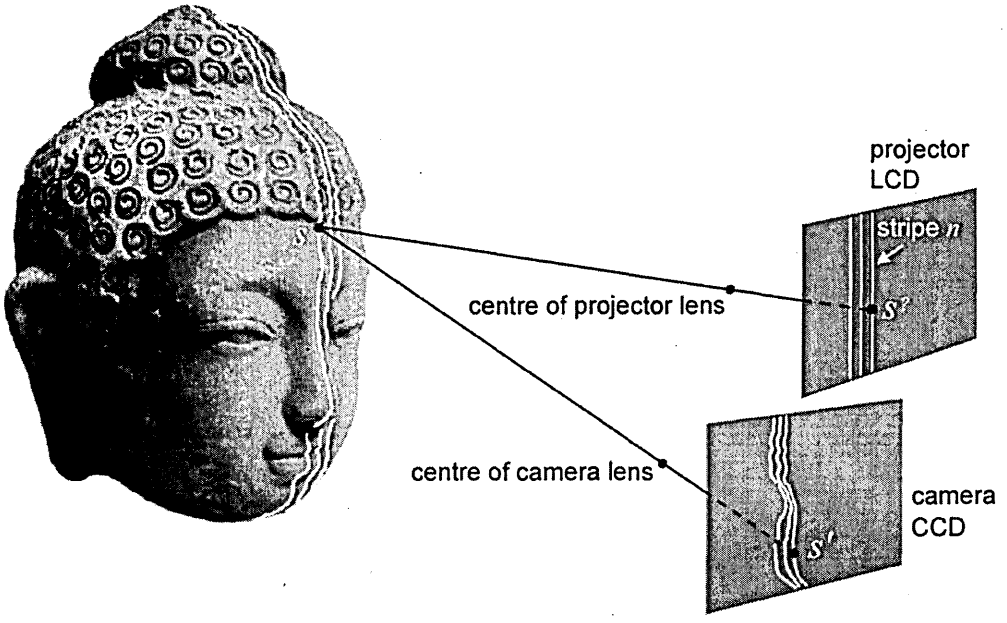

Figure 5.1.1: The Multiple Stripe Scanner.

Figure 5.1.1 shows the Multiple Stripe Scanner in general form, and this can be compared with the Single Stripe Scanner shown in Figure 4.1.1, Chapter 4. Three stripes are shown, which are projected onto the target surface and recorded by the video camera. Light at s' in stripe n is cast onto the target surface at s and onto the CCD image plane at s'. Referring to Section 5.1, in order to find the surface point $s = (x,y,z)$ we need to measure the parameters of $s' = (h,v,n)$, where (h,v) gives the horizontal and vertical location of s' in the image plane, and n is the index

of the stripe on which s' sits. This process is described by the scanning function $scan(h,v,n) = (x,y,z)$.

Some initial observations can be made:

- It is not possible to say exactly where $s^?$ is located on stripe n; hence the notation with a question mark. If a dot pattern were used which was vertically discrete, then the position of $s^?$ would be locatable, but there would need to be separation between the dots. This would mean that some vertical resolution would be lost compared with the continuous stripe which can be sampled on each pixel row (see Section 1.4).

- For the stripe to be considered as continuous, the sheet of light leaving the projector must contain an even distribution of light rays. In practice this means that the pixels in the projector LCD must not be discernible in the recorded image.

- Ideally the target should be a diffuse, Lambertian surface which scatters the rays sufficiently to ensure that enough light arrives at the sensor so that it can be accurately measured.

- In addition, the sensor must be able to efficiently separate projector light from ambient light and system noise.

These observations are discussed in more detail in relation to the Single Stripe Scanner, in Section 4.1.

5.2. Spatial Measurements

The scanning apparatus is shown in Figure 5.2.1a, viewed "from above", down the negative *Y*-axis, and in Figure 5.2.1b, viewed "from the side", down the *X*-axis. The setup is similar to that for the Single Stripe Scanner, with an LCD projector replacing the laser. The *camera axis* is the line from L_C, the centre of the camera lens, to the origin O, giving distance D_C. The *projector axis* is the line from L_P, the centre of the projector lens, to the origin O, giving distance D_P. The camera and projector are positioned so that the camera and projector axes, and the middle rows

of the camera and projector pixels, are all located in the *X-Z* plane. The camera and projector axes intersect at the origin *O*, giving angle θ. D_C, D_P and θ are the extrinsic parameters of the system.

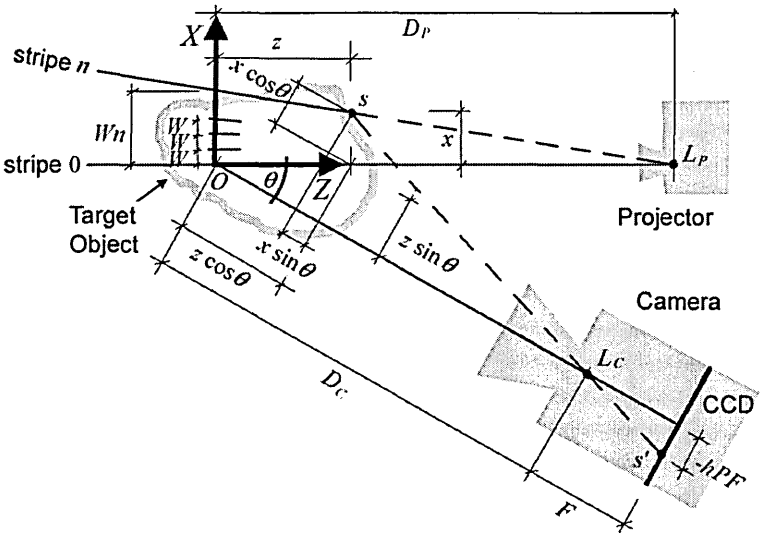

Figure 5.2.1a: The Multiple Stripe Scanner viewed "from above".

Evenly spaced, vertical (i.e. parallel to the *Y*-axis) stripes are projected which intersect the *X*-axis, and the distance between intersections is *W*. A point *s* on the surface of the target object reflects a beam (denote by the dashed line) in stripe *n* through the camera lens at L_C and onto the image plane at *s'*, where it is sensed by the camera CCD. The camera CCD is at distance *F*, the *focal length*, from the centre of the lens L_C, and the size of one square pixel in the CCD is given as *PF*.

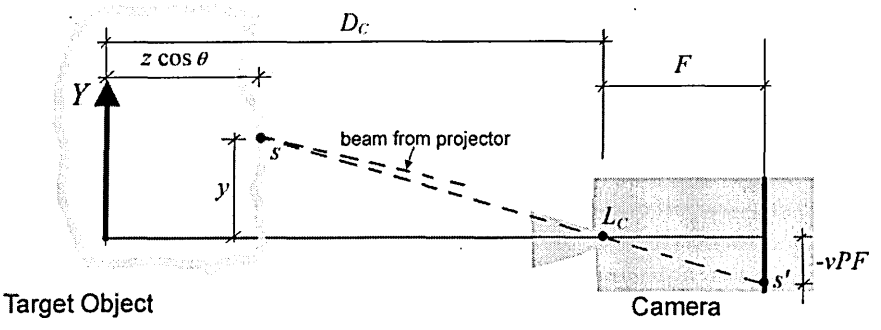

Figure 5.2.1b: The Mulitple Stripe Scanner viewed "from the side".

Therefore in the diagram it can be seen that if point s' is $-h$ pixels horizontally from the centre of the CCD, then the distance in system space is given as $-hPF$. Note that the h and v values are given relative to the centre of the CCD, and that the negative or positive sign must be observed. Similarly, when viewed in Figure 5.2.1b down the X-axis, if point s' is $-v$ pixels vertically from the centre of the CCD, then the distance in system space is given as $-vPF$.

The process of calculating the position of s in the target space, based on the position of s' in the image space, is essentially the same as for the Single Stripe Scanner. The additional variable is the stripe index n, and so referring to Figure 2a:

by similar triangles,

$$\frac{Wn}{D_P} = \frac{x}{D_P - z}$$

(note that by construction $D_P > 0$ and $D_P > z$)

and,

$$\frac{-hPF}{F} = \frac{z\sin\theta + x\cos\theta}{D_C - z\cos\theta + x\sin\theta}$$

(note that by construction $F > 0$ and $D_C > (z \cos \theta - x \sin \theta)$

therefore

$$x = Wn - z\frac{Wn}{D_P}$$

and

$$z\sin\theta + x\cos\theta = -D_C hP + zhP\cos\theta - xhP\sin\theta .$$

Combining,

$$z\sin\theta + Wn\cos\theta - z\frac{Wn}{D_P}\cos\theta = -D_C hP + zhP\cos\theta - WnhP\sin\theta + z\frac{Wn}{D_P}hP\sin\theta ,$$

therefore

$$z(\sin\theta - hP\cos\theta - \frac{Wn}{D_P}(\cos\theta + hP\sin\theta)) = -D_C hP - Wn(\cos\theta + hP\sin\theta),$$

therefore

$$z = \frac{D_C hP + Wn(\cos\theta + hP\sin\theta)}{hP\cos\theta - \sin\theta + \frac{Wn}{D_P}(\cos\theta + hP\sin\theta)}$$

Eq. 5.2.1

Also from Figure 5.2.1a, by similar triangles,

$$\frac{x}{Wn} = \frac{D_P - z}{D_P}$$

therefore

$$x = Wn - z\frac{Wn}{D_P}$$

Eq. 5.2.2

Now referring to Figure 5.2.1b, by similar triangles,

$$\frac{D_C - z\cos\theta}{y} = \frac{F}{-vPF}$$

therefore

$$y = vP(z\cos\theta - D_C)$$

Eq. 5.2.3

In these equations h and v give the position of the sensing pixel relative to the centre of the CCD, and n is the index of the stripe containing the sensed beam. The intrinsic parameters are constants W and P, where PF is the width and height of one square pixel (see Section 4.1), as well as the radial coefficient κ_1 which is described below in Section 5.6.1. The intrinsic and extrinsic parameters are found by calibration, and h and v are found by measurement in the image space; but the identification of stripe index n as seen in the image space must be found using Indexing Methods. *Note that Eq. 5.2.1 reduces to Eq. 4.1.1 when n = 0, thus maintaining consistency with the Single Stripe Scanner.*

5.3. Relative and Absolute Values of *n*

It is often easier for the indexing methods to find only the relative value of *n,* in other words that a stripe with index n_i is three stripes to the left of stripe n_{i+3}. If the stripes planes were parallel to each other and to the Y-axis, this relative value would be sufficient to calculate the geometric surface model, as *n* would have a linear relationship with *x, y* and *z*. This would mean that if all values of *n* were increased by say 5, the model would simply translate to another position in the system space, and not affect the overall shape and size of the object. However, in the scanning system as shown in Figure 5.2.1a the stripe planes radiate from the centre of the projector lens. The relationship between *n* and *z* is not linear, as can be seen from Eq. 5.2.1, and any relative change in the values of *n* would cause a distortion of the shape and size of the model. **Therefore the *actual value* must be found for *n*.**

5.4. Limiting Value for *z*

Although $z < D_P$ by construction, there is no negative limit for *z*, and there will be cases where as the denominator of Eq. 5.2.1 approaches zero, *z* approaches infinity in the negative direction. At infinity this condition would be given by:

$$\text{if } hP\cos\theta - \sin\theta + \frac{Wn}{D_P}(\cos\theta + hP\sin\theta) = 0$$

$$\Rightarrow hP\cos\theta = \sin\theta - \frac{Wn}{D_P}(\cos\theta + hP\sin\theta)$$

$$\Rightarrow hP(\cos\theta + \frac{Wn}{D_P}\sin\theta) = \sin\theta - \frac{Wn}{D_P}\cos\theta$$

$$\Rightarrow hP = \frac{\sin\theta - \frac{Wn}{D_P}\cos\theta}{\cos\theta + \frac{Wn}{D_P}\sin\theta}$$

$$\Rightarrow hP = \frac{\tan\theta - \frac{Wn}{D_P}}{1 + \frac{Wn}{D_P}\tan\theta}$$

Eq. 5.3.1

We know that s is always located on a stripe, and Figure 5.3.1 illustrates the condition where s is situated infinitely far along stripe n, such that the projected and reflected beams are parallel. From the figure:

$$\frac{Wn}{D_P} = \tan\alpha$$

therefore from Eq. 5.3.1:

$$hP = \frac{\tan\theta - \tan\alpha}{1 + \tan\theta\tan\alpha} = \tan(\theta - \alpha)$$

Eq. 5.3.2

which is confirmed diagrammatically in Figure 5.3.1. However, as the projected and reflected

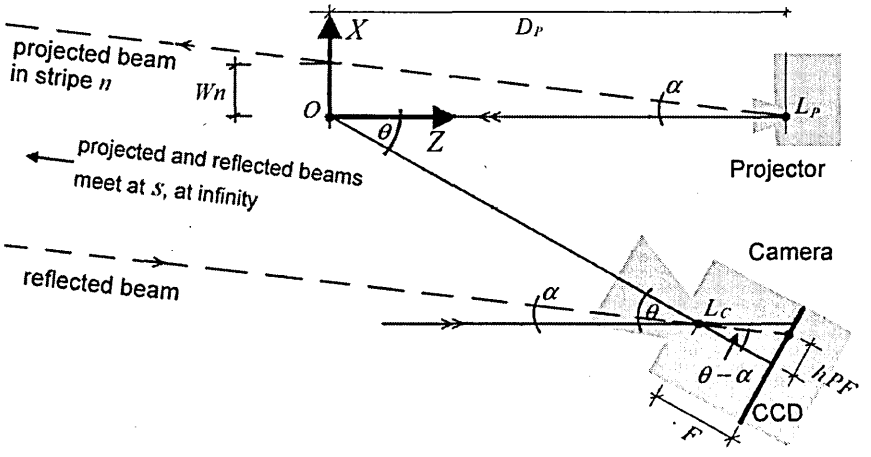

Figure 5.3.1: Condition where the target surface is infinitely far away.

beams by definition must intersect, they can never be parallel, and so the condition for $z = -\infty$ will never be met.

These equations are calculated with s located at specific octants within the 3D Cartesian coordinate system. If the polarity of all parameters is observed, the equations will also be true for locations of s throughout the space.

5.5. The Data Structures

Figure 5.5.1 shows the six principal data structures which are used to process the image data and to measure and visualise the target surface. Only three stripes are shown.
In summary, the process for creating these data structures is:

1. The input data set is the recorded image, which is a pixel array of luminance values.

2. The stripes are recognized as local peak values along each row.

3. Indexing methods then give stripe index values to these peaks.

4. This data is then rearranged to give a set of indexed stripes, each stripe n having a sequence of numbers which represent the horizontal pixel position of each stripe for every row r.

5. The subpixel estimator then adjusts these horizontal positions to give greater accuracy. Referring to Figures 5.2.1a and 5.2.1b, this array gives the values for point $s' = (h, v, n)$ in the recorded image.

6. Finally, each point (h,v,n) is projected into the system space using Eqs. 5.2.1, 5.2.2 and 5.2.3 to give Cartesian coordinates $s = (x,y,z)$ for each surface point (refer again to Figures 5.2.1a and 5.2.1b). Thus we finally have an $R \times N$ array of (x,y,z) vertices, describing the 3D position of each vertex on each row r of each stripe n.

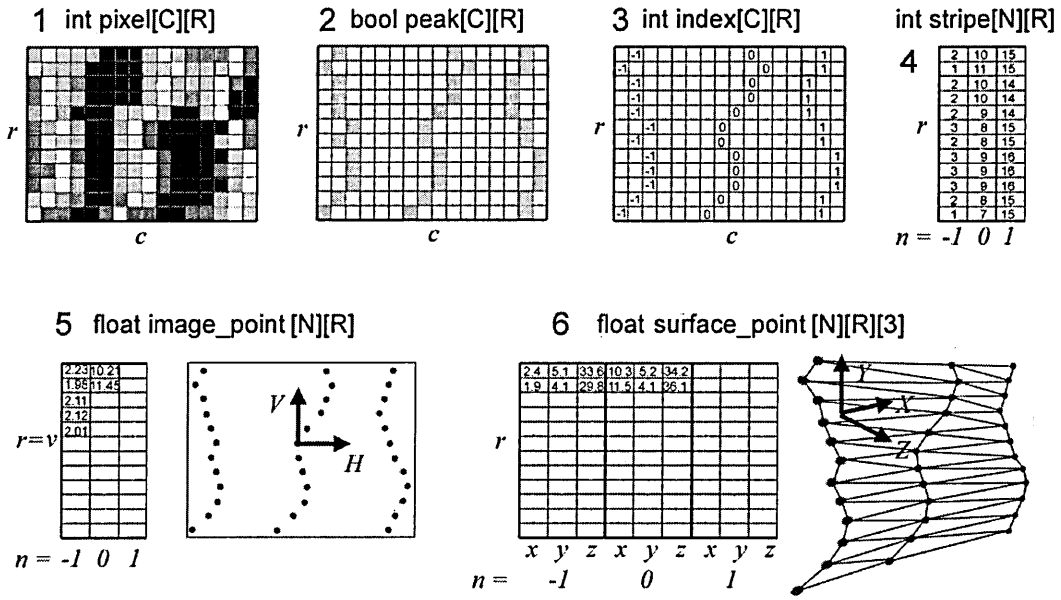

Figure 5.5.1: The six data structures.

Notice in the visualisation of (6) that the points have been connected to give a polyhedral surface. It is an important advantage of structured light scanning that the data structure carries an implicit ordering of the point cloud, which enables accurate initial surface reconstruction to be performed.

Looking at Figure 5.5.1 it can be seen that the relation between (1) and (5) is the transformation of the pixel array onto the image plane, expressed as a precise location of the centre of each stripe on each row. If the two diagrams were suitably scaled they could be superimposed to show this. In (6) the image plane is then transformed onto the target surface.

These data structures and processes are now discussed in detail.

5.5.1. The Pixel Array

The current system uses a standard monochrome video camera synchronised to the PAL standard, at a bandwidth of 5.5 MHz. This gives 575 visible horizontal lines of picture transmitted every 1/25 seconds, starting and ending in half lines. These half lines are padded by the digitally sampled video capture card, thereby giving 576 rows of pixels. To maintain the 4:3 aspect ratio each row has 4 x 576 / 3 = 768 pixels. Each pixel is a single 8 bit sample thereby

giving 256 discrete levels of luminance.The CCD is therefore represented as a $C \times R$ array of pixels, with luminance $l(c,r) \in [0,255] \subseteq \mathbb{N}$. It should be noted that the image which is sensed on the camera CCD is actually sampled twice, firstly by the CCD and secondly by the video capture card. At present it is assumed that this sampling can be considered as a single process which occurs at the CCD. It is also assumed that the output of each pixel sensor is a "fair assessment" of all the light received over that sensor's area. So far in the investigations there has been no evidence that this is not true.

Five important aspects of the way the image is stored by the pixel array are focus, aperture control, ambient light, unwanted artefacts, and other noise.

a. **Focusing issues.** If the image is not sharply focussed, the stripe will be spread more widely, in both the horizontal and vertical directions. Experiments with the Single Stripe Scanner show that for small amounts of defocusing this will have negligible effect on the horizontal position as set by the subpixel estimator, but will act as a smoothing convolution in the vertical direction, giving reduced resolution.

b. **Aperture Control.** This has a similar effect to defocusing, in that the stripe is widened in both directions if the aperture is greater and the luminance levels are therefore higher. The problem here is that the peak levels can attempt to push beyond 255, in which case clipping of a number of adjacent pixels occurs, making the centre of the stripe more difficult to find. The practical solution is to ensure that luminance levels are maintained within an upper limit, set at $l(c,r) < 200$ for the investigations.

c. **Ambient Light.** Ideally there should be no ambient light, so that the only signal recorded by the CCD sensors is from the reflection of the projected stripes. However, the peak data will be found using thresholding techniques to reduce the contribution of ambient light. Two limits are used: a lower limit typically of around $l(c,r) > 100$, below which the data is ignored, and an amplitude limit of typically around 8, representing the minimum difference between a peak in the white stripe and an adjacent trough in the dark stripe (see below in Section 5.5.2).

d. **Unwanted Artefacts.** These take the form of aliasing patterns in the recorded image due to the combination of sampling by the camera, projector and video capture card.

These effects can be reduced by subtly defocusing the devices and moving their relative positions.

e. **Other Noise.** The noise introduced by the camera and the video capture card for the shadow area to the left of the face in the sample recording "face3.bmp" is in the range of $102 < l(c,r) < 122$, a noise level of 20 out of 256 luminance levels. It seems likely that a similar amount of noise will also be present in the "stripy" parts of the recording, and the significance of this is discussed in Section 4.3.1 in relation to the subpixel estimator.

5.5.2. The Peak Array

The peak array maps in one-to-one fashion with the pixel array, and recognizes the stripes in the pixel array by finding the brightest pixel in every stripe, in every row. This is defined by the peak finder $peak(c,r)$ such that

$$l(c\text{-}1,r) \leq l(c,r) > l(c\text{+}1,r) \Rightarrow peak(c,r) = \text{TRUE}.$$

All other values in the peak array are marked as FALSE. To reduce the effects of ambient light and noise, a *threshold* luminance is set below which no pixel can be considered as a peak. Later work also included an *amplitude threshold* representing the minimum permissible difference in luminance between a peak and its neighbouring troughs (see below) on the same pixel row.

A *trough array* is also included in the data structures, which represents the "black stripes" which form in the gaps between the white stripes. As long as the (white) stripes are spaced such that their black counterparts are of approximately similar width, the results for troughs are as valid and accurate as for peaks. As expected, the conditions for a trough are that

$$l(c\text{-}1,r) \geq l(c,r) < l(c\text{+}1,r) \Rightarrow trough(c,r) = \text{TRUE}.$$

5.5.3. The Index Array

The index array takes some or all of the peaks and labels them with the index of the stripe which that peak helps to form. The index array maps in one-to-one fashion with the peak array (and the

pixel array), and there will be peak elements which are not indexed, as well as pixels which are not peaks, where *peak*(*c,r*) = FALSE. Pixels which are not peaks, and peaks which are not indexed, are given an arbitrary integer value NULL which is outside the range of stripe indices. In other words every index array element [c][r] which is not indexed will have a value of NULL. The indexing methods comprise a large proportion of this thesis; at this stage we simply state that the indexer *index*(*c,r*) = {*n*, NULL}.

5.5.4. The Stripe Array

Most of the indexing methods use algorithms based on connectivity and movement in the pixel space. Once the stripe index has been established, the data structures use a more compact representation where the gaps representing the non peaks are removed. This is achieved in the stripe array, where the row *r* dimension is still used, but the second dimension is now the stripe index *n*. The integer values of this array now represent the horizontal pixel position of a peak in stripe *n* for row *r*. Once again, if the horizontal position of a peak is undetermined it is given the arbitrary value outside the stripe index range. Therefore

$$stripe(n,r) = c,\ (index(c,r) = n) \wedge (peak(c,r) = \text{TRUE}).$$

Looking at the stripe array data in Figure 5.5.1, it can be seen that each column represents a stripe, with the values representing the horizontal displacement along each row.

5.5.5. The Image_Point Array

From the stripe array where *stripe*(*n,r*) = *c* the image_point array is created where *image*(*n,v*) = *h*, given that $v = r$ and $h = c + \delta c$. The value of δc is given by the subpixel estimator, discussed in Section 4.3.1.

This is essentially the data structure for the stripes as recorded in the continuous image space, with each stripe *n* represented as a sequence <*S'*> of points $s' = (h,v)$.

5.5.6. The Surface_Point Array

The parametric equations for $s = (x,y,z)$, Eqs. 5.2.1, 5.2.2 and 5.2.3, are then used to find the projection of s' to s using values h, v and n as stored in the image_point array.

In order to preserve the order of the surface points, the surface_point array is ordered as a two dimensional array of rows and stripe indices, each element containing the triple (x,y,z) representing the surface point location.

5.6. Calibration

The intrinsic and extrinsic parameters of the system are calculated by calibration, and the geometric model requires that this is done accurately. As this thesis is chiefly concerned with the Indexing Problem, which is a graph topological issue, accurate calibration is not normally important. However, some of the investigations into the relationship between vertex connections and geometric measurement (see Section 6.2) require at least the reassurance that accurate measurement can be achieved. Therefore an overview of the calibration process is given here.

The geometric model requires adjustment and measurement of the system in order to calculate surface point positions using the *scan()* function. We need firstly to calculate values for the radial coefficient κ_1 as described in Section 4.2; secondly to adjust the position of the system in line with the assumptions made in Section 5.2; and thirdly to provide values for the system constants W, P, D, C and θ as defined in Section 5.2.

5.6.1. Evaluation of Radial Coefficient κ_1

The Tsai Camera Model [Tsai, 1987] retrieves the undistorted location x_u of a point in the image plane, from its distorted location x_d :

$$x_u = x_d(1+\sum_{i=1}^{N}\kappa_i r^{2i})$$

Eq. 5.6.1

where κ_I is the radial coefficient, and r^{2i} is the square of the distance of the distorted point from the centre of the image plane, given by:

$$r^{2i} = (x_d^{\,2} + y_d^{\,2})^i .$$

Eq. 5.6.2

The value of N gives the length of the series, but work by Tsai and others has stated that $N = 1$ can be used in most cases.

In order to calculate κ_I a calibration chart is used, as seen in Figure 5.6.1.

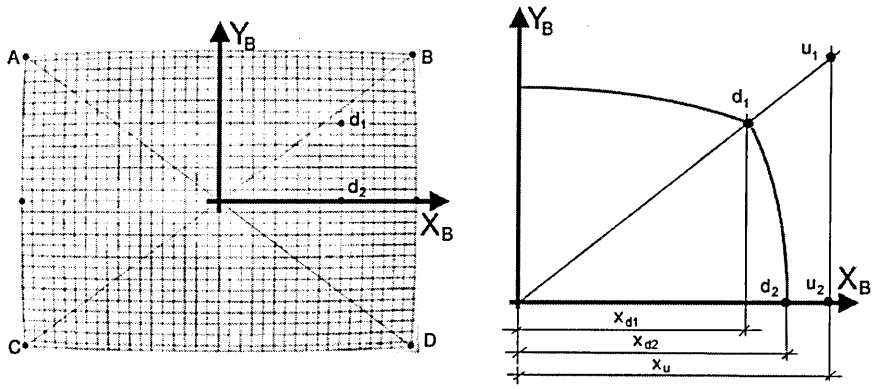

Figure 5.6.1: Calibration chart showing lens distortion.

5.6.2. Positioning the calibration chart

Figure 5.6.1 shows, on the left, a recorded image of a calibration chart comprising gridlines 1 cm. apart. We need to ensure that the chart is central and orthogonal to the camera, in other words (given the camera axes XYZ shown in Figure 5.2.1a and 5.2.1b), that

X_B is parallel to X,

Y_B is parallel to Y,

and X_B, Y_B and $-Z$ intersect at the same point.

The above positioning requirements are achieved ***to an accuracy of one pixel*** by adjusting the calibration chart and inspecting the resulting bitmap image so that (referring to Figure 5.6.1)

points A, B, C and D are each the same number of pixel rows and columns from their nearest corner in the bitmap image.

5.6.3. Formulating radial coefficient κ_1

In Figure 5.6.1, on the left we see an arbitrary point d_1 at the intersection of a horizontal and a vertical grid line, and point d_2 where that vertical grid line d_1-d_2 intersects the X_B axis. On the right we see in exaggerated form that line d_1-d_2 is curved by lens distortion. Recalling Tsai`s Camera Model

$$x_u = x_d(1+\sum_{i=1}^{N}\kappa_i r^{2i})$$

if $N=1$ we can state that for points d_1 and d_2

$$x_u = x_{d1}(1+\kappa_1 r_1^{\ 2})$$

$$x_u = x_{d2}(1+\kappa_1 r_2^{\ 2})$$

therefore

$$\kappa_1 = \frac{x_{d2}-x_{d1}}{x_{d1}r_1^{\ 2}-x_{d2}r_2^{\ 2}}$$

recalling

$$r^{2i} = (x_d^{\ 2}+y_d^{\ 2})^i$$

we have

$$\kappa_1 = \frac{x_{d2}-x_{d1}}{x_{d1}^{\ 3}+x_{d1}y_{d1}^{\ 2}-x_{d2}^{\ 3}-x_{d2}y_{d2}^{\ 2}}.$$

Eq. 5.6.3

Similarly we can find a point d_3 where d_1 projects onto the Y_B axis. Armed with Eq. 5.6.3 we can collect a set of results for κ_1 based on points on the grid in the recorded image. These are presented in the Chapter 9.

We will also examine the differences in values for κ_1 with different zoom and focus settings for the same lens.

5.6.4. Lens distortion with the projector

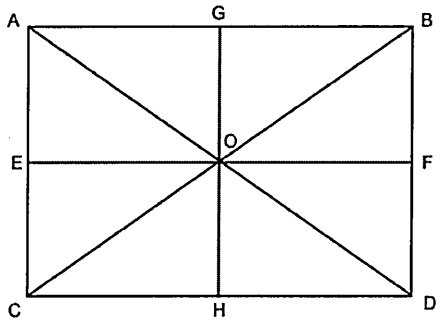

Figure 5.6.4: The projector calibration image.

In order to evaluate lens distortion for the projector lens, we project the calibration image onto a target plane, as shown in Figure 5.6.4. We adjust the position of the planar surface so that, when measuring by hand the distance between points A, B, C and D,

$$AB = CD,$$
$$AC = BD,$$
$$AD = BC.$$

For the LCD projector used in these investigations, we then measure the distance between points E and F and between points G and H, and find that

$$AB = CD = EF,$$
$$AC = BD = GH.$$

This shows that, for the projector being used, there is no radial distortion within the measurement system used here. (The manufacturers have corrected for distortion by applying inverse distortion to the output image).

5.6.5. Adjustment of System Positions

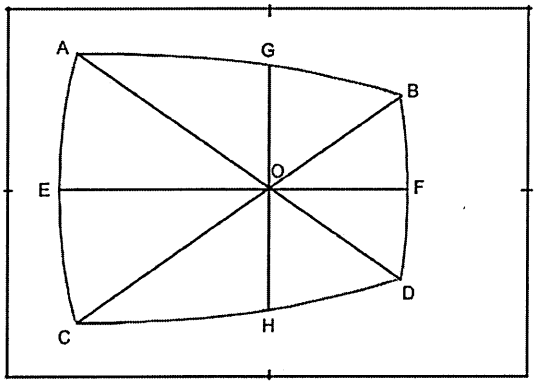

Figure 5.6.5: Calibration chart viewed in bitmap image.

In Section 5.2 we stated the following assumptions which would be made:

> "The camera and projector are positioned so that the camera and projector axes, and the middle rows of the camera and projector pixels, are all coplanar with the *X*-*Z* plane."

In order to adjust the system in line with these assumptions, the projection of Figure 5.6.4 is recorded in the camera (see Figure 5.6.5), which is positioned such that

line EF is centred on a single row of pixels,

line GH is centred on a single column of pixels, and

point O is at the central pixel in the bitmap image.

5.6.6. Evaluation of Constants *W, P, D_P, D_C* and *θ*

Referring to Figure 5.2.1a:

W, the spacing between stripes when the target plane coincides with the *X-Y* plane, is found by measurement of a field of stripes ideally extending over the whole width of the projected image. Then, if *I* is the width of the projected image and *N* is the number of stripes in the image

$$W = \frac{I}{N}$$

Eq. 5.6.6

D_P is the distance between the origin and L_P, the centre of the projector lens. It is found by

1. Measuring the size of the calibration image when the target plane is placed as in Section 5.2 at position $Z = 0$ units
2. Measuring the size of the calibration image when the target plane is placed as in Section 5.2 at position $Z = X$ units, where X is a known dimension,
3. By similar triangles evaluating D_P.

D_C is the distance between the origin and L_C, the centre of the camera lens. It is found by

1. Placing the calibration chart as in Section 5.2 at the origin.
2. Marking on the calibration chart the edge of the image frame as viewed through the camera.
3. Measuring the physical dimensions of the marked frame.
4. Moving the calibration chart X units closer to the canera lens.
5. Repeating steps 2 and 3.
6. By similar triangles evaluating D_C.

P is a constant associated with the size of a pixel, such that *PF* is the size of one square pixel where *F* is the focal length of the camera. Using the similar triangles for calculating D_C above, we can say that

$$\frac{D_C}{M} = \frac{HPF}{F}$$

Eq. 5.6.7

where H is the number of pixels in a complete row and M is the width of the marked frame on the calibration chart. This gives

$$P = \frac{D_C}{HM}$$

Eq. 5.6.8

θ is the angle between the camera axis OL_C and the projector axis OL_P. It is found by measuring the distance (say D) between the centre of the projector lens and the centre of the camera lens. The triangle with lengths D, D_P and D_C will then give the angle θ. **We will also formulate an expression to calculate θ using the ratios EO:OF shown in Figure 5.6.5.**

5.6.7. Ongoing adjustments during system use.

Constants W, P, D_P, D_C and θ will not normally be expected to change during operation. The area of movement is likely to be due to slight disorientation of the camera, and this will be checked prior to each scanning session by employing the adjustments described in Section 5.6.5, "Adjustment of System Positions".

5.7. Summary of Data Structures and Processes

This chapter has described the data structures which are used in the investigations to collect and process the data in a way which is the most useful for the proposed investigations. The scanning function $scan(h,v,n) = (x,y,z)$ is the key method which gives the location (x,y,z) of a point on the target surface. The location (h,v) of the point as seen in the recorded image is estimated to subpixel accuracy, and the stripe index n is found by the indexing methods.

The scanning function also includes constants in the form of intrinsic and extrinsic parameters, which are calculated by calibration, and methods for calibration are described in

this chapter. In particular the value of the Radial Coefficient is found, which adjusts the spatial position of points in the recorded image space.

In order to find *h, v,* and *n,* the following data structures and processes are used:

- pixel array `pixel[C][R]` where $l(c,r) \in [0,255] \subseteq \mathbb{N}$ gives the luminance of pixel (c,r),
- peak array `peak[C][R]` where $peak(c,r) \in \{\text{TRUE, FALSE}\}$ gives the peaks in the pixel array,
- index array `index[C][R]` where $index(c,r) \in \{n, \text{NULL}\}$ gives the index of each peak,
- the index array is reorganized as stripe array `stripe[N][R]` where $stripe(n,r) = c$ gives the column position of stripe n for row r,
- image point array `image_point[N][R]` where $image(n,v) = h$ and (h,v) is a subpixel adjustment of (c,r).

Thus *h, v* and *n* are found, and the scanning function $scan(h,v,n) = (x,y,z)$ then populates the array `surface_point[N][R][3]` which gives the (x,y,z) vertex at row r, stripe n.

Chapter 6: Geometric Constraints

6.1. Original Contributions

We take the methods using Epipolar Geometry and Rectification and adapt them to investigate the use of Dot and Stripe Patterns in Structured Light Scanning. In particular we introduce the ***Unique Horizontal Constraint*** and determine how far this can be successfully used to solve the Stripe Indexing Problem. Furthermore, we describe a method of rectifying the system by spatial positioning (Section 6.9).

There are two main uses of geometric constraints: firstly to assist in the calibration process and secondly to help in the investigation of indexing issues.

6.2. Using Geometric Constraints for System Calculations and Calibration

In Chapter 5, Section 5.2, parametric equations are given for the solution of the triple (x,y,z) representing a vertex on the target surface. Included in the spatial setup of the apparatus is the constraint that: "*the camera and projector are positioned so that the camera and projector axes, and the middle rows of the camera and projector pixels, are located on the X-Z plane*". This not only simplifies the parametric equations but also makes the calibration setup more manageable, because it is easier to align parts of the apparatus horizontally or vertically rather than at a specific angle.

However, this constraint does not help in solving the Indexing Problem. Figure 6.2.1 shows a detail from the recorded image where the system is set up without using what we will know to be the Epipolar Constraint. If the stripe which is marked as bright white is traced from the bottom of the image, when it leaves the head it is picked up again on the background wall at location "a". (We can guess where the stripe is picked up because the shadow matches the shape of the top edge of the head at that point). The stripe continues to the top of the image. Two locations on this stripe are denoted by "b" and "c", "b" being on the background wall and "c" being on the face of the object; and both locations are on the same pixel row.

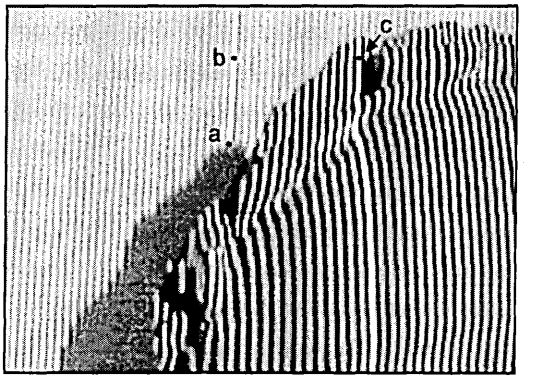

Figure 6.2.1: Detail from recorded image showing disconnected stripe.

Therefore it is possible for the same stripe to appear more than once along the same pixel row, and it is tempting to assume that, for any arbitrary setup of the apparatus, a particular pattern element can appear anywhere in the recorded image, depending on the depth of the target surface. In fact the pattern element will always appear somewhere along a curved line which is unique to each element, and it would be possible to find the equation of that line for each element in the projected pattern. However, finding the curved line for each pattern element would be an unwieldy process and it is possible to employ geometric constraints to simplify the task - the objective here being to reduce the possible locations in the recorded image of a particular pattern element. In fact the Epipolar Constraints described below ensure that each unique curved line is actually a straight line.

6.3. Epipolar Geometry and Rectification

The Epipolar Constraint is commonly used in Stereo Vision to reduce the problem of feature-matching from a two dimensional search of an area in the viewing plane, to a one dimensional search along a line [Faugeras, 1993] [Zhang, 1996]. In other words, once a feature has been found at a particular point in, say, the left image, the same location is then found in the right image and a search for the corresponding feature is undertaken horizontally from that location.

Figure 6.3.1 shows the two image planes of a stereo vision system, with a baseline intersecting the two focal points f_L and f_R. A point p in the scene is projected onto the image

planes at points i_R and i_L. An epipole is defined as the point at the intersection of an image plane with the baseline, which gives epipoles e_L and e_R in this system. (This assumes that the image planes are infinitely large, and that the figure shows only the rectangular parts which are visible through the lens). Furthermore any point, say i_R, on an image plane will define a plane in conjunction with the baseline. This plane, the epipolar plane, intersects the image planes giving two lines, the epipolar lines l_L and l_R; but as an epipole is also on the intersection of the image plane and the baseline, it follows that each epipole is on every epipolar line for that image plane.

In E^3 space we can therefore visualise an infinite set of epipolar planes, each radiating from the common baseline, and an infinite set of epipolar line pairs (l_L ,l_R), one left and one right, each radiating across the image plane from its epipole.

The importance of this is that any point in the scene will be projected onto an epipolar line in one image, and onto its coplanar epipolar line pair in the other image. Therefore if a left and right image point are not on coplanar epipolar lines, they cannot represent the same scene point. A function for the epipolar line at any one image point can be calculated if the system geometry is accurately known.

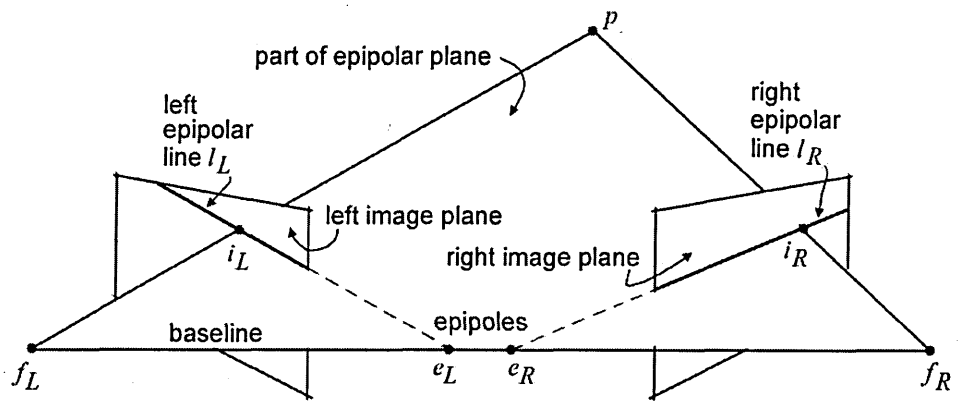

Figure 6.3.1: Epipolar geometry for stereo vision system.

To avoid the need to calculate the epipolar lines for every potential matched pair of image points, the recorded images can first be "rectified". In Figure 6.3.2 (top) we trace some of the epipolar lines in the pair of images; in Figure 6.3.2 (bottom) the images are transformed so that those epipolar line traces are now horizontal. This will give the effect that all the epipolar planes are parallel, stacked on top of each other, and therefore a scene point will appear

somewhere (shown at i_L and i_R) on the "same horizontal line" in both images; a much easier search. This one dimensional displacement we call the Horizontal Constraint.

The disadvantage of this rectification process is that, because it requires the processing of the digital image, some artefacts such as aliasing will occur. In addition, the resolution of the original recording will need to be reduced to fit the new shape in the old rectangular format. These problems will in turn reduce the accuracy of the final model. *In fact the reason for using "software rectification" in stereo vision methods is chiefly one of convenience - there are many library functions available in computer graphics resources which can perform this task. The computational complexity of transforming the image is not necessarily lesser than the computational complexity of calculating epipolar lines. Other work on the computation of epipolar rectification is detailed in Sections 2.5 and 2.9.*

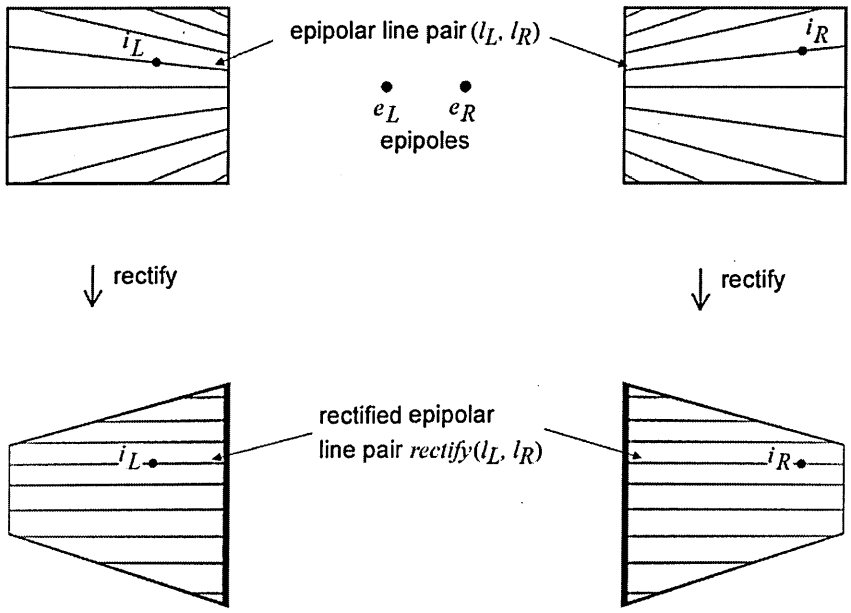

Figure 6.3.2: Rectification of stereo images.

6.4. Using Epipolar Geometry for Structured Light Systems

The Epipolar Constraint can be extended to deal with structured light systems, where one of the cameras is replaced by a projector (Figure 6.4.1). A point i_P on the projector image plane is

reflected from a surface point s on the target object and recorded on the camera image plane at point i_C. Using epipolar geometry, i_P will lie on an epipolar line, which is coplanar with the epipolar line in the camera image plane on which i_C must lie.

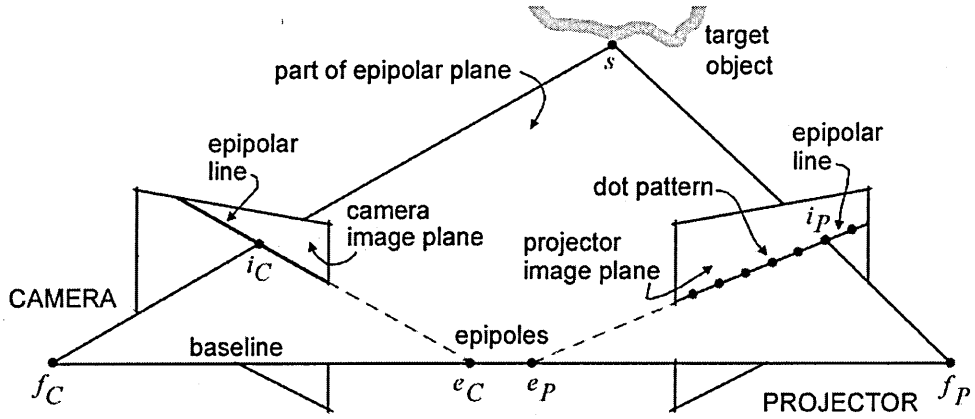

Figure 6.4.1: Epipolar geometry for a Structured Light System.

For this investigation, in order to compare the projected and recorded images, the camera and projector are positioned symmetrically as shown in Figure 6.4.1. This means that, as seen in Figure 6.4.2, the projector image and the reverse (i.e. reflected) camera image can be superimposed, and the epipolar line pairs will coincide.

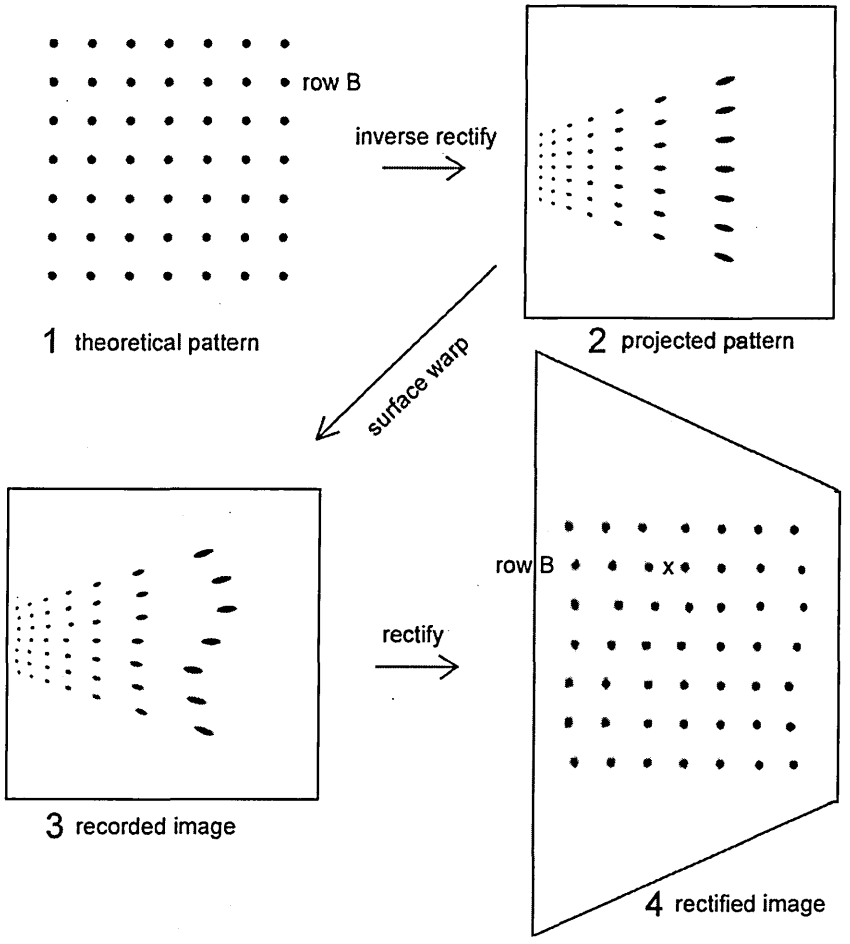

Figure 6.4.2: Projecting an inversely-rectified pattern.

Once again the system can be rectified, so that any point in the projected pattern will always be displaced horizontally in the recorded image. Figure 6.4.2 shows one method of applying the Epipolar Constraint and rectification to a horizontal pattern of dots. The theoretical pattern of dots (1) is transformed using an inverse of the rectification transformation to produce the pattern which will actually be projected (2). The inversely rectified dot pattern is then projected onto the target surface and recorded on the camera image plane (3). Finally, the recorded image is rectified (4), so that, as can be seen, the displacement of any dot by surface warping results in a one dimensional horizontal displacement in the rectified image. *Note that for ease of comparison between projected and recorded patterns, the recorded image(3) shows the projection onto the back of the camera image plane.*

By this method all the dots in one row in the theoretical pattern will also be in the same row in the rectified recording. In Figure 6.4.2(4) if a dot is visible at location X in row B, this could correspond to any of the dots in row B in the theoretical dot pattern (1). This means that with this method the Horizontal Constraint does not enable us to uniquely identify a particular dot, only to say that it lies on a particular line.

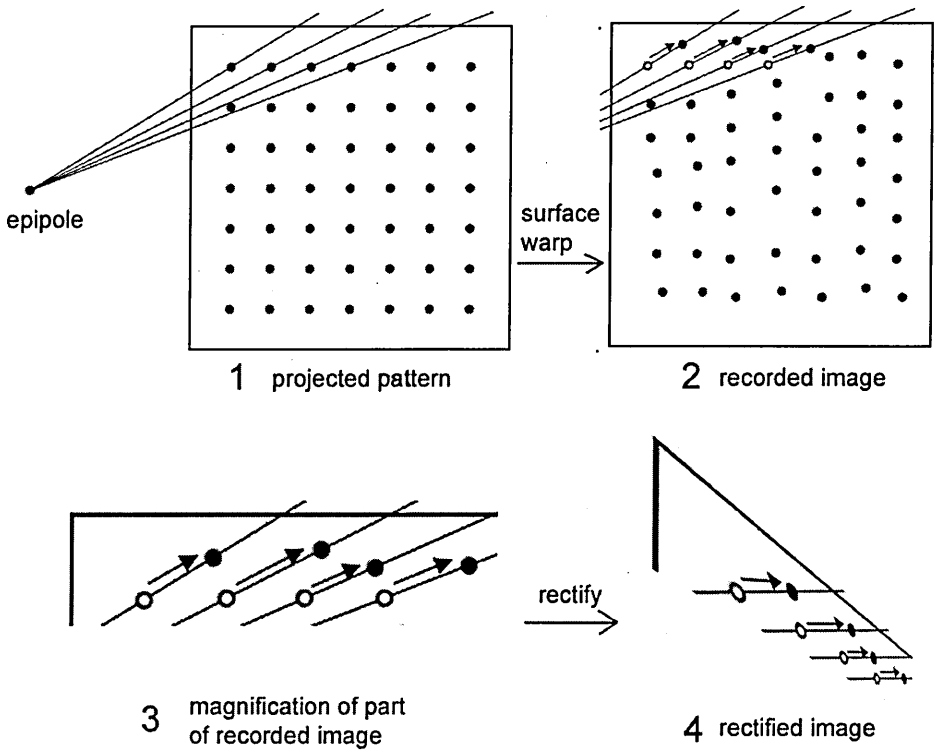

Figure 6.4.3: Projecting an orthogonal dot pattern.

Further implementations of the Epipolar Constraints are shown in Figure 6.4.3. (Again the recorded image refers to the back of the camera image plane, so that the epipolar lines are identically positioned in both the projected pattern and the recorded image). Here the dots are projected in an orthogonal pattern, without inverse rectification (1). Four epipolar lines are shown, intersecting the four leftmost dots on the top row of the pattern. Unlike the method shown in Figure 6.4.2, the epipolar lines will not coincide with the horizontal rows (apart from at the centre of the image).

The recorded image, showing the warped dot pattern, is shown at (2), and the four dots under consideration are displaced from their original positions in the dot pattern (shown as white discs) to their new positions in the recorded image (shown as black dots). The part of the image containing these four dots is magnified at (3), and shows that the arrowed displacement occurs along their respective epipolar lines. It is possible to arrange the dot pattern so that none of the epipolar lines coincide, so that each dot is assigned to a unique epipolar line in the recorded image. This is the Unique Constraint, and as every epipolar line can be predefined with an index and a parametric equation, it follows that in theory each dot in the recorded image can be correctly indexed.

If the recorded image is then rectified in the manner shown in Figures 6.4.2 and 6.3.2, after rectification (Figure 6.4.3 (4)) the horizontal displacement of each dot will be along different parallel lines. This would mean that each dot could be identified uniquely from its horizontal position in the recorded image. We call this the Unique Horizontal Constraint. However, this rectification would again introduce the problems of reduced resolution and aliasing artefacts.

6.5. Requirements for the Unique Horizontal Constraint

As described above, this constraint will assign a unique horizontal line to each dot in the projected pattern, and therefore the system will need to be able to discriminate one horizontal line for each dot in the pattern. This may be difficult to achieve, as the following discussion suggests.

Subpixel estimators (see Section 4.3.1) typically resolve projected patterns as seen in the recorded image to one eighth of a pixel. This means that it would be theoretically possible to consider eight horizontal lines *per* pixel, and this therefore gives a maximum number of dots in the projected pattern as eight times the available number of vertical pixels. A commercial CCD of, say, 1000 x 1000 pixels would then give 8000 dots which could be arranged, say, in a 100 x 80 grid. This is a small set of vertices for the size of the pixel array, when compared with a typical set for vertical stripes (see results in Chapter 9) of 144000 for a 768 x 576 array. To make the Unique Horizontal Constraint method worthwhile there would need to be a great improvement in indexing accuracy, which would conversely mean that other methods would need to be very poor at indexing.

A further problem with using an orthogonal pattern is that if there were a row of dots at the centre of the projector image plane, then the epipolar lines for that row will all coincide. Therefore it is important to ensure that no row is located at the centre of the pattern. Furthermore there may be alternative patterns which are more efficient at evenly spacing the epipolar lines, and a random patterning should also be considered, *but these investigations are outside the scope of this thesis.*

Practical systems may not need to be so rigorous in implementing the Unique Horizontal Constraint. Local uniqueness, where there are unique epipolar lines over a smaller group of dots, may suffice. However, because the accuracy of the dot location needs to be very high, the accuracy problems stated above when using software rectification may be an overriding factor. Therefore this investigation firstly looks at ways of implementing the Epipolar Constraint, where the epipolar lines are not parallel, for an orthogonal dot pattern.

6.6. Implementing the Epipolar Constraint for Orthogonal Dot Patterns

One method of implementing the Epipolar Constraint would be to firstly calculate a gradient for each epipolar line (which could be stored in a look-up table), and then to find the dot which intersects the line precisely - i.e. within a given distance. This task is fairly simple as the dot position could be easily found as a gradient with respect to the epipole, and the computational complexity would be in the order of $O(n^2)$ where n gives the number of dots in the pattern.

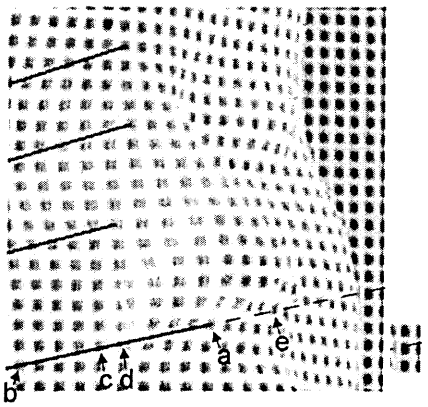

Figure 6.6.1: Finding epipolar lines corresponding to specific dots.

Figure 6.6.1 shows a recorded image: a part of the target surface viewed from the camera on which a grid of dots has been projected, without rectification, using a dot pattern. Four epipolar lines are superimposed on the image, and these lines connect to individual dots; the line connecting dot "a" is shown continuing as a dashed line. Visually it appears that the epipolar line is not unique to dot "a"; it also crosses close to the centre of dots "b" and "e" and possibly, depending on the accuracy of the system, other dots such as "c" and "d".

Tests were carried out to find the centres of these dots and the equation of the epipolar line, and Section 6.7 gives results to show that it is not possible to predict which of these five dots lies exactly on the epipolar line.

Despite this ambiguity, the constraint will still reject many of the dots as potential matches. However, the search will improve if ideas of vertex connections, whereby a prediction is made of the likely order of the dot pattern, are introduced into the design.

6.7. Combining Connection Issues with the Epipolar Constraint

In the previous section the usefulness of epipolar constraints was examined. A stage was reached where it is possible to find a group of dots which lie close to an epipolar line, but because of accuracy limitations it is not possible to single out one particular dot as being the epipolar point for a particular epipolar line. However, the five dots a, b, c, d and e of Figure 6.6.1 appear to be randomly ordered along the epipolar line; no account is taken of the likely order of the dots as defined by the projected pattern. To investigate this subject an alternative method has been designed, which uses the fact that two adjacent dots in the recorded image are likely to be adjacent in the projected pattern.

Figure 6.7.1 again shows, on the left, a part of the target surface viewed from the camera on which a grid of dots has been projected. It is assumed that a correspondence has been established whereby dot a_0 represents index n in the projected pattern. Recalling Section 5.1 where $scan(h,v,n) = (x,y,z)$, the task of the Indexing Problem is to find corresponding indices for most or all of the recorded dots.

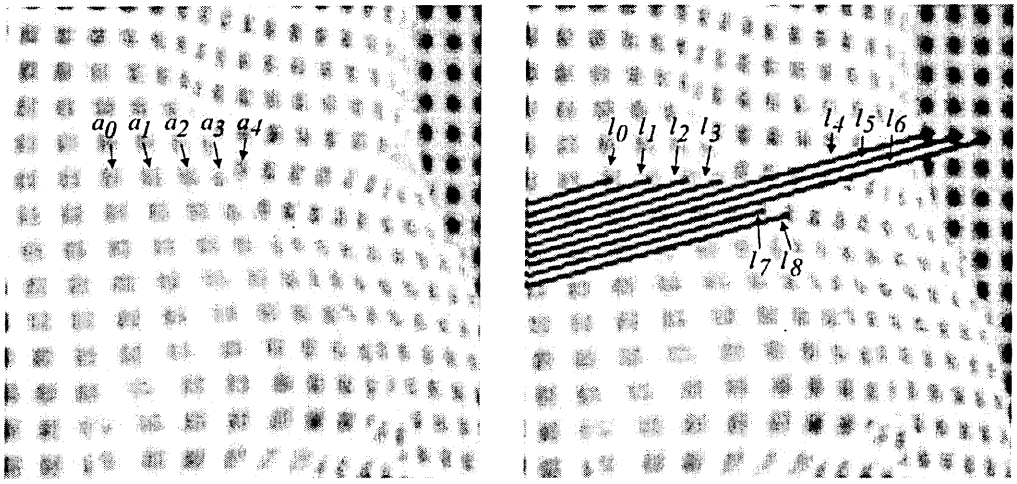

Figure 6.7.1: Theoretical positions of epipolar lines for specific dots.

Firstly, an intuitive attempt is made to solve the problem by eye. Starting from dot a_0 in Figure 6.7.1 (left), the dots a_1, a_2, and a_3 appear to follow along the row; but it is less obvious that dot a_4 continues in the same row. In Figure 6.7.1 (right) the epipolar line l_0 is drawn which connects to a_0. From there the subsequent lines l_1 to l_8 are drawn **based on their theoretical position in the projected pattern**. In other words, if a_1' is the dot in the projected pattern corresponding to dot a_1 in the recorded image, then the location and gradient of line l_1 in the recorded image will be identical to that of l_1' in the projected pattern, and this can be precalculated in a look-up table.

Starting from epipolar line l_0 the first four lines connect correctly to their dots, but l_4 misses a_4 completely. The connections for lines l_4, l_5 and l_6 are distant from the dots under consideration; this may be due to a miscorrespondence or may be a correct correspondence to a surface with a widely different depth. However, lines l_7 and l_8 connect with dots which are much closer to their "likely" position, depending upon how this "likeliness" is defined. Whether, on arrival at the disputed dot a_4, the search should then stop or carry on to test further consecutive epipolar lines as has been done here, is a design question for the indexing algorithms.

Implementing an algorithm based on the above intuitive method requires the following methods:

- recognizing dots in the recorded image,
- estimating the location of the centre of each dot,

- finding the nearest dot in the next row or column,
- finding the epipolar line for each dot,
- comparing that line with its corresponding epipolar line in the projected pattern.

6.7.1. Recognizing Dots in the Recorded Image

In its simplest form this is achieved by searching for a maximum (or minimum as shown in the figure which shows black dots for convenience) brightness level within a grid of pixels. If the grid size is approximately equal to the spacing of the dots, then this will usually find the brightest peak in each dot. A threshold level can be set by hand or automatically to ignore background noise, and a minimum amplitude between dots and "holes" will further reduce the effect of noise. These processes have previously been dealt with in the section describing the Single Stripe Scanner, in this case to find the centre of a stripe in the horizontal dimension.

6.7.2. Estimating the Location of the Centre of Each Dot

This uses the subpixel estimator as used by the Single Stripe Scanner, but a delta value is found in both the horizontal and vertical direction, so that the location of a dot with its maximum brightness at pixel (c,r) is given by (h,v), where $h = c+\delta c$, $v = r+\delta r$.

6.7.3. Finding the Nearest Dot in the Next Row or Column

This is a connection issue which will be dealt with fully in Chapter 7. For the present it is sufficient to find the next peak to the left, the right, above or below, within a given range of pixels.

6.7.4. Finding the Epipolar Line for Each Dot

The issue here is how to express the epipolar lines. As each line passes through the epipole, which is found by precalibration of the system, a suitable description is as a gradient from the epipole.

6.7.5. Comparing Corresponding Epipolar Lines

The gradients of epipolar lines for each projected pattern element n (that is, each dot) are stored either in a look-up table or as a function of their position in the pattern. This gradient is then compared with the gradient of the epipolar line of the recorded dot which is currently a candidate for matching.

6.8. An Algorithm for Dot Indexing

The dot indexing algorithm is implemented in these steps:

A. Starting with a square of d by d pixels in the vicinity (0,0) of a_0, where d is the spacing between the dots, find the brightest pixel:

$$[c_{max},r_{max}] = \arg\max_{c \text{ in } [0,d],\, r \text{ in } [0,d]} f(c,r).$$

B. Calculate the subpixel location (h,v) of the centre of the dot using the subpixel estimator.

C. Construct a rectangle $2d$ pixels wide by $0.5d$ pixels high, to the right of the current subpixel location, and find the brightest pixel:

$$[c_{max},r_{max}] = \arg\max_{c \text{ in } [h,\, h+2d],\, r \text{ in } [v,\, v+0.5d]} f(c,r).$$

D. Calculate the subpixel location as for step B.

E. Calculate the epipolar gradient for the new dot location.

F. Compare with the corresponding gradient in the look-up table.

G. If the two gradients are equal, within an error margin, repeat from step C.

H. Otherwise, search the look-up table for a near neighbour which has a similar gradient.

I. If a near neighbour is found, repeat from step C.

J. Otherwise, stop.

This process can be implemented with a more comprehensive search algorithm to give a wider cover.

6.9. Rectification by Spatial Positioning

To avoid the need to rectify the recorded image, it is possible to achieve the Horizontal Constraint by suitably positioning the camera in relation to the projector. This idea is suggested by [Faugeras, 1993], although no publication of this method has been found. Figure 6.9.1 shows the Structured Light System of Figure 6.4.1, but with the camera moved so that its image plane is parallel to the baseline. Recalling that the epipole is defined as the point of intersection of the image plane and the baseline, it follows that the epipole e_C must be at infinity. Therefore all the lines which radiate from the infinitely distant epipole e_C must be parallel to each other, and this includes ALL the epipolar lines in the camera image plane AND the baseline (which by definition also intersects e_C).

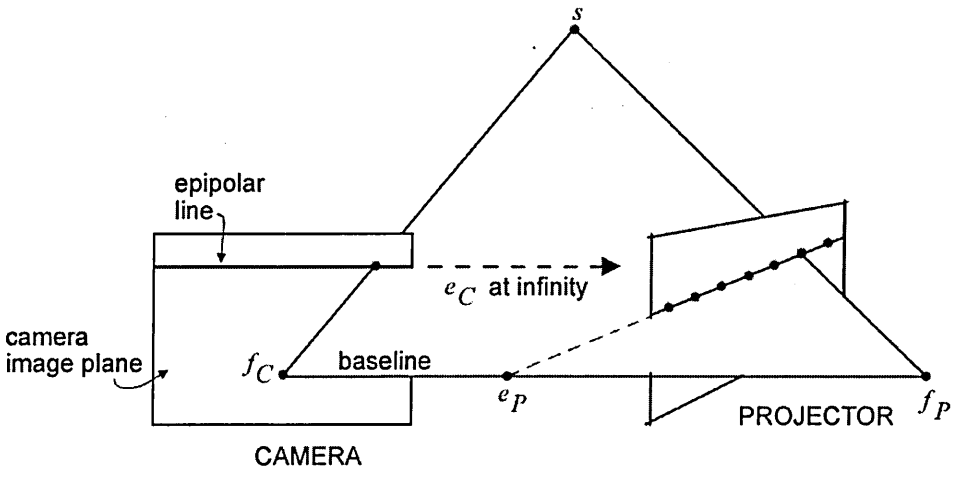

Figure 6.9.1: Rectification by spatial positioning.

The camera and projector are positioned so that these parallel lines are horizontal in relation to the camera image plane. Thus we have achieved the Horizontal Constraint, without the need for a computed rectification of the recorded image. This means that in Figure 6.9.1 all the dots shown on the epipolar line in the projector pattern will appear somewhere along the epipolar line shown in the camera image plane.

In theory this spatial rectification can be achieved simultaneously for both the camera and projector epipoles, in which cases the camera and projector image planes would be parallel. Assuming that the camera and projector axes were normal to their respective image planes, then the two axes would be parallel. The disadvantage of this configuration would be that because the camera and projector axes are parallel, the viewing volume may be restricted and could not be altered to suit a particular scanning apparatus without destroying the "simultaneous rectification".

The rectified configuration of Figure 6.9.1 can also employ the Unique Horizontal Constraint if a dot pattern is designed which gives uniquely positioned epipolar lines in the projector plane. As stated above, the only advantage of the Horizontal Constraint and the Unique Horizontal Constraint is to avoid the need for a look-up table of precalculated epipolar gradients, and the disadvantage, even now that software rectification is not necessary, is that a dramatically reduced set of vertices will be produced (see Section 6.5).

We will now examine the use of stripe patterns and then make comparisons with the above issues raised by dot patterns.

6.10. Using Stripe Patterns

So far the projection patterns have taken the form of dots in different arrangements. Dots have a two dimensional location in the image space, and therefore can be used to create the Unique Horizontal Constraint. However, as we have seen this constraint requires a high resolution for a small number of pattern elements; practical algorithms may abandon this in favour of methods requiring only the Horizontal Constraint, which allows for more pattern elements to be included in a single image. If a pattern of vertical stripes is used the vertical position of the elements is lost, but this does not invalidate the Horizontal Constraint, which can still be achieved using rectification or spatial positioning.

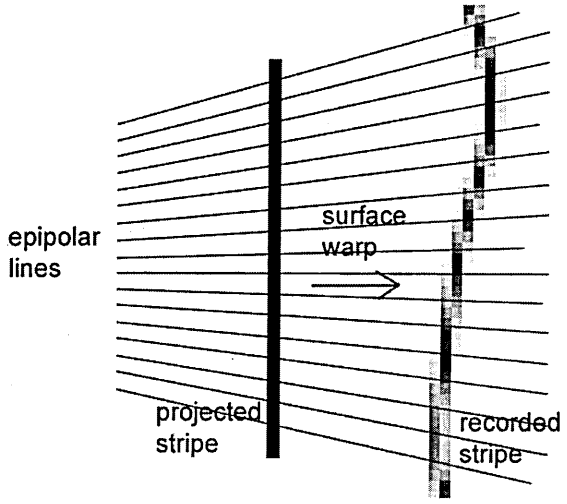

Figure 6.10.1: Projection of vertical stripe.

Figure 6.10.1 shows a vertical projected stripe as viewed on the projector plane and its warped, digitised transformation as viewed on the reverse side of the camera plane. Referring back to Figure 6.4.1, showing the reverse side of the camera plane enables the two images to be superimposed, in which case the projector epipolar lines and their corresponding camera epipolar lines will coincide, as shown.

The advantage of using a continuous pattern such as a stripe rather than discrete dots is that the stripe crosses every pixel row in the recorded image. In other words, the vertical sample size is equal to the number of pixel rows in the recorded image, not the number of rows of dots in the pattern. Therefore the density of samples is much higher for continuous stripes. The disadvantage is that the Unique Horizontal Constraint cannot be applied, because we cannot define epipolar lines in the projected pattern as we do not know the vertical position of any pattern elements. (In fact, because the practical projector is a Liquid Crystal Display which is made up of individual pixels, the projected pattern elements do exist, but when considering vertical stripes it is assumed that they are lost in the acquisition process).

However, these epipolar constraints are still useful with vertical stripe patterns. Figure 6.10.2 shows four vertical stripes in a recorded image which has been rectified, either by software or spatially. The stripes are broken by an occlusion, but because the Horizontal Constraint is valid, each of the stripes must continue horizontally, as shown in the figure. This principle will be used in algorithms in Section 7.10 to help solve the Indexing Problem.

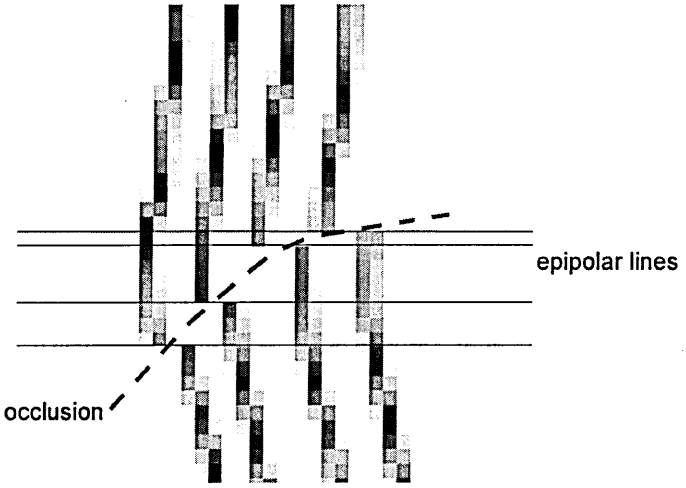

Figure 6.10.2: Stripes "broken" by occlusion.

6.11. Comparison between Dot and Stripe Patterns

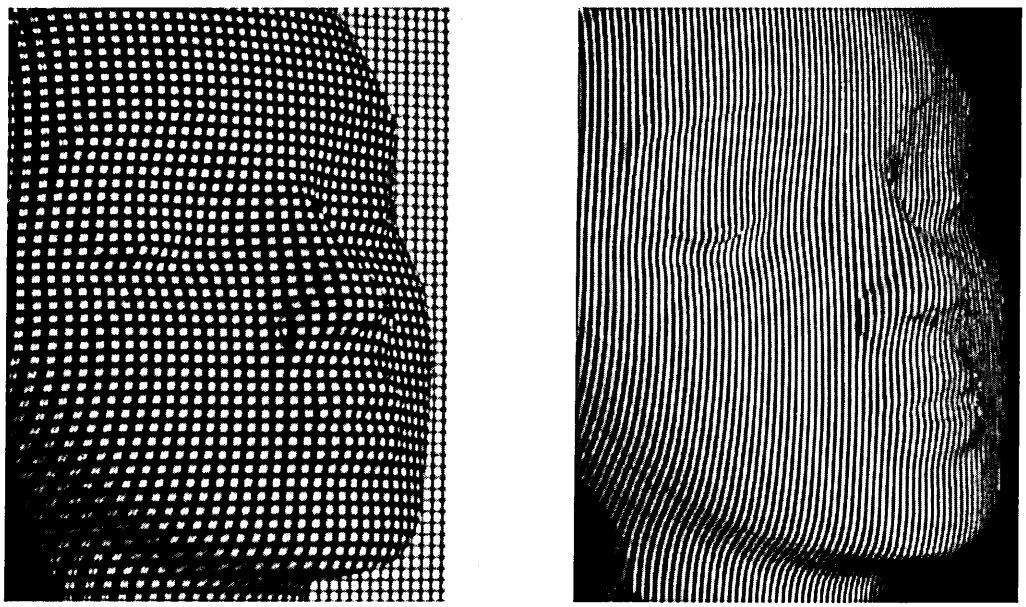

Figure 6.11.2: Dot and stripe projections.

Figure 6.11.2 shows a target object with, on the left, a projected dot pattern and, on the right, a projected stripe pattern. The horizontal spacing of the dots is twice the spacing of the stripes,

and vertically the spacing between the dots is 10 pixels. This means that per unit area in the recorded image there will be 10 x 2 white stripe elements for every dot element, and if the black stripes are indexed as well as the white stripes, there will be 10 x 2 x 2 = 40 stripe elements for every dot element. Attempts to increase the density of the dot pattern have so far failed to give good results; methods of indexing using stripes are the subject of further investigation in Chapter 7.

6.12. Summary of Investigation into Geometric Constraints

The investigations in this chapter can be categorized as follows:

- **System calculations and calibration.** Alignment of the camera and projector in the same plane simplifies calibration and parametric calculations.

- **Epipolar configuration.** Ensures that all points on one epipolar line in the projector image plane are on the same epipolar line in the camera image plane. This is the Epipolar Constraint.

    - **Software rectification.** Adds to the epipolar configuration the constraint that all epipolar lines in the camera image are horizontal. This gives the Horizontal Constraint, whereby a projected pattern element must be found on a definable horizontal line in the recorded image. The disadvantage is that software rectification reduces the resolution of the image.

    - **Pattern rectification.** The orthogonal dot pattern is inversely rectified before it is projected, so that again the Horizontal Constraint is enforced. In addition, whichever row a dot is on in the prerectified image , it will be on the same row in the recorded image.

    - **Spatial rectification.** Achieves the same result as software rectification, but without the loss of resolution.

- **Unique configuration.** This is a special case of the Epipolar Constraint which is only applicable to horizontally and vertically discrete elements such as dots. Each pattern element is assigned a unique definable epipolar line in the recorded image. Thereby if the position of the dot can be found accurately enough, its index will explicitly follow. This is the Unique Constraint. The disadvantage of this configuration is that the accuracy of distinction between epipolar lines must be very high, severely reducing the size of the set of vertices.

    - **Unique horizontal configuration.** If the recorded image is rectified, then the epipolar lines in the recorded image will all be horizontal, so that every dot will occupy a unique horizontal position in the recorded image. This is the Unique Horizontal Constraint. The added disadvantage is that software rectification will further reduce the system resolution.

From this investigation it has been decided, by a combination of logical inference and analysis of the test results, that the following methods will be investigated further:

- spatial rectification, on the grounds that the Horizontal Constraint will be a significant help in the searching algorithms, and that the accuracy problems associated with software rectification will be avoided,

- horizontal configuration using stripes, which cannot employ the Unique Constraint.

For this thesis no further investigations will be made into the use of dot patterns with the Unique Horizontal Constraint, as the resolution obtained by the dot pattern is so low (typically one fortieth in Figure 6.11.2) compared with the stripe pattern.

Chapter 7: Indexing Definitions and Algorithms

7.1. Original Contributions

In this chapter we make numerous contributions to the solution of the Indexing Problem. In the peak array, ***N-, S-, E- and W-Neighbours*** are defined, leading to definitions of ***Stripe Paths*** and ***Neighbourhoods***. These types form the basis of descriptions of ***Patches***, and are used as ***Patching Conditions*** in the novel algorithms which have been designed to successfully index the stripes.

These novel indexing algorithms comprise ***Scanline***, ***Scan and Search*** and ***Flood Fill*** methods, which adapt and extend existing search methods used in Pattern Recognition and Segmentation applications. Here we also pose the ***Size v. Accuracy Question***: "For a given scanning application, what is the relative importance of indexed patch size compared with indexing accuracy", and this question must be addressed by the application designer.

7.2. Introduction

Stripe indexing requires the assignment of the correct index to every part of every stripe, as seen in the recorded image. If the pattern is carefully coded, such as by colour or width, then a scanline search is usually sufficient to extract the correct index. However, as discussed in the Introduction, this thesis investigates how far it is possible to solve the Indexing Problem using uncoded light patterns, which will require more thorough pattern recognition techniques. In this chapter we start by implementing scanline and other simple recognition methods, and then extend these algorithms to search in other ways through the recorded image data. In particular, we introduce definitions and categories of connections and connected areas, and finish the chapter with some thoughts on the problems of occlusions, and the importance of geometric measurements.

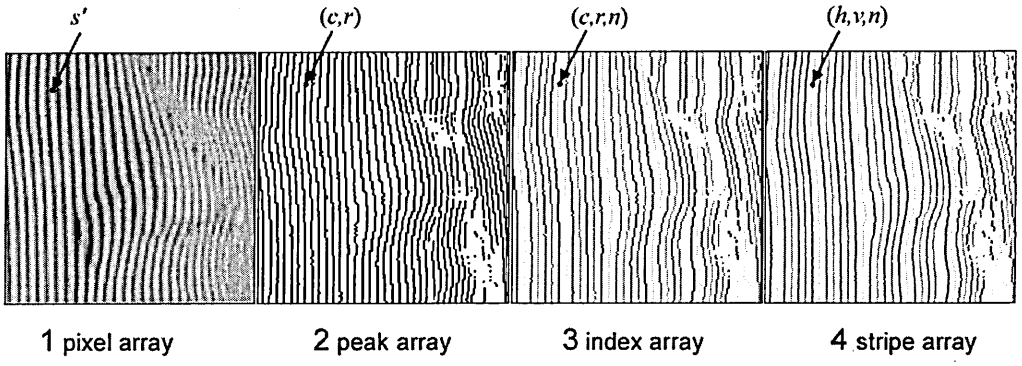

Figure 7.2.1: Processing the recorded image.

Figure 7.2.1 shows, at (1), the pixel array representing part of a typical recorded image, showing the deformed pattern of vertical stripes. As described in Section 5.1, in order to measure the target surface at a point $s=(x,y,z)$ the location $s'=(h,v)$ of its projection onto the image plane must be calculated, together with the stripe index n at that point. Figure 7.2.1 also shows the process of calculating h, v and n: firstly the peaks (c,r) at the centre of each stripe for each row are found (2), then the peaks are given their stripe index n (3) and thirdly the subpixel estimator and lens distortion function calculate the more exact locations (h,v), together with their index n (4). Note that the peaks in (2) have not necessarily been found over the whole image, due to constraints imposed if the stripes are considered to be indistinct, and note also that the indexing (3) is incomplete; a carefully study will show incorrect indexing in the right hand part of the image.

The indexing process may use either the discrete peak array or the (almost) continuous (h,v) array, or a combination of both. Initially the indexing methods will use the discrete peak array, in other words the index array (3) will be derived from the peak array (2); consideration of the usefulness of the (h,v) array will follow later in this chapter.

7.3. Searching and Scanning Approaches

Three initial approaches are taken in order to solve the Indexing Problem.

- Looking again at Figure 7.2.1 (1), we first consider the way in which the stripes would be indexed intuitively by eye. The obvious method would be to search for a **connection** along one stripe until some kind of **stopping boundary** occurs, and then look for **adjacent stripes**, either left or right, and increase or decrease the index number appropriately. In the Figure 7.2.1 example, this approach is likely to work well until the area around what we know to be the nose is reached, at which point the stripes become less distinct, and the route of the stripes becomes **ambiguous**. Fortunately in this image there is a possibility of correctly deducing the indices by using the top row as a starting point, where the stripes "appear" to follow in the correct order. However, it is not immediately clear what logic is used to decide that the top row has this validity.

- The most obvious mechanical indexing method is to scan along each row in turn, starting from say the bottom left, indexing each peak as it is met, and then incrementing the index. The index would be reset at the start of the next row. The downfall of this method can be seen in Figure 7.2.1, where the leftmost stripe disappears out of frame for part of its length, and there is also a hint of another stripe near the bottom. This means that, for instance, the first "complete" stripe on the left would have a changing index along its length.

- However, a possible compromise can be easily seen: if firstly the leftmost complete stripe is correctly identified, and the scanline indexing continues from there, then most of the stripes will be correctly indexed.

We will call these three approaches the *search* method, the *scanline* method and the *scan and search* method. To investigate further we must implement suitable searching algorithms, and this requires an understanding of the concepts described above in boldface: connections, stopping boundaries, adjacent stripes and ambiguity.

7.4. Graph Definitions

At this stage reference is made to graph theory [Bondy and Murty, 1976], and the definitions of *connectivity, connected graphs, adjacent vertices* and *paths.*

Definition 7.4.1 *A connected graph is one in which there is a path from any vertex to any other vertex.*

Definition 7.4.2 *Vertex connectivity is the minimum number of vertices whose deletion from the graph disconnects the graph.*

Definition 7.4.3 *Adjacent vertices are two vertices which have an edge between them.*

The idea of ***adjacency*** will be used to describe whether or not vertices are adjacent. The distinction between "connected" and "adjacent" is important: the former requires a connection between every vertex in the graph, whereas the latter requires only a connection between the two vertices under consideration. Continuing the ideas of connectedness and adjacency we define a *path* and a *connected path.*

Definition 7.4.4 *A path is a sequence of successively adjacent vertices.*

Definition 7.4.5 *A connected path is a path which visits every vertex in the graph.*

The representation of the recorded image data as a graph is confusing as there is a sense of "adjacency" between the pixels, and another sense of "adjacency" between the peaks. However, once the stripe indexing has been established, the mesh of vertices can be represented as a graph in both the two dimensional image plane and the three dimensional surface model. Therefore care will be taken not to misappropriate the graph terms in this chapter.

7.5. Definitions of Neighbours

The concept of direct (d-)neighbours and indirect (i-)neighbours is generally used in segmentation algorithms to search around a pixel array [Pavlidis, 1982], so that the connection can be made between neighbouring pixels. In this thesis we extend this concept of connections to include pixels which are not necessarily d- or i-neighbours, but may be n-, s-, e- or w-neighbours. Thus paths may cross over and ignore some d- and i-neighbours, to search and find areas which have a valid connection.

The raw data for this study is the peak array $C \times R$ obtained from the pixels in the recorded image. Recalling Figure 5.5.1, Chapter 5, this array uses the luminance $l(c,r)$ of a pixel to give local peaks along each row r, such that $l(c\text{-}1,r) \leq l(c,r) > l(c+1,r) \Rightarrow peak(c,r) = \text{TRUE}$; otherwise $peak(c,r) = \text{FALSE}$.

Figure 7.5.1 shows part of the $C \times R$ array in which pixels are marked as TRUE if they are peaks (coloured grey). The array can be considered as a square tessellation of pixels, each pixel representing a square with four corners and four sides. For these initial definitions we make two important assumptions:

- firstly that because the vertical stripes are sampled on every row there will be a peak for each visible stripe on each row, and

- secondly that the Horizontal Constraint (see Section 6.3) is enforced whereby a stripe can only exist, at most, at one location on each row.

Definition 7.5.1 *Any two pixels are called direct neighbours (d-neighbours) if they share a side, and indirect neighbours (i-neighbours) if they share one and only one corner.*

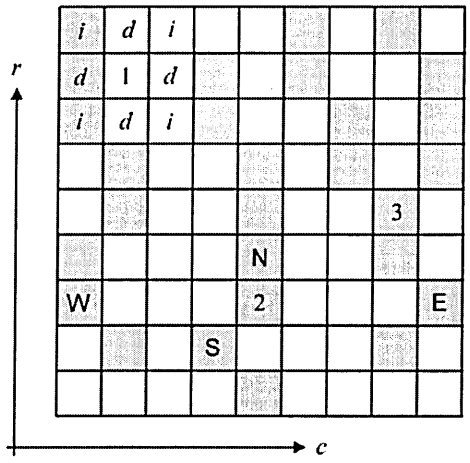

Figure 7.5.1: Moving around the peaks data.

As all pixels in the captured image are situated in a two dimensional Euclidean plane, then each pixel may have maximally eight neighbours (four direct and four indirect) as seen in Figure 7.5.1, at position 1. Now we make definitions referring specifically to the peaks in the array, and

these will be called the ***peak neighbours***, comprising ***n-neighbours***, ***s-neighbours***, ***w-neighbours*** and ***e-neighbours***.

Definition 7.5.2 *A northern neighbour (n-neighbour) of a current peak is defined as a d- or i-neighbour in the row incremented by one. A southern neighbour (s-neighbour) of a current peak is defined as a d- or i-neighbour in the row decremented by one.*

Definition 7.5.3 *A western neighbour (w-neighbour) is defined as the nearest peak in the same row which has a lesser column index. An eastern neighbour (e-neighbour) is defined as the nearest peak in the same row which has a greater column index.*

The definition of n- and s-neighbours assumes that the stripe cannot move laterally by more than one pixel as it moves up or down, i.e. that the "slope" of the stripe is never less than |45|°. Figure 7.5.1 shows, relative to peak 2, its n-neighbour N, its s-neighbour S, its w-neighbour W and its e-neighbour E. Also, at this stage no limit is given for the "nearness" of two w- and e-neighbours; in other words, the maximum distance between stripes is not defined.

7.6. Stripe Paths and Neighbourhoods

We now formally define a ***stripe path*** and a ***neighbourhood***.

Definition 7.6.1 *A stripe path is a sequence of peaks* $\{p_1, p_2, \ldots, p_n\}$ *such that each peak in the sequence* $\{p_1, p_2, \ldots, p_{n-1}\}$ *has one and only one northern neighbour.*
Consequently sequence $\{p_2, p_3, \ldots, p_n\}$ has one and only one southern neighbour.

Definition 7.6.2 *A neighbourhood is a set of peaks each of which has at least one peak neighbour in the set.*

In graph theory terms a neighbourhood is therefore a *connected subgraph* of the graph of all peaks.

7.7. Boundaries

The idea of a neighbourhood implies that there is a boundary around the neighbourhood, beyond which none of the pixels are peak neighbours. From Definitions 7.2 and 7.3 there are four peak neighbour types: n-neighbours, s-neighbours, e-neighbours and w-neighbours. A peak which has **all four** peak neighbour **types** we call a *full neighbour*, and a peak which has **between one and three** neighbour **types** we call a *boundary neighbour*. Therefore in Figure 7.9.1, which repeats the array of Figure 7.5.1, peak 2 is a full neighbour and peaks 4, 5 and 6 are boundary neighbours. Peak 3 is an interesting case; it has four neighbours but only three types, as there are two n-neighbours. If the Horizontal Constraint is enforced, whereby each stripe can only be in one location along any row, then there cannot be two valid n-neighbours. This is an example of *ambiguity*, and the indexing algorithm will have to decide which of the two possible *connections* is correct. If one were chosen, then by implication the other n-neighbour would become "disconnected".

These ideas on boundaries are summarised by stating that the ***boundary of the neighbourhood is the set of all boundary neighbours***, that the ***open neighbourhood is the set of all full neighbours***, and that the ***closed neighbourhood is the set of all full neighbours and boundary neighbours***.

7.8. Connections and Patches

The preceding paragraph introduces the idea of a "connection", which implies a stronger link between two peaks than mere neighbourliness, and that a full neighbour within a neighbourhood may be "disconnected" from its neighbours. This in turn implies that there may be a smaller area within the neighbourhood in which all neighbours are "connected" under the terms of the imposed conditions. This smaller, more constrained area we call a **patch**. To avoid confusion with the graph theory sense of "connected" (see Definition 7.4.1) which requires the connection of **every** vertex in the graph, if two neighbours comply with the imposed **Patching Conditions** they are said to be **patched**. No formal definitions will be made of patches and Patching Conditions, as these concepts are subject to change during the investigations, but their general form will be as outlined here.

7.9. Summary of Connection Definitions

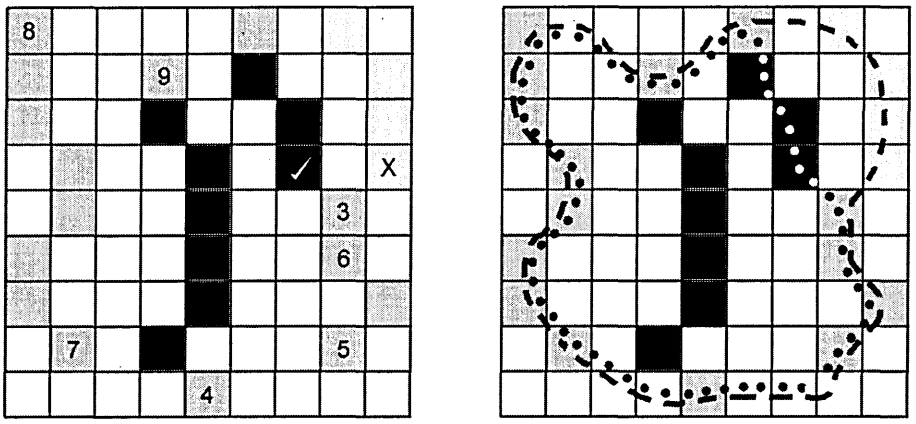

Figure 7.9.1: Illustration of patches, boundaries and neighbourhoods.

Figure 7.9.1 illustrates the formal and informal definitions made in this section, using the pixel array of Figure 7.5.1. It is assumed that the Patching Conditions have determined that peak 3 connects to its left-hand n-neighbour (marked with a tick) and that its right-hand n-neighbour is therefore disconnected (marked with a cross). The types illustrated are:

- full neighbours (coloured black) which have a full complement of n-, s-, e- and w-peak neighbour types,
- boundary neighbours (dark grey and light grey) which have one, two or three peak neighbour types,
- a closed neighbourhood (bounded by the dashed line), containing all the full and boundary neighbours,
- an open neighbourhood containing only the full neighbours (coloured black),
- the unpatched part of the neighbourhood (light grey),

- the patch (bounded by the dotted line) of neighbours which fulfil the Patching Conditions, and

- stripe paths between peaks 7 and 8, peaks 4 and 9, and peaks 5 and 6. Note that this last stripe path does not include peak 3, which disobeys the "one and only one northern neighbour" rule in Definition 7.6.1.

It is also assumed that the "starting point" for the connections is somewhere within the patch; if the starting point is in the top right corner of the array then the patch will comprise the four light grey peaks.

The distinction between the larger *neighbourhood* containing all the peak neighbours, and the smaller *patch* containing only the patched peaks, can be clearly seen. The patch has a greater "validity" due to its constraint by the Patching Conditions, and therefore there should be more confidence in the measurement data produced. One of the main tasks of the indexing algorithms is to devise the Patching Conditions so that the validity of any patch is as high as possible.

We now have formal and general definitions of neighbours, peaks, stripe paths, patches and boundaries, and these will be used to inform the stripe indexing algorithms. Referring back to Section 7.3, these algorithms are categorized as *scanline* methods, *search* methods, and *scan and search* methods.

7.10. Scanline Methods

Referring to Section 5.5.3, the task of the stripe indexing algorithms is to label each location in the peak array with its correct index or NULL if there is no peak at that location (*i. e. peak*(*c,r*) = FALSE).

The basic Scanline Method is:

```
1.   for(r = 1 to R) {
2.        n = 1;
3.        for(c = 1 to C) {
```

```
4.              if(peak[c][r]) {
5.                   index[c][r] = n;
6.                   increment n;
7.              }
8.              else
9.                   index[c][r] = NULL;
10.        }
11.   }
12.   return;
```

The indexing is performed along each row in turn, resetting the stripe index at the start of the next row.

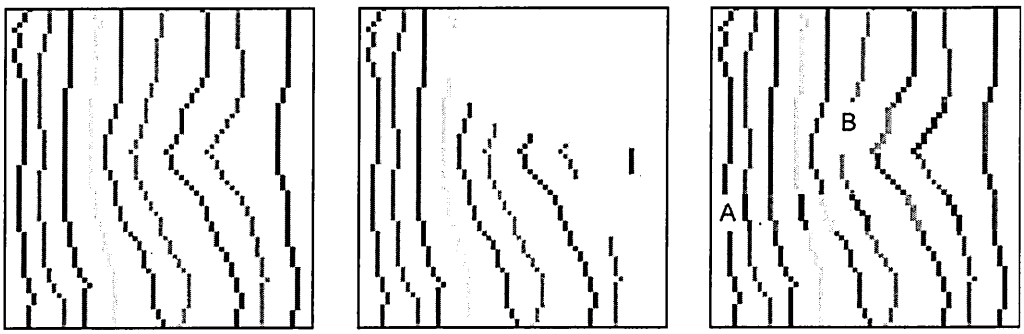

Figure 7.10.1: Indexing using Scanline methods.

Ideally, for this method to work correctly, all the stripe paths (Definition 7.4) should extend over the full range of rows, so that each stripe would contain the sequence of peaks $\{p_1, p_2, \ldots, p_R\}$ as shown in Figure 7.10.1 (left). If not then the method will still work if, as each row is scanned, there are no peaks subsequent to a missing peak (Figure 7.10.1, centre). Otherwise, if say there are parts of the first stripe missing as at A, or there is a "hole" in the middle of the patch as at B, then the indexing will be incorrect from those points onward (Figure 7.10.1, right).

In practice the requirement for full length stripes may be difficult to attain. Not only would the target surface have to be free from holes, occlusions and non-diffuse reflection, but also the field of vision of the camera would need to ensure that no stripe disappeared "off the sides" of the recorded image. This thesis takes the scanline method as a starting point to investigate methods of indexing where some of these problems do occur.

7.11. Patching Constraints

An intuitive investigation of Figure 7.10.1 (right) suggests a simple improvement to avoid the incorrect indexing: the scan should stop as soon as the stripe path is broken because of the missing northern neighbour for the leftmost stripe, at location A. The Scanline Method is modified to ensure that the current peak has a northern neighbour by testing for a TRUE peak at (c-1,r+1), (c,r+1) and (c+1,r+1). A new line is inserted after line 4, and we call this line 4b:

```
if(NOT(peak[c-1][r+1] OR peak[c][r+1] OR peak[c+1][r+1])) return;
```

which ensures that the method stops immediately if there is no northern neighbour above any peak. In Figure 7.10.1 (left) this would have the effect of only indexing the rows below area A, but the advantage would be that all the calculated indices would be correct.

This new line 4b can be thought of as a **Patching Constraint,** which has the effect of **reducing the indexed patch size but increasing the indexing accuracy**. Different scanning applications will require different balances between patch size and indexing accuracy, and this issue is formulated as the **Size *v.* Accuracy Question**:

The Size v. Accuracy Question. *For a given scanning application, what is the relative importance of indexed patch size compared with indexing accuracy?*

7.12. Scan and Search Methods

The example of Figure 7.10.1 (right) when indexed using the Scanline Method, produces an obviously false indexing, with the index changing along the length of what we can see to be a single stripe path. This strongly suggests that a search along each stripe path, in turn, would be appropriate. This is achieved by the Scan and Search Method which scans for successive peaks along a starting row, and then at each peak on that row searches up and down to mark the stripe path. The method uses the following steps, which are illustrated in Figure 7.12.1 (left):

1. Find a starting peak. Set the stripe index to zero. The start point can be found by hand or automatically.

2. Trace the current stripe northwards until a boundary (i.e. a missing n-neighbour) is met.

3. Return to start and trace the current stripe southwards until a boundary (i.e. a missing s-neighbour) is met.

4. Increase the stripe index, move to the next peak eastwards and repeat 2 and 3.

5. Repeat 4 until boundary (i.e. a missing e-neighbour) is met.

6. Return to start and repeat 4 and 5 moving westwards, decreasing the stripe index each time.

7. Renumber the stripes to start at one.

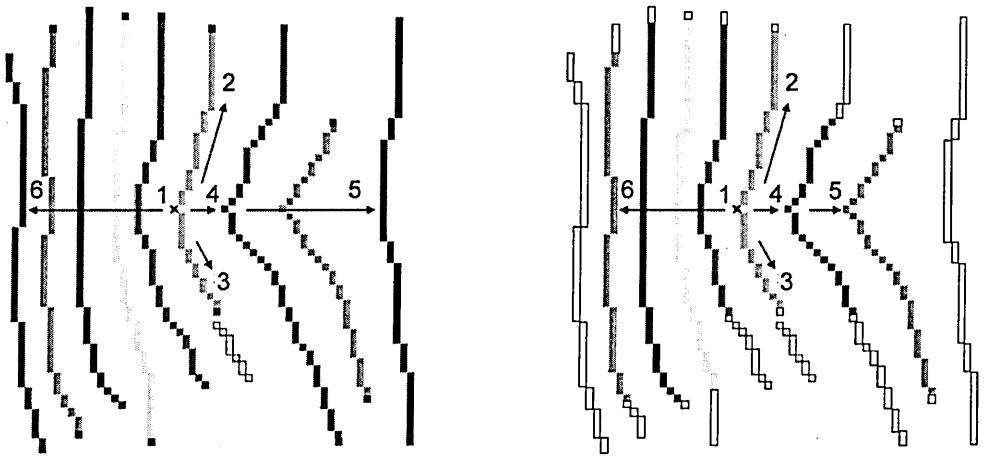

Figure 7.12.1: Indexing using Scan and Search methods.

In Figure 7.12.1 the numbers refer to the algorithm steps, and the stripe indices are given arbitrary colours.

One effect of this process is that once a boundary has been reached, the stripe will not be "picked up" again, as can be seen from the uncoloured part below the number "3". Therefore the indexed patch size will be reduced, and whether this is a suitable result is again a subject of the Size *v.* Accuracy Question.

Also in this method it can be seen how much the indexed patch size will vary, depending on how the Patching Conditions are defined. By allowing the search to flow along the stripe until a missing northern or southern neighbour is met (black peaks), the indexed patch size is as shown in Figure 7.12.1 (left). However the search will stop much more quickly if the boundary is *any* Boundary Neighbour (see Figure 7.12.1, right) and consequently the indexed patch will be much smaller. Here the Full Neighbours (apart from those below the number "3") are indexed, and the Boundary Neighbours are not indexed.

7.13. Live Boundaries *versus* Premarked Boundaries

The new line 4b in the Scanning Method acts as a "boundary tester", detecting unpatched peaks, and at the same time stopping the method at that point. The boundary is therefore detected simultaneously with the other actions, and this we call a ***live*** boundary.

In the Scan and Search Methods it may be more appropriate to employ an initial "pass" through the pixel array simply to mark the boundaries, and then to run the indexing method using these ***premarked*** boundaries as stopping conditions. Certainly this will be the case in the second example, shown at Figure 7.12.1 (right), where the boundary test to find Full and Boundary Neighbours is more complex than simply finding a n-, s- e- or w- boundary.

The decision to premark the boundaries is here taken because of the complexity of the boundary test, but in some situations live boundary finding will produce different results from premarking. In this example the results will not be affected if a pre-marked boundary is used rather than a live one, but in later methods this issue will be important.

7.14. Resolving the Size *v.* Accuracy Question

How can a decision be made which will resolve the Size *v.* Accuracy Question? It is a simple task to run each method and count the total indexed peaks, and then compare the results for each method. However, although this will allow a comparison of the indexed patch size it will not indicate the accuracy of the method, that is whether an individual peak has been correctly indexed.

In the results (Section 9.5.1) the accuracy of the indexing method is calculated with reference to a "template", a similar surface which has been indexed by hand with prior

knowledge of the surface. Each calculated peak index is compared with its known counterpart in the template image. This is a "black box" approach, whereby an accuracy measure is achieved without any understanding of what constraints in the algorithm design have contributed to the indexing results. Alternatively, a "white box" approach is to look for logical reasons for the imposition of a Patching Condition based on, say, a physical characteristic of the surface such as a hole or occlusion. These questions are generalised as:

The White Box Question. *Is there a straightforward logical reason for the inclusion of this Patching Condition?*

The Black Box Question. *Do the test results show that the conditions under consideration will produce more accurate indexing?*

Ideally the design should start with the first question, and the second should follow as a validation of the first or, *in extremis*, as a desperate search for more usable constraints.

Posing these two questions with respect to Figure 7.12.1, there appears to be a strong logical, physical case for ignoring the stripe below the number "3": it will be shown in proceeding sections that missing stripe portions are often the result of occlusions and other surface characteristics of the target object. However, there is no such logical case for not indexing the other peaks in the right diagram of Figure 7.12.1; the decision appears to be purely arbitrary. Therefore the second, "black box", question is posed, and only a strong correspondence between indexing accuracy in the test data output would convince the designer to abandon all the peaks.

7.15. Search methods

Pursuing the goal of indexing as many peaks as possible requires a more mobile search than is provided by the Scanning Methods, which stop as soon as a boundary is found in a horizontal or vertical direction.

A useful analogy is with computer "Paint" programs, which attempt to fill a bounded area with a colour. If a scanline algorithm is used many of the nooks and crannies will be missed, and to overcome this the classic **Flood Fill** recursive function is used [Hill, 1990]. A typical version is as follows:

```
1.   floodFill(x, y) {
2.        if(NOT(getColour(x,y) = WHITE) return;
3.        setColour(x, y, NEW_COLOUR);
4.
5.        floodFill(x, y+1); // north
6.        floodFill(x+1, y); // east
7.        floodFill(x, y-1); // south
8.        floodFill(x-1, y); // west
9.   }
```

A call to this function will paint the pixel (*x,y*) with the new colour (line 3) if it is currently white (line 2). It will repeat the process moving north (line 5) until a non-white pixel is found, then try east, south and west. The program is very effective and will find every pixel in the bounded area; the drawback is that the recursive calls can build up to a prohibitively large stack of functions. Usually a queuing system is employed to reduce the stack size.

The Flood Fill function has been extensively adapted in this thesis, to pass parameters of `stripe, row, heading` and `index`:

```
1.   floodFill(stripe, row, heading, index) {
2.        if(heading==NORTH) goNorth();
3.        if(heading==EAST) goEast(); index++;
4.        if(heading==SOUTH) goSouth();
5.        if(heading==WEST) goWest(); index--;
6.
7.        if(boundary || alreadyIndexed) return;
8.
9.        indices[stripe][row] = index;
10.
11.       floodFill(stripe, row+1, NORTH, index);
12.       floodFill(stripe+1, row, EAST,  index);
13.       floodFill(stripe, row-1, SOUTH, index);
14.       floodFill(stripe-1, row, WEST,  index);
15.  }
```

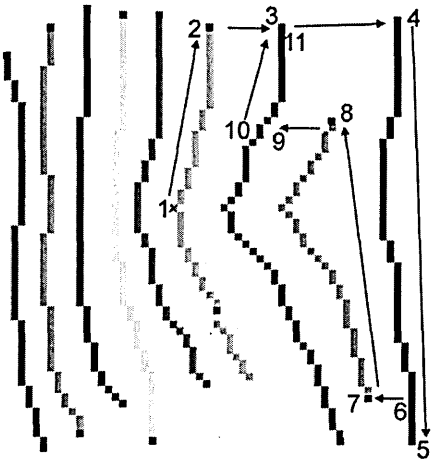

Figure 7.15.1: The Flood Filler.

The Flood Filler will move north if patching conditions allow, otherwise east, otherwise south, or finally west. If it cannot move in any direction, the current *floodFill*() function is taken from the stack revealing the parameters of the previous position and this is repeated until a previous position is returned to which has a valid move. In Figure 7.15.1 when the algorithm arrives at 5 it cannot move to the north, south, east or west, and peels the stack back until it arrives at previous position 6, whence it can move west to 7.

This method will index the stripe path missed in Figure 7.12.1 (left). It is not bound by horizontal and vertical motion, and will find both sections of the indexed stripe. This, of course, may not be desirable when asking Question 1 in Section 7.14; the physical reason for the disconnection may be a hole in the surface, or may result from a more complex reason such as an occlusion or step.

However, these scanning and searching methods have now armed us with a variety of ways of stripe indexing, which will be developed as we attempt to find logical physical reasons for the connections and disconnections presented in the recorded image data.

7.16. Absolute and Relative Indexing

In Section 5.3 it was stated that for this scanning system, in which the stripe planes radiate from a central point - the centre of the projector lens - the values of n must be found absolutely, rather than relatively to each other. In order to find the absolute indices using these methods, the

leftmost or rightmost stripe must be visible in at least part of the recorded image. Therefore the process will start, usually during the peak finding method, by labelling the leftmost or rightmost stripe with the correct index.

If in a particular application it is not appropriate to have a visible edge to the stripe pattern, then some other approach must be taken, such as a specially marked central stripe (perhaps with extra brightness) which can be easily detected by the algorithms.

7.17. Summary of Connection Methods and Questions

In this section we have investigated ways in which the recorded stripe patterns, as represented by **peaks** in the pixel array, can be described. Firstly we have defined a **neighbourhood** of **neighbouring** peaks, and then introduced the concept of **patches**, whereby a **patch** of connected neighbours is formed with a more constrained boundary than the neighbourhood.

This patch is then indexed, using **scanline, scan and search,** and **search** methods, and these methods will produce an indexed patch which may well have some wrongly indexed peaks. The first question posed, the Size *v.* Accuracy Question, is "for a given scanning application, what is the relative importance of indexed patch size compared with indexing accuracy?"

It is shown that the setting of the patching conditions has a considerable influence over the indexed patch size and the indexing accuracy. In order to give some basis for the choice of boundary conditions, two further questions are posed: firstly the **White Box Question** "is there a straightforward logical reason for the inclusion of this patching condition?", and secondly the **Black Box Question**, "do the test results show that the conditions under consideration will produce more accurate indexing?".

7.18. Implementation of Connection Methods

The three methods described in this section: *Scanline*, *Scan and Search* and *Flood Filler*, have been implemented using the standard `face3.bmp` recorded image. The results of these implementations are discussed in detail in Section 9.4.2, but here we look at one particular area of the recorded image which produces different outputs depending on the methods used and the patching conditions employed.

Figure 7.18.1 (left) shows part of the `face3.bmp` (partly shown in Figure 1.6.2) recorded image, around what we know to be the nose. The peak array (Figure 7.18.1, right) is shown with the northern and southern disconnections marked by dark pixels.

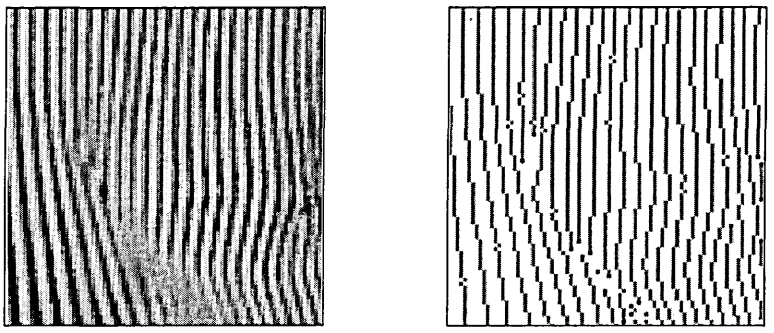

Figure 7.18.1: Northern and southern disconnections.

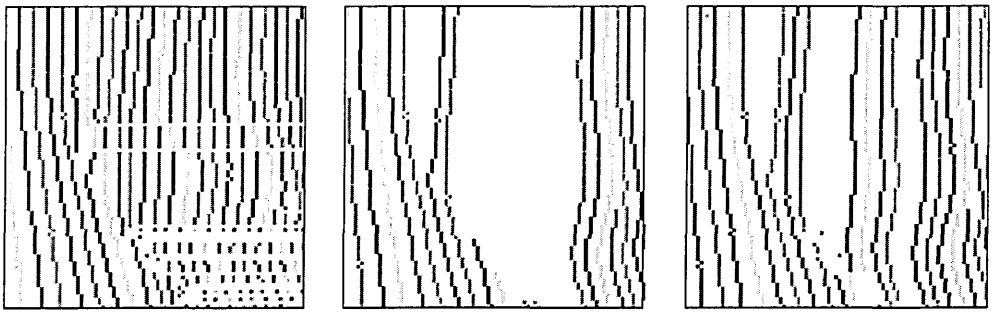

Figure 7.18.2: Indexing with the Scanline method (left), the Scan and Search method (centre) and the Flood Fill method (right).

In Figure 7.18.2 the peak array is then used to index the stripes (shown with an arbitrary colouring scheme) by using each of the three approaches described above: a modified Scanline Method (left), a Search and Scan method (centre), and a Flood Fill method (right). Here we compare the results with our intuitive estimate of how the indexing should work, based on our prior knowledge of the shape of the face.

- The Scanline Method is modified to start with a correctly identified leftmost stripe. Problems begin to occur when the indexing count becomes "out of phase" from row to row, giving the classic effect, shown also in Figure 7.10.1 (right), of an apparent stripe path with a changing index number.

- The Scan and Search method introduces patching conditions which stop the search when confronted by a disconnection, but even here some mistakes are made. The two stripes to the left of the large central space flow across what we know (with prior knowledge) to be an occlusion at the nose. Also, the third stripe from the right seems to end for no obvious reason, when reading from the bottom row.

- The Flood Fill Method provides a greater coverage of indexed peaks, and two additional "incorrectly" indexed stripes to the right centre of the image. *Note that the Flood Fill method would index every stripe, except that a limit has been imposed to prevent stack overflow (see Section 7.15).*

Asking the White Box and Black Box questions of these results, the Black Box question is answered by the tests and results shown in Section 9.6.3; in general the Scan and Search methods give smaller indexed patches with greater accuracy, and the Flood Fill methods give greater indexed patch size with lesser accuracy.

However, we have very little evidence so far to answer the White Box Question, "is there a straightforward logical reason for the inclusion of this boundary constraint?". We can see that the disconnections relate in some way to occlusions in the surface, and thereby confuse the algorithms, but no logical relation has been found between the disconnections and the occlusions. Another obvious effect, seen in the original recording (Figure 7.18.1 (left) is that the stripe spacing varies across the image. This observation suggests that there may be some relation between stripe width and surface shape which can be used profitably.

Therefore, two specific questions are posed which may provide "white box" answers to the Indexing Problem":

The Stripe Spacing Question. *Can the spacing between stripes in the recorded image be used to improve the indexing algorithms?*

The Occlusion Question. *Can an understanding of occlusions be used to improve the indexing algorithms?*

These questions will be investigated in Chapter 8.

Chapter 8: Occlusions

8.1. Original Contributions

To overcome the well-known problems caused by occlusions [Park *et al.*, 2001], we make specific definitions which relate occluded areas on the target surface to the resulting recorded image: ***Near Occlusion Points***, ***Far Occlusion Points***, ***Projector Occlusion Points*** and ***Camera Occlusion Points***. From these definitions four ***boundary types*** are categorized: the ***Concave Pair***, the ***Contour Single***, the ***Concave Tuple*** and the ***Contour Tuple***. The investigation then looks specifically at how Concave Pairs can be identified in the recorded image, and thence designs a novel algorithm to ***detect and mark the occlusion boundary***. This is added to the Patching Conditions, giving improved levels of indexing accuracy.

These investigations relate to occlusions which are not horizontal. In addition, a theoretical instance has been designed which shows that it is possible to have ***exactly the same spacing between stripes, above and below a horizontal occlusion***. This means that spacing between stripes cannot be readily used as an indicator of the presence of a horizontal occlusion.

8.2. Introduction

In Chapter 7 methods are described which produce indexed patches, from which the 3D surface model can be derived. The comparative success of these methods, measured in terms of indexed patch size and indexing accuracy, is detailed in the test results of Section 9.6.3. However, the methods do not so far attempt to provide logical relationships between the stopping conditions of the algorithms, and physical characteristics of the target surface. In this section we will investigate how far an analysis of occlusions can be used to give logical reasons for the algorithmic stopping conditions, and thence improve either the indexed patch size, or the indexing accuracy, or both. We will also look at whether the spacing of stripes in the recorded image can be used to determine the shape of the surface, and in particular the presence of occlusions.

8.3. Defining Occlusions

In the machine vision context occlusions are of two types: camera occlusions in which one part of an object is hidden from the camera view by another object, and shadow occlusions in which one part of an object is hidden from the light source by another object. As discussed in Section 2.6, stereo vision techniques have detected camera occlusions, and particularly their boundaries, by comparison of the image pair. Research on shadow occlusions has derived surface shape from an estimation of the projection function from the light source, often by moving the light source to obtain triangulation. Of particular interest to this thesis are the investigations by Koenderink and others [Koenderink, 1984] on the subject of *contours*, and how they can be used to derive solid shape. This section will incorporate these three subjects - occlusion boundaries, shadows and contours - into the general investigation on occlusions.

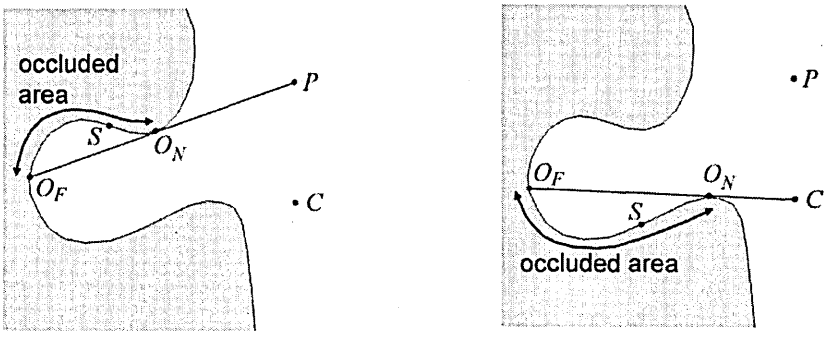

Figure 8.3.1: Projector and Camera Occlusions.

Figure 8.3.1 (left) shows a section through a target object, the surface under consideration being marked by a black outline. A camera is positioned at C and an omnidirectional light source is projected from point P onto the target surface. A line is drawn from P which grazes the surface at point O_N, and continues to point O_F, where the line intersects the surface at a second point. This line is a supporting line, which for a smooth surface will be tangential. The part of the surface between O_N and O_F is *occluded,* hidden from the projection point P. Although the figure shows a 2D section, the principle can be extended over the 3D surface, so that a continuous set of these occluded sections will form an *occluded area.* We call this occluded area a *projector occlusion,* O_N a *near occlusion point* and O_F a *far occlusion point.* Projector occlusions are, of course, generally called *shadows.*

In Figure 8.3.1 (right) the surface is analysed to find occlusions from the camera viewpoint. A supporting line is drawn from the camera view point C to *near occlusion point* O_N and *far occlusion point* O_F. Here the occluded area is that part of the surface which is hidden from the camera viewpoint, and we call this a *camera occlusion.* Note that it is quite possible for some parts of the target surface to be occluded from both the camera and the projector.

If we generalise the camera position and projector position as viewpoints for these two types of occlusion, then we can define *near* and *far occlusion points*:

Definition 8.1. A *near occlusion point* is a point on the surface of the target object, such that a straight line from that surface point to the viewpoint is supporting to the surface at that *near occlusion point.*

Definition 8.2. A *far occlusion point* is a point on the surface of the target object, such that a straight line from that surface point to the viewpoint is supporting to the surface at a *near occlusion point.*

The occluded areas have been described as being "hidden" from their respective viewpoints, either the camera or the projector. This means that a line segment between the viewpoint and any surface point S (in Figure 8.3.1) within the occluded area must also intersect the surface at the point or points causing the occlusion. Therefore surface points within occluded areas can be defined:

Definition 8.3. A *projector occlusion point* is a point on the surface of the target object such that a straight line segment from that point to the projector viewpoint will intersect the target surface at one or more other points.

Definition 8.4. A *camera occlusion point* is a point on the surface of the target object such that a straight line segment from that point to the camera viewpoint will intersect the target surface at one or more other points.

8.4. Defining Occluded Areas

The areas containing projector and camera occlusions can take different forms. For any given surface there may be many occluded areas, each bounded by a line of near and far occlusion points. There may also be "unoccluded" areas within an occluded area. In general it can be said that **a projector occlusion is a subset of the complete set of projector occlusion points on the target surface**, and **a camera occlusion is a subset of the complete set of camera occlusion points on the target surface**; but should occlusions include the near and far occlusion points? The sense of the supporting lines described in Definitions 8.1 and 8.2 suggests that a beam of light will actually illuminate the far occlusion point, and will also just illuminate the near occlusion point. On that basis **the projector occlusion excludes both the near and far occlusion points.** For consistency it can also be assumed that **the camera occlusion excludes both the near and far occlusion points.**

In order to specify occluded areas more accurately, it is necessary to investigate the ways in which the areas are bounded.

8.5. Boundaries of Projector Occlusions

Figure 8.5.1 (left) shows a target surface, a cup and saucer, illuminated by a light source at *P*. The projector occlusions, or shadows, are coloured grey. Some of the near occlusion points (marked with a ring) and some of the far occlusion points (marked with a black disk) are shown, each with their associated supporting line, numbered 1 to 4. If the target surface is continuous and there is an infinitely dense set of surface points, then every occluded area will be bounded by a continuous line of near and far occlusion points. In Figure 8.3.1 the supporting lines are each associated with a pair of near and far occlusion points, but this association is not always the case, as can be seen on the cup and saucer where, for instance, supporting line 3 is associated with two near occlusion points and one far occlusion point. In fact the four supporting lines in the figure illustrate four types of association between a supporting line and near or far occlusion points, which we call *boundary types*:

1. The **concave pair** (O_F, O_N), if the supporting line grazes the surface at ONE near occlusion point and intersects the surface at a far occlusion point.

2. The **contour single O_N**, if the supporting line grazes the surface at ONE near occlusion point but then does not intersect the target surface.

3. The **concave tuple** $(O_F, O_{N1}, O_{N2}, \ldots)$,if the supporting line grazes the surface at MORE THAN ONE near occlusion point and intersects the surface at a far occlusion point.

4. The **contour tuple** $(O_{N1}, O_{N2}, \ldots)$ if the supporting line grazes the surface at MORE THAN ONE near occlusion point but then does not intersect the target surface.

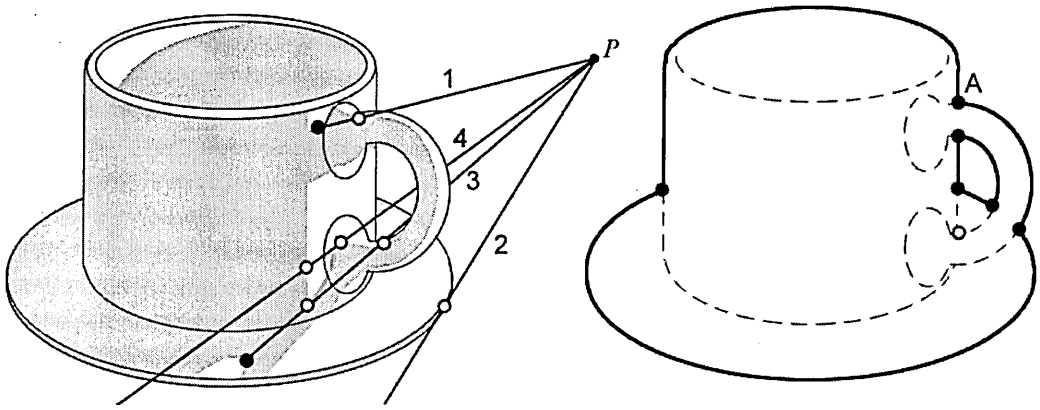

Figure 8.5.1: Occlusion boundary types 1, 2, 3 and 4.

Each member of the set L of all possible supporting lines (which intersect P) will be associated with one of these four boundary types T_1, T_2, T_3 and T_4. Each element $t \in \{T_1, T_2, T_3, T_4\}$contains one or more near and far occlusion points which are members of the set B of all the surface points which make up the occlusion boundaries. Therefore **the complete set of surface points which comprise occlusion boundaries can be derived from the complete set of supporting lines.**

8.6. Boundaries of Camera Occlusions

In Figure 8.5.1 (right) the cup and saucer are marked with camera occlusion boundaries. The supporting lines from the camera viewpoint will, because by definition they all intersect the

viewpoint, be projected onto the image plane as points. In other words they will all be seen "end on" in the scene. This also means that **all the near and far occlusion points belonging to one supporting line will be coincident.** For instance, the supporting line at A includes two near occlusion points, one grazing the handle and one grazing the side of the cup. The line does not intersect the surface and is therefore of boundary type 4: a contour tuple.

In the figure the thick black line shows the *contour boundaries* of the camera occlusions, comprising the complete set of surface points which are members of the contour singles and contour tuples, and the dotted line shows the *concave boundaries* comprising the complete set of surface points which are members of the concave pairs and concave tuples. The tuple types are rare, only occurring eight times in this case, and they are separately marked with a ring for concave tuples and a black disc for contour tuples. This observation is true in general; tuples are single instances where from the viewpoint two contour lines from different parts of the surface happen to meet, whereas concave pairs and contour singles will form continuous lines of points.

8.7. Applying Occlusion Categorization to Structured Light Projection

So far the light source used has been a uniform omnidirectional set of beams, thus illuminating every unoccluded part of the target surface. When a structured light pattern is projected, some parts of the unoccluded target surface will not be illuminated, corresponding to the dark part of the pattern. However, the bright parts of the pattern will still exhibit the same categories of projector occlusion as defined above, and the camera occlusion definitions will still apply.

Recalling that the key task of this thesis is to solve the Stripe Indexing Problem, it is important to be able to find out where a particular stripe, or other pattern element, is "picked up" again after it is obstructed by a projector or camera occlusion. This "picking up" is precisely what happens between near and far occlusion points. If they form a concave pair then the pattern stops at the near point and is picked up again at the far point; if there is a contour single then the stripe is lost forever having visited the near occlusion point.

Also, having found a near point any far point must occur along the same supporting line, and *vice versa*. Therefore if the geometry of the system is sufficiently known to enable the supporting line to be calculated from a given near occlusion point, then far occlusion points and occlusion boundaries may consequently be determined.

We will now look more closely at the concave pair boundary type, when applied to structured light patterns.

8.7.1. Finding Concave Pairs

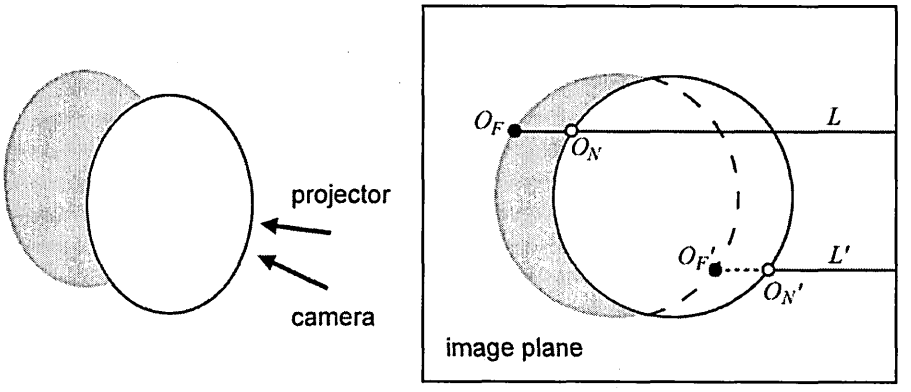

Figure 8.7.1: Occluded areas showing Concave Pairs.

Figure 8.7.1 (left) shows a target surface: a circular disc suspended over a flat plane. The surface is illuminated by a light source which produces a shaded area as shown. Figure 8.7.1 (right) shows the same target surface , viewed in the image plane of the camera, **when the system is set up with the Horizontal Constraint** as described in Section 6.3. With this constraint all the beams of light from the projector will appear to be directed horizontally with respect to the image plane. A supporting line L from the projector grazes the edge of the disc at O_N and intersects the surface at O_F , giving a concave pair (O_N, O_F). If the surface is continuous, an infinite set of concave pairs will mark the boundary of the projector occlusion, which in this case will be in two parts: the circular shadow on the flat surface, and the dark underside of the disc. In fact in this rather special case all the near occlusion points will be on the edge of the disc, and all the far occlusion points will be on the surface of the flat plane, bounding the circular shadow.

A second supporting line L' gives near and far occlusion points O_N' and O_F'. Now analysing the occlusions from the camera viewpoint, O_N' is at both a near and far occlusion point, and O_F' is a surface point within the camera occlusion. Therefore O_F' is hidden from view; all we know is that it must be positioned somewhere along L' or its extension, due to the Horizontal Constraint.

We can extend these observations to look at stripe patterns on a target surface. Figure 8.7.2 (left) shows the target object of Figure 8.7.1, this time lit by a stripe pattern, indexed from 1 to 7. At "a" the stripe is broken by a projector occlusion, with near and far occlusion points at A_N and A_F. In the image this appears as a horizontal translation of the disconnected stripe, in the direction of the arrow. A similar situation also occurs at "b". At "c" although the near occlusion point C_N is visible, its pair C_F is hidden by the camera occlusion. Again, all we know is that it must be translated horizontally to the left. A similar situation also occurs at "d".

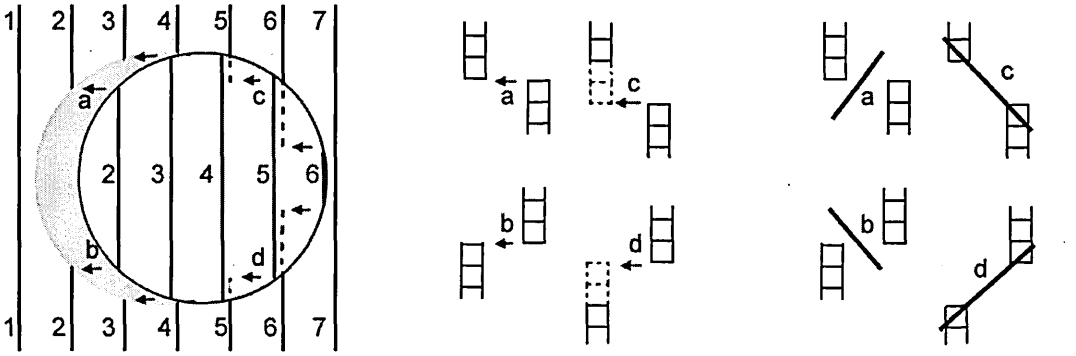

Figure 8.7.2: Categorization of Concave Pairs.

In Figure 8.7.2 (centre) parts of the image are magnified to pixel level, and the four typical situations described are classified: at "a" a *high projector occlusion* (h.p.o.), at "b" a *low projector occlusion* (l.p.o.), at "c" a *high camera occlusion* (h.c.o.) and at "d" a *low camera occlusion* (l.c.o.). This illustrates that in theory if the pixel representation of the near and far occlusion points are on adjacent pixel rows but with a horizontal translation, the occlusion is likely to be an h.p.o or l.p.o. Alternatively, if the pixel representation of the near and far occlusion points are not on adjacent pixel rows, suggesting that there are some missing pixels, then the occlusion is likely to be an h.c.o or l.c.o. Also, for cases a and c the continuation of the stripe is from high on the left to low on the right, and for cases b and d the continuation is from low on the left to high on the right. These observations are summarised as follows:

a. The High Projector Occlusion (h.p.o.) has no missing pixels, and the continuation is from high on the left to low on the right.

b. The Low Projector Occlusion (l.p.o.) has no missing pixels, and the continuation is from low on the left to high on the right.

c. The High Camera Occlusion (h.c.o.) has at least one missing pixel, and the continuation is from high on the left to low on the right.

d. The Low Camera Occlusion (l.c.o.) has at least one missing pixel, and the continuation is from low on the left to high on the right.

Thus a categorization of occlusions has been established which can be used to find occlusions and connections between stripe elements. At this stage two provisos are important: firstly that this is a theoretical investigation and practical examples are unlikely to be so precise due to, for instance, imperfect sampling of the pattern elements, and secondly that horizontal and vertical occlusions are not included in these categories.

In Figure 8.7.2 (right) the likely position of the occlusion boundary for cases a, b, c and d is marked with a thick black line. The four boundary lines will each have a gradient approximately in the direction as shown, sloping upwards from left to right for a and d, and downwards from left to right for b and c. Conditions where the boundaries are horizontal or vertical will be investigated later, in Section 8.8.

8.7.2. Incorporating Occlusions in the Data Structure

In Section 5.5 the system data structures were defined, prior to the investigation into occlusions. Now an occlusion data type `int occlusion[C][R]` is added, derived from the peak array `bool peak[C][R]`. Recalling that *peak(c,r)* is TRUE if a peak is present at that location, then:

$$occlusion(c,r) = \begin{cases} N \text{ if } \neg(peak(c,r)) \\ D \text{ if } \neg(peak(c-1,r-1) \vee peak(c,r-1) \vee peak(c+1,r-1)) \\ U \text{ if } \neg(peak(c-1,r+1) \vee peak(c,r+1) \vee peak(c+1,r+1)) \\ C \text{ otherwise} \end{cases}$$

Eq. 8.7.2

Thereby peaks, i.e. TRUE pixels, are labelled as disconnected going up (*U*), disconnected going down (*D*), or connected (*C*). Non peaks are denoted as *N*.

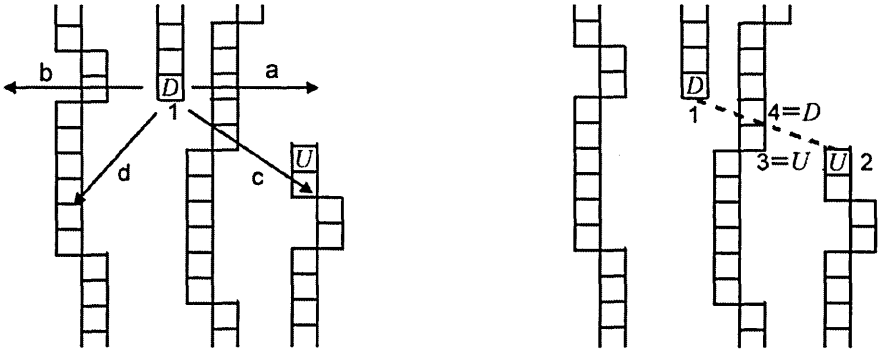

Figure 8.7.3: Occlusion detection and marking.

Figure 8.7.3 (left) shows a practical example of occlusion detection and marking, using the occlusion array and the cases shown in Figure 8.7.2. From the *D* peak at position 1 we look for a *U* peak complying with case a, b, c or d. Note that we can cross fully connected stripes (that is a continuous sequence of type *C* peaks) in our search. In Figure 8.7.3 (right) we have found the *U* peak at position 2, and this configuration is now compared to the cases shown in Figure 8.7.2. Because there are missing pixel rows between the *D* peak and the *U* peak it must be a camera occlusion, and because the continuation is from high on the left to low on the right it must a high camera occlusion (case c). The occlusion boundary is drawn as a dotted line connecting the *D* peak with the *U* peak. This occlusion line now crosses a seemingly connected stripe, and we therefore know that this is a falsely connected stripe, and that there must be a break approximately in the region of the occlusion line. We therefore mark *U* and *D* peaks at positions 3 and 4.

This occlusion boundary can be confirmed intuitively by eye. It seems to be a reasonable interpretation that there are three indexed stripes, with the stripe ending at 1 resuming at 3, and the stripe broken at 4 resuming at 2.

All the peaks labelled as *U* and *D* can now be used as stopping conditions in the connectivity algorithms, and this investigation has therefore achieved one of its objectives - to find a logical relationship between the stopping conditions and the physical characteristics of the target surface.

Referring to Section 7.13, the occlusions can be marked either as Live Boundaries or Premarked Boundaries. An implementation of occlusion marking is shown in Figure 8.7.4, using a detail from the test object (with the difficult area around the nose occlusion). The recorded stripes are shown on the left, with the peak data in the centre image, and the occlusion

boundaries on the right. Here the *D* disconnected peaks are marked with a black square, the *U* disconnected peaks with an outlined white square. The occlusions are estimated to be of case c type, and the occlusion boundaries are marked accordingly with a black line, which follow what we know with prior knowledge to be the nose occlusion. These occlusion boundaries are then added to the stopping conditions of the algorithms. This method is used in Section 9.6.3 to obtain the results with occlusion detection.

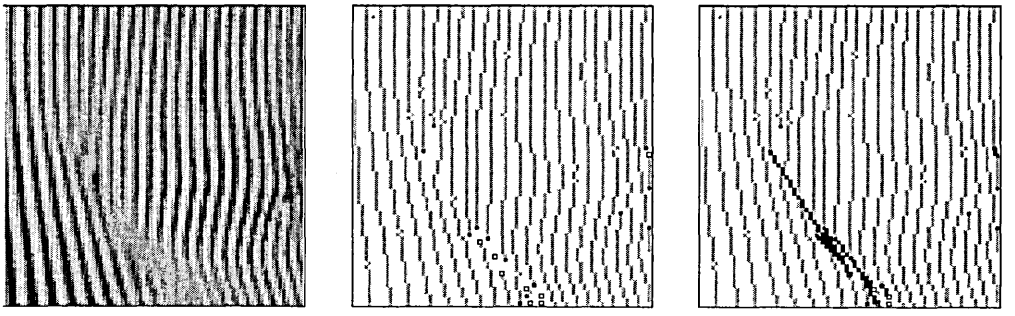

Figure 8.7.4: Occlusion marking on recorded data.

8.8. Vertical and Horizontal Occlusions

Vertical occlusions can be considered as "vertical" from the camera viewpoint, from the projector viewpoint, or from both viewpoints. In the latter case the occlusion boundary formed by the line of near occlusion points will be parallel to the Y -axis of the system. A study of practical examples shows that the effect of vertical occlusion is to increase or decrease the translation of the stripe at its disconnection, and this does not in itself affect the validity of the stopping conditions described above.

Horizontal occlusions or steps, however, present a more serious challenge. In the Introduction (Section 1.7) a synthetic example was suggested which illustrates the problem of ambiguous interpretation, and this case is repeated here in Figure 8.8.1. The hypothetical recorded image is shown at (a), and two possible indexing results at (b) and (c). These two results are visualised at (d) and (e) with the coloured stripes superimposed. As was stated in the Introduction, "This synthetic example suggests that there will be cases where apparently continuous stripes are in fact discontinuous, and also that if this area is the full extent of the image it may be impossible to decide which interpretation is correct.".

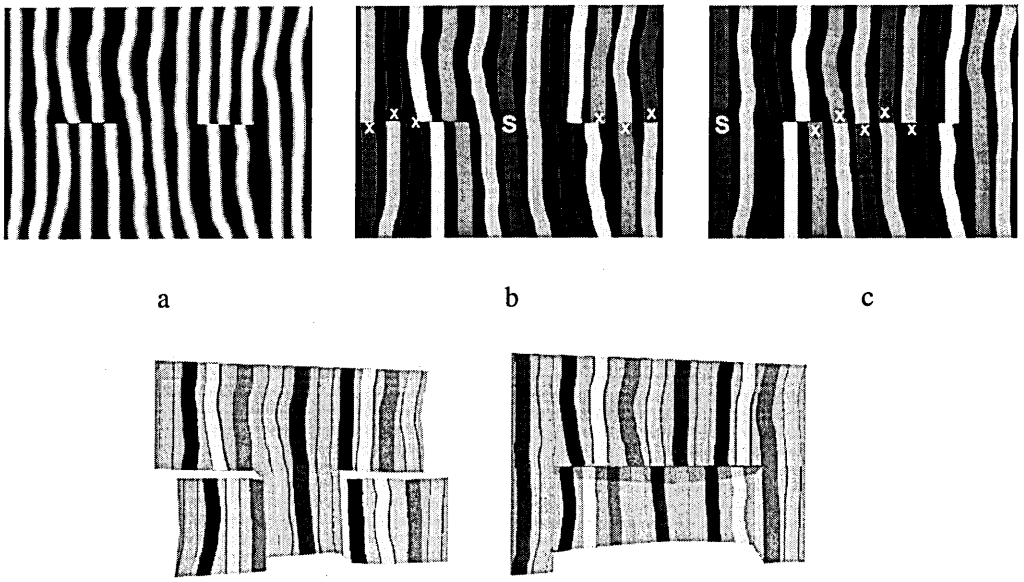

Figure 8.8.1: (a) Synthetic pattern of stripes indexed starting at the centre (b) and the left (c). The resultant shapes for (b) and (c) are shown in (d) and (e).

The problem arises because of the horizontal occlusion. If the occlusions were not horizontal, it would be possible to detect the occlusion by the disconnections and translations of the stripe, using the methods suggested above. However, here there are seven stripes - two at the left, two at the right, and three in the centre - which appear to be continuous. **The question arises as to whether it is possible to have exactly the same spacing between two stripes, above and below a significant (i.e. not small) horizontal occlusion?** If it is not possible, then this synthetic example would be unachievable, and there would always be a change in stripe width on either side of the occlusion. If this change were detectable in practice, then it would help to solve ambiguities caused by horizontal occlusions.

8.9. The Significance of Stripe Spacing in the Recorded Image

In order to answer the question as to whether it is possible to have exactly the same spacing between two stripes, above and below a significant (i.e. not small) horizontal occlusion, a theoretical example is now given.

Figure 8.9.1 shows the scanning system as described in Figure 5.2.1a, looking at the "horizontal" plane containing *C,* the centre of the camera lens, *P*, the centre of the projector lens, and *O* the origin of the Cartesian Coordinate System. The Horizontal Constraint (see Section 6.3) is enforced, so that $\angle OPC = 90°$, and the image plane is given an arbitrary position normal to the camera axis *OC.*

A theoretical planar target surface, surface 1, is shown which intersects *O* and is normal to the projector axis *OP*. The constraints of the scanning system are such that all the stripe planes will intersect the *X* -axis (and in this case surface 1) with an equal spacing of *W*. Two surface points S_1 and S_2 are shown where stripes $n = 4$ and $n = -4$ intersect the target surface. (As the model is three dimensional, S_1 and S_2 will actually represent straight lines at the intersection of the surface plane with the two stripe planes, but for this demonstration we assume that S_1 and S_2 are arbitrary points on those lines).

Now a second planar surface, surface 2, is constructed which is parallel to surface 1. It is positioned such that surface point S_4 is at the intersection of line CS_2 and stripe plane $n = -7$.

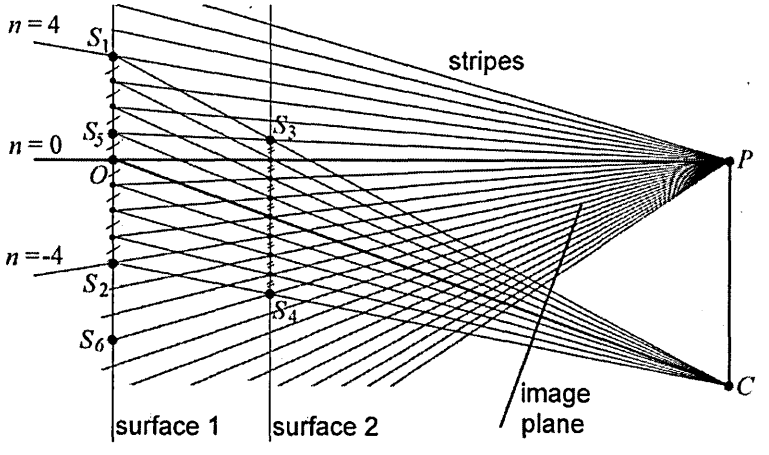

Figure 8.9.1: Evenly spaced stripes projected onto image plane from surfaces 1 and 2.

Point S_3 is at the intersection of stripe $n = 1$, and if line CS_3 is projected on to surface 1, it will intersect at point S_1. It can be seen diagrammatically that this will only be true if line length S_1S_2 = line length S_5S_6 (a spacing of $8W$) and this is proved as follows:

by similar triangles

$$\frac{\overline{S_1S_2}}{\overline{S_3S_4}} = \frac{\overline{CS_1}}{\overline{CS_3}},$$

$$\frac{\overline{S_5S_6}}{\overline{S_3S_4}} = \frac{\overline{PS_5}}{\overline{PS_3}},$$

$$\frac{\overline{CS_1}}{\overline{CS_3}} = \frac{\overline{PS_5}}{\overline{PS_3}}.$$

$$\therefore \frac{\overline{S_5S_6}}{\overline{S_3S_4}} = \frac{\overline{S_1S_2}}{\overline{S_3S_4}}$$

$$\therefore S_5S_6 = S_1S_2.$$

Eq. 8.9.1

Because S_1 and S_3 are collinear with C they will appear at the same point in the image plane; similarly S_2 and S_4 will appear at the same image point. Recalling that the model is three dimensional, if S_1 and S_3 are considered to be vertical lines denoting stripes reflected from surfaces 1 and 2 respectively, then those two lines will coincide in the image plane, as will the other "pairs" such as (S_2, S_4). In other words, **the image recorded using surface 1 as a target object will be identical to the image recorded using surface 2 as a target object.**

8.10. Visualisation of Theoretical Example

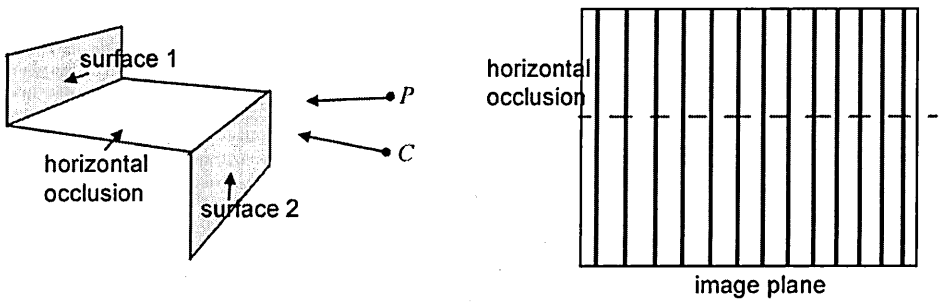

Figure 8.10.1: Visualisation of horizontal occlusion.

A possible visualisation of the theoretical instance given in Figure 8.9.1 is shown in Figure 8.10.1. On the left is a rendition of the target object, and on the right the stripe reflections as seen from the image plane. The stripe spacing seen on surface 1 will appear to be identical to the stripe spacing seen on surface 2. It is immediately apparent that the horizontal occlusion cannot be detected in the image view, as the stripes appear to continue with the same spacing on either side of the occlusion.

This example shows, by construction, that the spacing of stripes is not dependent on the depth z of the target surface. Therefore it is **not possible to solve the ambiguity problem, caused by horizontal occlusions, by measuring the stripe spacing at either side of the occlusion.**

8.11. Conclusions on Occlusion Detection and Marking

This chapter has looked at ways of defining occlusions from both the camera and projector viewpoint. In particular, for stripe projections, categories have been made of the likely position of the stripe elements which are close to occluded areas. This has enabled a method to be devised to predict and mark the position of occlusions, based on the way in which stripe elements are connected. This method can be used to produce occlusion boundaries, which will be used as stopping conditions in the indexing algorithms.

Horizontal occlusions present a separate problem, and a theoretical investigation has concluded that horizontal occlusions do not necessarily mean that the spacing of stripes above and below the occlusion will be different.

Chapter 9: Results and Evaluation

9.1. Original Contributions

In order to test and evaluate the results, all the indexing methods are executed using a *test object*, which is compared to a hand-produced *template* result. Three novel measures: ***total patch size***, ***accurate patch size*** and ***indexing accuracy*** are used to evaluate the new methods. These measures overcome the difficulty of assessing the success of scanning a surface with occlusions.

9.2. The Research Questions

In the Introduction, the two principal research questions were posed:

1. How can we correctly identify in the sensor a specific projected pattern element, expressed as an index? This is the Indexing Problem.

2. How far can the Indexing Problem be solved using an uncoded projection pattern?

In this chapter we evaluate the tests conducted to answer these questions. Some of the results, such as the outputs of indexing algorithms, are directly related to the research questions; other results test the data preprocessing and the geometric measurements which are needed as a precursor to the principal investigations.

Three stages in the scanning process are considered, and these have distinct methods of testing and evaluation:

1. **Recording the image as a pixel array**. This requires that the deformed stripe patterns are expressed clearly and distinctly in the most appropriate data structure.

2. **Calibrating the intrinsic and extrinsic parameters.** The system parameters are measured, and then their accuracy is evaluated by measuring objects of known dimensions.

3. **Patch size and indexing accuracy.** Isomorphic comparisons are made with known shapes.

9.3. Preprocessing and Geometric Measurement Methods

9.3.1. Presentation of the Data to the Peak Array

As discussed in Section 5.5.1, five aspects of the way the image is stored by the pixel array are identified: focus, aperture control, ambient light, unwanted artefacts and other noise. Tests were conducted with the Single Stripe Scanner to evaluate the contribution of these criteria.

The premise of these tests is that if the same surface lit by the same stripe is recorded under measured changes in the five criteria, then the outputs can be evaluated to show the effects of the criteria.

9.3.2. Calibration of the System Parameters

The calibration task is divided into two stages: firstly the intrinsic and extrinsic parameters are found by methods described in Section 5.6, and secondly the accuracy of those parameters is tested by scanning known objects such as planes and spheres. The problem here is that there are six parameters, and it may be difficult to decide which parameter is responsible for any errors in the output measurements. To deal with this problem it is important to be able to accurately and independently vary any one of the parameters before a test is run, and then evaluate the outputs.

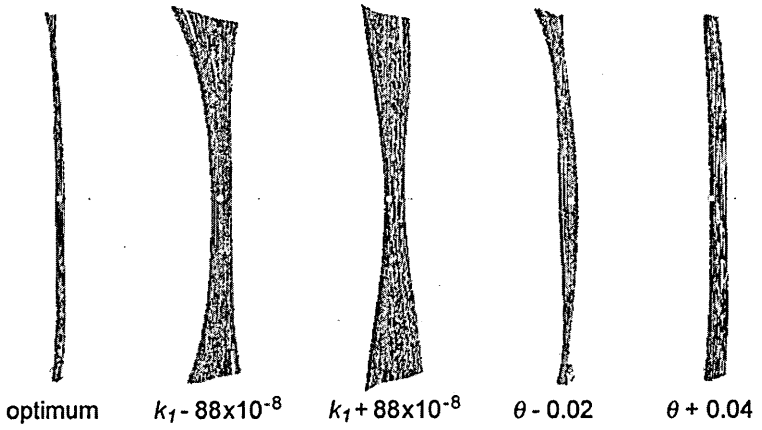

Figure 9.3.2: Visualisation of scanned surfaces with variations in parameters.

Figure 9.3.2 shows a visualisation of five outputs from the scanner, where the target object is a planar surface of approximately 270 mm x 350 mm. The line of sight of the visualisation is parallel to the target plane, so that the optimum output (left) should see a thin vertical line down the screen, whereas there is actually a slight curvature. Investigation into this curvature suggests that the most likely explanation is the inaccurate modelling of the camera lens using the radial function (see Section 5.6.1), and the internal software for the projector "keyhole" function (see Section 9.3.2). These effects are further discussed in Section 9.4.2.

For this thesis the primary goal is to solve the Indexing Problem rather than achieve very high geometric accuracy (except where necessary in relation to the Geometric Constraints), and therefore no attempt has been made to improve the modelling of the camera lens. There exists a large body of research on the subject of calibrating video camera systems (see Section 2.5).

The next two images show changes in the value of κ_1, the radial coefficient, and the two rightmost images show the result of changes in θ, the triangulation angle. These two parameters, and in particular κ_1, have proved to be the most critical for effecting the output measurements. The problem here is that whereas the other five parameters can in theory be found to very high accuracy, κ_1 is inevitably a compromise unless the power series (see Eq. 5.6.1) is extended beyond the first term. The most practical way of deciding upon the value of κ_1 is a heuristic one of modifying the value and then inspecting the output for linearity, either by eye (as seen in Figure 9.3.2) or by variance measures.

In order to provide control of the parameters during the scanning process, the C++ program **`striper.exe`** was written (see Appendix), and the outputs of this program are now shown, for the same planar surface as visualised in Figure 9.3.2.

The target surface is a white painted wall of which approximately 270 mm x 350 mm is visible in the recorded image, located at the origin of the scanning system, and oriented in the *Y-X* plane. The scanner comprises the system as described in Section 5.1, using a monochrome CCD camera (a Panasonic WV-BL200) and a PAL video capture card (a Matrix Vision MV delta) which interprets the output as a 768 x 576 array of pixels. As with the Single Stripe Scanner (see Chapter 4), there are two digitization processes, firstly to store the luminance levels in the CCD, and secondly to sample the serial analogue video output and produce the pixel array. The LCD projector (a Hitachi CP-S225) acts as a Windows 2000 VDU with the display size set at 800 x 600 resolution. A major feature of this projector is that the "keyhole" function interpolates the data received through the standard monitor port so that if the projector axis is not normal to, and central to, the projector screen, the image is rectified to project as an orthogonal rectangle. This means that a further sampling takes place, and experiments show that there is only one setting for the keyhole function which interpolates the display without introducing further artefacts. The recorded image is seen in Figure 9.3.3. Here the curving of the stripes because of radial distortion is clearly visible, and aliasing artefacts cause the *moiré* banding seen in the right half of the image. A discussion of this issue is discussed below in Section 9.4.1.

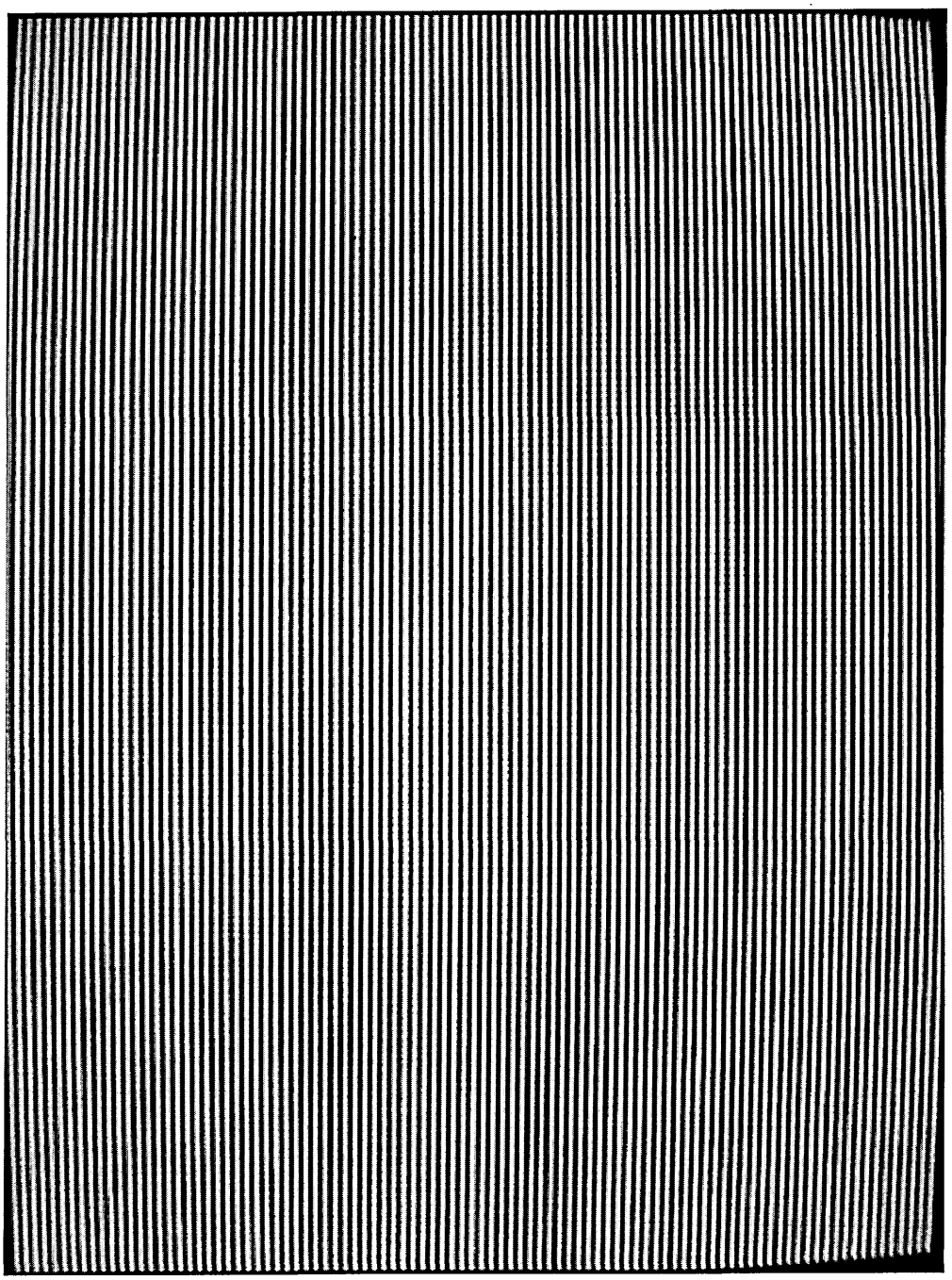

Figure 9.3.3: Recording of planar wall.

Figure 9.3.4 shows a visualisation of this data using the program **`striper.exe`**.

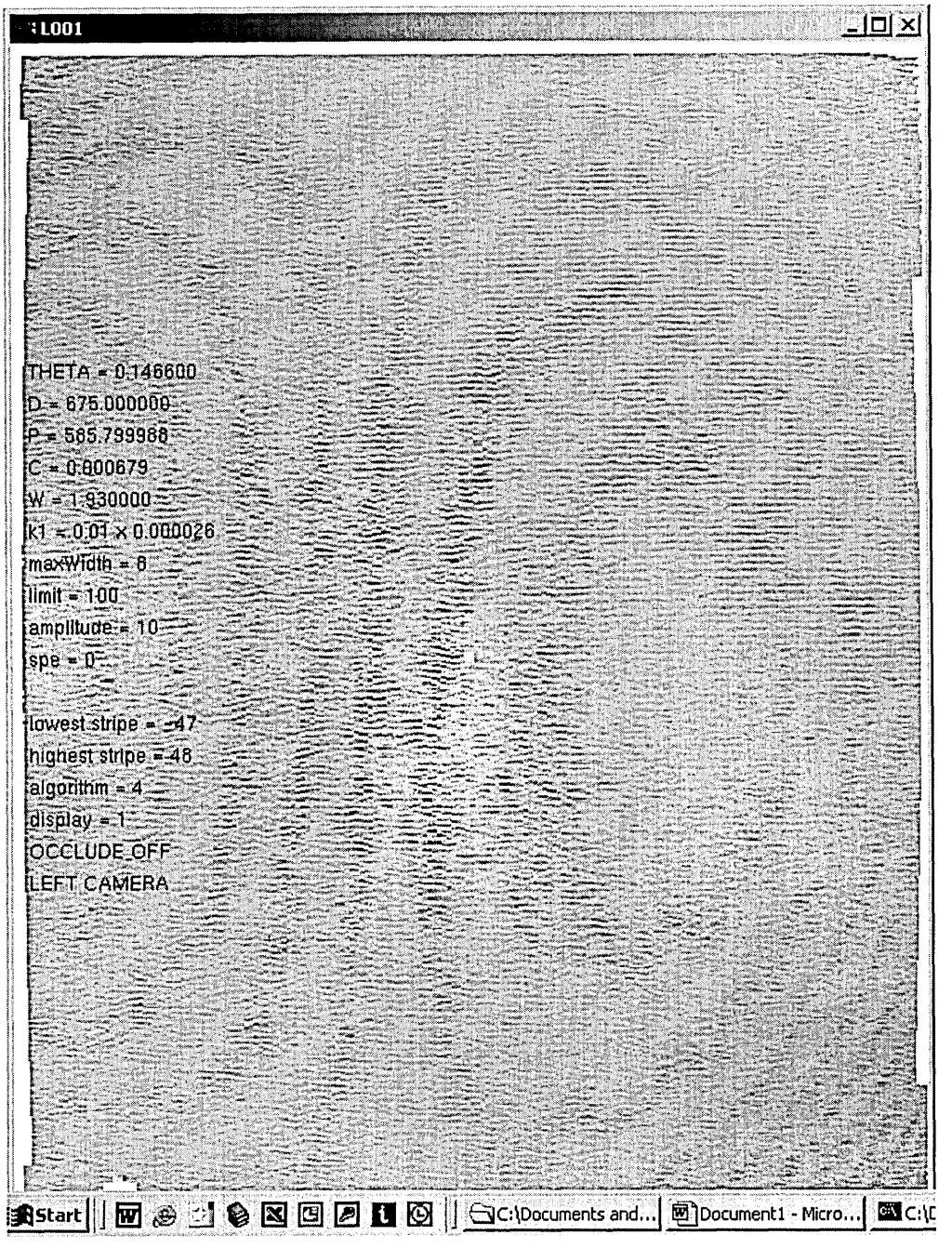

Figure 9.3.4: Visualization of surface of planar wall.

The visualisation is a polyhedral surface representing the array of surface points as described in Section 5.5.6. In the program **`striper.exe`** the surface is rendered using the standard Phong Illumination Model [Phong, 1975] from a single light source, with a high specular reflection component in order to emphasise irregularities in the surface. The statistical report to the left of the figure lists the parameters which can be changed by the user, and the parameters of relevance to the calibration process are THETA, D, P, C, W and k1. These represent the system parameters θ, D_C, D_P, P, W and κ_1.

9.4. Evaluating the Scanning of a Planar Surface

9.4.1. Aliasing Artefacts

The visualisation of Figure 9.3.4 illustrates the aliasing artefacts which accumulate during the scanning process. The recorded image, seen in Figure 9.3.3, has artefacts on the right half of the image, but very few on the left; but when this data is processed to produce the visualisation of Figure 9.3.4 it is clear that artefacts have been added to the left part of the picture and that there is now a cross pattern of *moiré* banding on the right. The most likely cause of these is the inverse radial distortion function, which adds a "spherical warping" to the data. Further factors which may add to the aliasing artefacts are the subpixel estimator (see Section 4.3.1), and the computing process which may introduce errors caused by type conversion and arithmetic operations. Therefore six factors have now been described which contribute to producing unwanted artefacts and increasing the "noise" in the output data:

1. The discrete pixel pattern produced by the LCD projector.
2. The sensing process on the camera CCD.
3. The sampling method of the video capture card.
4. The subpixel estimator.
5. The inverse radial distortion function.
6. Numerical operations.

Possible future work to reduce these artefacts could include "supersampling" [Baker and Kanade, 2001] [Cheeseman *et al.*, 1994] and other methods of inter-pixel interpolation.

9.4.2. Comparison of Results for Single and Multiple Stripe Scanners

In Section 4.4.1 the Single Stripe Scanner was tested using a planar surface as the target object, and recording 21 successive stripes as the object rotated on the turntable. These results are now compared to those obtained using the Multiple Stripe Scanner when indexed using the Flood Fill algorithm (see Section 7.15). The recorded image from the planar surface as seen in Figure 9.3.3 is used. From the array `surface_point[N][R][3]` (as described in Section 5.5), the central stripe $n = 0$ is chosen for evaluation. Table 9.4.2 shows the results, which can be compared to Table 4.4.1. Each test uses a sample range from r = START to r = END, giving depths `surface_point[0][START][z]` to `surface_point[0][END][z]`. It is assumed that the depths should all be equal, as the planar surface is vertical (i.e. normal to the projector axis). The fourth column shows the arithmetic mean of the error between the calculated depth and the assumed depth for each sample in the range, and the fifth column gives the standard deviation of those sample errors. The sixth column repeats the Standard Deviation results for the Single Stripe Scanner, from Table 4.4.1 as a comparison (the sample ranges are slightly different).

test	START	END	mean error (mm) MSS	stand. dev. MSS	stand. dev. SSS
1	50	60	-0.032	0.104	0.122
2	350	360	-0.005	0.322	0.068
3	740	750	0.065	0.077	0.244
4	10	740	0.007	0.179	0.729
5	10	380	-0.024	0.174	0.629
6	400	770	0.040	0.174	0.773

Table 9.4.2: Mean error and standard deviation for planar surface.

The comparison between the Single Stripe Scanner and Multiple Stripe Scanner results shows that there is a marked improvement in the larger sample tests 4, 5 and 6 with the Multiple Stripe Scanner (MSS) compared with the Single Stripe Scanner (SSS). The most likely reason for this is the improved calibration techniques used.

Note from Section 4.4 that if the Mean Error values are significantly lower than the Standard Deviation values, this may be due to quantization noise, and this is certainly true in

Tests 1, 2, 4, 5 and 6, Table 9.4.2. Indeed, because in these tests the Mean Error values are very low, it suggests that quantization noise is now the major factor. These results for the central stripe $n = 0$ should be related to Figure 9.3.2 which visualizes the whole scanned surface, and shows much less linearity at the "corners" of the scan, where inaccurate modelling of the Radial Distortion Function (Section 5.6.3) is likely to have a greater effect. For the reasons stated in Section 9.3.2 these inaccuracies will not be investigated further in this thesis.

A crucial difference between the horizontal and vertical measurements which is not evident in Table 9.4.2 is the difference in spacing between vertices horizontally and vertically. Vertically, for both the single and multiple stripe scanner, the vertices are spaced at one per pixel as seen in the recorded image, i.e. for a 768 row CCD viewing the 350 mm test surface the spacing is 0.455 mm. For a 100 mm high surface the spacing would be 0.130 mm, a resolution of 1 part in 768, which is in line with typical industrial values. Similarly, the depth z as seen in test 1 has a mean error of 0.104 mm, which if compared to a viewing volume depth of 100 mm, gives a resolution of 1 part in 961. Horizontally the spacing is much wider - the width W between stripes as they intersect the X-axis of the system (which is where the test surface is placed) is 1.93 mm. This spacing is halved because the "black" stripes are also used, giving a horizontal spacing between vertices of 0.965 mm over a scanned surface which is 270 mm wide. This gives a horizontal resolution of 1 part in 280, a much lower value than for the other two dimensions. This is summarised as:

horizontal resolution in X	1:280
vertical resolution in Y	1:768
depth resolution in Z	1:961

9.4.3. Summary of Tests on System Geometry

To improve the horizontal resolution would require the stripes to be closer together, which would not only have implications in the accuracy of the subpixel estimator, but would also increase the likelihood of wrongly indexing adjacent stripes. Other results are in line with expectations both from industry standards and in comparison with the Single Stripe Scanner. It has also been shown that calibration can achieve the required accuracy by the methods described in Section 5.6.

Although geometric measurement does not directly relate to the Indexing Problem, it is important to fully understand the accuracy and limitations of the system measurements in order to improve the design of indexing algorithms, especially in relation to geometric constraints.

9.5. Results of Tests Using Indexing Algorithms

As described in the Introduction, the task of the indexing algorithms is to find the index value of stripe elements in the recorded image. The two principal measures of success are ***patch size*** - what proportion of the stripe elements have been indexed - and ***indexing accuracy*** - what proportion of the indexed stripe elements have been indexed correctly. The patch size measure is relatively easy to implement, being a ratio between the number of indexed elements and the number of peaks found in the recorded image (refer to Chapter 5 for definitions of data types and processes). However, the notion of "correct" indexing is harder to define, as there is no independent measure of the set of indices in the way that there is of the set of surface vertices, which can be compared to known objects such as planes or spheres. Indeed, simple forms such as planes and spheres are inappropriate as tests for indexing accuracy as they are likely to be indexed successfully, having none of the problem shapes such as occlusions. To show this in practice, three shape types are tested and evaluated: planar surfaces, spheres and "surfaces with occlusions".

9.5.1. Creating the Template

In order to test and evaluate shapes with occluded surface areas, the test object should include both camera and projector occlusions. A set of "correct indices" is then needed for this test object, against which to evaluate the algorithms. The test object is the 300 mm high sculpted head which is used throughout the investigations. The set of correct indices is called the ***template***, and this is created by the following process:

1. record the projected stripe pattern,

2. edit the recorded image by hand so that occlusion boundaries are clearly defined,

3. run the Flood Fill algorithm to index the stripes,

4. check the results by eye for any "intuitively obvious" indexing errors,

5. if errors are found, make appropriate changes and return to step 2,

6. store the completed index array as the ***template***.

This process effectively means that the template is created by hand using our prior knowledge of the object shape, such as the occlusion at the nose. The use of the Flood Fill algorithm is merely an aid to speed up what would be a laborious process by hand.

9.5.2. Comparing the Template to the Algorithm Outputs

The comparison between the template and the outputs produced by the tested algorithm uses the index array which, recalling Section 5.5.3, gives $index(c,r) = \{n, \text{NULL}\}$ where (c,r) is the column and row of the pixel to be indexed, n is the index at that pixel, and an unindexed pixel is given the value NULL. For the tests, the indices in the template are given by $template(c,r) = \{n, \text{NULL}\}$.

The indices are compared on a pixel-by-pixel basis between the template and the test outputs, and the following sets are then produced from the recorded image, represented by the pixel set $P = C \times R$:

- T, the "template set" of elements t, so that

$$t \in P = C \times R \bullet template(c,r) \neq NULL$$

- A, the "algorithm output" set of elements a, so that

$$a \in P = C \times R \bullet index(c,r) \neq NULL$$

- I, "corresponding indices", i.e. the set of elements i where the template and test pixels have the same non-NULL index, so that

$$i \in P = C \times R \bullet (template(c,r) = index(c,r)) \wedge (index(c,r) \neq NULL),$$

- M, "miscorresponding indices", i.e. the set of elements m where the template and test pixels have different non-NULL indices, so that

$$m \in P = C \times R \bullet (template(c,r) \neq index(c,r)) \wedge (index(c,r) \neq NULL) \wedge (template(c,r) \neq NULL)$$

- X, "non-corresponding indices", i.e. the set of elements x where a the test pixel has been indexed (is not NULL) and the corresponding template pixel has not been indexed (is NULL), so that

$$x \in P = C \times R \bullet (index(c,r) \neq NULL) \wedge (template(c,r) = NULL).$$

A further set F, the "full patch" of all peaks, whether indexed or not, is given by

$$f \in P = C \times R \bullet peak(c,r) = TRUE.$$

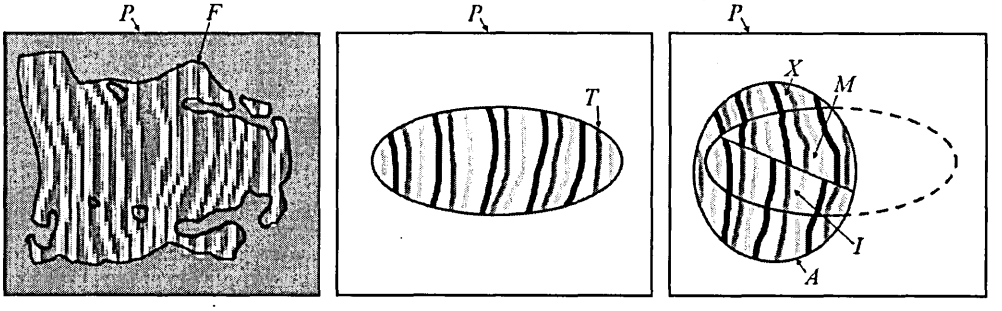

Figure 9.5.2: Indexed sets for template comparisons.

Figure 9.5.2 gives a visual summary, showing at (a) the recorded image stored as pixel set P, with the full set F of all peaks outlined in black. At (b) the recorded image has been indexed to produce the template set T, given an arbitrary colouring scheme as shown. The template set contains only those peaks which, with prior knowledge, we can be sure are part of the target surface, which in this case has an elliptical shape. Note that some of the peaks in (a) are rejected by the template. At (c) the same recorded image has been indexed using the algorithm under test, producing the test set A. The area of set I can be seen to have the same indexing values as the corresponding area in T, while in M the corresponding area of T is indexed differently. In the

area of set X there are no corresponding indexed elements in T. This synthetic example suggests some useful measures which can be found:

$$\text{total patch size } \% = \frac{100(\#A)}{\#F} = \frac{100(\#X + \#M + \#I)}{\#F},$$

Eq. 9.5.3

$$\text{accurate patch size } \% = \frac{100(\#I)}{\#F},$$

Eq. 9.5.4

$$\text{indexing accuracy } \% = \frac{100\#I}{\#A} = \frac{100(\#I)}{\#X + \#M + \#I}.$$

Eq. 9.5.5

These equations, particularly the *indexing accuracy* and *accurate patch size*, will be used as a starting point for evaluation of the indexing algorithms.

9.6. Testing the Indexing Algorithms

9.6.1. Planar Surfaces

The tests conducted in Section 9.4.2 to compare single and multiple stripe scanners use a planar surface indexed by the Flood Fill method, and it is possible to evaluate the outputs using the patch size and indexing accuracy equations Eqs. 9.5.2, 9.5.3 and 9.5.4. Table 9.6.1 gives the result for the planar surface using the Flood Fill method. Here the accurate patch size (*aps*) is very high, because very few peaks have not been indexed; only those at the extreme right and left in Figure 9.3.4. Also, the indexing accuracy (*ia*) is 100 %; every indexed peak has been done so successfully.

Test	**Occl-usion**	**Method**	***F***	***I***	***M***	***X***	***A***	***aps*** **%**	***ia*** **%**
7	NO	flood fill	147917	144077	0	0	144077	97.4	100

Table 9.6.1: Measures for Flood Fill method on planar surface.

This result shows that for a typical planar surface, given an adequate level of preprocessing as described in Section 5.5.1, the accurate patch size will be a very high proportion of the total recorded data, and indexing accuracy will be very high.

9.6.2. Spherical Surfaces

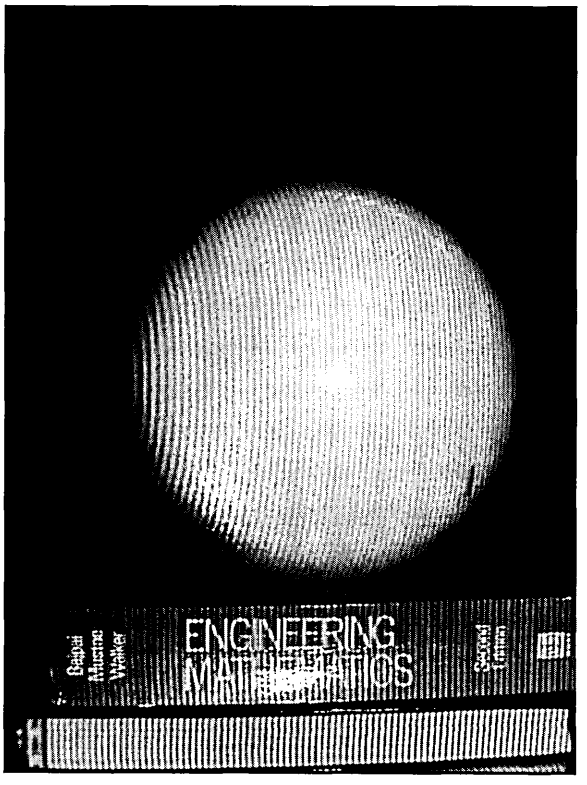

Figure 9.6.2: Recording of spherical surface.

Figure 9.6.2 shows a recorded image from the scanning of a white spherical light fitting, sitting on two textbooks. Although the sphere has a number of badly presented areas such as the highlights and marks on the surface, the visualisation of Figure 9.6.3 again shows a good degree of coverage and accuracy. The Flood Fill method was used, which stopped the search at the edge of the sphere, thus avoiding the textbooks. The highlight at the centre, and the two vertical marks to the top and bottom right, have not been indexed. The visualisation of Figure 9.6.3

shows to the left the view down the *X*-axis and to the right the view down the *Z*-axis, of the point cloud of vertices.

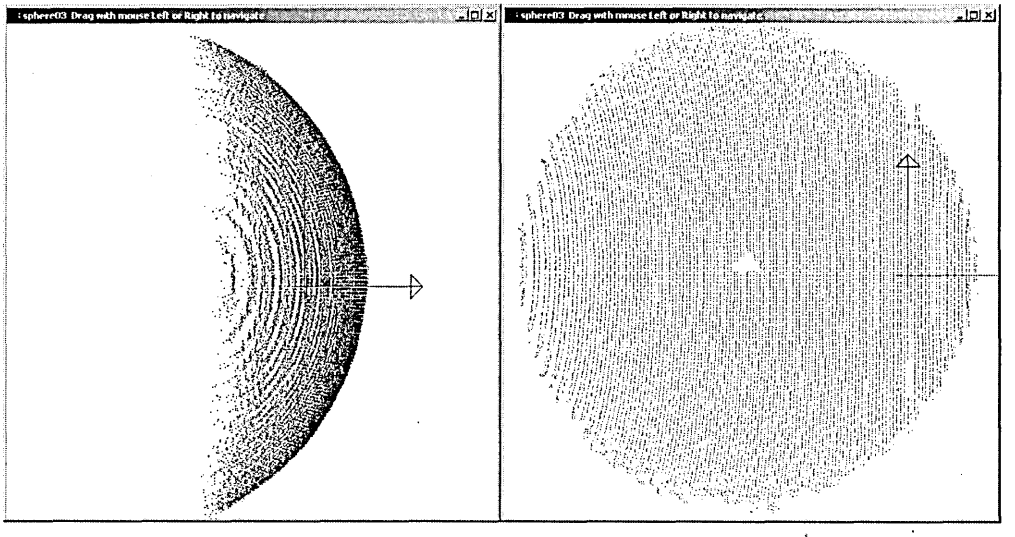

Figure 9.6.3: Visualisation of scanned spherical surface.

Table 9.6.2 shows the standard test method, where the full patch *F* excludes the surface of the text books. **As with the planar surface, for a part of a spherical surface, the accurate patch size has very good coverage and the indexing accuracy is very high.**

Test	**Occl-usion**	**Method**	***F***	***I***	***M***	***X***	***A***	***aps %***	***ia %***
8	NO	flood fill	32410	29493	0	371	29864	92.1	98.8

Table 9.6.2: Measures for scanned spherical surface.

A further observation is that the introduction of a coding scheme would be unlikely to improve the indexing accuracy, as there are no errors in set *M*, the miscorresponding indices.

9.6.3. Occluded Surfaces

In order to test and evaluate occluded surfaces the test object and template set are used. In Section 7.10 three types of algorithm were described: *Scanline*, *Scan and Search* and *Flood Filler*, and these are now tested using the evaluation method outlined above for the test object. The question posed at the end of Section 7.18 was whether an understanding of occlusions can help to improve the indexing algorithms. In Section 8.7 a method was devised for detecting and marking occlusions in the recorded image, and this method will also be tested.

Test	Occlusion	Method	*F*	*I*	*M*	*X*	*A*	*aps* %	*ia* %
9	NO	scanline	28400	19188	3004	781	22973	67.6	83.5
10	YES	scanline	28400	19201	2660	371	22232	67.6	86.4
11	NO	scans'rch	28400	18681	2304	362	21347	65.8	87.5
12	YES	scans'rch	28400	17332	0	0	17332	61.0	100.0
13	NO	floodfill	28400	26120	5188	14333	45641	92.0	57.2
14	YES	floodfill	28400	25668	2175	31	27874	90.4	92.1

Table 9.6.3: Testing algorithms with and without occlusion detection.

Table 9.6.3 shows the results of testing the indexing algorithms against the template, with and without the added constraints of the occlusion marker. These results are visualised in Figure 9.6.3 which shows the surface representation of the output model, with the arrows indicating the main areas where indexing errors occur.

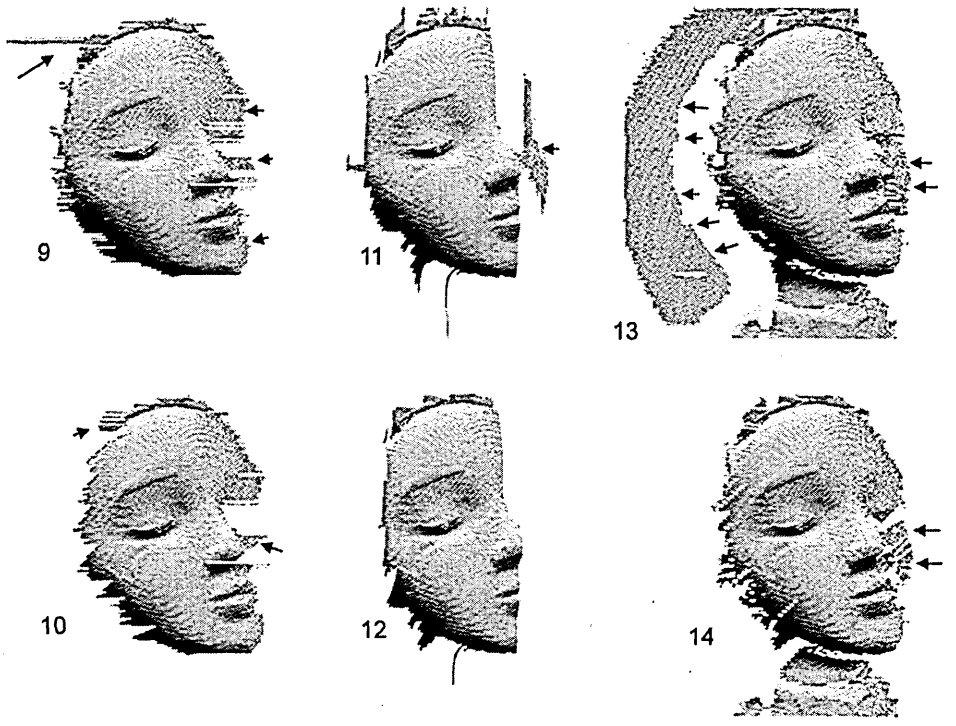

Figure 9.6.3: Visualisation of tests 9 to 14.

9.7. Evaluation of Tests on Occluded Surfaces

For a detailed description of the algorithms used here, refer to Section 7.10. Tests 7 and 8 relate to planar and spherical surfaces, and are discussed in Sections 9.4 to 9.6. The indexing accuracy for both surfaces is >98% and the accurate patch size is >92%. However, for tests 9-14 on occluded surfaces these measures are less impressive, and need further comment.

Test 9. The Scanline method searches along each row in turn until a stopping condition is met, starting from the centre of the image. This gives the characteristic horizontal gaps at the end of rows where an early stopping condition is found. Also, once an indexing error occurs on a particular row, it will continue up to the stopping conditions; this is the cause of all the arrowed errors in this test. The indexing accuracy (*ia* = 83.5%) is fairly good, although the accurate patch size (*aps* = 67.6) is low due to the limitations of the search, which cannot continue on a row after a stopping condition has been met.

Test 10. With the addition of occlusion detection some occlusions are found at the top left of the head, and around the nose, which slightly improve the indexing accuracy (*ia* = 86.4%).

Test 11. The Scan and Search method searches along the centre row, starting at the centre of the image, and each time the centre of a stripe is encountered, that stripe is searched from top to bottom. This method is not prone to the Scanline problem of the index changing along the length of a stripe. However the error which is arrowed is a typical problem whereby the search carries over an occlusion (at the nose) to produce a miscorrespondence. The results are similar to those for the Scanline method, with a slightly improved indexing accuracy (*ia* = 87.5) because of the issue described above.

Test 12. The occlusion detection finds this occlusion at the nose, thereby stopping the search at that point. **This gives the most significant result of all six tests - that the indexing accuracy is now 100%, although the patch size has dropped to 61.0%.**

Test 13. The Flood Fill method will cover every pixel within a bounded area (i.e. bounded by the stopping conditions). The disadvantage is that any "leak" across an occlusion will spill out to fill the whole of the wrongly indexed region, as seen in Figure 9.6.3. This means that the accurate patch size is very large (*aps* = 92.0), but the indexing accuracy (*ia* = 57.2) is very low; this is an opposite result to that of Test 12.

Test 14. The occlusion detection reduces the likelihood of a spillage over an occlusion, and greatly improves the results for indexing accuracy (*ia* = 92.1) while only slightly reducing the accurate patch size (*aps* = 90.4).

9.8. Summary of Tests and Evaluations

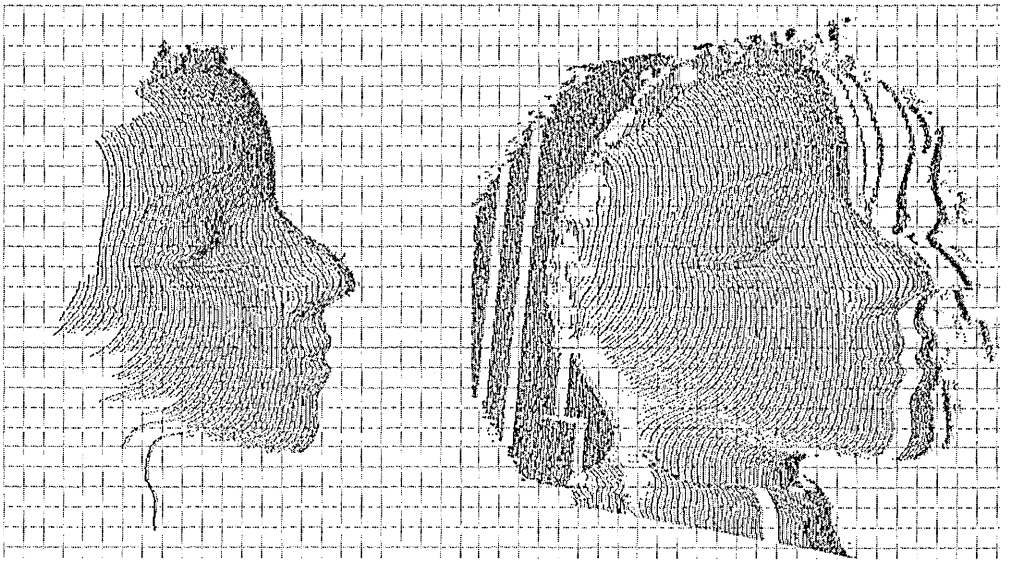

Figure 9.8.1: Comparison between tests 12 (left) and 13 (right).

As a final visualisation of the results, Figure 9.8.1 shows Test 12 (left) and Test 13 (right), visualised as point clouds of vertices. As discussed above, Test 12 gives a completely accurate but small patch, whereas Test 13 gives a larger patch but with many indexing errors. The errors are clearly seen as "fault lines" where a patch of surface is translated away from its true position - to the right of the face and in the background. These two models represent opposite answers to the "patch size *v.* indexing accuracy" question which is discussed in Section 7.1.1, and either may be valid depending on the application.

Occlusions will present indexing problems unless the algorithms are heavily constrained using occlusion detection as in test 12. However, surfaces with no occluded areas, such as the planar surface and sphere portion tested here can often be indexed with no errors, as long as the data is preprocessed in a satisfactory way.

9.9. Registration and Fusion

The 3D surface model typically consists of a "patch" (see Section 7.8) of vertices located in a Cartesian Coordinate System (Sections 5.2), and two separately created patches can each be considered to have their own local coordinate system. If these two patches represent parts of the same target object, it may be useful to attempt to piece them together, in the way that an archaeologist pieces together two shards of pottery from the same pot. The pottery shards must be from separate parts of the surface of the pot, possibly touching; however the two 3D patches may overlap. Piecing together the two patches requires two processes: ***registration***, whereby the optimal point-to-point image correspondence is found and the transformation parameters (rotation matrix and translation vector) are estimated such that the two images can be spatially aligned, and ***fusion***, whereby the two sets of vertices are combined in the same reconstructed surface. The combined registration and fusion processes are described by [Rodrigues *et al.*, 2002], and these methods, together with the *tripod* method of registration [Rodrigues and Robinson, 2004], are used in the examples given below. Further references to this large research area are given in Chapter 2, Section 2.2.

9.9.1. Categorization of patches

The scanning methods described in this thesis suggest three useful categories of relationship between patches:

- **Disoriented patches**, if there is no knowledge about the relative orientation. This will be the case if two scans have been made of the target object with the object (or the scanner) being moved freely by hand between the scans. The registration methods [Besl and McKay, 1992] [Rodrigues *et al.*, 2002] require a degree of overlap between the two patches, and a number of recognizable features which are common to both patches.

- **Intersecting patches**, if the two patches are from the same scan, i.e. the same recorded image, and some of the peaks are common to both patches. This will be the case if two different indexing methods have been used with, say, different Patching Conditions (see Section 7.11) and starting from different locations in the peak array. The same registration methods can be used if there are sufficient common features, but the

orientation can also be expressed more simply as a *relative indexing parameter.* If the indices of the two patches are correct then they will be in the correct relative orientation; otherwise the two patches can be moved into the correct orientation simply by incrementing or decrementing the relative indices of the patches. This is likely to be a simpler task than the general registration methods.

- **Disjoint patches**, if the two patches are from the same scan, i.e. the same recorded image, and none of the peaks are common to both patches. This will be the case if the Patching Conditions prevent any overlap between the two patches. Here it will be necessary to use the relative indexing parameter, due to the lack of an overlap and the consequent absence of any common features.

Figure 9.9.1 shows the registration (denoted by "R") and fusion (denoted by "F") of the test object using four source patches: *1a* and *1b* which are *intersecting* patches from the same recorded image, and *2a* and *2b,* which are *intersecting* patches from a second recorded image. *1a* and *1b* are therefore *disoriented* with respect to *2a* and *2b*.

9.9.2. Visual evaluation of registration and fusion examples

1a and *1b* are registered and fused to give patch *1f*, and *2a* and *2b* are registered and fused to give patch *2f*. *1f* and *2f* are then registered and fused to give the *final* patch, which includes vertices from all four source patches.

At this stage these fused patches have not been subjected to the evaluation methods described in this chapter; we simply note that visually the patches appear to fit together reasonably well. As described in Section 9.9.1, it should be an easier task to register the intersecting patches than the disoriented patches, and certainly a visual inspection cannot find any errors in the registration of intersecting patches *1a+1b*, or of *2a+2b*. However, there are similarly no obvious visual errors in the registration of the two disoriented patches *1f* and *2f.*

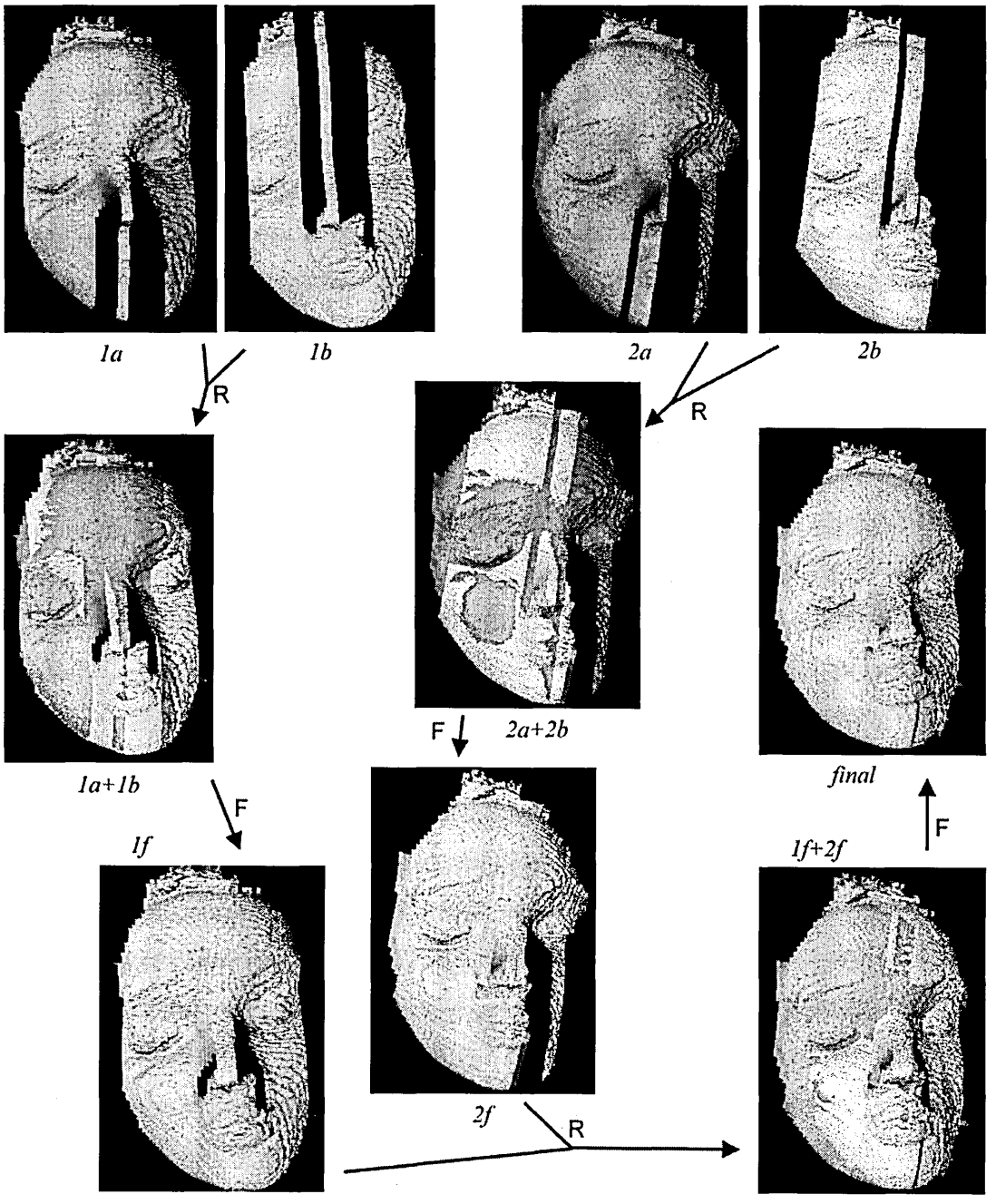

Figure 9.9.1: The registration and fusion of patches.

Chapter 10: Conclusions

The main objective of this thesis is to record three-dimensional surfaces, as quickly, fully and accurately as possible. This is a task which many people have tackled, and a large and varied toolkit of methods is now available to the Machine Vision community. It is unlikely that a revolutionary new process will emerge to replace all the existing methods, and so most workers, including ourselves, are looking at incremental developments of the current technologies.

Our particular approach has been to examine how far the Indexing Problem can be solved for Structured Light Scanners without recourse to coding schemes; this is not to make the coding tool redundant, but to show that in many cases it is unnecessary. Further than that, we begin an investigation into the relationship between surface shape and the recorded projection patterns, particularly when occlusions are present, which will have value beyond the confines of Structured Light Scanning.

10.1. Original Contributions

In order to conduct the investigations, the following original contributions have been made, which are discussed at the start of each relevant chapter:

10.1.1. Geometric Constraints

We take the methods using Epipolar Geometry and Rectification and adapt them to investigate the use of Dot and Stripe Patterns in Structured Light Scanning. In particular we introduce the ***Unique Horizontal Constraint*** and determine how far this can be successfully used to solve the Stripe Indexing Problem. Furthermore we implement a method of rectifying the system by spatial positioning. As far as is known, this method has not been used by previous researchers.

10.1.2. Connections

In the peak array, ***N-, S-, E- and W-Neighbours*** are defined, leading to definitions of ***Stripe Paths*** and ***Neighbourhoods***. These types form the basis of descriptions of ***Patches***, and are used as ***Patching Conditions*** in the novel algorithms which have been designed to successfully index the stripes.

These novel indexing algorithms comprise ***Scanline***, ***Scan and Search*** and ***Flood Fill*** methods, which adapt and extend existing search methods used in Pattern Recognition and Segmentation applications. Here we also pose the ***Size v. Accuracy Question***: "For a given scanning application, what is the relative importance of indexed patch size compared with indexing accuracy", and this question must be addressed by the application designer.

An open issue is the complexity of the algorithms. The estimated upper bound of the Scanline and Scan and Search algorithms is $O(n^2)$ where n is the maximum number of rows in the bitmap image. This value can be improved. The complexity of the Flood Fill algorithm is under investigation.

10.1.3. Measures of Success

In order to test and evaluate the results, all the indexing methods are executed using a *test object*, which is compared to a hand-produced *template* result. Three novel measures: ***total patch size***, ***accurate patch size*** and ***indexing accuracy*** are used to evaluate the new methods. These measures overcome the difficulty of assessing the success of scanning a surface with occlusions.

10.1.4. Occlusions

To overcome the well-known problems caused by occlusions, we make specific definitions which relate occluded areas on the target surface to the resulting recorded image: ***Near Occlusion Points***, ***Far Occlusion Points***, ***Projector Occlusion Points*** and ***Camera Occlusion Points***. From these definitions four ***boundary types*** are categorized: the ***Concave Pair***, the ***Contour Single***, the ***Concave Tuple*** and the ***Contour Tuple***. The investigation then looks specifically at how Concave Pairs can be identified in the recorded image, and thence designs a

novel algorithm to ***detect and mark the occlusion boundary***. This is added to the Patching Conditions, giving improved levels of indexing accuracy.

These investigations relate to occlusions which are not horizontal. In addition, a theoretical instance has been designed which shows that it is possible to have ***exactly the same spacing between stripes, above and below a horizontal occlusion***. This means that spacing between stripes cannot be readily used as an indicator of the presence of a horizontal occlusion.

10.2. The Research Questions

The principal research questions, described in the Introduction, are:

- How can we correctly identify in the sensor a specific projected pattern element, expressed as an index? This is the Indexing Problem.
- How far can the Indexing Problem be solved using an uncoded projection pattern?

In order to answer these questions, five primary investigation topics were educed:

- The use of geometric constraints, and in particular Epipolar Geometry, to alleviate the Indexing Problem.
- A comparison between the benefits of two dimensionally discrete patterns such as dots, and patterns such as stripes which are discrete in one dimension and continuous in the other.
- Algorithms for recognizing and indexing the pattern elements, and defining the boundaries of the target surface.
- Understanding the relevance of occlusions to the Indexing Problem.
- Testing the Success of the Indexing Methods.

These topics are described in chronological order based on their position in the development of the thesis; for instance an understanding of the geometry of the system was progenitive to the debate between using dot or stripe patterns. Similarly the development of the indexing algorithms depended on an understanding of the system geometry and the choice of projection patterns. Occlusion detection was developed as a result of problems found in the early indexing

methods, and all of the above required suitable measures of success in order to have confidence in their validity.

As a result of these investigations a number of findings have been made, which are listed here in order of importance rather than chronology:

1. that without the presence of occlusions, and given well-presented input data, the Indexing Problem can be successfully solved using uncoded patterns,

2. that in the presence of occlusions, errors will occur in the indexing process,

3. that if occlusion detection is included in the process, the errors will be reduced,

4. and that measures of *patch size* and *indexing accuracy* give a useful indication of the success of the methods under investigation.

Other findings are

5. that methods can be designed which give either large patch areas, or greater indexing accuracy,

6. that one-dimensionally discrete patterns such as stripes have advantages over two-dimensionally discrete patterns such as dots,

7. that epipolar geometric constraints can be used to simplify the calibration process and to assist the index searching process,

8. that the spacing between pattern elements cannot be successfully used to help solve the Indexing Problem.

10.3. Principal Investigation Findings

Now the principal investigation findings are discussed, with reference to the section of text in which they appear.

10.3.1. Finding 1:

"Without the presence of occlusions, and given well-presented input data, the Indexing Problem can be successfully solved using uncoded patterns" (Section 7.10).

The meaning of "successfully solved" is discussed below in Section 10.3.4; the meaning of "well-presented input data" refers to the assumptions made in Section 5.5.1. Here five important aspects of the way the image is stored by the pixel array are noted: focus, aperture control, ambient light, unwanted artefacts and other noise. Experiments have found that it is possible to set up the system so that these aspects do not affect the data output.

One application of 3D scanning will be where the target surface is smoothly undulating, with no occluded areas and few if any recognizable features. These surfaces are often unsuitable for stereo vision methods, which require a dense set of features in order to create the model, but the results here give good coverage using the two defined measures: patch size and indexing accuracy. For both the planar surface and the sphere portion both measures give patch sizes of > 90% and indexing accuracy of >98% (see Section 9.6).

More practical examples are shown in Figure 10.3.1, where on the left the car body part (courtesy Jaguar Cars Ltd.) gives a patch size of 93.0% and an indexing accuracy of 100%, and on the right the crumpled sheet of paper gives a patch size of 98.7% and an indexing accuracy of 99.4%.

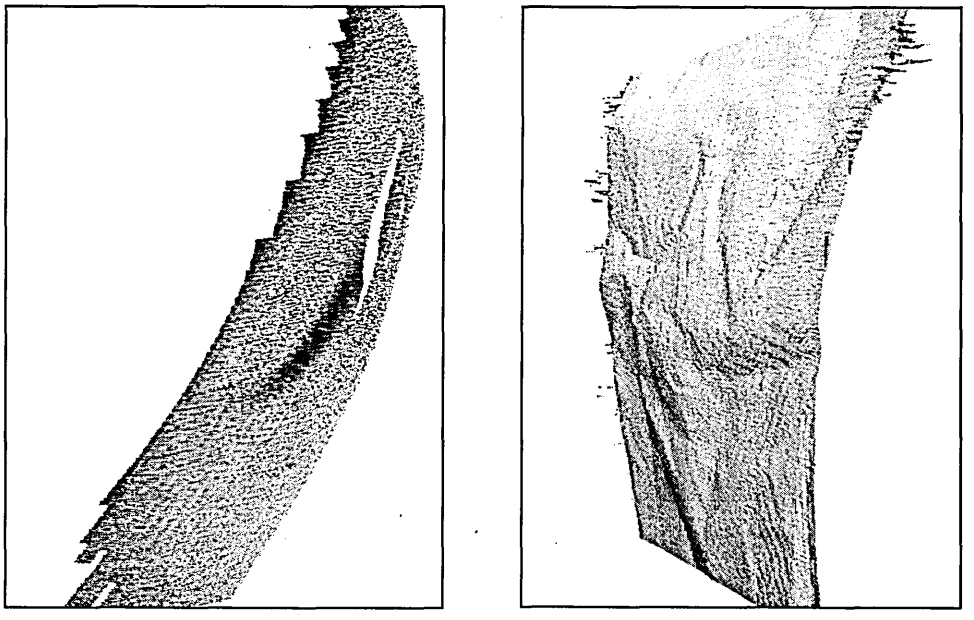

Figure 10.3.1: Visualisation of car body part (left, Courtesy jaguar Cars Ltd.) and crumpled paper sheet (right).

The car body part scan has an application in reverse engineering, where the main alternative method is to use optical markers and stereo vision, which limits the point cloud to the number of optical markers used. Whichever system is used, there is a major task of calibration to ensure that the geometric measurements are sufficiently accurate.

One proviso of the indexing methods used in this scanning system is that the indices must have absolute rather than relative values, and this is achieved by ensuring that the leftmost stripe is visible at some point in the recorded image. If a particular application does not allow this, then an alternative is to specially mark the stripe at the centre of the pattern, perhaps by making it brighter than the others. This is the first step towards combining uncoded and coded patterns, where a small amount of coding is used for a particular task.

This investigation shows that Structured Light Scanners are ideally suited to applications where there are smoothly undulating forms with no occlusions, and few recognizable features. The level of coverage and indexing accuracy will be close to 100%, so that the overall measurement accuracy will depend entirely upon the calibration process.

10.3.2. Finding 2:

"In the presence of occlusions, errors will occur in the indexing process" (Section 7.18)

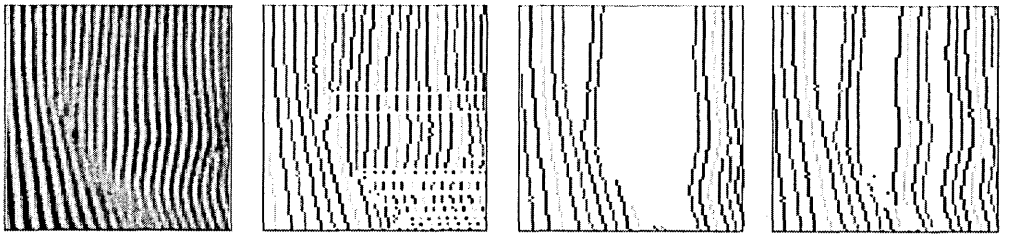

Figure 10.3.2: Indexing of image (a) using (b) Scanline, (c) Scan and Search and (d) Flood Fill methods.

A detail of the scanning results using the test object is shown in Figure 10.3.2. At (a) we see the recorded image, and then the indexing results using the Scanline method (b), the Scan and Search method (c) and the Flood Fill method (d). Each of these methods has errors due to the occlusion at the bridge of the nose: the Scanline method has a horizontal banding effect as the indices move out of phase with their "correct" stripe, the Scan and Search method correctly

stops at the edge of the nose apart from where the two stripes continue across the nose occlusion, and the Flood Fill method makes four errors in crossing the nose occlusion.

Here also the conflict between patch size and accuracy can be seen: the Scanline method has the largest correctly indexed patch but also the largest incorrectly indexed patch. The results in Section 9.6 show that it is possible to choose a particular method which is likely to give a larger patch area and lower indexing accuracy, or *vice versa.*

Overall, the methods adopted in this thesis, without occlusion detection, have all included errors if occlusions are present from either the camera or projector viewpoint.

10.3.3. Finding 3:

"If occlusion detection is included in the process, the indexing errors will be reduced". (Section 8.7.2)

The thesis has sought logical reasons for constraining the algorithms in a particular way, rather than simply a trial-and-error approach of trying a modification and hoping for better outputs. The main problem area has been identified as the presence of occlusions, and an investigation has been undertaken to understand how the pattern elements (particularly stripes) behave when they are projected onto occluded areas.

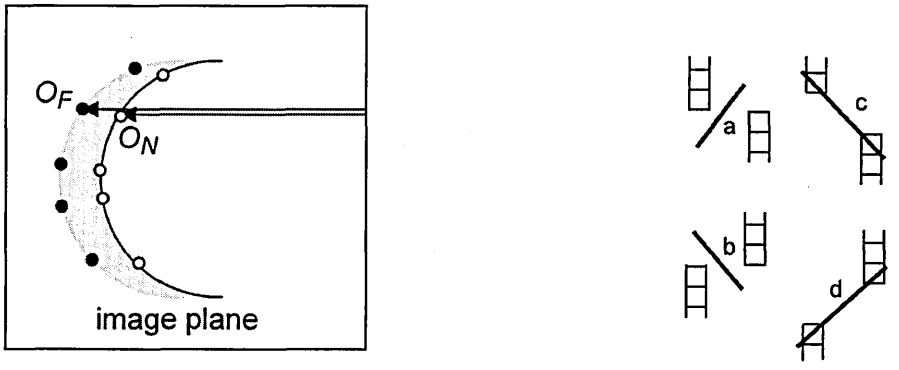

Figure 10.3.3: Concave Pair (left) and four categories (right).

The initial ploy is to constrain the system using the Epipolar Constraint, a new version having been devised for structured light methods, so that a particular pattern element must appear somewhere along a known horizontal line. Figure 10.3.3 (left) shows this situation, which

means that two closely positioned pattern elements, one of which, O_N, is on the "near" edge of an occlusion and one of which, O_F, is on the "far" edge, must be situated on adjacent rows in the recorded image. Given this "Horizontal Constraint", we have then defined more precisely where these near and far occlusion points will be, depending on the type of occlusion. Figure 10.3.3 (right) shows the four occlusion categories *a, b, c* and *d* which identify where the occlusion will be, based on the relative positions of the disconnected stripes. The likely occlusion boundaries are denoted with black lines.

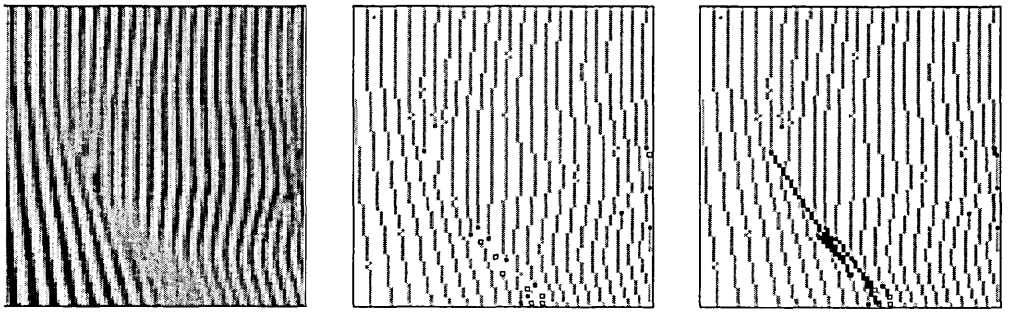

Figure 10.3.4: Recorded image (left), disconnections (centre) and marked occlusions (right).

Figure 10.3.4 shows the occlusion detector and marker in operation, with the recorded image of Figure 10.3.2 shown at (a). The peaks at the centre of each stripe are shown at (b), with downward disconnections *D* marked as black squares and upwards disconnections *U* marked as white squares. The occlusions, calculated by the method to be of category *c* rather than *a*, are marked in (c) as black lines, and these are then added to the stopping conditions of the indexing algorithms.

The results (see Section 9.6.3) show that the addition of these occlusion boundaries to the stopping conditions significantly improves the indexing algorithms, in terms of indexing accuracy. In Table 9.6.3 the indexing accuracy for the Scan and Search method improves from 87.5% to 100 %, and for the Flood Fill method from 57.2% to 92.1%. Conversely, the patch size is slightly reduced under the imposition of occlusion constraints, from 65.8% to 61.0% for the Scan and Search method, and from 92.0% to 90.4% for the Flood Fill method. Once again a decision will need to be made between patch size and indexing accuracy.

The work on occlusion definitions, detection and marking is in its infancy; even so it has produced good results, and some thoughts on future research are included below.

10.3.4. Finding 4:

"Measures of *accurate patch size* and *indexing accuracy* give a useful indication of the success of the methods under investigation". (Section 9.5.2)

A major issue in this thesis has been how to evaluate the success of the indexing methods when dealing with occluded surfaces. It is likely to be difficult to measure such a surface by hand as a comparison with the model created by the method under test. In the chapter on Results and Evaluation, a template data set has been used which is derived from the target object, the sculpted head, against which outputs from the indexing methods are compared.

Using the template as a comparison, two measures have been created which, in broad terms, separate the correctly indexed vertices from the incorrectly indexed vertices in the method under test. These two measures are *accurate patch size*, which gives the percentage of correctly indexed vertices compared with all the vertices which are present in the recorded image, and *indexing accuracy* which gives the percentage of correctly indexed patches compared with all the indexed vertices. The discussion in relation to the methods illustrated in Figure 10.3.1 gives an indication of the significance of the measures. The definition of "success" is to some extent arbitrary, but in these tests a level of >90% is appropriate for the *indexing accuracy*. The *accurate patch size* may be considerably lower depending on the application. If patch registration (see Section 10.6) is employed, then a level of < 50% may be acceptable.

Reference is made in the Introduction to the strategy of White Box testing and Black Box testing, and these two measures are especially useful in relation to Black Box testing. In this case changes are made to a method on a trial-and-error basis without necessarily having a logical reason for the change, and whether the change is permanently introduced is dependent on the patch size and indexing accuracy of the results. An example of this is seen in Figure 10.3.2 where it is not clear by logical deduction that the Scan and Search method will give greater indexing accuracy than the Flood Fill method. However, the tests and evaluations show this to be true.

Using the White Box strategy, it is more likely that there is a logical reason for the change, and it will often be provable using mathematics rather than testing. Examples of this are

the Horizontal Constraint (Section 6.3) and the investigation into the relationship between horizontal occlusions and stripe spacing (Section 8.9).

This investigation shows that the measures of *accurate patch size* and *indexing accuracy* can be used to usefully evaluate the indexing methods, and will give indications of which methods will be most suited to particular applications.

10.4. Other Relevant Findings

10.4.1. Dots *versus* Stripes.

The relative advantages of using a two-dimensionally discrete projection pattern such as dots or a grid, compared with a stripe pattern which is discrete in one dimension and continuous in the other, have been extensively investigated. The investigation has looked at how each pattern type can be used in conjunction with the Epipolar Constraints to simplify the search for pattern elements in the recorded image; it has concluded that in theory the dot pattern can give a unique range of positions for each dot in the recorded image, but that the resolution of the system in order to implement this condition would be unfeasibly high.

Countering the advantage of the discrete dot pattern is that the continuous stripes in say, the vertical direction, would allow a sampling of the stripe on each row in the pixel array, giving the maximum possible resolution.

It is concluded that for these reasons the stripe pattern is the most suitable for these applications.

10.4.2. The spacing of stripes on either side of a horizontal occlusion.

A theoretical investigation has been undertaken to decide whether the spacing between stripes will be inevitably be different on either side of a horizontal occlusion, which would help in the identification of the occlusion. It has been proven that this is not true, by finding a spatial setup

of the system in which the stripes will be projected identically onto the image plane from two differently positioned surface planes.

10.5. Summary

This thesis has shown that Structured Light Scanners using uncoded stripe patterns can be successfully used to measure 3D surfaces quickly and accurately. A particular application is where the surfaces are unoccluded and featureless, and this type of surface may be difficult for other scanning types to measure quickly, i.e. in a single video timeframe.

The method used by this system requires the successful indexing of all vertices, and in this quest definitions and algorithms have been created to recognize pattern elements in the image data. A problem is acknowledged in the presence of occluded surfaces, but by defining, detecting and marking occlusions many of these obstacles can be overcome.

Care has been taken to ensure that the testing and evaluation of the methods gives a useful interpretation of the outputs. This has shown that a choice can be made between measuring a large patch with some errors, and measuring a smaller patch with a high degree of accuracy.

10.6. Future Research

A number of areas have been identified where problems still exist and further research is required, such as the detection of occlusions, the incorporation of a limited amount of coding, and the joining together of disconnected patches.

10.6.1. Occlusion handling

The investigations relating to occlusions are at a comparatively early stage, and it is likely that further work in this area will provide still more improvements in the accuracy of the model. Whether further investigations can completely solve the occlusion problem, and avoid the need for some degree of coding, is an open question. Research is needed to develop the occlusion

detection methods so that more complex boundaries than simple straight lines can be found. Some initial attempts have been made to group together adjacent straight line boundaries and form an "envelope", which would mark an occlusion with greater validity, and enable curved lines.

A related problem which has not yet been tackled occurs when the stripes are so close together in the recorded image that they cannot be separated. This is not strictly an occlusion, but is found where the camera axis is almost tangential (i.e. supporting) to the target surface. This in turn affects the design of the stripe pattern, and how closely spaced the stripes should be. It may be possible to predict the likely path of the stripes even if they cannot be separated, as it is likely that it will be known how many stripes are contained in the specific area.

10.6.2. The introduction of a limited coding scheme.

At the same time it will be appropriate to introduce a limited amount of coding to solve the cases where the Indexing Problem remains intractable. In particular, when using the multiple stripe patterns, an "absolute indexing" method is needed (see Section 5.3). It may be that if, say, every tenth stripe is projected with greater brightness, this could be detected in the algorithms and used as an absolute marker.

10.6.3. Registration and fusion of surface patches.

In many cases it will be desirable to register two or more patches from different scanning operations, and typically this is achieved using algorithms such as the Iterative Closest Point (I.C.P.) method. It may be possible to combine the Indexing methods used here, with the I.C.P., to simplify what is often a time-consuming process. A simpler registration problem is to combine two patches from the same scan, where the stopping conditions have forbidden a connection between them. This would mean that the relationship between the two sets of indices would be unknown, but it would be one of a small number of discrete possibilities. Methods may be devised to choose the most likely relationship and thereby successfully register the patches. Some initial results of registration and fusion are given in Section 9.9.

10.6.4. Recording motion.

Finally, a longer-term goal is to incorporate this work into the modelling of moving surfaces, for instance by recording a speaking person whose face is illuminated by the projected pattern, and then deriving the point cloud for each successive frame. The modelling would not need to be done in real time if the recording was of a resolution comparable to that used in this thesis. And a possible advantage would be that comparisons and interpolations from frame to frame may be used to improve the indexing accuracy.

The 3D recording of moving surfaces would be beneficial to many applications in media, industry and medicine.

References

[Agrawala *et al.*, 1995]

Maneesh Agrawala, Andrew C. Beers, and Marc Levoy, *3D Painting on Scanned Surfaces,* Proceedings of 1995 Symposium on Interactive 3D graphics, Monterey, California, April 9-12, 1995.

[Alboul *et al.*, 2004]

Lyuba Alboul, Gilberto Echeverria and Marcos Rodrigues, *Total Absolute Curvature as a Tool for Modelling Curves and Surfaces*, Grid Generation: Theory and Applications, Moscow, 28 June-1 July, 2004.

[Arman and Aggarwal, 1993]

Farshid Arman and J. K. Aggarwal, *CAD-based vision: object recognition in cluttered range images using recognition strategies*, CVGIP: Image Understanding, Vol. 58 No. 1, pp.33-48, July 1993

[Armstrong *et al.*, 1998]

S. Armstrong, A. C. Kokaram, and P. J. W. Rayner. *A Bayesian framework for reconstructing color image data,* SPIE Conference on Bayesian Inference for Inverse Problems, pp. 21-28, July 1998.

[Ashbrook *et al.*, 1998]

A. P. Ashbrook, R. B. Fisher, C. Robertson and N. Werghi, *Segmentation of Range Data into Rigid Subsets using Surface Patches*, Proceedings of International Conference on Computer Vision, Bombay, pp. 201-206, January 1998.

[Baker and Kanade, 2001]

Simon Baker and Takeo Kanade, *Super-Resolution: Reconstruction Or Recognition?* Proceedings of the 2001 IEEE-EURASIP Workshop On Nonlinear Signal and Image Processing, 2001.

[Baribeau *et al.*, 1992]

R. Baribeau, M. Rioux, and G. Godin. *Color reflectance modeling using a polychromatic laser range sensor,* IEEE Trans. on Pattern Analysis and Machine Intelligence, 1992.

[Becker, 1995]

Markus Becker, *Signal processing for reduction of speckles in light stripe systems*, 3D Imaging and Laser-Based Systems for Metrology and Inspection, Photonics East'95, 1995.

[Belhumeur and Mumford, 1992]

Peter N. Belhumeur and David Mumford, *A Bayesian Treatment of the Stereo Correspondence Problem Using Half-Occluded Regions,* Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 1992.

[Beraldin *et al.*, 1992]

J.-A. Beraldin, M. Rioux, F. Blais, G. Godin and R. Baribeau, *Model-based calibration of a range camera*, proceedings of the 11th International Conference on Pattern Recognition pp. 163-167, The Hague, The Netherlands, August 30-September 3, 1992.

[Beraldin *et al.*, 2001]

J. Beraldin, C. Atzeni, G. Guidi, M. Pieraccini and S. Lazzari, *Establishing a Digital 3D Imaging Laboratory for Heritage Applications: First Trials*, Proceedings of the Italy-Canada 2001 Workshop on 3D Digital Imaging and Modeling Applications, Padova, Italy, April 3-4, 2001

[Besl and McKay, 1992]

Paul J. Besl and Neil D. McKay, *A Method for Registration of 3-D Shapes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14 No. 2, pp. 239-256, February 1992.

[Beyer, 1991]

H. A. Beyer, *An introduction to photogrammetric camera calibration,* invited paper, Seminaire Orasis, St. Malo, France, September 23-27, 1991.

[Blais, 2004]

F. Blais, *Review of 20 Years of Range Sensor Development*, Journal of Electronic Imaging, 13(1), pp. 231-240, NRC 46531, January 2004.

[Blake *et al.*, 1995]

A. Blake , D. McCowen , H. R. Lo and P. J. Lindsey, *Trinocular Active Range-Sensing*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15 No. 5, pp. 477-483, May 1993.

[Bondy and Murty, 1976]

J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications,* Elsevier North-Holland 1976.

[Braga-Neto and Goutsias, 2003]

Ulisses Braga-Neto and John Goutsias, *A multiscale approach to connectivity*, Computer Vision and Image Understanding, Vol. 89, Issue 1 , pp. 70-107, January 2003.

[Bregler *et al.*, 2000]

Chris Bregler, Aaron Hertzmann, and Henning Biermann, *Recovering Non-Rigid 3D Shape from Image Streams,* Proceedings of CVPR '00, 2000.

[Brown, 1971]

D.C. Brown, *Close-range camera calibration*, Photogrammetric Engineering, 1971, pp. 855-866.

[Busboom and Schalkoff, 1996]

Axel Busboom and Robert J. Schalkoff, *Direct surface parameter estimation using structured light: A predictor-corrector based approach*, Image and Vision Computing, 14(5): pp. 311-321, June 1996.

[Castellani *et al.*, 2002]

U. Castellani, S. Livatino and R. B. Fisher, *Improving Environment Modelling by Edge Occlusion Surface Completion*, Proceedings of International Symposium on 3D Data Processing Visualization and Transmission (3DPVT), Padova, Italy, pp. 672-675, June 2002.

[Cheeseman *et al.*, 1994]

P. Cheeseman, B. Kanefsky, J. Stutz, and R. Kraft. *Subpixel Resolution from Multiple Images.* Lunar and Planetary Science XXV, pp. 241-242, 1994.

[Cheeseman *et al.*, 1996]

P. Cheeseman, B. Kanefsky, R. Kraft, J. Stutz and R. Hanson, *Super-Resolved Surface Reconstruction from Multiple Images*, in Maximum Entropy and Bayesian Methods, G. R. Heidbreder (ed.), pp. 293-308, Kluwer, the Netherlands, 1996.

[Chen *et al.*, 1999]

C. S. Chen, Y. P. Hung and J. B. Cheung. *Ransac-based darces: a new approach to fast automatic registration of partially overlapping range images*, IEEE Trans. Pat. Anal. and Mach. Intel. 21(11), pp. 1229-1234, November 1999.

[Chen and Li, 2003]

S. Chen and Y. Li, *Self-recalibration of a colour-encoded light system for automated 3-D measurements*, Measurement Science and Technology, Vol. 14, No. 1, pp. 33-40, January 2003.

[Chen and Medioni, 1992]

Yang Chen and Gérard Medioni, *Object modelling by registration of multiple range images,* Image and Vision Computing, Vol. 10 No. 3, pp.145-155, April 1992.

[Chen and Medioni, 2001]

Q. Chen and G. Medioni, *Building 3-D human facemodels from two photographs,* Journal of VLSI Signal Processing Systems February 2001, Vol. 27 Issue 1-2, pp.127-140, 2001.

[Chen and Stockman, 1996]

Jin-Long Chen and George C. Stockman, *Determining Pose of 3D Objects With Curved Surfaces*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.18 No.1, pp.52-57, January 1996.

[Cipolla and Blake, 1992]

Roberto Cipolla and Andrew Blake, *Surface shape from the deformation of apparent contours,* International Journal of Computer Vision, Vol. 9 No. 2, pp.83-112, November 1992.

[Clark *et al.*, 1997]

J. Clark, E. Trucco and H.F. Cheung, *Using light polarization in laser scanning*, Image and Vision Computing, 15, pp. 107-117, 1997.

[Clarke, 1994]

T.A. Clarke, *An analysis of the properties of targets used in digital close range photogrammetric measurement*, Proceedings of SPIE Vol. 2350 Videometrics III, Boston, Massachusetts, pp. 251-260, November 1994.

[Clarke and Fryer, 1998]

T.A. Clarke and J.G. Fryer, *The Development of Camera Calibration Methods and Models*, Photogrammetric Record, 16(91), pp. 51-66, April 1998.

[Clarke *et al.*, 1998]

T.A. Clarke, L. Li and X. Wang, *3D-Net -a new real-time photogrammetric system.* ISPRS Vol. XXXII, Part 5, pp. 29-34, 1998.

[Cordier *et al.*, 2003]

F. Cordier, H. Seo, and N. Magnenat-Thalmann, *Made-to-measure technologies for online clothing store,* IEEE Computer Graphics and applications, pp. 38-48, 2003.

[Curless and Levoy, 1995]

Brian Curless and Mark Levoy, *Better Optical Triangulation through Spacetime Analysis,* Proceedings of International Conference on Computer Vision, pp. 987-994, Boston, June 1995.

[Curless and Levoy, 1996]

B. Curless and M. Levoy, *A Volumetric Method for Building Complex Models from Range Images*, SIGGRAPH `96, August 1996.

[Daley and Hassebrook, 1998]

Raymond C. Daley and Laurence G. Hassebrook, *Channel capacity model of binary encoded structured light-stripe illumination*, Applied Optics, Vol. 37, No 17, 10 June 1998.

[Davis *et al.*, 2002]

James Davis, Stephen R. Marschner, Matt Garr, and Marc Levoy, *Filling holes in complex surfaces using volumetric diffusion*, First International Symposium on 3D Data Processing, Visualization, and Transmission, Padua, Italy, June 19-21, 2002.

[Dell'Acqua and Fisher, 2002]

Fabio Dell'Acqua and Robert Fisher, *Reconstruction of planar surfaces behind occlusions in range images*, IEEE Trans. Pat. Anal. and Mach. Intel., 24(4), pp 569-575, April 2002.

[DePiero and Trivedi, 1996]

Fred W. DePiero and Mohan M. Trivedi, *3-D Computer Vision Using Structured Light: Design, Calibration, and Implementation Issues*, in Advances in Computers, Vol. 43, pp 243-278, 1996.

[Devernay *et al.*, 2002]

F. Devernay, O. Bantiche and È. Coste-Manière, *Structured light on dynamic scenes using standard stereoscopy algorithms*, RR 4477, INRIA, June 2002.

[Dornaika and Chung, 2001]

F. Dornaika and R. Chung. *An Algebraic Approach to Camera Self-calibration*, Computer Vision and Image Understanding , Vol. 83, No. 3, pp. 195-215, September 2001.

[Edelsbrunner, 2001]

Herbert Edelsbrunner, *Geometry and Topology for Mesh generation,* Cambridge University Press, 2001.

[Egmont-Petersen *et al.*, 2002]

M. Egmont-Petersen, D. De Ridder and H. Handels, *Image processing using neural networks -- a review,* Pattern Recognition, Vol. 35, No. 10, pp. 2279-2301, 2002.

[Ensisco *et al.*, 1999]

R. Enciso, J. Li, D. Fidaleo, T-Y. Kim, J-Y. Noh, and U. Neumann, *Synthesis of 3D Faces,* International Workshop on Digital and Computational Video (DCV'99), December 1999.

[Faugeras, 1993]

O.D. Faugeras, *Three-Dimensional Computer Vision, a Geometric Viewpoint,* MIT Press, 1993.

[Faugeras and Hebert, 1986]

O D Faugeras and M Hebert, *The representation, recognition, and locating of 3-d objects,* International Journal of Robotics Research, Vol. 5, No. 3, pp. 27-52, Fall 1986.

[Faugeras *et al.*, 2001]

Olivier Faugeras, Quang-Tuan Luong and Theo Papadopoulo, *The Geometry of Multiple Images,* MIT Press, 2001.

[Ferreira *et al.*, 2002]

João Filipe Ferreira, Jorge Lobo and Jorge Dias, *Tele-3D Developing a Handheld Scanner Using Structured Light Projection*, RECPAD 2002-12th Portuguese Conference on Pattern Recognition, Aveiro, Portugal, June 27th-28th, 2002.

[Fisher and Naidu, 1996]

R. B. Fisher and D. K. Naidu, *A Comparison of Algorithms for Subpixel Peak Detection*, in Sanz (ed.) Advances in Image Processing, Multimedia and Machine Vision, Springer-Verlag, Heidelberg.

[Fisher, 2001]

R. B. Fisher, *Projective ICP and Stabilizing Architectural Augmented Reality Overlays*, Proceedings of International Symposium on Virtual and Augmented Architecture (VAA01), pp. 69-80, Dublin, Ireland, June 2001.

[Fitzgerald *et al.*, 1999]

W.J. Fitzgerald, S.J. Godsill, A.C. Kokaram, and A.J. Stark, *Bayesian methods in signal and image processing,* Oxford University Press, 1999.

[Fofi *et al.*, 2000]

D. Fofi, J. Salvi and E. Mouaddib, *Euclidean Reconstruction by means of an Uncalibrated Structured Light Sensor*, V Ibero American Symposium on Pattern Recognition, pp. 159-170, Lisbon (Portugal), September 2000.

[Forsyth and Ponce, 2002]

David A. Forsyth and Jean Ponce, *Computer Vision -- A modern approach,* pp. 120-124, Prentice-Hall, August 2002.

[Fryer *et al.*, 1994]

J. Fryer, T. Clarke and J. Chen, *Lens distortion for simple C-mount lenses*, International Archives of Photogrammetry and Remote Sensing, 30(5), pp. 97-101, 1994.

[Fua and Miccio, 1999]

P. Fua and C. Miccio, *Animated Heads from Ordinary Images: A Least Squares Approach*, Computer Vision and Image Understanding, Vol. 75, No. 3, pp. 247-259, 1999.

[Fusiello *et al.*, 2000]

A. Fusiello, E. Trucco and A. Verri, *A compact algorithm for rectification of stereo pairs,* Machine Vision and Applciations, 12(1), pp16-22, 2000.

[Gillies *et al.*, 2004]

Marco Gillies, Daniel Ballin, and Balzs Csanad Csaji, *Efficient Clothing Fitting from Data,* Journal of WSCG, Vol. 12, No. 1, pp. 129-136, 2004.

[GOM, 2004]

GOM mbH, *Quality Control and 3D-Digitizing using Photogrammetry and Fringe Projection*, http://www.gom.com/pub/publications/quality-en.pdf.

[Grimson, 1991]

W. Eric L. Grimson, *Object recognition by computer: the role of geometric constraints,* MIT Press, Cambridge, MA, 1991

[Guhring, 2001]

Jens Guhring, *Dense 3-d surface acquisition by structured light using off-the-shelf components*, Videometrics and Optical Methods for 3D Shape Measurement, 4309, pp. 220-231, 2001.

[Hall-Holt and Rusinkiewicz, 2001]

Olaf Hall-Holt and Szymon Rusinkiewicz, *Stripe Boundary Codes for Real-Time Structured-Light Range Scanning of Moving Objects*, Proceedings of the Eighth International Conference on Computer Vision (ICCV 2001), July 2001.

[Heseltine *et al.*, 2002]

Thomas Heseltine, Nick Pears and Jim Austin, *Evaluation of Image Pre-Processing Techniques for Eigenface Based Face Recognition*, Proceedings of the Second International Conference on Image and Graphics, SPIE Vol. 4875, 2002.

[Heseltine *et al.*, 2004]

Thomas Heseltine, Nick Pears and Jim Austin, *Three-Dimensional Face Recognition: A Fishersurface Approach*, ICIAR (2) pp. 684-691, 2004.

[Hill, 1990]

F. S. Hill Jr., *Computer Graphics using OpenGL*, Prentice Hall, New Jersey, 1990.

[Hilton *et al.*, 1996]

A. Hilton, A.J. Stoddart, J. Illingworth and T. Windeatt, *Reliable Surface Reconstruction from Multiple Range Images*, Proceedings ECCV 1996, April 1996.

[Hilton, 1997]

Peter J. Hilton, *Image surface roughness using correlated speckle grain pairs*, DICTA/IVCNZ97, Massey University, New Zealand, pp 349-354, December 1997.

[Horowitz and Pavlidis, 1976]

Steven L. Horowitz and Theodosios Pavlidis, *Picture Segmentation by a Tree Traversal Algorithm*, Journal of the Association for Computing Machinery. Vol. 23, No. 2, pp. 368-388, April 1976,.

[Hu and Stockman, 1989]

Gongzu Hu and George Stockman, *3-D Surface Solution using Structured Light and Constraint Propagation*, IEEE Trans. on PAMI 11(4), pp. 390-402, April 1989.

[Huang and Menq, 2001]

J. Huang and C. H. Menq, *Automatic Data Segmentation for Geometric Feature Extraction from Unorganized 3-D Coordinate Points,* IEEE Transactions on Robotics and Automation, Vol. 17, No. 3, pp. 268-279, 2001.

[Huang and Menq, 2002]

J. Huang and C. H. Menq, *Combinatorial manifold mesh reconstruction and optimization from unorganized points with arbitrary topology*, Computer-Aided Design, 2002, Vol. 34, No. 2, pp. 149-165, February 2002.

[Iketani *et al.*, 1998]

A. Iketani, A. Nagai, Y. Kuno, and Y. Shirai, *Detecting persons on changing background*, Proc. 14th International Conference on Pattern Recognition, pp.74-76, 1998.

[Isgro and Trucco, 1999]

F. Isgro and E. Trucco, *Projective Rectification without Epipolar Geometry*, Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, pp. 94-99, Fort Collins, Colorado, USA, June 1999.

[Ishigawa and Geiger, 1998]

H. Ishikawa and D. Geiger, *Occlusions, Discontinuities, and Epipolar Lines in Stereo*, Fifth European Conference in Computer Vision (ECCV'98), 1998.

[Jeong *et al.*, 2002]

M.H. Jeong, Y. Kuno, N. Shimada, and Y. Shirai, *Recognition of Shape-Changing Hand Gestures,* IEICE Transactions on Information and Systems, Vol. E85-D, pp.1678-1687, 2002.

[Jarvis, 1983]

R. A. Jarvis, *A perspective on range-finding techniques for computer vision*, IEEE Trans. Pattern Analysis and Machine Intelligence, 5(2), pp. 122-139, 1983.

[Jiang and Bunke, 1992]

X.Y. Jiang and H. Bunke, *Fast segmentation of range images into planar regions by scan line grouping.* Technical report IAM 92-006, Institute for Computer Science, University of Bern, Switzerland, April 1992.

[Jiang and Bunke, 1999]

X. Jiang and H. Bunke, *Edge detection in range images based on scan line approximation,* Computer Vision and Image Understanding, 73(2), pp. 183-199, 1999.

[Jonannesson and Thorngren, 2004]

Mattias Johannesson and Hakan Thorngren, *Advances in CMOS Technology Enables Higher Speed True 3D-Measurements,* https://www.machinevisiononline.org/ public/articles/ivp1.pdf

[Kawai *et al.*, 1998]

Yoshihiro Kawai, Toshio Ueshiba, Yutaka Ishiyama, Yasushi Sumi and Fumiaki Tomita, *Stereo Correspondence Using Segment Connectivity*, Proceedings of the 14th International Conference on Pattern Recognition-Vol. 1, p. 648, August 16-20, 1998.

[Kriegman and Ponce, 1990]

D. J. Kriegman and Jean Ponce, *On Recognizing and Positioning Curved 3-D Objects from Image Contours*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12 No. 12, pp.1127-1137, December 1990.

[Karara, 1989]

H. M. Karara, *Non-Topographic Photogrammetry*, American Society of Photogrammetry and Remote Sensing, Falls Church, VA, 1989.

[Kimura and Saito, 2001]

Makoto Kimura and Hideo Saito, *Interpolation of three views based on epipolar geometry*, Proceedings of SPIE Vol. 4310, pp. 218-227, Visual Communications and Image Processing 2001.

[Koch *et al.*, 1998]

R. Koch, M. Pollefeys and L. Van Gool, *Automatic 3D Model Acquisition from Uncalibrated Image Sequences*, Proceedings CGI'98 (Computer Graphics International), 1998.

[Koenderink, 1984]

Jan J. Koenderink, *What does the occluding contour tell us about solid shape?,* Perception, Vol. 13, pp. 321-330, 1984.

[Koenderink and van Doorn, 1976]

J. J. Koenderink and A. J. van Doorn, *Geometry of Binocular Vision and a Model for Stereopsis,* Biol. Cybernetics 21, pp. 29-35, 1976.

[Krishnamurthy and Levoy, 1996]

V. Krishnamurthy and M. Levoy, *Fitting smooth surfaces to dense polygon meshes,* SIGGRAPH '96, pp. 313-324, 1996.

[Lee and Oh, 2003]

Jong S. Lee and Paul Y. Oh, *A Study on the Planar Rectification of Self-Calibrated Stereo Images,* International Conference on Computer, Communication and Control Technologies (CCCT), Vol. 3, pp. 465-470, Orlando, FL, July 2003.

[Lei and Hendriks, 2001]

B. J. Lei and Emile A. Hendriks, *Multi-step View Synthesis with occlusion handling*, Proceedings of VMV 2001, Stuttgart, Germany, November 21-23, 2001.

[Levoy *et al.*, 2000]

Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk, *The Digital Michelangelo Project: 3D scanning of large statues*, Computer Graphics SIGGRAPH 2000 Proceedings, 2000.

[Li and Li, 2003]

N. Li and Y. F. Li, *Feature Encoding for Unsupervised Segmentation of Color Images*, IEEE Transactions on System Man and Cybernetics, Part B, Vol. 33, No. 3, pp. 438-447, June 2003.

[Li and Liu, 2003]

Y. F. Li and Z. Liu, *Information Entropy Based Viewpoint Planning for 3D Object Reconstruction*, IEEE Transactions on Robotics, 2003.

[Li and Zhang, 2004]

Y. F. Li and B. Zhang, A *Method for 3D measurement and reconstruction for active vision*, Measurement Science and Technology, Vol. 15, No. 11, pp. 2224-2232, Nov. 2004.

[Lim and Menq, 1997]

E. M. Lim and C. H. Menq, *Integrated Planning for Precision Machining of Complex Surfaces; Part 1: Cutting-path and Feedrate Optimization*, International Journal of Machine Tools and Manufacture, Vol. 37, No. 1, pp. 61-75, 1997.

[Lin *et al.*, 2002]

Chung-Yi Lin, Sheng-Wen Shih, Yi-Ping Hung and Gregory Y. Tang, *New Approach to Automatic Reconstruction of a 3-D World Using Active Stereo Vision,* Computer Vision and Image Understanding, Vol. 85, No. 2, pp. 117-143, Feb 2002.

[Liu *et al.*, 2000]

Zicheng Liu, Zhengyou Zhang, Chuck Jacobs, Michael Cohen, *Rapid Modeling of Animated Faces From Video,* In Proceedings of The Third International Conference on Visual Computing (Visual 2000), pp. 58-67, Mexico City, September 2000.

[Livingstone *et al.*, 1993]

F. R. Livingstone, L. King, J.-A. Beraldin and M. Rioux, *Development of a real-time laser scanning system for object recognition, inspection, and robot control*, SPIE Proceedings, Telemanipulator Technology and Space Telerobotics, Boston, MA., Vol. 2057. pp. 454-461, September 7-10, 1993.

[Liu and Rodrigues, 2002]

Y. Liu and M.A. Rodrigues, *Geometric Analysis of Two Sets of 3D Correspondence Data Patterns for the Registration of Free-Form Shapes,* Journal of Intelligent and Robotic Systems, Vol. 33, No. 4, pp. 409-436, 2002.

[Maas, 1998]

H.-G. Maas, *Image sequence based automatic multi-camera system calibration techniques,* International Archives of Photogrammetry and Remote Sensing Vol. 32, Part V, 1998.

[Marek and Stanek, 2002]

Zimanyi Marek and S. Stanek, *Facing Virtual Habitat: Range-Images Face Reconstruction and Real-faced Avatars*. In LEBERL, F. - FERKO, A. (Eds.) 2002. East-West Vision 2002. ISBN 3-85403-163-7, pp. 253-255,Vienna, 2002.

[McCullum *et al.*, 1996]

B. C. McCallum, W. R. Fright, M. A. Nixon and N. B. Price, *A Feasibility Study of Hand-held Laser Surface Scanning*, Image & Vision Computing New Zealand, pp. 103-108, Lower Hutt, August 1996.

[McCullum *et al.*, 1998]

B. C. McCallum, M. A. Nixon, N. B. Price and W. R. Fright, *Hand-held Laser Scanning In Practice*, Proceedings of Image and Vision Computing NZ, The University of Auckland, pp. 17-22, October 1998.

[Marr and Poggio, 1979]

D. Marr and T. Poggio, *A computational theory of human stereo vision*, Proceedings of the Royal Society of London, B, p. 204, pp. 301-328, 1979.

[Maruyama and Abe, 1993]

M. Maruyama and S. Abe, *Range Sensing by Projecting Multiple Slits with Random Cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.15 No.6, pp.647-651, June 1993.

[McIvor, 1997]

Alan M. McIvor, *An Alternative Interpretation of Structured Light System Data.*, DICTA/IVCNZ97, Massey University, pp 41-46, New Zealand, December 1997.

[McIvor and Valkenburg, 1997]

Alan M. McIvor and Robert J. Valkenburg, *Substripe Localisation for Improved Structured Light System Performance*, DICTA/IVCNZ97, Massey University, New Zealand, pp. 309-314, December 1997.

[Michelson and Morley, 1887]

Albert A. Michelson and Edward W. Morley, *On the Relative Motion of the Earth and the Luminous Ether,* The American Journal of Science, No. 203 Vol. XXXIV, November 1887.

[Naemura *et al.*, 2001]

Takeshi Naemura, Tatsuya Yoshida and Hiroshi Harashima, *3-D Computer Graphics Based on Integral Photography,* Opt. Express, 8, pp. 255 - 262, February 2001.

[Ngai *et al.*, 1999]

A. Nagai, Y. Kuno and Y. Shirai, *Detection of Moving Objects against a Changing Background,* Systems and Computers in Japan, Vol.30, No.11, pp.107-116, 1999.

[Narayanan *et al.*, 1995]

P.J. Narayanan, P. Rander and T. Kanade, *Synchronous capture of Image Sequences from Multiple Cameras*, tech. report CMU-RI-TR-95-25, Robotics Institute, Carnegie Mellon University, December, 1995.

[Narayanan *et al.*, 1998]

P.J. Narayanan, P. Rander and T. Kanade, *Constructing Virtual Worlds Using Dense Stereo*, Proceedings of the Sixth IEEE International Conference on Computer Vision (ICCV'98), pp. 3-10, January 1998.

[Nayar *et al.*, 1995]

S. K. Nayar, M. Nigochi, M. Watanabe and Y. Nakagawa, *Focus Range Sensors,* Proceedings of 7th International Symposium on Robotics Research (ISRR), Herrsching, Germany, October 1995.

[Nyquist, 1928]

Harry Nyquist, *Certain topics in telegraph transmission theory*, AIEE Trans., Vol. 47, pp. 617–644, Jan. 1928.

[Park *et al.*, 2001]

Johnny Park, Guilherme N. DeSouza, and Avinash C. Kak, *Dual-Beam Structured-Light Scanning for 3-D Object Modeling,* Third International Conference on 3D Digital Imaging and Modeling (3DIM 2001), May 2001.

[Pavlidis, 1982]

Theo Pavlidis, *Algorithms for Graphics and Image Processing,* Springer-Verlag, Berlin-Heidelberg, 1982.

[Pellegrini *et al.*, 2000]

S. Pellegrini, G.S. Buller, J.M. Smith, A.M. Wallace and S. Cova, *Laser-based distance measurement using picosecond resolution time-correlated single photon counting*, Measurement Science and Technology, 11, pp. 713-716, 2000.

[Phong, 1975]

Phong, B., *Illumination for computer generated pictures*, Communications of the ACM, 18(6), pp. 311-317, June 1975.

[Pollefreys *et al.*, 1998]

M. Pollefeys, R. Koch and L. Van Gool, *Self-Calibration and Metric Reconstruction in Spite of Varying and Unknown Internal Camera Parameters*, Proceedings of ICCV'98 (International Conference on Computer Vision), Bombay, 1998.

[Pritchett and Zisserman, 1998]

P. Pritchett and A. Zisserman, *Wide baseline stereo matching*, Proceedings of International Conference on Computer Vision, 1998, pp. 754-760, 1998.

[Proesmans *et al.*, 1996]

M. Proesmans, L. Van Gool and A. Oosterlinck, *One-shot active range acquisition*, Proceedings ICPR'96 (International Conference on Pattern Recognition), Vienna, Vol. C, pp.336-340, 1996.

[Proesmans and Van Gool, 1998]

M. Proesmans and M, L. Van Gool, *Reading between the lines - a method for extracting dynamic 3D with texture*, Proceedings of ICCV'98 (International Conference on Computer Vision), Bombay, 1998.

[Pushpakumara, 1996]

D.D.A. Pushpakumara, R.M Gooch and T.A. Clarke, *Real time image processing and data communication for a 3-D measurement system*, 15th National Information Technology Conference, Computer Society of Sri Lanka, pp. 1-9, 1996.

[Ramamoorthi and Arvo, 1999]

Ravi Ramamoorthi and James Arvo, *Creating Generative Models from Range Images,* Computer Graphics (SIGGRAPH 99 Proceedings), 1999.

[Ramamoorthi and Hanrahan, 2001]

Ravi Ramamoorthi and Pat Hanrahan, *A Signal-Processing Framework for Inverse Rendering in Computer Graphics,* SIGGRAPH 2001 Proceedings, 2001.

[Reveret and Essa, 2001]

L. Reveret and I. Essa, *Visual Coding and Tracking of Speech Related Facial Motion,* Proceedings of Workshop on Cues in Communication, held in Conjunction with IEEE CVPR 2001, Kauai, Hawaii, December 2001.

[Rioux *et al.*, 1998]

M. Rioux, F. Blais, J.-A. Beraldin and P. Boulanger, *Range imaging sensors development at NRC laboratories*, Proceedings of the IEEE Workshop on Interpretation of 3D Scenes, Austin, TX. pp. 154-160, November 27-29, 1989.

[Robinson *et al.*, 2004]

A. Robinson, L. Alboul and M.A. Rodrigues, *Methods for Indexing Stripes in Uncoded Structured Light Scanning Systems*, Journal of WSCG, 12(3), pp. 371-378, 2004.

[Rocchini *et al.*, 2001]

C. Rocchini, P. Cignoni, C. Montani, P. Pingi and R. Scopigno, *A low cost 3D scanner based on structured light,* Computer Graphics Forum (Eurographics 2001 Conference Proc.), Vol. 20 (3), 2001, pp. 299-308, Manchester, 4-7 September 2001.

[Rodrigues and Liu, 2002]

M.A. Rodrigues and Y. Liu, *On the Representation of Rigid Body Transformations for Accurate Registration of Free Form Shapes*, Robotics and Autonomous Systems, an International Journal, Vol. 39, Issue 1, pp. 37-52, 30 April 2002.

[Rodrigues *et al.*, 2002]

M.A. Rodrigues, R. Fisher and Y. Liu, *Registration and Fusion of Range Images,* Special Issue of CVIU (Computer Vision and Image Understanding), Vol. 87, Issues 1-3, pp. 1-131, July 2002.

[Rodrigues and Robinson, 2004]

M.A. Rodrigues and Alan Robinson, *Image Processing Method and Apparatus, Rotational Position Estimator and 3D Model Builder (Tripods),* UK Patent Application 0402420.4, February 4th, 2004.

[Rodrigues *et al.*, 2004 a]

M.A. Rodrigues, Alan Robinson and Lyuba Alboul, *Apparatus and Methods for Three Dimensional Scanning, Multiple Stripe Scanning Method,* UK Patent Application No. 0402565.6, February 5th, 2004.

[Rusinkiewicz and Levoy, 2001]

S. Rusinkiewicz and M.Levoy, *Efficient Variants of the ICP Algorithm*, Third International Conference on 3D Digital Imaging and Modeling (3DIM), 2001.

[Rusinkiewicz *et al.*, 2002]

Szymon Rusinkiewicz , Olaf Hall-Holt and Marc Levoy, *Real-Time 3D Model Acquisition,* Proceedings of the 29th annual conference on Computer graphics and interactive techniques, San Antonio, Texas, pp. 438 - 446, 2002.

[Sa *et al.*, 2002]

Asla Sa, Paulo Cezar Carvalho and Luiz Velho,*(b, s)-BCSL : Structured Light Color Boundary Coding for 3D Photography,* Proceedings of 7th International Fall Workshop on Vision, Modeling and Visualization, 2002.

[Sa *et al.*, 2003]

Asla Sa, Paulo Cezar Carvalho and Luiz Velho, *Recovering Registered Geometry and High Dynamic Range Texture with Coded Structured Light*, Proceedings of The 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, 2003.

[Salvi *et al.*, 2002]

J. Salvi, X. Armangue and J. Batlle, *A comparative review of camera calibrating methods with accuracy evaluation*, Pattern Recognition, July 2002, Vol. 35, (7), pp. 1617-1635(19), 2002.

[Salvi *et al.*, 2004]

J. Salvi, J. Pagés and J. Batlle, *Pattern Codification Strategies in Structured Light Systems,* Pattern Recognition 37(4), pp 827-849, April 2004.

[Sawada *et al.*, 1983]

H. Sawada, J. Fujii, K. Kato, M. Onoe and Y. Kuno, *Three- dimensional reconstruction of the left ventricle from multiple cross sectional echocardiograms: Value for measuring left ventricular volume*, British Heart Journal, Vol.50, pp.438-442, 1983.

[Scharstein and Szeliski, 2003]

D. Scharstein and R. Szeliski. *High-accuracy stereo depth maps using structured light*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), Vol. 1, pp. 195-202, Madison, WI, June 2003.

[Segall *et al.*, 2001]

C.A. Segall, R. Molina, A.K. Katsaggelos and J. Mateos, *Bayesian High-resolution Reconstruction of Low-resolution Compressed Video*, 2001 IEEE International Conference on Image Processing (ICIP2001), Vol. 2, pp. 25-28, Thessalonica (Greece), 2001.

[Senior *et al.*, 2001]

A.Senior, A.Hampapur, Y-L Tian, L. Brown, S. Pankanti and R. Bolle, *Appearance Models for Occlusion Handling*, Proceedings of Second International workshop on Performance Evaluation of Tracking and Surveillance systems in conjunction with CVPR'01, December 2001.

[Shen and Menq, 2001]

T. S. Shen and C. H. Menq, *Automatic Camera Calibration for a Multiple-Sensor Integrated Coordinate Measurement System*, IEEE Transactions on Robotics and Automation, Vol. 17, No. 4, pp. 502-507, 2001.

[Shen and Menq, 2002]

T. S. Shen and C. H. Menq, *Digital Projector Calibration for 3D Active Vision Systems*, ASME Journal of Manufacturing Science and Engineering, Vol.124, pp. 126-134, 2002.

[Sinlapeecheewa and Takamasu, 2002]

C. Sinlapeecheewa and K. Takamasu, *3D Profile Measurement by Color Pattern Projection and System Calibration*, IEEE ICIT'02, pp. 405-410, Bankok, December 2002.

[Subbarao and Liu, 1998]

M. Subbarao and Y. F. Liu, *Computational Algorithm for unified focus and defocus analysis for 3D scene recovery*, proceedings of SPIE, Vol. 3457, Optical Science, Engineering, and Instrumentation, July 1998.

[Tanimoto and Pavlidis, 1977]

Steven L. Tanimoto and Theodosios Pavlidis, *The Editing of Picture Segmentations Using Local Analysis of Graphs,* Communications of the ACM, Vol. 20, No. 4, April 1977.

[Trucco and Fisher, 1994]

Emanuele Trucco and Robert B. Fisher, *Acquisition of consistent range data using local calibration*, Proceedings IEEE International Conference on Robotics and Automation, (IEEE Comp.Soc. Press, Los Alamitos), pp 3410-3415, 1994.

[Trucco *et al.*, 1994]

E. Trucco, R. B. Fisher and A.M. Fitzgibbon, *Direct Calibration and Data Consistency in 3-D Laser Scanning,* Proceedings British Machine Vision Conference BMVC94, York, pp 489-498, September 1994.

[Trucco *et al.*, 1998]

E. Trucco, R.B. Fisher, A.W. Fitzgibbon and D.K. Naidu, *Calibration, data consistency and model acquisition with a 3D laser striper*, International Journal of Computer Integrated Manufacturing, 11(4), pp. 293-320, 1998.

[Tsai, 1987]

R. Y. Tsai , *A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses*, IEEE J. Robotics Automat., Vol. RA-3, No. 4, pp. 323-344, 1987.

[Umasuthan and Wallace, 1996]

M. Umasuthan, and A.M. Wallace, *Outlier removal and discontinuity smoothing of range data*, IEE Proceedings: Vision, Image and Signal Processing, 143(3), pp. 191-200, 1996.

[Vaillant and Faugeras, 1992]

Régis Vaillant and Olivier D. Faugeras, *Using Extremal Boundaries for 3-D Object Modeling*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.14 No.2, pp.157-173, February 1992.

[Valkenburg, 1998]

R. J. Valkenburg, *A Bayesian approach to camera system calibration/spatial intersection*, Proceedings IVCNZ98, pp. 11-16, Auckland, New Zealand, December 1998.

[Valkenburg and McIvor, 1998]

R.J.Valkenburg and A.M.McIvor, *Accurate 3D measurement using a Structured Light System*, in *Image and Vision Computing*, 16(2):99-110, February 1998.

[Verri and Trucco, 1998]

A. Verri and E. Trucco, *Finding the epipole from uncalibrated optical flow*, Image and Vision Computing, 17, pp. 605-609, 1998.

[Vedula *et al.*, for publication 2005]

S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, *Three-Dimensional Scene Flow,* IEEE Transactions on Pattern Analysis and Machine Intelligence, for publication in 2005.

[Wallace *et al.*, 2000]

A.M. Wallace, P. Csakany, G.S. Buller and A.C. Walker, *3D imaging of transparent objects*, Proceedings of British Machine Vision Conference, pp. 466-475, Bristol, September 2000.

[Watt, 2000]

Alan Watt, *3D Computer Graphics,* Addison-Wesley, London, 2000.

[Wilburn *et al.*, 2002]

Bennett Wilburn, Michael Smulski, Hsiao-Heng, Kelin Lee and Mark Horowitz, *The Light Field Video Camera,*Proceedings of Media Processors 2002, SPIE Electronic Imaging, 2002.

[Winkelbach and Wahl, 2002]

S. Winkelbach and F. M. Wahl, *Shape from Single Stripe Pattern Illumination*, Pattern Recognition (DAGM 2002), Lecture Notes in Computer Science 2449, pp. 240-247, Springer, Zürich, 2002.

[Wood *et al.*, 2000]

Daniel N. Wood , Daniel I. Azuma , Ken Aldinger , Brian Curless , Tom Duchamp , David H. Salesin , Werner Stuetzle, *Surface light fields for 3D photography*, Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 287-296, July 2000.

[Wyngaerd and Van Gool, 2002]

Joris Vanden Wyngaerd and Luc Van Gool, *Automatic crude patch registration: toward automatic 3D model building,* Computer Vision and Image Understanding archive
Vol. 87, Issue 1-3, July 2002.

[Yang *et al.*, 1995]

J. Yang,, N. L. Lee and C. H. Menq, *Application of Computer Vision in Reverse Engineering for 3D Coordinate Acquisition*, Proceedings of the Symposium on Cocurrent Design and Process Engineering, pp. 157-173, ASME International Mechanical Engineering Congress and Exposition, San Francisco, CA, November 12-17, 1995.

[Zhang, 1994]

Z. Y. Zhang, *Iterative point matching for registration of free-form curves and surfaces*, International Journal of Computer Vision, 13(2), pp 119-15, Oct. 1994.

[Zhang, 1996]

Z. Zhang, *Determining the Epipolar Geometry and its Uncertainty: A Review*, Research Report, No.2927, INRIA Sophia-Antipolis, July 1996.

[Zhang and Faugeras, 1992]

Zhengyou Zhang and Olivier D. Faugeras, *Three-dimensional motion computation and object segmentation in a long sequence of stereo frames*, International Journal of Computer Vision, Vol.7 No.3, pp.211-241, April 1992.

[Zhang *et al.*, 2002]

Li Zhang, Brian Curless and Stephen M. Seitz, *Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming,* 1st International Symposium on 3D data processing, visualization and transmission, Padova, Italy, June 19-22, 2002.

Appendix

A1. Description of Main Program `striper.cpp`

The program **`striper.cpp`**, which creates and displays the surface model from the recorded image, is presented here in pseudocode. Descriptive rather than actual program elements are shown italicised.

The program begins with the main() function which reads in the parameter list, which has been created by the calibration process, parses the recorded image file as the pixel array, and displays the requested visualisation. The keyboard and mouse functions can change the parameters, the indexing algorithms, the subpixel estimator, the occlusion detector, and other actions, during the display.

Note that main() also calls init() which in turn calls reCalculate(), which conducts all the system calculations.

```
main() {

    readParameters()            // read text file containing values for θ, D_C, D_P, P, W,
                                   and κ_1.

    resetStripes()              // initialise depth, indices, peaks, trace and boundary
                                   arrays.

    enter .bmp filename         // read in the recorded image

    parse(filename)             // fill the pixel array from the recorded image

    glutMainLoop() {            // cycles through main loop

        display()               // calls the display function
```

```
        init()                  // initialising

            reCalculate()       // performs all the calculations

        event handler           // handles keyboard and mouse events

        glutMainLoop()          // cycles through main loop
    }
}
```

The display() function visualises the data in various forms, depending upon which mode has been selected at the keyboard. Apart from the point clouds (modes 5 and 6), the other modes show the *column, row* mapping of the pixel array, enabling a comparison between the model with each of the four indexing algorithms with occlusion marking turned on of off. A special `trace[column][row]` array, not described in the thesis, is created to visualise the tracing of the Flood Fill, Scan, and Search and Search algorithms, labelling peaks as *N, S, E, W, U, D,* or *C* as appropriate.

```
display() {

    colour = 0              // start with zero colour index

    drawStripes() {         // draw the stripes in selected mode (1 to 8)

        1. polyhedrals      // displays each peak as a lit surface element

        2. original bitmap  // displays recorded image

        3. coloured stripes // displays coloured stripes over the recorded image,
                               with position modified by subpixel estimator
                               (which can be selected as 5-pixel, 3-pixel or 1-
```

pixel), and lens distortion function.

4. undistorted stripes // displays coloured stripes on a white background, without lens distortion function.

5. point cloud in Y-Z // displays the cloud of vertices projected onto the *Y-Z* plane.

6. point cloud in X-Z // displays the cloud of vertices projected onto the *X-Z* plane.

7. trace // displays the trace array, with values:
UNTRACED_PEAK
NORTH
SOUTH
EAST
WEST
U
D
C
NON_PEAK

8. boundaries // displays the boundary array with values:
UNTESTED_PEAK
UP_BOUNDARY
DOWN_BOUNDARY
LEFT_BOUNDARY
RIGHT_BOUNDARY
OCCLUSION
CONNECTED
UNKNOWN

}

These eight display modes are shown on the following pages. Note that some of the detail may be missing if the rendition is monochrome.

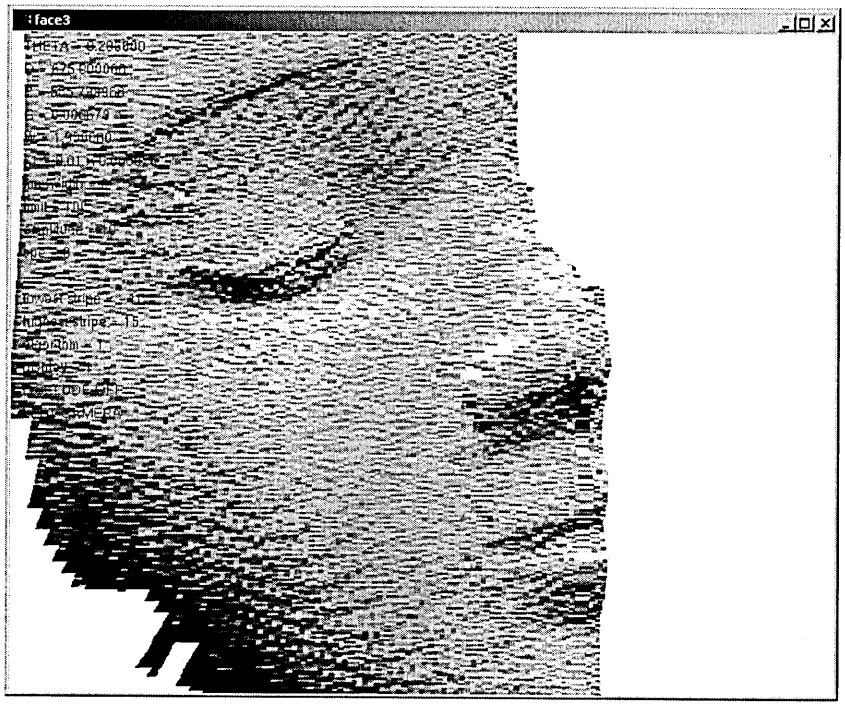

Figure A1: Display mode 1, showing polyhedral surface (detail).

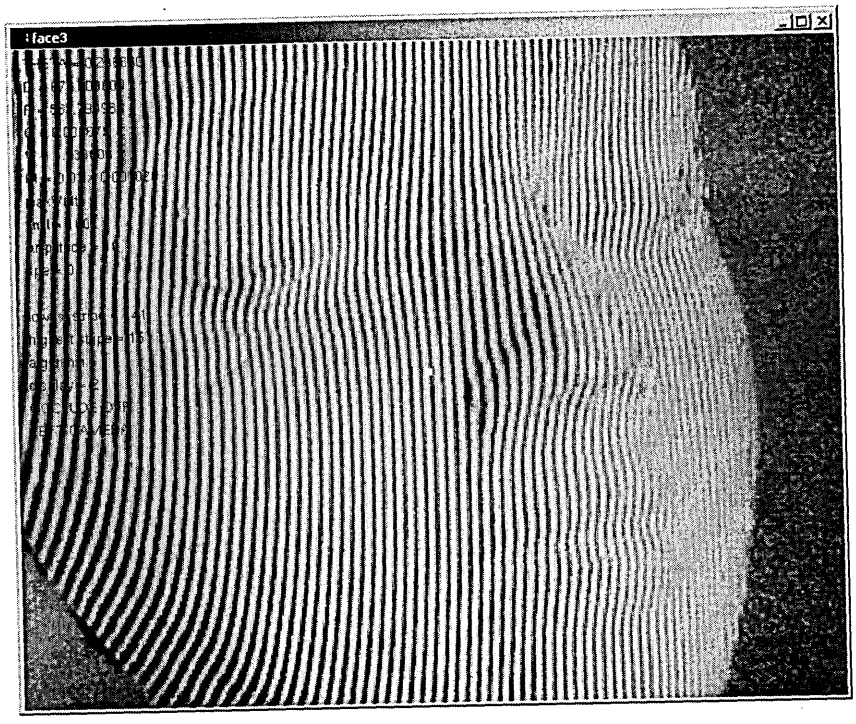

Figure A2: Display mode 2, showing original recorded image (detail).

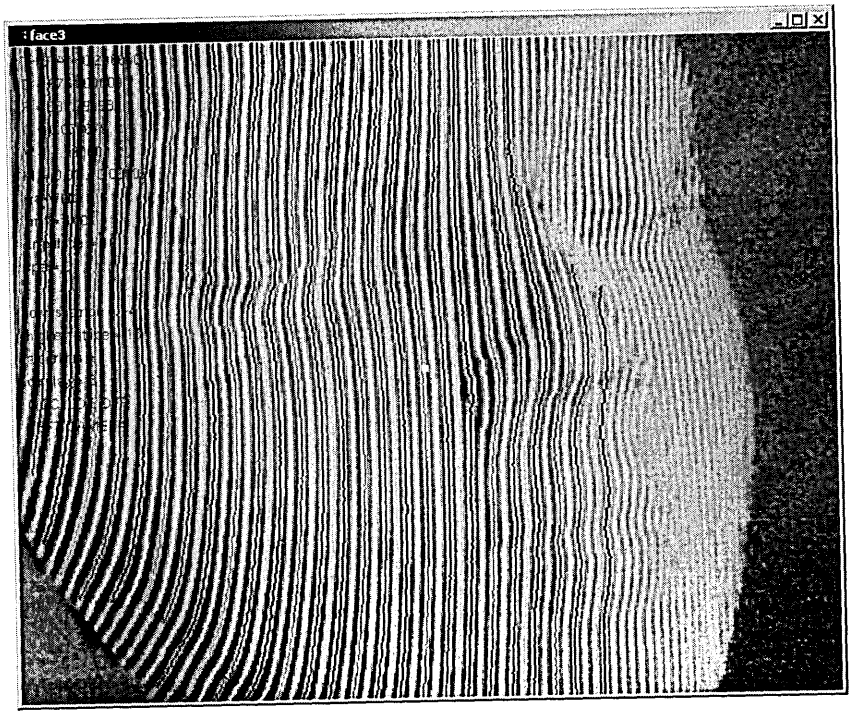

Figure A3: Display mode 3, showing indexed peak stripes (detail).

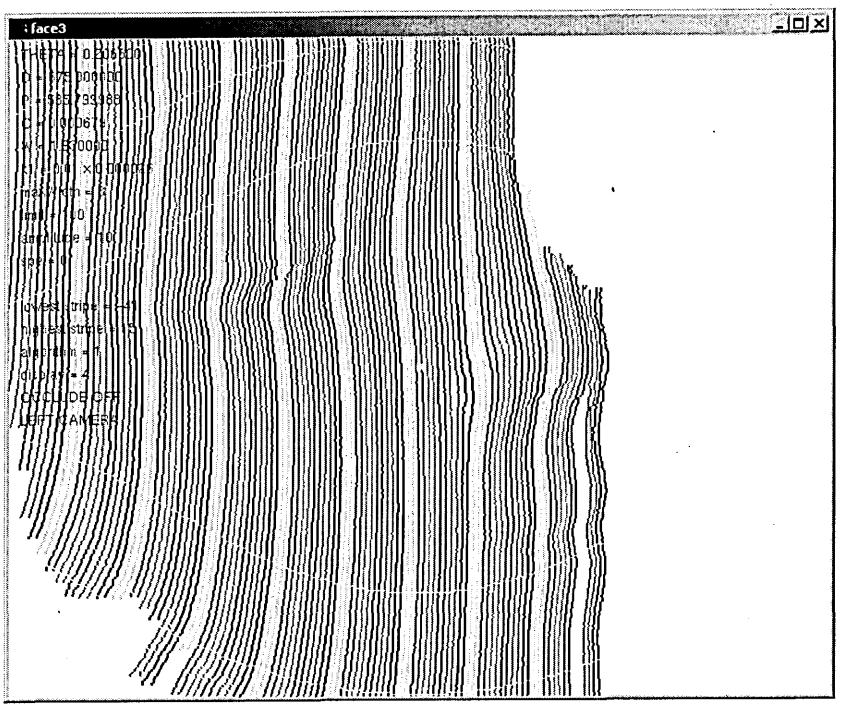

Figure A4: Display mode 4, showing indexed peak and trough stripes (detail).

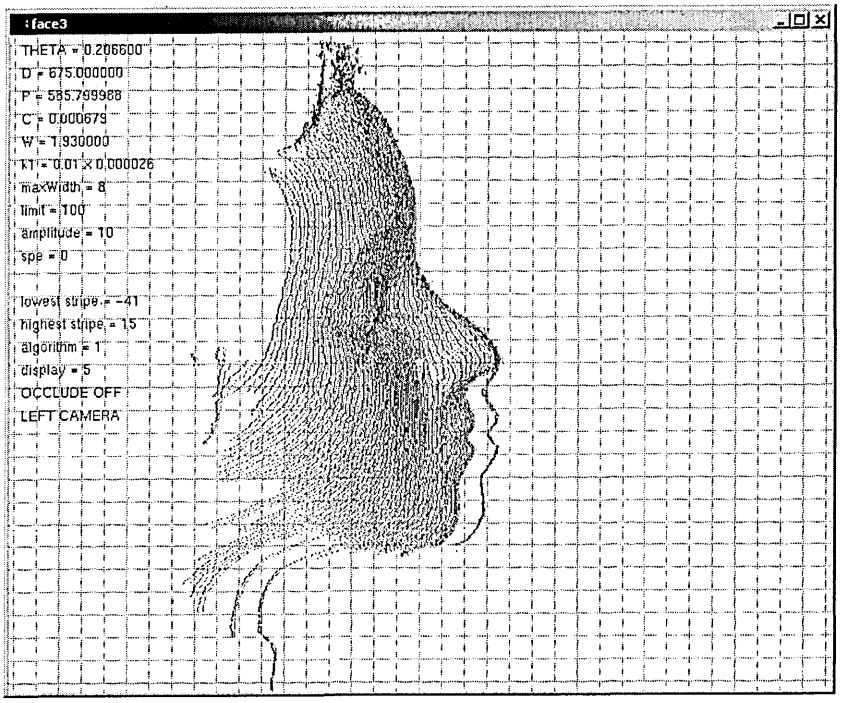

Figure A5: Display mode 5, showing point cloud projected onto Y-Z plane.

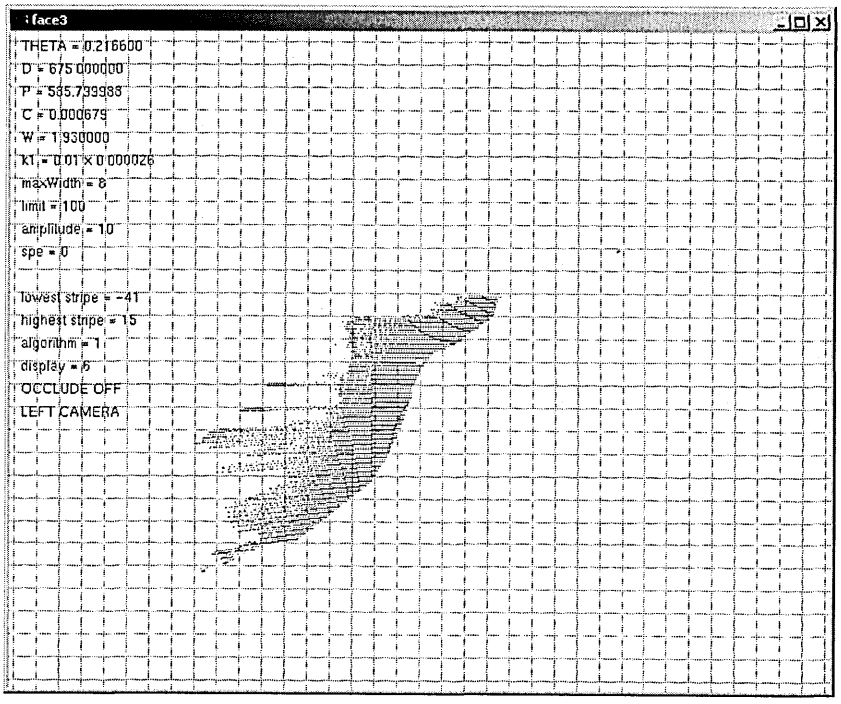

Figure A6: Display mode 6, showing point cloud projected onto X-Z plane.

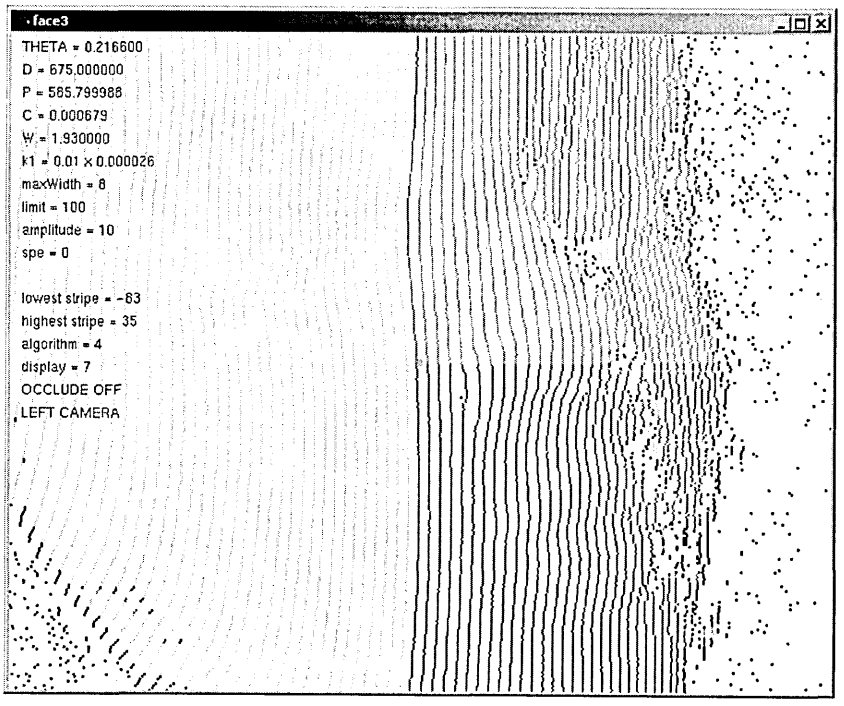

Figure A7: Display mode 7 showing trace array depicting N, S, E, and W directions of search.

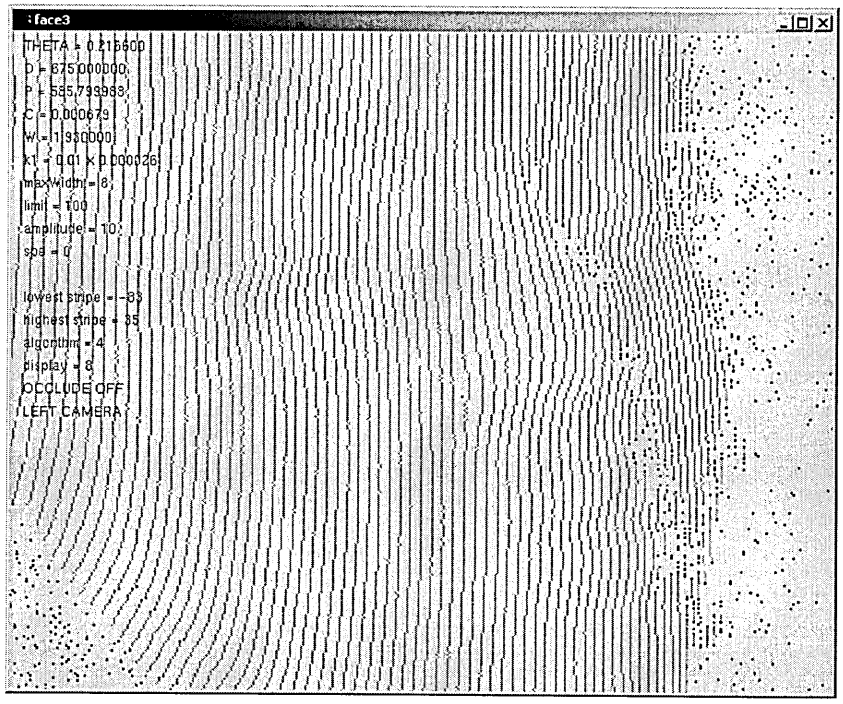

Figure A8: Display mode 8, showing peaks (green), U disconnections (red), D disconnections (blue) and unknown connections (yellow) (detail).

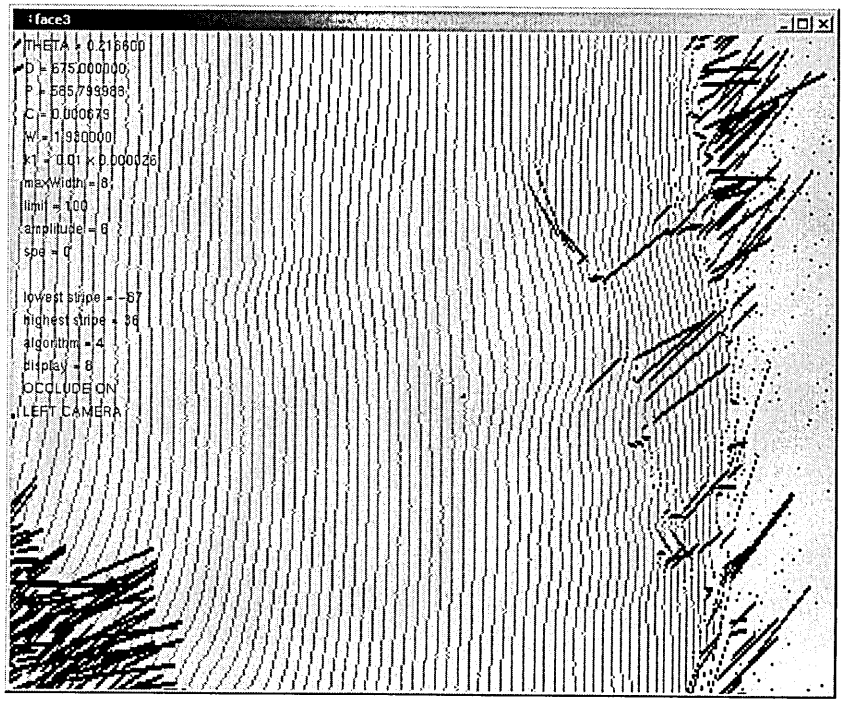

Figure A9: Display mode 8, with the occlusion boundaries marked black (detail).

A2. The reCalculate() Function

The reCalculate() function calls the selected algorithm (1, 2 or 4) which indexes some or all of the peaks. This index set is then compared with the template set (if the scanner is in test mode) and the results are outputted in data.txt.

From the index set, the vertex set, derived from the white (peak) stripes and the black (valley) stripes is calculated, and is outputted as a .scn file.

```
reCalculate() {

    x = mouseX, y = mouseY        //  set the search start with a mouse click

    case 1: scanSearch(x,y)       //  if algorithm is type 1 (see below)

    case 2: scanLine(x,y)         //  if algorithm is type 2 (see below)

    case 3: not used

    case 4: floodFill(x,y)        //  if algorithm is type 4 (see below)

    findPeaks()                   //  find local maxima in pixel array, populate
                                      boundary array, and if occlusion=ON, connect
                                      occlusions (see below)

    output data.txt file          //  data for the template comparison

    subpixel estimator:           //  finds the adjusted x value of each peak and for each
        for each indexed peak:        trough using the subpixel estimator
            x1 = setPeak(x,y)
            x2 = setvalley(x,y)

    calculate vertices:           //  use the standard scanning formula, with x1 = h.
```

for each indexed peak:

(x,y,z)=calculate(x1,v,n)

outputs vertices as a .scn file

scanfile(*filename.scn*) //

}

A3. The Indexing Algorithms

Because of the importance of the indexing methods, they are included here in full. The general form of the methods is described in Chapter 7, and the utility methods used here are detailed below.

```
//////////////  algorithm 1 ////////////////////

void scanSearch(int xFirst, int yFirst) {
     resetStripes();
     findPeaks();

     // Find first stripe eastwards
     xFirst = goEast(xFirst, yFirst);

     // Draws first stripe then next right stripe, Stops at right edge
     int x = xFirst;
     int y = yFirst;
     stripes = 0;
     while(edgeTest(x, y) && test(x, y, stripes)) {
          followStripe(x, y);
          stripes++;
          x = goEast(x, y);
     }

     // Draw stripes stopping at left edge
     x = goWest(xFirst, y);
     stripes = -1;
     while(edgeTest(x, y) && test(x, y, stripes)) {
          followStripe(x, y);
          stripes--;
          x = goWest(x, y);
     }
}

// index a single stripe
void followStripe(int xStart, int yStart) {
     int y=yStart;
     int x=xStart;
```

```
        setStripe(x, y);
        while(edgeTest(x, y) && test(x, y, stripes)) {
                y++;
                x = centreStripe(x, y);
                if(x>-1) {
                        setStripe(x, y);
                        trace[x][y] = 3;
                }
        }
        y=yStart;
        x=xStart;
        while(edgeTest(x,y) && test(x, y, stripes)) {
                y--;
                x = centreStripe(x, y);
                if(x>-1) {
                        setStripe(x, y);
                        trace[x][y] = 4;
                }
        }
}


//////////////////  algorithm 2 ///////////////////

void scanline(int xFirst, int yFirst) {
        resetStripes();
        findPeaks();

        // Find first stripe eastwards
        xFirst = goEast(xFirst, yFirst);      int x=xFirst;
        int y=yFirst;

        // successively find peaks going right then going left.
        // Move to next peak above and continue to top edge
        while(y<iHeight) {
                x = centreStripe(x, y);
                stripes=1;
                setStripe(x, y);
                lookRight(x, y);
                stripes=1;
                lookLeft(x, y);
                y++;
```

```
        }
        x=xFirst;
        y=yFirst;

        // repeat to bottom edge
        while(y>0) {
            y--;
            x = centreStripe(x, y);
            stripes=1;
            setStripe(x, y);
            lookRight(x, y);
            stripes=1;
            lookLeft(x, y);
        }
}


////////////////////// algorithm 4 //////////////////////

// includes queue to avoid stack overflow

void floodFill(int x, int y) {
        resetStripes();
        findPeaks();
        qFirst=1;
        qLast=0;
        flood(x, y, NORTH, 0);
        do {
            int xF = qX[qFirst];
            int yF = qY[qFirst];
            int hF = qH[qFirst];
            int iF = qI[qFirst];
            qFirst = (qFirst+1)%stack;
            flood(xF,   yF+1, NORTH, iF);
            flood(xF+1, yF,   EAST,  iF+1);
            flood(xF,   yF-1, SOUTH, iF);
            flood(xF-1, yF,   WEST,  iF-1);
        }
        while(qFirst != (qLast+1)%stack);
}

void flood(int x, int y, int heading, int index) {
```

```
      if(heading==NORTH || heading==SOUTH) {
            if(peaks[x-1][y]) x--;
            else if(peaks[x+1][y]) x++;
            else if(peaks[x][y]);
            else return;
      }
      if(heading==EAST) {
            x = goEast(x,y);
            if(boundary[x][y]>0) return;
      }
      if(heading==WEST) {
            x = goWest(x,y);
            if(boundary[x][y]>0) return;
      }
      if(indices[x][y]<5000 || index>400 || index<-400 || !(edgeTest(x,y)) ||
!test(x, y, index))
            return;
      indices[x][y] = index;
      trace[x][y] = heading+1;
      qLast = (qLast+1)%stack;
      qX[qLast] = x;
      qY[qLast] = y;
      qH[qLast] = heading;
      qI[qLast] = index;
      return;
}
```

A4. Utility methods used in indexing algorithms

```
// return false if (x,y) is within margin if image frame

bool edgeTest(int x, int y) {
     if(x<MARGIN || y<MARGIN || x>width-MARGIN || y>iHeight-MARGIN)
          return false;
     else return true;
}


// return false if the edge of the image or a boundary is found

bool test(int x, int y, int i) {
     if(!edgeTest(x, y)) return false;
     else if(boundary[x][y]>0) return false;
     else return true;
}


// return false if the edge of the image or a boundary is found

bool test(int x, int y) {
     if(!edgeTest(x, y)) return false;
     else if(boundary[x][y]>0) return false;
     else return true;
}


// return the leftmost peak in 3 adjacent pixels

int centreStripe(int x, int y) {
          if(peaks[x-1][y]) return x-1;
          else if(peaks[x+1][y]) return x+1;
          else if(peaks[x][y]) return x;
          else return -1;
}


// return the next peak to the right, before a boundary
```

```
int goEast(int x, int y) {
      x++;
      while(test(x, y) && edgeTest(x, y) && !peaks[x][y]) x++;
      return x;
}


// return the next peak to the left, before a boundary

int goWest(int x, int y) {
      x--;
      while(test(x, y) && edgeTest(x, y) && !peaks[x][y]) x--;
      return x;
}


// find all successive western peaks

void lookLeft(int x, int y) {
      while(x>1 && test(x, y, stripes)) {
            x = goWest(x, y);
            stripes--;
            setStripe(x, y);
      }
}


// find all successive eastern peaks

void lookRight(int x, int y) {
      while(x<width-1 && test(x, y, stripes)) {
            x = goEast(x, y);
            stripes++;
            setStripe(x, y);
      }
}
```

A5. Finding the peaks, boundaries and occlusions

The findPeaks() function looks along each row in turn and marks as true all local maxima, if the brightness limit and amplitude level are reached. Then the boundaries are found and occlusions marked.

```
void findPeaks() { // find all local maxima if greater than limit
  int x;
  int centre, right;
  int bottom;

  for(int y=0;y<iHeight;y++) {
    x=0;
    while(x<width-1) {
      do {
        centre = image[x][y];
        right = image[x+1][y];
        x++;
      }
      while(!(right>centre) && x<width-1);
        bottom = image[x-1][y]; // bottom of trough
      do {
        centre = image[x][y];
        right = image[x+1][y];
        x++;
      }
      while(!(right<centre) && x<width-1);
      x--;

      // check that brightness is above limit
      // check that brightness-bottom > amplitude
      if(image[x][y]>limit && (image[x][y]-bottom)>amplitude) {
        peaks[x][y] = true;
        boundary[x][y] = UNTESTED_PEAK;
      }
    }
  }
  findBoundaries();
  if(occlude) connectOcclusions();
}
```

The findBoundaries() function finds the N, S, E and W neighbours for peak (x,y). If there is no north or south neighbour the peaks are marked as D or U respectively. If west or east neighbours are outside the maxWidth limit, they are marked as L and R boundaries respectively. Otherwise if the peak is not near the image edge, it is marked as connected (C).

```
void findBoundaries() {
  int xTop, xBot, xLeft, xRight;

  for(int x=0;x<1000;x++) {
    for(int y=0;y<1000;y++) {
      if(peaks[x][y]) {
        xTop = centreStripe(x, y+1); // N neighbour
        xBot = centreStripe(x, y-1); // S neighbour
        xLeft = goWest(x, y);        // W neighbour
        xRight = goEast(x, y);       // E neighbour
        if(xTop==-1) boundary[x][y] = UP_BOUNDARY;                    // U
        else if(xBot==-1) boundary[x][y] = DOWN_BOUNDARY;             // D
        else if((x-xLeft)>maxWidth) boundary[x][y] = LEFT_BOUNDARY;   // L
        else if((xRight-x)>maxWidth) boundary[x][y] = RIGHT_BOUNDARY; // R
        else if(edgeTest(x,y)) boundary[x][y] = CONNECTED;            // C
      }
    }
  }
}
```

The connectOcclusions() function finds D and U peaks, and then draws lines between the nearest (U,D) pair as long as they are less than 50 pixels apart. These lines are expressed by marking the associated boundary elements along each line as an OCCLUSION.

```
void connectOcclusions() {
  bool connected;
  int c,r,rr;
  for(int x=0;x<1000;x++) {
    for(int y=0;y<1000;y++) {
      if(boundary[x][y]==DOWN_BOUNDARY) { // look for D boundary
        connected = false;
```

```
    for(int n=1;n<50;n++) {
      for(int j=0;j<n;j++) {
        for(int k=0;k<n;k++) {        // now look for U
          if(!connected && boundary[x+j][y-k]==UP_BOUNDARY) {// first U
            boundary[x+j][y-k] = OCCLUSION;
            connected = true;
            c = j;
            r = k;
          }
        }
      }
    }
    for(int cc=1;cc<c;cc++) {          // draw the occlusion line
      rr = int(float(cc)*float(r)/float(c));
      if(peaks[x+cc][y-rr])
        boundary[x+cc][y-rr] = OCCLUSION;
      if(peaks[x+cc][y-rr-1])
        boundary[x+cc][y-rr-1] = OCCLUSION;
    }
  }

  if(boundary[x][y]==UP_BOUNDARY) { // look for U boundary
    connected = false;
    for(int n=1;n<50;n++) {
      for(int j=0;j<n;j++) {
        for(int k=0;k<n;k++) {      // now look for D
          if(!connected && boundary[x+j][y+k]==DOWN_BOUNDARY){ //first D
            boundary[x+j][y+k] = OCCLUSION;
            connected = true;
            c = j;
            r = k;
          }
        }
      }
    }
    for(int cc=1;cc<c;cc++) {          // draw the occlusion line
      rr = int(float(cc)*float(r)/float(c));
      if(peaks[x+cc][y+rr])
        boundary[x+cc][y+rr] = OCCLUSION;
      if(peaks[x+cc][y+rr+1])
        boundary[x+cc][y+rr+1] = OCCLUSION;
    }
```

```
        }
      }
    }
}
```

Thus the boundary array is marked with occlusions, which can then be used in the test() function (see above) as a stopping condition.