An “Optical Paintbrush” for Virtual Reality Modelling

Ben Southall

MSc Information Technology: Knowledge Based Systems
Department of Artificial Intelligence
University of Edinburgh
1996

Abstract

Virtual reality applications are becoming increasingly common and more sophisticated. However, many virtual environments suffer from a very artificial feel, mainly due to impoverished descriptions (often based on CAD models) of scenery and objects. More sophisticated models which have realistic shape and texture can be obtained by building 3D models directly from range data, and life-like textures can be mapped on to the surface of the model using colour images of the object taken at known viewpoints. This dissertation describes work carried out in improving the performance of a prototype hand-held optical surface scanner. The focus of the report is on noise reduction, principally gaining a deeper understanding of the sensor's global position measurement system and the implementation of a variety of morphological and smoothing operators for the purpose of improving surface description.

Acknowledgements

Firstly I would like to thank my supervisor, Bob Fisher, for his encouragement, help and enthusiasm for the project. I also wish to thank all of the members of the Machine Vision group for their support and friendship during my stay in the vision lab. In particular I thank Anthony Ashbrook for proof-reading this dissertation and useful discussion on many subjects, Maurizio Pilu for LaTeXtips and advice on experimental procedure and data analysis, and finally Andrew Fitzgibbon for his help on circumventing the bugs in the **g++** compiler and for advice on avoiding the pitfalls of C++.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With current improvements in computer graphics algorithms and hardware, "virtual reality" applications are becoming both more common and more accomplished. However, despite advances, environments are typically quite sparse and represent the real world poorly; the reason for both of these is the cost of constructing realistic models of environmental features. Most current techniques use simple primitives to represent objects, so complex objects must be constructed from many simple ones; this is time-consuming and of only limited effectiveness. Even when a model of the geometry of an object has been constructed, there is still the problem of texture-mapping colours and patterns onto the surface. Solutions to this shading problem are being pursued by the computer graphics community (e.g. using fractals and repetitive textures), but at present do not provide realistic colours and lighting. Another deficiency of current computer modelling is that all objects look the same – there is no scope for adding everyday "wear and tear" to modelled features, so environments tend to look too pristine and clinical.

It is obvious that a system that is simple to use which can produce realistically shaped, properly shaded objects would be of great use to those working with virtual reality; a prototype of a system to satisfy these needs had been constructed (see Section 1.1 and Chapter 2), and initial results looked promising. The noise on such models in order to remove dents and protrusions caused by outliers in the raw data.

1.1 The "Optical Paintbrush"

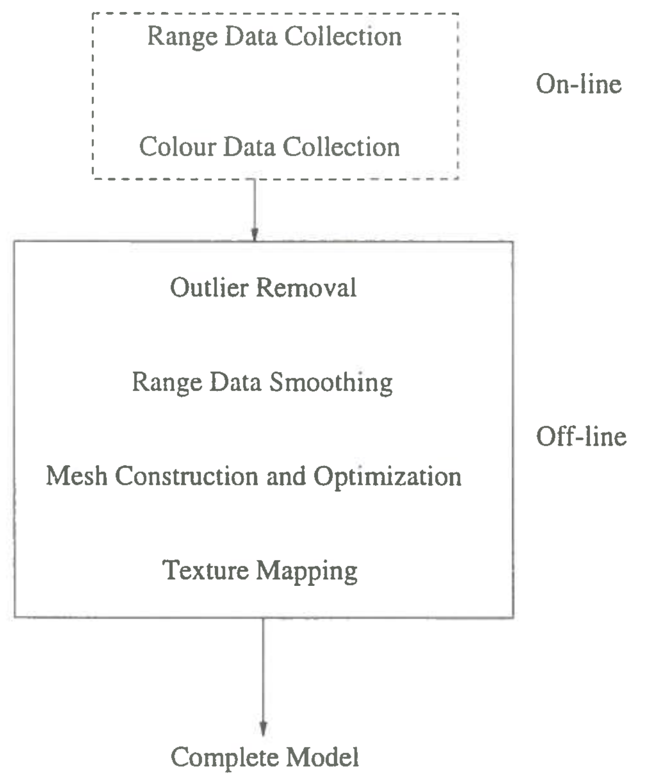

Figure 1.1: System data flow

The prototype modelling system that has been constructed is based around a hand-held laser-stripe ranger. The user runs the stripe over the object being scanned as if painting the object with the stripe, hence the name "optical paintbrush". The full model building process has two stages; on-line and off-line. The first stage is the collection of surface information. This stage is performed in real-time, or 'on-line', the user painting the object with the stripe, which is kept

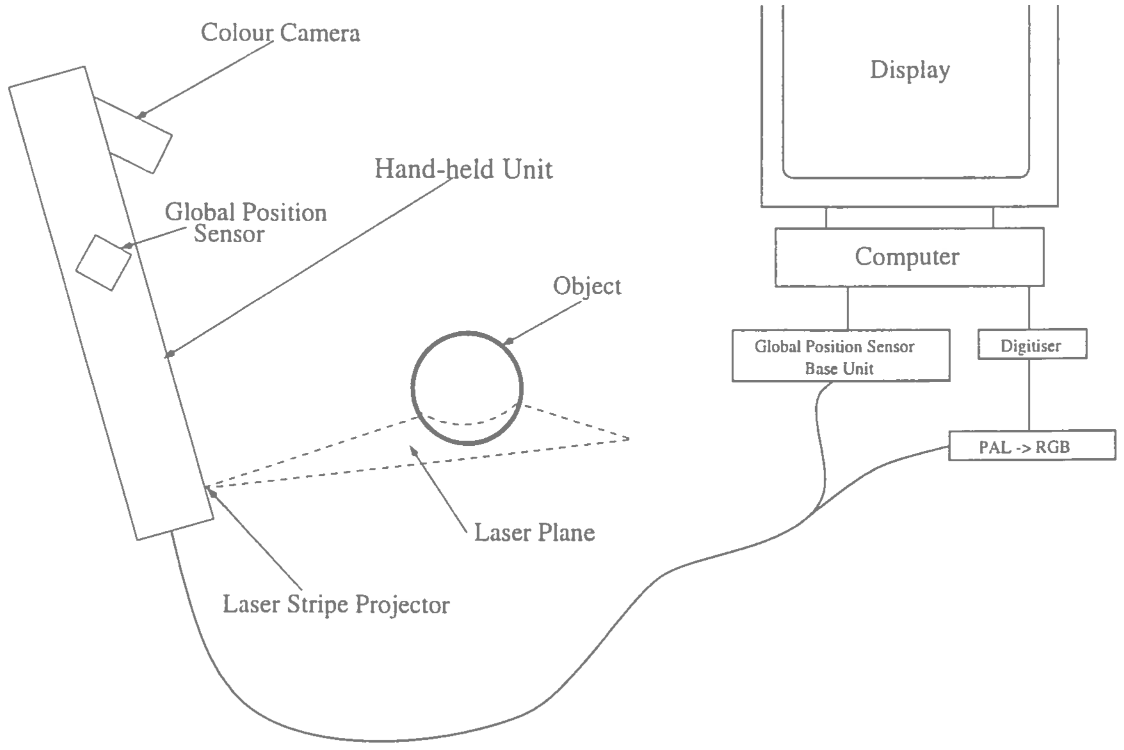

Figure 1.2: System schematic

the user then switches the laser projector off, turns on a local light source and takes still frames of different views of the object – these stills are used in the texture mapping phase later. After the user has performed these actions, off-line processing of the data is performed to reconstruct the object's surface from the set of data points collected. The colour frames captured earlier are now pasted on to the reconstructed surface, and the model is complete. Figure 1.1 outlines the process. Currently the models are stored in RenderWare format (a commercially available 3-D graphics library).

A schematic of the system (adapted from [Fisher *et al.* 96]) is seen in Figure 1.2. The camera image of the stripe on the object's surface is sent back to the computer (via decoding and digitisation software) along with the output of the global position sensor. This global position reading allows the stripe image to be coherent, model of the object's surface can be built up. Two stages of surface modelling take place; an initial generation of points from the scanned range data, followed by a mesh construction and optimisation algorithm [Hoppe *et al.* 92].

A fuller description of both hardware and software is found in Chapter 2.

1.2 Problems Addressed

Although the prototype system is working, there are several aspects of its performance which could be improved;

- The global position sensor suffers from noisy position and angle estimates, and it is possible that some of this noise is systematic. A better understanding of the sensor's operation could lead to improved noise characteristics.
- Filtering the input data before surface reconstruction begins, to remove noise points which will later cause surface distortions. The current preprocessing algorithm is also slow, with running times of up to one hour on a typical data set.
- Various hardware changes – adding more position sensors, improving the light source used in the colour acquisition stage, adding more laser stripes to increase speed of data collection.
- Improved real-time display of points as they are collected, to give the user a clearer picture of any portions of the surface they may have missed.
- Offering alternative output formats, e.g. VRML.

Due to time limitations, it was unrealistic to attempt to solve all of these problems, so the issues of noise reduction prior to surface reconstruction and analysis of the global position sensor, with the aim of increasing the faithfulness of both the geometric model and the placement of texture on the model were

The major source of noise at the time of data collection is the Flock of Birds global position sensor. This is a 6 degree-of-freedom electromagnetic sensor which measures the position of a remote unit relative to its base unit. There is random noise on the translational and angular measurements, both of which lead to inaccurate placement of range images in the 3D frame. Imprecise placement of textures captured from different positions, resulting in blurring of a texture, overlap of two separate textures or an untextured gap appearing between two patches of colour, suggest a systematic error in the sensor readings, although this could be due to random errors on the position of the small number of colour images typically used ($\approx$ 10 images is typical).

It was decided to perform experiments to study both the translational and angular measurement performance, and to attempt to correct any systematic error found (see Chapter 3).

1.2.2 Outlier Removal

In the prototype, there was a problem with outlier data points, i.e. those points which are far from the rest of the data set, and which, if not removed can lead to unwanted surface deformations after reconstruction. Currently the majority of outliers are removed by hand, a tedious and sometimes imprecise process, ripe for automation. A related problem is that of gaps in the data – noise can not only result in spuriously placed points, but also in missing out points which should be in place.

To combat both of these problems, a set of morphological operators were implemented and investigated (Chapter 4).

1.2.3 Data Smoothing

Currently, the outlier removal is performed (by hand) on the output

[Fisher *et al.* 96]) is highly sensitive to clusters of outlier points in the initial data set, so these errors are passed on to the surface reconstruction process. By both performing outlier removal on the initial data set as (see above) and implementing a more efficient and effective noise smoothing algorithm, higher quality models should result in less time.

1.3 Report Outline

In Chapter 2, a more detailed background of the "Optical Paintbrush" and its software is given, along details of other 3D data collection systems. Chapter 3 covers the experiments done with the Flock of Birds position sensor, and the application of the results from these experiments. In Chapter 4, range data noise reduction is tackled. Results of the final reconstructions, conclusions and recommendations for further work are presented in Chapters 5 and 6. The appendices include the procedures for system calibration, results from the Flock of Birds experiments and an outline of the 3D data structure used in the work of Chapter 4.

Chapter 2

Previous Work

Here, the existing prototype hand-held modeller (the "optical paintbrush") is discussed, along with a brief outline of some notable existing 3D data acquisition systems. A short introduction to laser-stripe ranging (as used in the hand-held) is provided for the uninitiated.

2.1 3D Range Data Acquisition Systems

To date ranging systems have largely addressed two problems; in robotics to guide mobile robots through the world or manipulators through the workspace; and in the use of 3D range images in industry – for inspection, identification or automatic acquisition of CAD models to replace traditional engineering drawings. More recently, with increasing interest in virtual reality, systems for building models of general, free form objects have started to appear. The navigation/guidance application aims to build coarse models of the world (maps) as opposed to finer models of individual objects, and will not be covered here.

Two industrial systems that use typical 3D data-collection methods are briefly described; the 3D workplace sensor developed at INRIA, and the ranger used to collect data for the Imagine II system from Edinburgh. It should be noted that only the data acquisition methods are covered here here – as complete vision

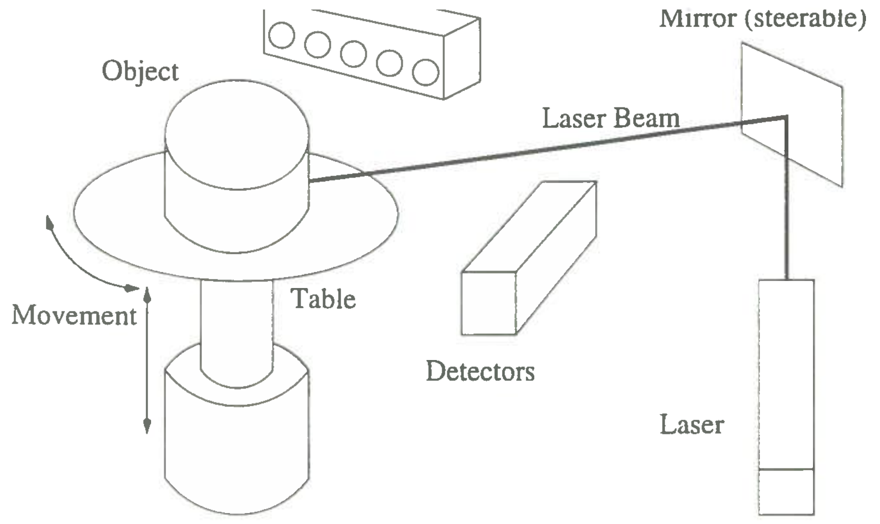

Figure 2.1: The INRIA workplace sensor (schematic)

The INRIA system [Faugeras *et al.* 83] uses a single laser beam as part of an active triangulation system. The setup is illustrated in Figure 2.1. The laser beam is used to illuminate (via a steerable mirror) a single point on the object's surface. This point is detected by both banks of diodes, and its position in 3D space calculated by the position of the point in each detector's field of view. To gather data from the entire surface, two mechanisms are used to scan the laser beam; the first is the steerable mirror, which can direct the beam horizontally, the second a movable table upon which the object is placed. The table can rotate (bringing previously occluded areas into view) and translate vertically, so the entire height of the object may be scanned.

There are many drawbacks to this system; by using a point of light, only one range point may be gathered per sensor sample; the mirror and table combination cannot bring the top (or bottom) of an object into view, and for the triangulation to work it is important to keep the geometry of the system well calibrated.

The Edinburgh range sensor [Naidu & Fisher 90] tackles the speed issue by using a laser stripe, generated by spreading a beam into a 'sheet' of laser light.

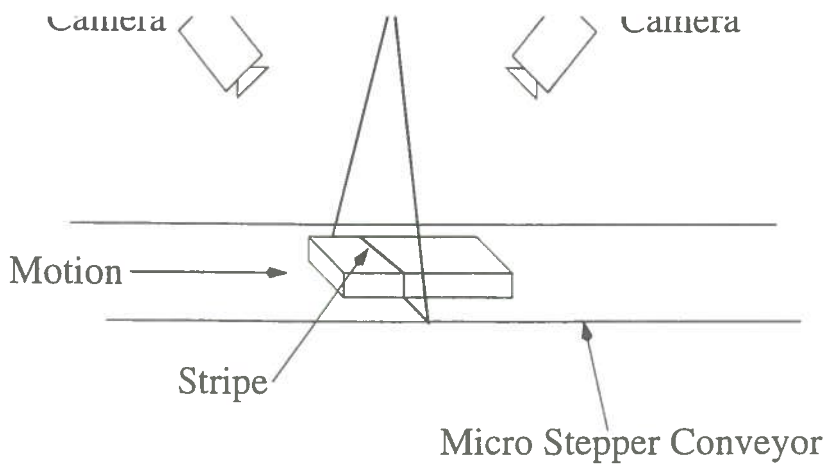

Figure 2.2: Edinburgh sensor hardware (schematic)

along the stripe, using laser-stripe triangulation (see Section 2.2). Laser-stripe triangulation requires only one camera, but the Edinburgh sensor uses a pair of cameras (as seen in Figure 2.2) as part of a noise reduction strategy which compares the range image from each camera, removing as noise any points which do not coincide (within bounds) in both images. The part being inspected is placed on a conveyor like linear micro-stepper which enables the surface to be captured one stripe at a time, with known movement between stripes. To capture all aspects of an object, it is run through the system several times, posed differently on each run, and the images of each view are registered using the iterated closest point technique [Eggert *et al.* 96]. Again, the physical setup of the system is highly calibrated, and although data collection is good (low noise, high density data being acquired rapidly), the need to take multiple views is inconvenient and time-consuming. It is also worth noting that neither the Edinburgh ranger nor the INRIA sensor captures colour information, although sensors exist which do.

The Edinburgh and INRIA scanners were made with industrial part recognition/inspection in mind, not virtual reality modelling, where firms involved are less likely to want to invest in calibrated workspace areas than industrial which uses laser stripe triangulation to collect range data. The camera/laser pair are mounted on an articulated arm which provides the position and orientation of the striper as the user moves it around the object. To scan the rear of an object, a turntable is used to keep the setup calibrated, so that the new data can be correctly registered with the other view(s). Colour acquisition systems (which use a second camera) are also available, but as an added extra.

2.2 Laser Stripe Triangulation

Laser stripe triangulation is a popular structured light technique dating back to the early 1970s [Shirai & Suwa 71], and is used by the "optical paintbrush" system to obtain range images, so it is appropriate to provide an outline of how this technique works (adapted from [Fisher 96]). Figure 2.3 illustrates a simple triangulation set-up, showing the structured light plane, the image plane and the camera origin. The structured light is a laser plane; as its position and orientation relative to the camera is known, an equation of the form $\mathbf{x}.\mathbf{n} = d$ can be written, where $\mathbf{x}$ is any point in the plane, $\mathbf{n}$ the plane normal and d the perpendicular distance from the plane to some arbitrary but known origin.

In this example a pinhole camera model is used, where all rays pass through the camera origin – in reality the model is more complex, but the basic idea of transforming from a ray in the world to a pixel on the image plane is unchanged. The points in the diagram $\mathbf{X_i}$ and $\mathbf{C_i}$ are 3-D vectors, $\mathbf{P_i}$ the 3D co-ordinate of a pixel in the image plane.

Where the laser plane intersects a surface, a stripe is formed. This stripe is seen as a curve on the image plane. Each point on the stripe $\mathbf{X_i}$ has a corresponding projection $\mathbf{P_i}$ on the image plane. Using the pin-hole camera model, a ray passing through $\mathbf{X_i}$ and $\mathbf{P_i}$ also passes through the camera centre, $\mathbf{C_i}$, hence the following equation can be written:

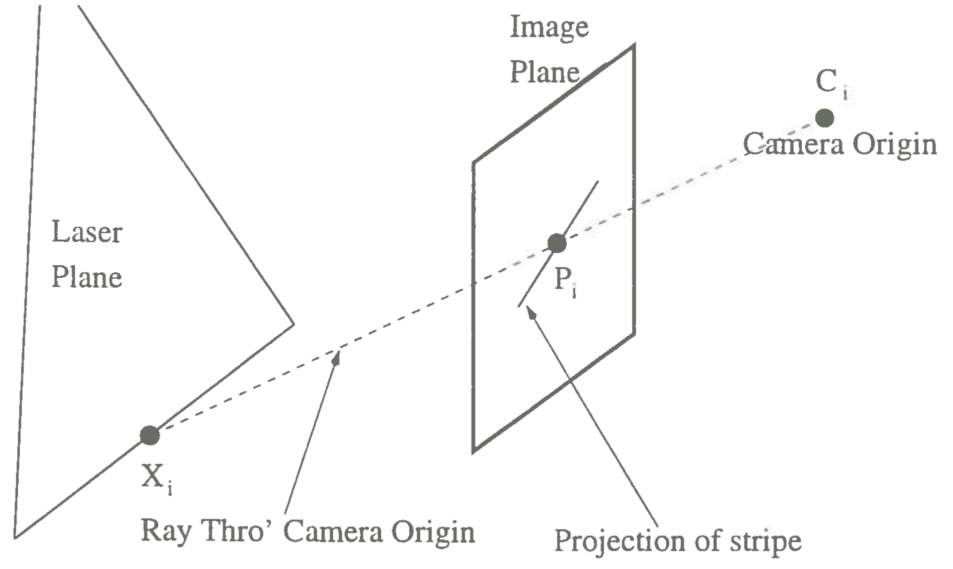

Figure 2.3: Laser plane triangulation

This simply states that the point $\mathbf{X_i}$ lies on the same ray as $\mathbf{C_i}$ and $\mathbf{P_i}$, at an unknown distance λ (if $\mathbf{C_i}$ and $\mathbf{P_i}$ are normalised, otherwise λ is a scale-factor)from the camera origin. To solve for λ, the above expression for $\mathbf{X_i}$ is substituted in to the plane equation like so:

$$(\mathbf{C_i} + \lambda(\mathbf{P_i} - \mathbf{C_i})).\mathbf{n} = d$$

A range image can be built up once $\mathbf{X_i}$ is found (using λ):

$$\mathbf{X_i} = \mathbf{C_i} + \left[\frac{d - \mathbf{C_i}.\mathbf{n}}{(\mathbf{P_i} - \mathbf{C_i}).\mathbf{n}}\right](\mathbf{P_i} - \mathbf{C_i})$$

Where $\mathbf{n}$, d and $\mathbf{C_i}$ are known in advance through calibration.

2.3 The Edinburgh Hand-Held Sensor

This description of the hardware and software used in the prototype system is largely based on [Fisher *et al.* 96]. The software component can be broken component where the gathered data is processed and the final model constructed. The sequence of use is geometry data collection followed by colour collection and then processing.

2.3.1 The Hardware

The prototype has the following hardware components whose use is described in more detail in the subsequent sections.

- "Flock of Birds" 6 degree of freedom (x, y, z,yaw,pitch,roll) position sensor. It provides the position of a remote unit (attached to the hand-held unit) relative to its base unit. A full discussion of its performance is the subject of Chapter 3.
- Panasonic CD1 colour CCD camera, with 7.5 mm focal length lens.
- Wide bandwidth light source – 20W halogen bulb, used during colour acquisition.
- DASM-VIP8 frame grabber. This digitises RGB signals and performs stripe detection.
- PAL-RGB converter: takes camera output and converts it into RGB signals for the frame-grabber. Electrocraft model TYPE-PD-84.
- Structured light projector. A 5 mW, 635 nm wavelength laser projector, emitting a stripe with near linear profile along its length.

For surface display, the RenderWare real-time 3D graphics library (from Criterion Software) is used.

On-line Range Data Acquisition

The sensor uses laser-stripe triangulation as described above with some changes. In the above description, all the components were fixed in calibrated positions, whereas the hand held unit is obviously free to move, so although the principle remains the same some adjustments to the formulae must be made. The Flock of Birds global position sensor will provide position and orientation of the remote unit within a one metre radius of its base unit. The remote unit is attached to the hand-held striper such that there is a fixed transformation between the remote unit, the camera and the laser plane. Rewriting the striper equations in terms of co-ordinates relative to the remote unit is trivial once this transformation is known. The flock of birds sensor now returns the transformation between the base unit and the remote unit which allows the global position of every range point to be calculated. At time t, the global position of x_{global} of a range point x_{remote} relative to the remote unit is given by

$$x_{global} = R_t \times x_{remote} + T_t$$

Where R_t and T_t denote the rotation and translation of the remote unit respectively. Each point captured can now be stored in the global co-ordinate frame, correctly positioned with respect to other collected data.

Two calibration processes (described in Appendix A) are used to calculate the transformation between the remote unit and the camera/laser plane and the camera parameters for the triangulation process.

Off-line Pre-Processing

The acquisition stage gives a ‘cloud’ of points which is centred on the surface, but with many outliers, largely caused by random ‘glitches’ in the global position reading. A surface fitting algorithm, used to fit a triangulated mesh to the surface implied by the data, is described below and in [Hoppe *et al.* 92]. However processing is required to remove these. There is also a large amount of random noise on the data, caused by fluctuations on the position and orientation readings given by the global position sensor.

A clustering algorithm is used for pre-processing by the prototype system. The algorithm (described below) essentially clusters the points into spheres and then rejects any sphere with too few points in it as being an outlier. The output of the algorithm is the average point position from within each sphere, the aim of this averaging being to smooth the random noise.

```
// make initial ball centre list
   ball_centre_list = []

// make initial ball centres
   for each point P,
      if within distance D of any ball centre
         then ignore P
         else add to ball_centre_list

// assign points to balls
   for each point P,
      assign to ball in ball_centre_list with closest centre

// delete outlier balls
   delete all balls with less than T range points

// form output points
   average points in each ball
```

This process is both time-consuming – with a large number of points and a small radius D, it can take up to an hour to run (on a Sun SPARC 10/51 required to remove the final outliers.

Off-line Surface Reconstruction

After the pre-processing stage, the resulting data set (a smoothed version of the original set containing fewer points) is passed to the Hoppe [Hoppe *et al.* 92, Hoppe 94] algorithm. This algorithm aims to estimate the underlying surface of the data set by approximating a signed distance function $f(\mathbf{p})$ to the surface. If a point $\mathbf{p}$ is inside the surface, $f(\mathbf{p})$ is positive; outside the surface f will be negative, and if $\mathbf{p}$ is actually on the surface, then the distance function will evaluate to zero. This signed distance function is used by a contour-tracing algorithm which fits a triangulated mesh to the data points approximating the surface.

It is important to note that $f(\mathbf{p})$ is never explicitly quantified – this method does not perform a function fit, and this allows free-form surfaces of arbitrary topology to be recovered. The signed distance function is, instead, approximated as a distance from a local tangent plane to the point $\mathbf{p}$ (Figure 2.4). The local tangent planes are calculated using principal components analysis (PCA) to produce the least-squares best-fit plane to the k nearest neighbours of point $\mathbf{x}_i$ ($Nbhd(\mathbf{x}_i)$), the parameter k being user-defined. Increasing k enlarges the area over which the normal is calculated, which has the effect of smoothing out local variations in the normals. This can lead to local noise being smoothed, but unfortunately it also means sharp changes in surface orientation (i.e. corners) will be smoothed over too. The local plane centre $\mathbf{o}_i$ is the mean position of $Nbhd(\mathbf{x}_i)$, the local normal $\mathbf{n}_i$ is the eigenvector corresponding to the smallest eigenvalue from the PCA.

With the set of local tangent planes calculated, $f(\mathbf{p})$ can now be estimated for any point $\mathbf{p}$ by

$$f(\mathbf{p}) = (\mathbf{p} - \mathbf{o}_i).\mathbf{n}_i$$

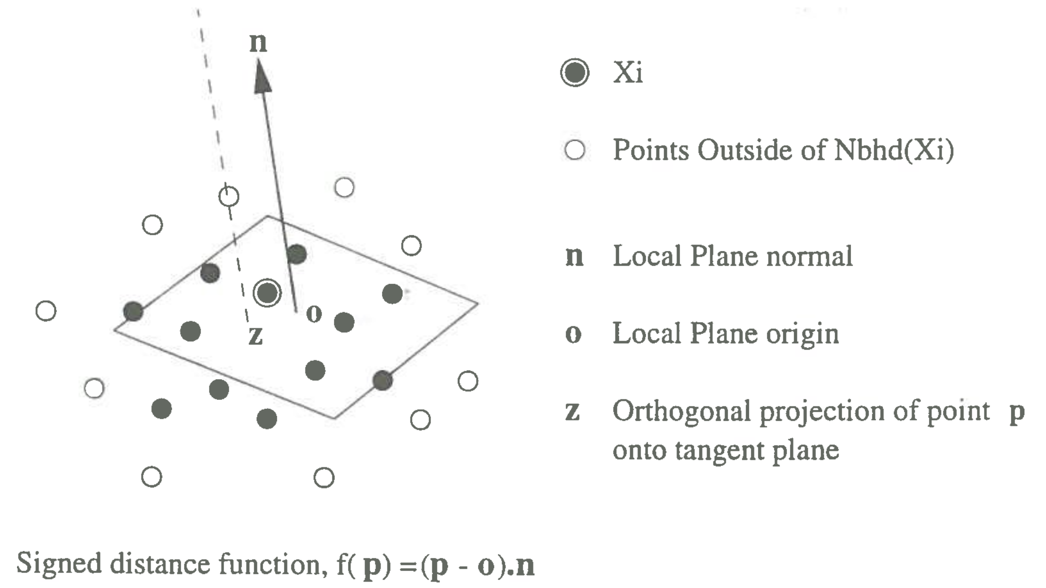
n
o
z
Xi
Points Outside of Nbhd(Xi)
n Local Plane normal
o Local Plane origin
z Orthogonal projection of point p onto tangent plane
Signed distance function, f(p) =(p - o).n

Figure 2.4: Local tangent plane

mal. However, it should be noted that this assumes that all surfaces considered are unbounded (i.e. have no edges) – which obviously will not always be true (consider a cube, for example). In these cases, it is important to judge whether a point $\mathbf{p}$ can possibly lie on the surface or not.

If the sample data set is ρ dense (i.e. any sphere centred in the data set with radius ρ will contain at least one sample point) and δ noisy (the random noise on any sample has maximum size $|\delta|$) then any point which is further than a distance $\rho+\delta$ from the data cannot belong to the surface. When assigning a value of $f(\mathbf{p})$ to a point in space $\mathbf{p}$, if the projection of $\mathbf{p}$ onto the local tangent plane is further than $\rho+\delta$ from the local origin $\mathbf{o}_i$, then $f(\mathbf{p})$ is said to be undefined.

Now a definition of $f(\mathbf{p})$ has been established, a mesh approximating the surface may be built up. For this purpose, Hoppe uses a variant on the "Marching Cubes" algorithm [Lorenson & Cline 87, Wyvill *et al.* 86]. Space is divided into a cubic grid, and for each cube vertex $\mathbf{v}$, $f(\mathbf{v})$ is evaluated. Any cube with $f(\mathbf{v})$ defined at each vertex is placed on a queue for visitation by the marching the value given by $f(\mathbf{v})$ assigned to it. If any edge $\mathbf{e}$ connects two vertices with different signs of $f(\mathbf{v})$, then the surface must intersect that edge. In this way, a triangular mesh is constructed whose vertices are the positions of intersection of surface and edge – these positions calculated by (linear) interpolation of the values of f at the two vertices. The algorithm then 'marches' on to triangulate the next cube on the list.

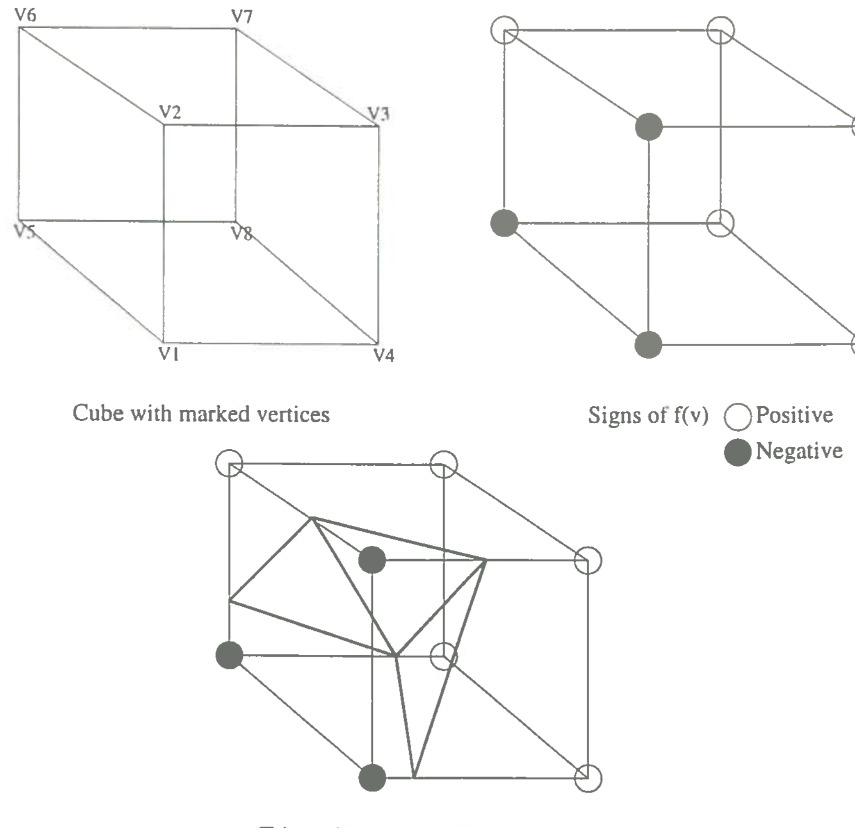

Figure 2.5: Mesh triangulation

The marching cubes algorithm produces the initial triangulated mesh which is then passed, along with the original data points given to the Hoppe algorithm, to a mesh optimisation stage, which aims to balance accuracy to the data and conciseness of mesh description, using an energy minimisation process described

On a typical data set from the clustering algorithm (1000-3000 points), the Hoppe algorithm takes about 1 minute to run. However, it is often necessary to repeat the process if there are still outliers in the clustering stage output – these must be removed by further hand editing for best results.

Colour Acquisition

The process of colour grabbing is as follows: the laser is switched off, and the halogen bulb is switched on. Enough colour frames are then taken to cover the entire object – attaching to each colour sample the global position sensor reading so that the view may be positioned correctly in space.

The texture mapping algorithm currently used maps colour and not surface reflectance onto the geometric model. In general, a reflectance recovery algorithm will lead to more realistic texture mapping, but in this case direct colour measurement from the camera is viable for several reasons.

- The halogen bulb light source near the camera optical axis eliminates most shadows that might otherwise be seen by the camera.
- If the scanner is approximately the same distance from the surface for each colour shot then surface illumination, from ambient light supplemented by the halogen bulb, will be approximately constant.
- The automatic gain control on the camera works well enough to provide fairly constant colour signals
- Only regions of the surface whose normals are within 45 degrees of the camera's optic axis are colour mapped – if the normals are pointing toward the camera, shading effects are reduced.

Although the colour mapping routine is currently used, a surface reflectance algorithm has been proposed [Fisher & Gionis 96], but has yet to be fully integcorrespondence in the following way:

```
For each pixel(i,j)
  if (ray through (i,j) intersects with model)
     if (model surface normal at point of intersection is
         oriented less than 45 degrees from view direction)
           give point of intersection RGB colour value of pixel(i,j)

If more than one colour estimate is given for a point, average them.

Blur the colour values with Gaussian mask to reduce noise

Propagate colour values into any uncoloured surface areas.
```

The pixel-ray correspondence is that used in the range data collection, known from the camera calibration, and the model surface normals can be found using the geometry of the model. The size of the colour-mapping cells on the object surface was 1 mm^2.

2.4 Weak Points of the Sensor

There are many improvements which could be made to the system, as mentioned in Chapter 1, which range from improving real-time display of data as it is collected to bettering the performance of the texture-mapping. The problem addressed by this work is that of noise reduction on the geometry of the models, both by gaining further insight in to the performance of the Flock of Birds sensor and to implement a more efficient and effective data smoothing algorithm to pre-process the range data for the Hoppe algorithm.

Chapter 3

Fine Calibration of the Flock of Birds Sensor

Presented here are details of the work undertaken to understand and finely calibrate the Flock of Birds electromagnetic position sensor. The sensor is central to the data capture phase – it provides information on the global position of the camera and laser plane, thereby allowing each stripe to be located with respect to its neighbours and a surface of data points to be built up. It also enables registered placement of colour information in the texture-mapping phase.

As with any sensor system, the Flock of Birds sensor has its imperfections, and experiments were designed to further the understanding of these flaws and hopefully to compensate for them. First is a brief introduction to the sensor and its specifications, then details of experiments made, conclusions drawn and how this new information was integrated into the whole system and the results of this integration.

3.1 Specifications

The Flock of Birds sensor consists of a base unit and a single remote unit (although more may be added, see [Asc95]), both feeding into a control unit which sensor only operates within one hemisphere – measurements made outside of this region will be unreliable). From the received signal, it is possible to calculate the three-dimensional translation and rotation relative to the base unit to a certain degree of accuracy within a specified range from the base unit. Table 3.1 shows certain characteristics of the sensor; for fuller details consult [Asc95].

Parameter	Value
Positional Range	±1m (any direction)
Angular Range	±180^o Azimuth & Roll
	±90^o Elevation
Positional Accuracy	2.5mm RMS
Positional Resolution	0.75mm
Static Angular Accuracy	0.5^o RMS
Angular Resolution	0.1^o

Table 3.1: Flock of Birds specifications

As mentioned in Chapter 2, the Flock of Birds sensor provides the global position information which aligns each stripe in space and allows pasting of the colour data onto appropriate areas of on object. Initial use of the range-scanner highlighted two significant undocumented problems worthy of investigation. Firstly there is an occasional sensor 'glitch' resulting in an inaccurate camera and laser-plane position being sent to the computer, which leads to a stripe being positioned incorrectly in space. Incorrectly placed range data will, if not removed, lead to unwanted protrusions or concavities in the reconstructed surface. The second effect was noticed in the texture-mapping phase. Separate colour frames which should contain some overlap were often positioned several millimetres apart – the consistency of such errors suggested some type of systematic error could be present in the sensor.

In order to reduce the effects of the random and systematic errors noted above.

The experiments below are split into two categories which study the translational and angular performance of the sensor. To minimise disturbance of the transmitted magnetic field, all experiments described were performed on a wooden table, with no metallic objects in the close vicinity of the sensor. It should also be noted that the sensor is configured to operate in the hemisphere where z is negative. It is also important to note that the (x, y, z) origin is not known precisely, so distance measurements given are therefore relative to a nearby origin. Figure 3.1 shows the axes of both the base unit and the remote unit.

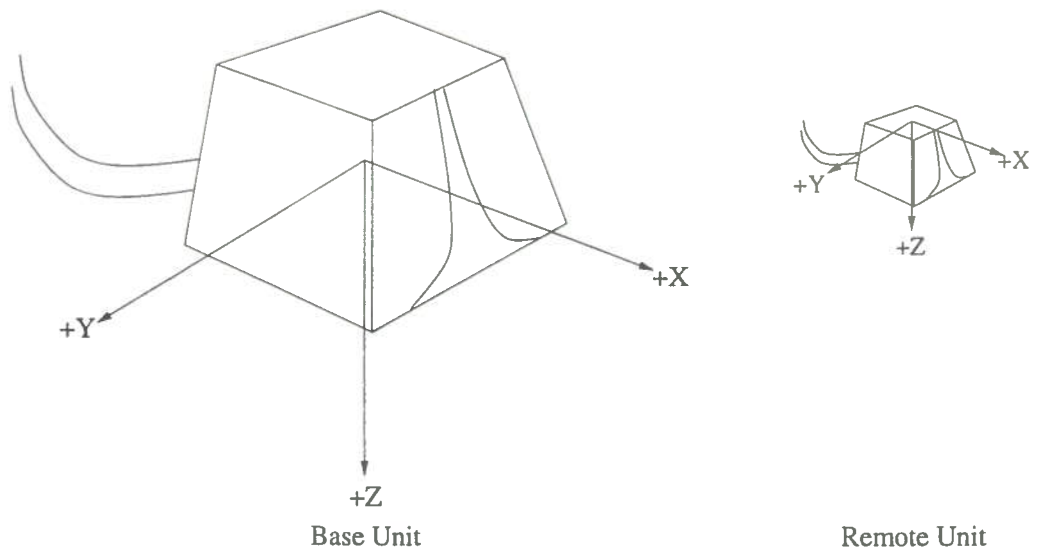

Figure 3.1: The Flock of Birds axes

3.2.1 Translational

The tests described below were undertaken in order to better quantify potential systematic and random errors in the measurement of translations of the remote unit.

This experiment is illustrated in Figure 3.2, and involved holding the remote unit at each of the nine positions shown, $x = 10$ to $x = 90$ cm, where x is measured globally, the global x axis parallel to the Flock of Birds base unit x axis, with global and Flock of Birds base unit origins approximately coincident. Four hundred samples were taken at each position, and the mean and standard deviation of the x measurements calculated, which are shown in Table B.1. For each measurement, global y was $0cm$ and global z $-2.5cm$. Standard deviation versus mean X location is plotted in Figure 3.3. The standard deviation of the measurements increases with distance from the base unit, but is generally very low; the worst reading at 90 cm is only just over $0.2mm$. The standard deviation is directly related to the accuracy – if it is assumed that the noise on the data is Gaussian, over 99% of the measurements will lie within ±3 standard deviations of the mean. In the worst case described above, this will give an accuracy of within 1.2mm, which is better than the manufacturer's specification. However, the dynamic tests described below revealed further noise.

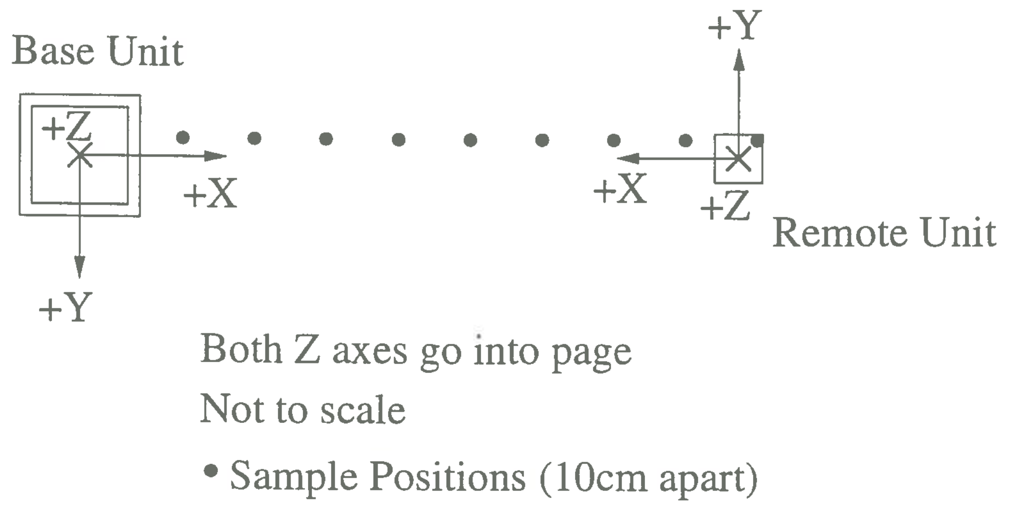

Figure 3.2: The static position experiment (schematic, plan view)

For the samples taken at $x = 80cm$ and greater, 600 samples were taken and of the curve in Figure 3.4 is approxim

behaviour of many real world systems to a step change in state, such as switching on a power source. The graph also shows the measurement quantisation of the sensor.

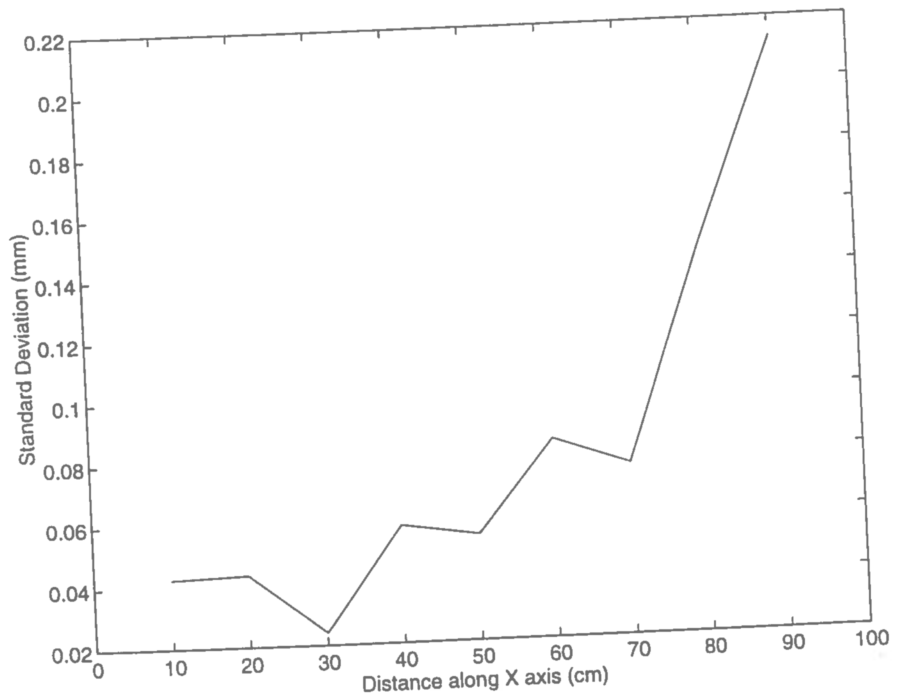

Figure 3.3: Translational standard deviation *vs* x location

Plane Fitting

These experiments were carried out to find how well points gathered from a plane actually fitted that plane, and to investigate the symmetry of the sensor. The base unit was placed on a table and the remote sensor moved on planes parallel to the table at various heights, along the 4 directions shown in Figure 3.5. Points are sampled at regular $2cm$ steps along each of the directions A,B,C and D. Due to space restrictions, to sample the full extent of each direction, it must be measured in three sweeps, also illustrated in Figure 3.5.

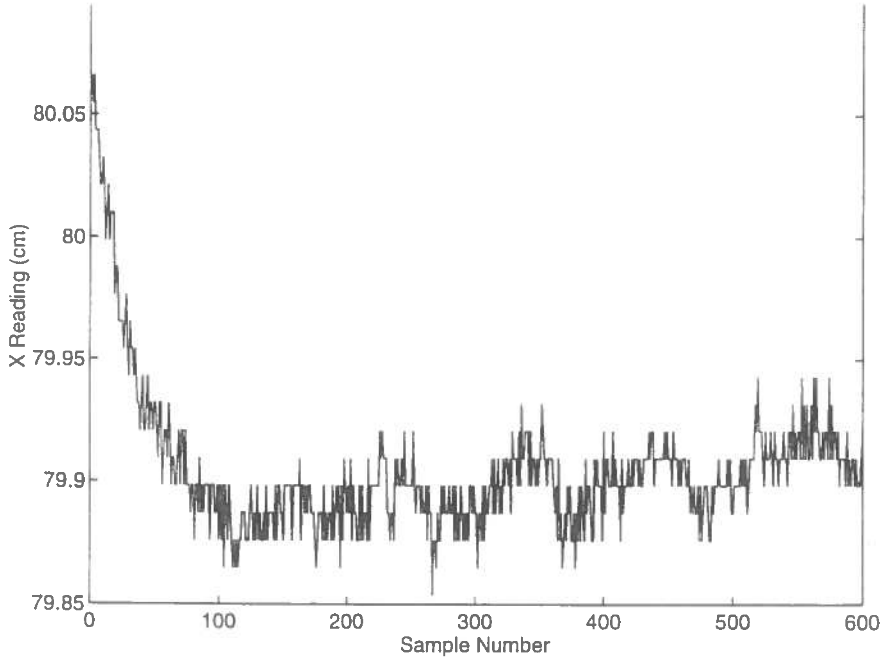

Figure 3.4: Reaching a steady state

The raw data gathered from one such test on a plane with z measurement of $-6cm$ is shown in Figure 3.6. The data along each direction is shown in Figure 3.7, each graph (a) to (d) plots three curves corresponding to the three sets of measurements taken in each direction. There are two points to note about the plot; the first is that although the data was gathered from a plane, there is obviously an underlying curve – this shows a systematic (as opposed to random) error in the readings. Secondly, there is a region around the z axis where the data is very irregular, and certainly cannot be said to lie in (or even near) the same plane as the rest of the points.

The graphs shown in Figure 3.7 show both the curve and the 'spike' noise in the centre of the field. The curves also show that the field is essentially symmetrical about the z axis. Performing the tests with the remote unit at various angles to the plane revealed that the curve of the field is independent of

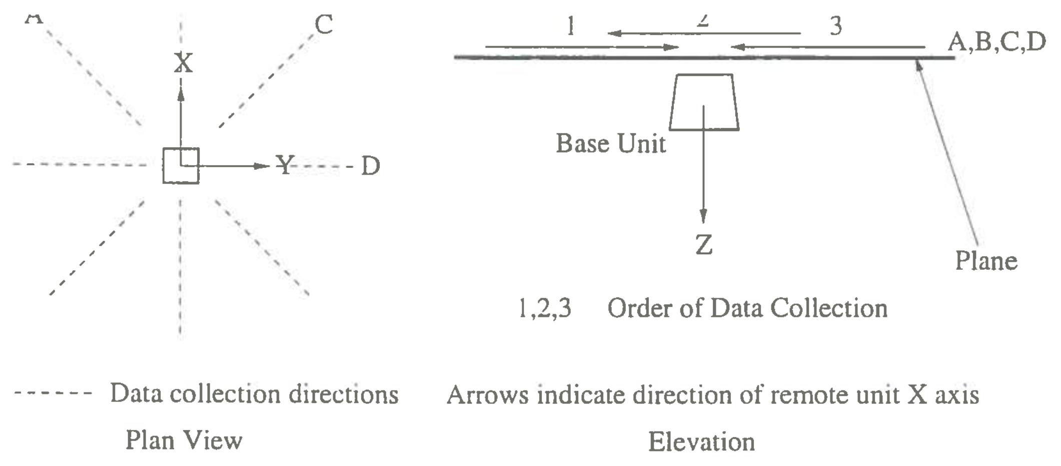
C
X
Y
D
Data collection directions
Plan View
1
3
A,B,C,D
Base Unit
Z
Plane
1,2,3 Order of Data Collection
Arrows indicate direction of remote unit X axis
Elevation

Figure 3.5: Plane data collection

x, y and z.

For the plane fitting, it was decided to gather more dense data than at 2cm intervals, in order to improve the statistical accuracy of any conclusions drawn. The procedure was altered slightly to reduce experimental error – instead of placing a plane above the sensor base unit, the base unit was placed upside down at various positions above the table top (Figure 3.8), which it was hoped would provide a very rigid, flat surface (although it must be mentioned that the flatness of the table top is not guaranteed). For this reason, it appears that the curve of the data is reversed, but this is simply a measurement artifact. It was decided only to use data from the region away from the z axis, in order to avoid the 'spiky' central region and the step discontinuity which would adversely effect the plane fit, so for each plane, 8 sample sets were taken, i.e. one set from each side of the z axis in directions A-D from Figure 3.5. Planes at 0, 4, 11 and 23 cm above the table top were measured. These planes will be referred to as planes 1, 2, 3 and 4 respectively.

Figure 3.9 shows a typical dense sample plot, where the remote unit was moved at a constant pace along a straight line in a plane. As can be seen,

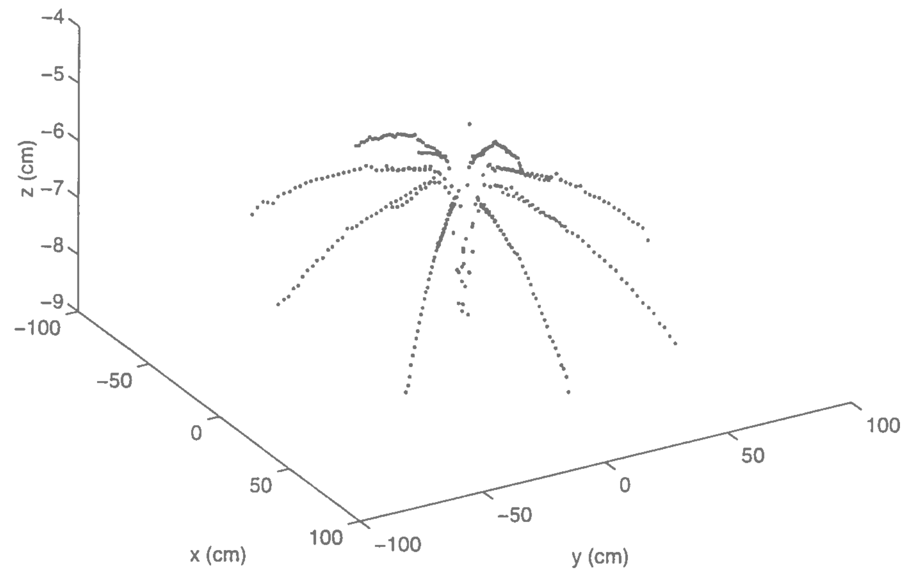

Figure 3.6: Raw data output from a plane ($z = -6cm$)

carried out above, possibly due to small irregularities in the electromagnetic field. The magnitude of the noise seen here is more in line with the manufacturer's specifications, and the systematic error is also visible.

All the points gathered from a plane were used in a least-squares plane fitting algorithm ([Faugeras 93], Section 10.7) which gave an equation for each plane in the form

$$\mathbf{n}.\mathbf{x} + d = 0 \tag{3.1}$$

where $\mathbf{n}$ is the best fit plane normal vector (in each case all but coincident with the z axis), and d the perpendicular distance from the plane to the origin. The residual of each data point was then calculated using the formula

$$r = \mathbf{n}.\mathbf{x}_{\mathbf{data}} + d \tag{3.2}$$

where r is the calculated residual for each data point $\mathbf{x}_{\mathbf{data}}$. Figure 3.10 (a) plots

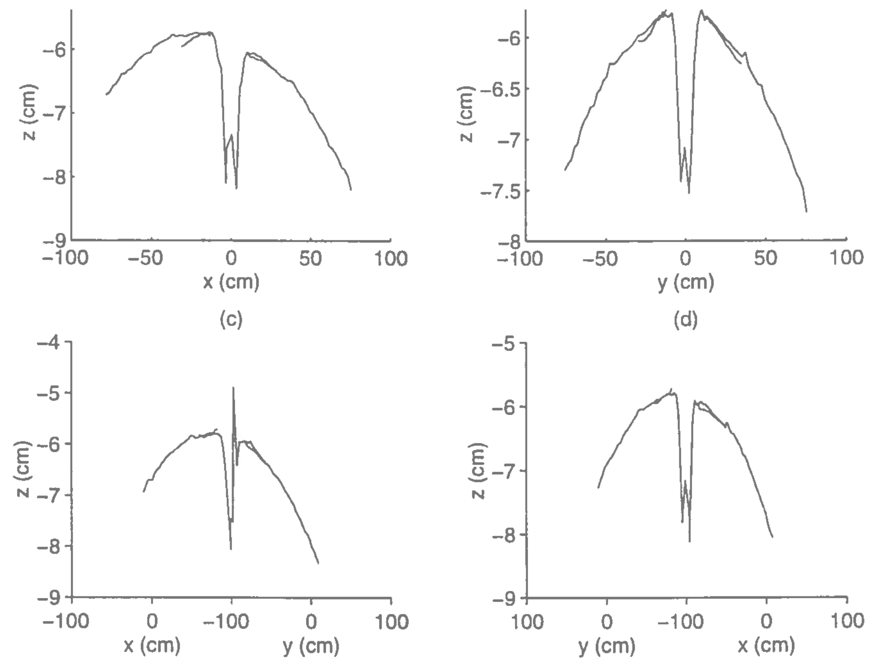

Figure 3.7: Views of the data along: (a) x,(b) y,(c) $x = y$,(d) $x = -y$

1. From inspection of the data, it was judged that the systematic error could be approximated by a second-order polynomial (Figure 3.10 (b)) in the radius from the z axis, which could be fitted to the residual data using least-squares methods. Such an equation will have the form

$$\hat{r} = aR^2 + bR + c \qquad (3.3)$$

where $\hat{r}$ is the residual model and R the radius from the z axis, i.e. $\sqrt{x^2 + y^2}$, and a, b and c are determined by the least-squares fit.

Data Correction

With the fitted plane normals being very close to the z axis in orientation, it is acceptable to simply calculate the value of the correction formula (Equation 3.3) at the given (x, y) position and subtract this from the z reading. The effectiveness

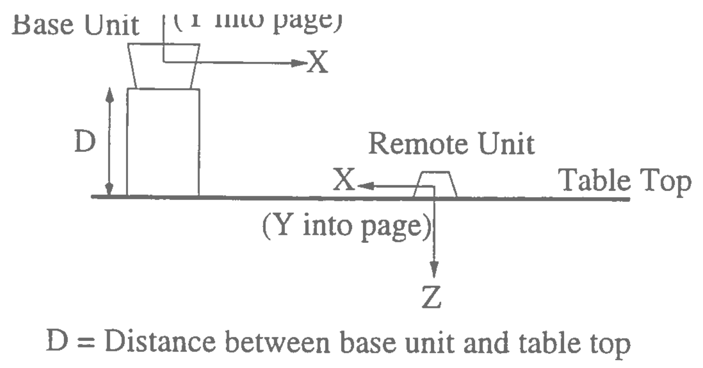

Figure 3.8: New experimental procedure

newly corrected data. The result for each sample set from Figure 3.10 (a) is shown in Figure 3.11. Figure 3.12 histograms the magnitude of the residual; (a) showing the corrected data, (b) the uncorrected. The histograms make the benefits of the correction quite obvious – the range of values taken by the residual is reduced and the noise now has a Gaussian distribution, centred on zero.

The curve-fitting and correction procedures outlined above were then generalised for planes at different levels. A single correction curve was fitted to the residuals from all four fitted planes, and is given by

$$\hat{r} = 0.0003R^2 - 0.0079R - 0.2899 \tag{3.4}$$

To check the effectiveness of the Equation 3.4, it was used to correct the data from all four test planes and the residuals from the general correction compared to the residuals from the plane-specific correction formula. The results are summarised in Table 3.2 which shows the standard deviation of the set of residuals for both the general and the plane-specific method (i.e. correction formulae of the form of Equation 3.3 fitted for each of the planes 1,2,3 and 4) for each plane. It can be seen from the table that the standard deviation resulting from the plane-specific and general methods is very similar for planes 2,3 and 4 – only for plane 1 is there a large decline in performance. Although it would be possible to use the

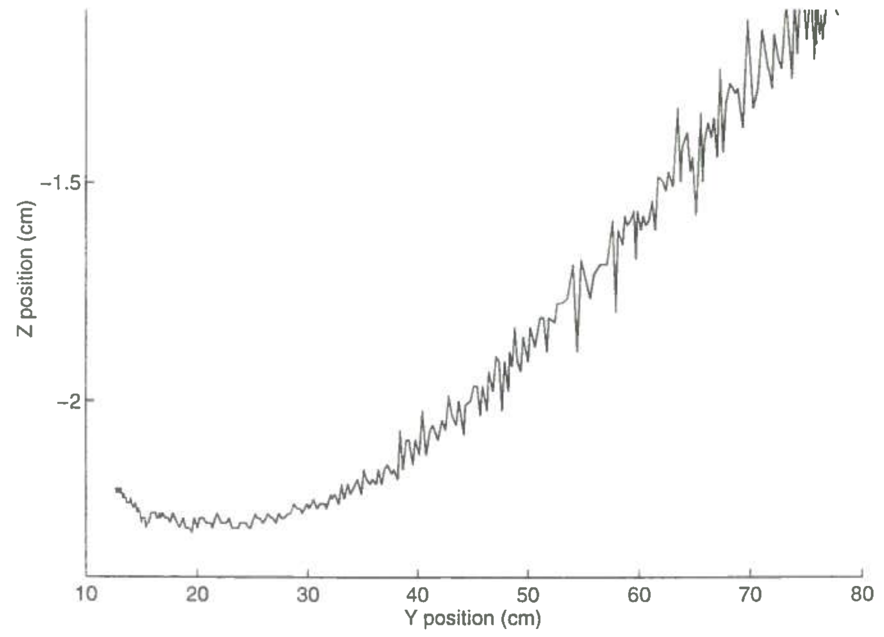

Figure 3.9: Plot of raw dense data (along the y axis, plane 1)

effective correction, this is not straightforward – a relationship between the curves at different heights is not obvious. For this reason, the general curve suggested above (Equation 3.4) was used.

Plane	Standard Deviation (cm)	
	Plane-Specific	General
1	0.0582	0.1151
2	0.1113	0.1117
3	0.0935	0.0964
4	0.0682	0.0772

Table 3.2: Comparison of plane-specific and general correction

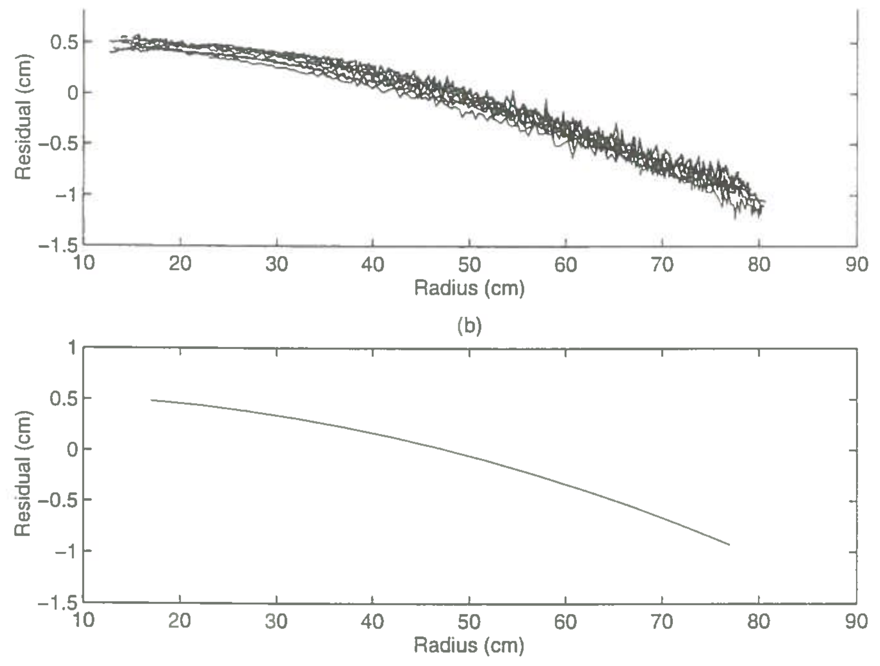

Figure 3.10: (a) Plane residuals *vs* radius from z axis, (b) model of residual

3.2.2 Angular Measurements

The tests described here were performed to identify any irregularities in the angular measurements taken by the sensor. Tests were undertaken looking at both the noise on static measurements and the linearity of the sensor's measurements.

Static Angular Accuracy

As with the translational tests, the aim of this experiment was to check the variation in sensor reading whilst the remote unit was held stationary. The basic procedure was the same as that shown in Figure 3.2, but samples are taken between $x = 15cm$ and $x = 75cm$, and the quantity measured was not position but the rotation matrix providing the transformation between base and remote units. This rotation matrix was then used to rotate a known vector [1 0 0] and the standard deviation of the angles of this rotated vector with respect to the

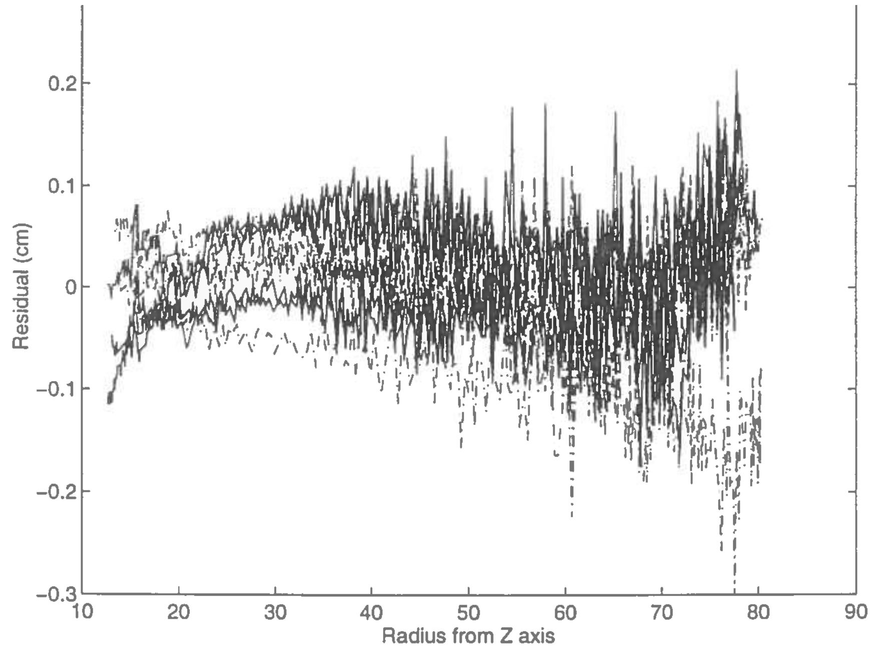

Figure 3.11: Residuals from corrected data

shows the full results, which are also plotted in Figure 3.13. Unlike the translational measurements, the standard deviation does not get obviously worse over the range of the sensor. The worst case has a standard deviation of 0.4223^o, which will give an error of c.$\pm 1.3^o$ – this is worse than the manufacturer's specification of 0.5^o.

Linearity

There are two aspects to the linearity here; whether the sensor held at a constant orientation gives a (near) constant reading as it is moved away from the origin, and whether the measurement of a known (10^o) rotation remains constant over the working range. A simple experiment was devised whereby the sensor is stepped over 180^o in 10^o increments (see Figure 3.14) at seven different radii from the z axis (15 - 75 cm in 10cm steps). Again, the rotation matrices sampled at each

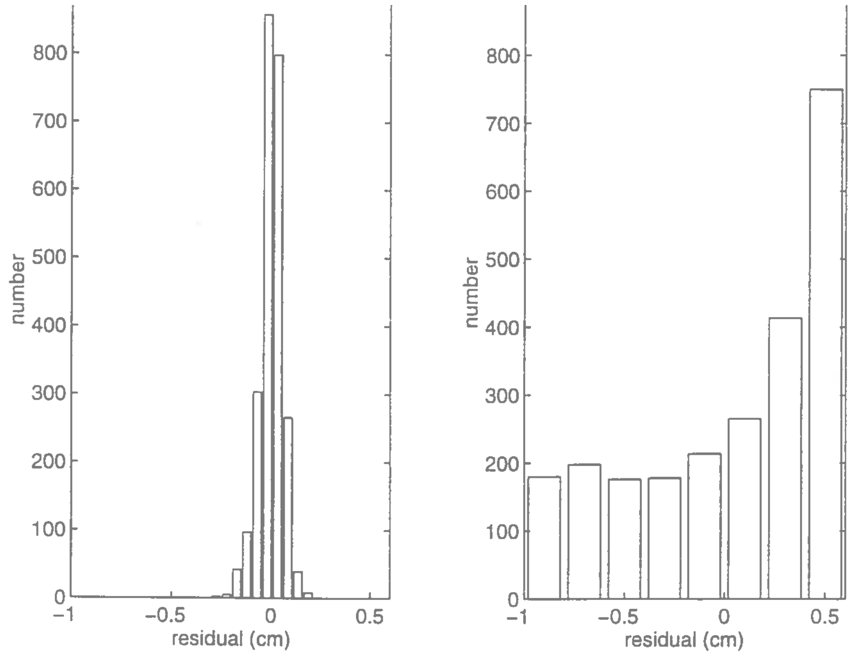

Figure 3.12: Residual histograms (a) corrected data, (b) uncorrected

orientation are used to rotate a known vector, and the angles measured are the angles between the rotated vector and the unrotated vector. Figure 3.15 shows the plots of (a) all seven sensor tests and (b) the residual of each measurement from a line fitted to the test data. It should be noted in (a) that the tests are not coincident (although the scale of the graph hides this), i.e. they do not all run from exactly $90 - 180^o$, and this is due to the tests taking place at slightly different orientations at each distance. What is obvious from the graphs is that the readings are fairly linear. The plot of the residual shows that there is little to indicate a systematic error – the values seem to be quite randomly distributed around zero. In this case, no systematic correction need be made, but if possible, the average of several readings should be passed on to the stripe positioning software, as this will reduce the random effects of the noise.

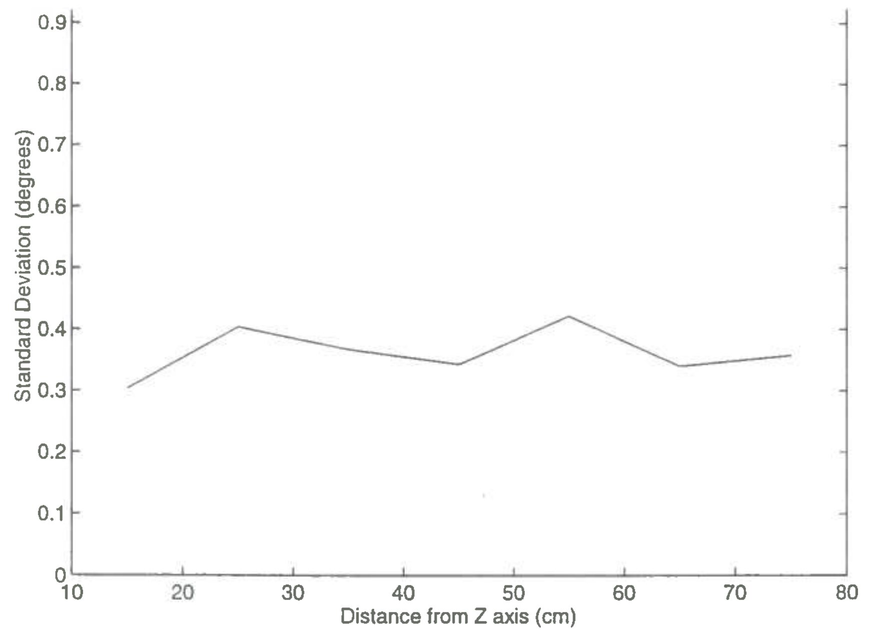

Figure 3.13: Angular standard deviation *vs* location

3.3 The Sensor with the Hand-Held

The remote unit was next attached to the hand-held unit, and the experiments leading to the derivation of a correction formula (Equation, 3.3) were repeated, although due to the fact that the remote unit is now fixed to the hand-held, the procedure had to be altered slightly (Figure 3.16). Planes were sampled with the base unit at the following heights D above the table (in cm): 14.5, 17.8, 21, 23, 33.6, 40.5, 44.3, 51.5. A new function was then fitted to all of the residuals from these data sets, resulting in the following correction formula

$$\hat{r} = -0.0001R^2 - 0.0042R + 0.4231 \qquad (3.5)$$

which was then incorporated into the stripe collection software. Unfortunately, the results were less than satisfactory – Figure 3.17 shows the effect of adding the correction given in Equation 3.5 on striping data from the plane $z = 0$ (i.e.

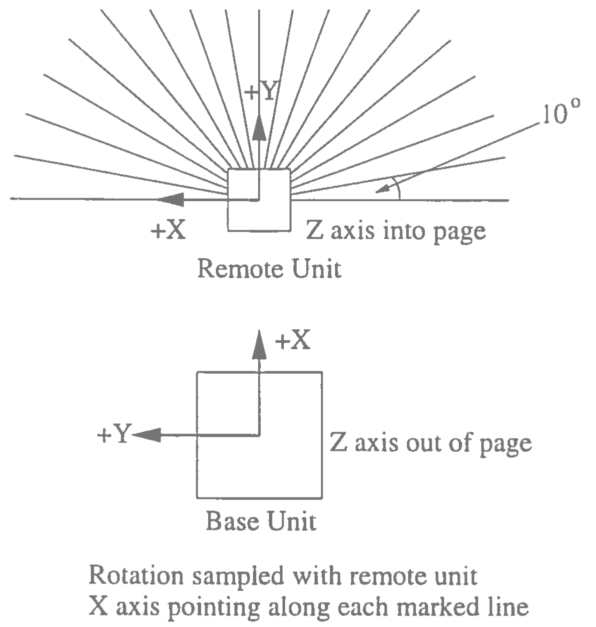
+Y
10°
+X
Z axis into page
Remote Unit
+X
+Y
Z axis out of page
Base Unit
Rotation sampled with remote unit
X axis pointing along each marked line

Figure 3.14: Angular test procedure

revealing that the correction makes the spread of the data about twice as bad as without the correction).

The reason for the poorer performance is that the model assumes that a single correction formula will perform well over the whole workspace. The results given in Table 3.2 suggested that this was the case when the remote unit was not mounted on the striper. Further study of data collected when the remote was attached showed why the model was unsuitable; Figure 3.18 shows (a) the correction curve which was fitted to the set of residuals from all of the planes , and (b) the curve fitted to the residuals from individual planes, the graphs plainly showing that the sign of the curvature of the residual changes with z position.

To cater for this change in curvature, the option of adapting the model to change with z reading was investigated. If the coefficients of the correction polynomial (Equation 3.3) change with z in a way that can be modelled, then it would be possible to to make a residual model of the following form:

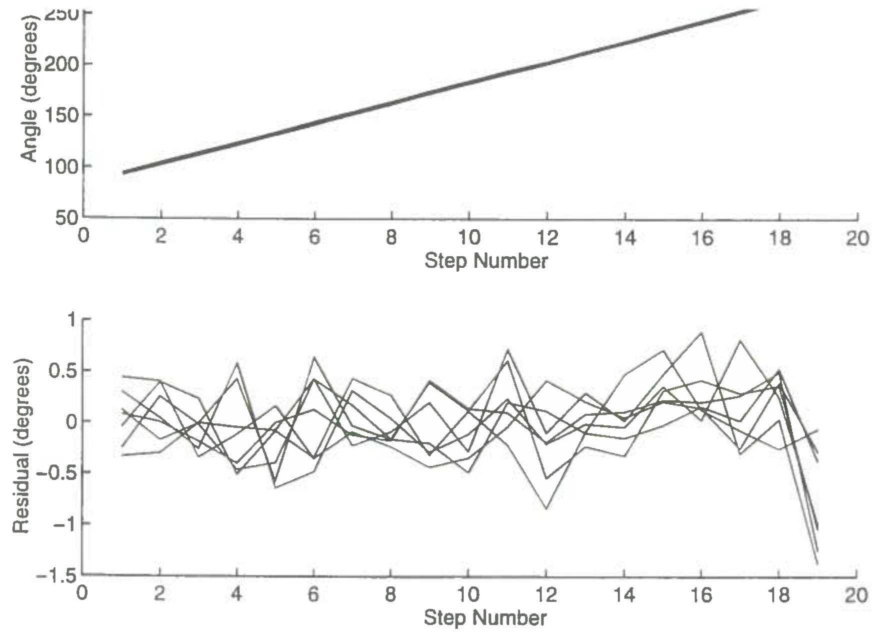

Figure 3.15: (a) measured angle (b) residual

where the coefficients A, B and C are all functions of the z reading. To look for such a relationship, the values of the coefficients a, b and c from the equation (of the form of Equation 3.3) found for each plane were plotted against the z value of their corresponding planes. These graphs are given in Figures 3.19,3.20 and 3.21, and show no well-behaved change with z reading; so a model of the form of Equation 3.6 cannot be constructed. The random scatter of the graphs is unlikely to be the consequence of random noise on each experiment; to produce the model for each plane, hundreds of data points were gathered, so the distribution of the data used to fit each equation should not be dominated by noise.

It is difficult to conclude what causes this variation, as very little information is given about the physics of the Flock of Birds device. The manufacturers have been contacted for advice, but, at the time of writing, they have yet to reply.

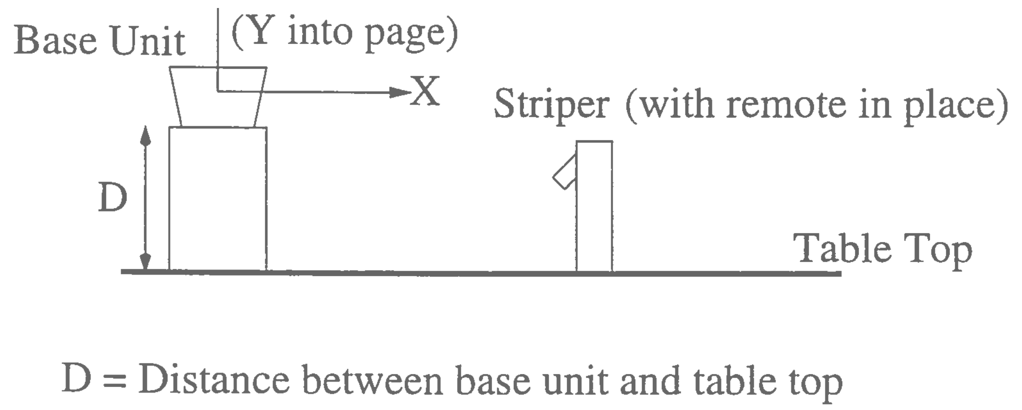
Base Unit
(Y into page)
X
Striper (with remote in place)
D
Table Top
D = Distance between base unit and table top

Figure 3.16: Procedure with hand-held unit

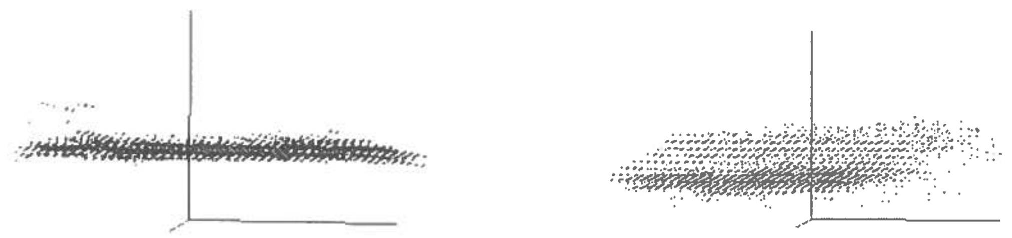

Figure 3.17: (a) Plane without correction (b) with correction

3.4 Conclusions on Correction

Unfortunately, although the problem of correcting the systematic position error initially looked quite straightforward, with a simple quadratic equation required to counter the error, the situation is more complex than this. Possibly the only effective way to address the problem is to sample the entire work space at fine, regular intervals and construct an interpolation table based on these results. This is a lengthy task, and sadly there was not enough time to examine it further.

The angular performance of the sensor does not appear to have a systematic error, but is very noisy. Variations of $\pm1.3^o$ will result in errors of up to $\pm4.5mm$

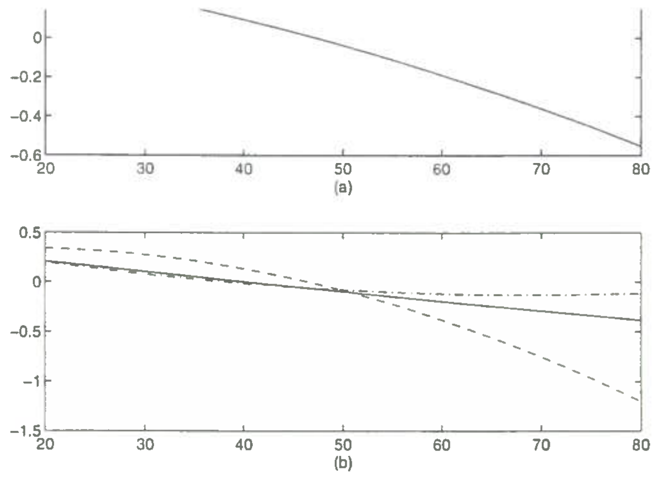

Figure 3.18: Correction curves (a) fitted to all planes (b) fitted to plane $z = -14.5cm$ (dashed), $z = -27cm$ (solid) and $z = -40.5cm$ (dash-dot)

and object. Such effects can only be countered by averaging repeated readings, although this brings its own problems, as the sensor is always being moved and successive measurements are unlikely to be from the same place. However, the sensor takes approximately 70 samples per second, so collecting three readings, for example, will only take c. $40ms$, in which time the sensor is unlikely to have moved far (to maximise data density, the user must move the sensor slowly). Obviously, repeated readings will also help reduce the effect of random noise on the translational measurements as well as rotational ones.

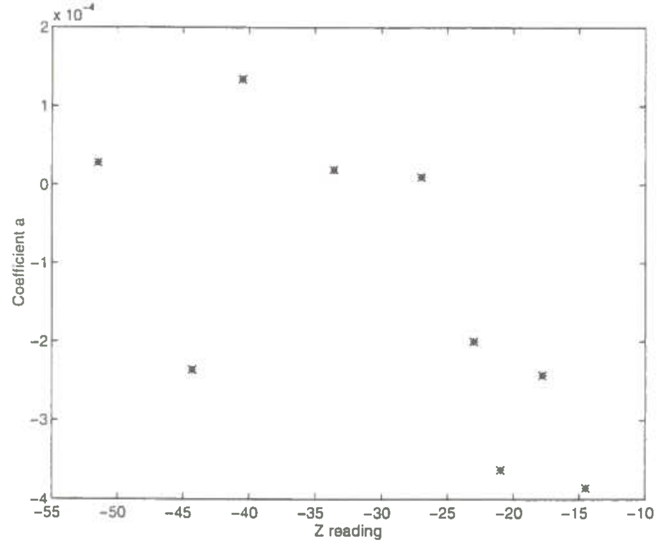

Figure 3.19: Model coefficient a *vs* z reading

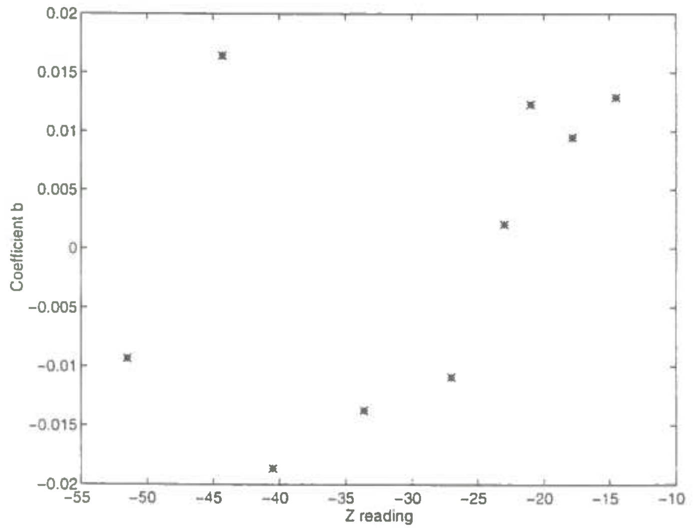

Figure 3.20: Model coefficient b *vs* z reading

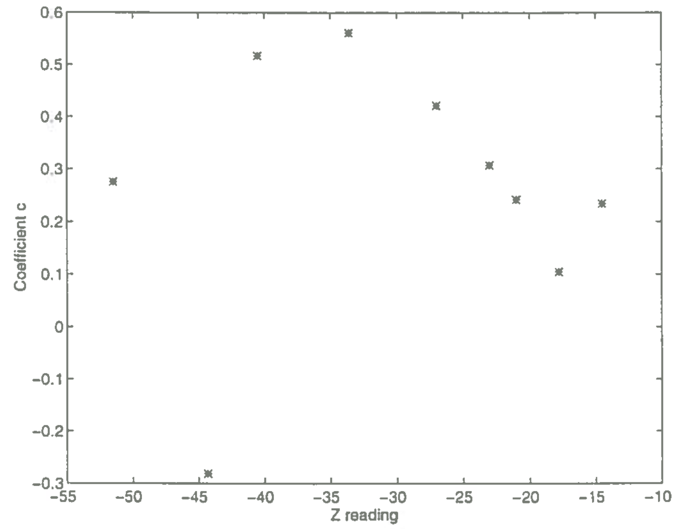

Figure 3.21: Model coefficient c *vs* z reading

Chapter 4

Cleaning the Range Data

The second major front on which noise can be attacked is at the data preprocessing stage, where the raw range data is ‘cleaned up’ prior to being fed to Hoppe’s surface reconstruction algorithm. The aims of this cleaning process are two-fold, to remove outliers from the range data and also to smooth the random noise on the surface data. Outlier points (those which lie far from the bulk of the data) cause unwanted distortions to the surface, whereas the random noise element can lead to surface smoothness being lost.

The work described here can be separated into two parts; the basic development of algorithms in two dimensions with tests performed on simulated data, and the extension of these algorithms into three dimensions.

4.1 The 2D Case

Before attempting to design algorithms to clean the full set of real 3D range data, the two dimensional case was studied. There are two reasons for this; the availability of the MATLAB program provided a ready-made environment for rapid development and testing of methods, and the two dimensional case is simpler than 3D for the implementation of algorithms (which could later be extended to 3D).

To design and test algorithms the basic problem must be identified and in the 2D case, with real data unavailable, some method of simulating data must be devised. In the real system, co-ordinates of points are captured off the surface of an object, building up what is in essence a three dimensional binary image, i.e. any position in space either contains a point (be it from the surface itself or a spurious noise point) or is empty. This trivially scales down to 2D, where points are captured from a line rather than a surface, the image still being binary. The real 3D objects can be modelled by simulated outlines in 2D.

The next step is to model the sampling process which captures the data points. the hand-held system gathers stripes from the surface, the inter-stripe spacing being determined by the speed of the stripe movement across the surface (which the user aims to keep constant) and the frequency of image capture (which is fixed). In 2D this stripe can be simplified to a point, although this implicitly assumes that all stripes taken are parallel, which in general is untrue. Ignoring the stripe structure means that algorithms will not take advantage of all the information available, but will not lead to any algorithms developed with the model over-performing and degrading in the 3D case, which is definitely to be avoided. The slightly irregular spacing of the sample points can be simulated by sampling the outline data at regular intervals with a little random noise added. This is illustrated in Figure 4.1, where the sample centres indicate the regular sample points, and the bounds the region in which the sample could actually be taken.

Now the image type and sampling process have been modelled, there just remains the simulation of the noise to perform. The main source of noise is that from the Flock of Birds sensor, which results in surface points being incorrectly positioned in space; this is modelled by adding noise to the position of each outline point sampled. Some random points are added to model the outlier noise.

A typical result of the sampling process is illustrated in Figure 4.2, (a) showing

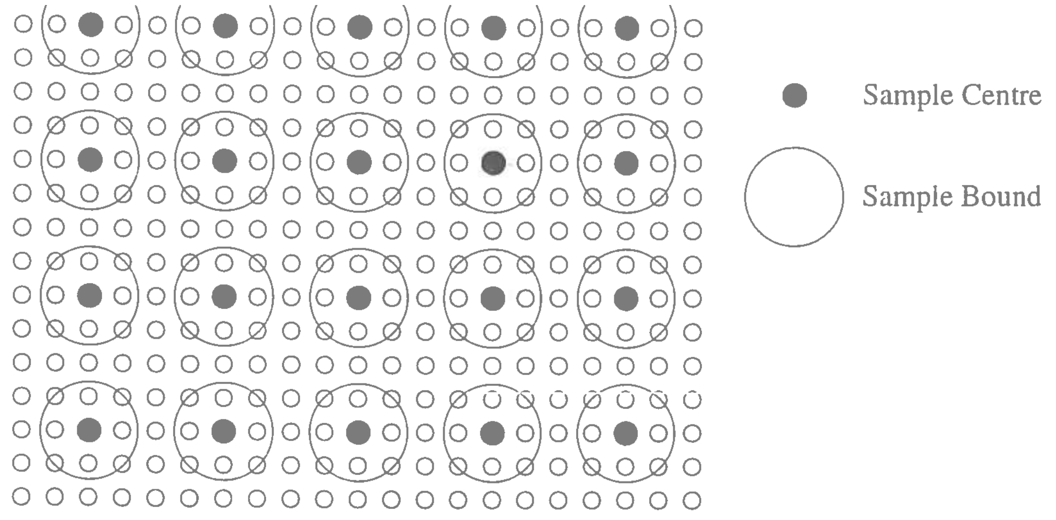

Figure 4.1: The near-regular sample model

4.1.2 Algorithms in 2D

With a sample model completed, some algorithms could be designed to perform the tasks of outlier removal and noise smoothing with the aim of producing a smoothed, thin version of the original data with no outliers.

Outlier Removal

To remove stray points distant from the bulk of the data, a morphological thinning operator was used. The basis of the thinning operator is that a set of features is extracted from the image, and then this set is removed from the image. Most commonly, the features extracted are the outsides of thick edges and corners (hence the 'thinning' name), but in this case it is the outlier points which are to be removed.

The initial feature extraction is performed by the 'hit-or-miss' transform. Unlike most other morphological operators, the hit-or-miss examines both image foreground *and* background to study their relationship. A structuring element pair (B below) composed of two separate elements (F, $\bar{F}$) is used to probe an

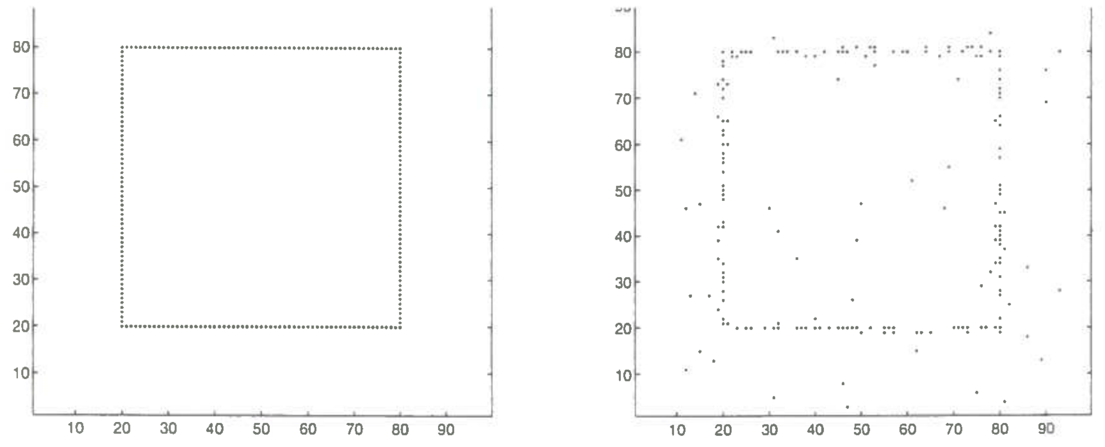

Figure 4.2: (a) Original outline (b) sample

image set A. E and F are translated to each point in A, and any such point at which E fits inside A and F outside A is passed through to the transform's output. More formally, this can be expressed:

$$A \circledast B = (A \ominus E) \cap (A^c \ominus F)$$

where $\circledast$ is the hit-or-miss operator and $\ominus$ the morphological erosion operator. A^c is the complement of set A. A suitable pair B for detecting outliers is shown in Figure 4.3, the outlier model being a single occupied point surrounded by a neighbourhood of empty points (zeros). This size of this neighbourhood is set according to noise and sample density, as in Section 2.3.2.

$\boxed{1}$

E

0	0	0	0	0
0	0	0	0	0
0	0		0	0
0	0	0	0	0
0	0	0	0	0

F

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

B

Figure 4.3: The structuring element pair

original image, simply performed by taking the difference of the original image and the set of outliers. The thinning of an image S by a pair B can therefore be described:

$$S \otimes B = S - (S \circledast B)$$

where $S \otimes B$ is the set-theoretic difference between S and $S \circledast B$. A good introduction to morphological image processing which covers all major operators is given by [Dougherty 92].

Results of the process are illustrated in Figure 4.4, where (a) shows the sampled data (with the outliers circled) and (b) the results of the outlier removal – any point which is alone at the centre of a 5×5 neighbourhood has been removed.

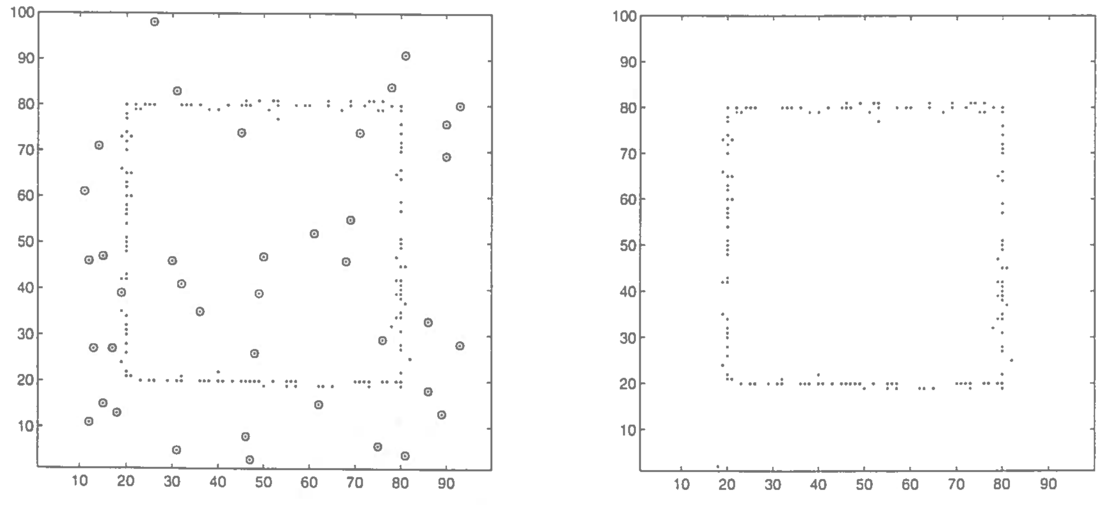

Figure 4.4: (a) Sampled data (outliers circled) (b) outliers removed

Smoothing and Thinning

After the outliers have been removed, the next stage is to reduce the noise and thin the data. The morphological thinning algorithm described above could be used, but to produce a fully thinned skeleton from a data set takes several iterations ning algorithm to three dimensions would involve the addition of even more iterations with new patterns, making the speed situation worse.

A standard and effective method for 2D edge detection is the Canny edge detector [Canny 86]. This algorithm first performs a Gaussian smoothing on the intensity data of an image and then takes the two-dimensional first spatial derivative of the grey-scales to highlight areas where the derivative is large. These peaks in the derivative space correspond to rapid spatial changes in grey-scale, most likely to be edges. The peaks are then non-maximally suppressed leaving thin lines corresponding to edges in the image. The final phase of the algorithm is the tracking of continuous lines to produce edge descriptions.

As stated above, the images here are not grey-scale but binary, so the Canny detector is not obviously applicable. However, it is possible to adapt it in the way shown below to get encouraging results which led to a 3D implementation. An important aspect of the data collected in the 3D system is that it is essentially two dimensional, in that it represents the surface on an object, not its volume. Similarly, the 2D simulated data is essentially one dimensional, so any given local region of the data should also be one dimensional. Figure 4.5 shows data (•) and a cluster of noise points (⋄) not filtered out by the outlier removal algorithm.

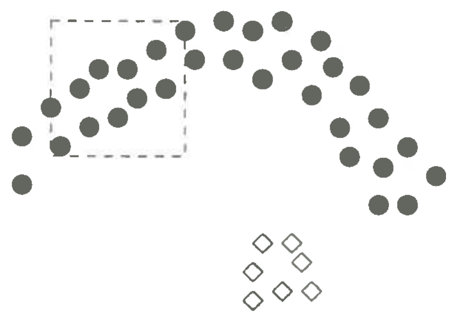

Figure 4.5: Line data and noise

In local regions (such as that marked by the square) the data points display direction; it is two dimensional. If the dimensionality of the data surrounding each image pixel can be quantified and a grey-scale image built of these numbers, then a Canny-like edge detector can be used to extract strongly one-dimensional regions and to reject regions which are essentially two-dimensional and therefore likely to be noise.

Fortunately, the well developed method of principal components analysis (PCA, explained in [Ross 96]) exists for assessing the intrinsic dimensionality of data sets. PCA makes use of the N×N covariance matrix (where N is the dimension of the space the data occupies) describing the data presented to it. The eigenvectors of this matrix describe the principal directions of the data, and the relative sizes of the eigenvalues indicate the dimensionality of the data. In the 2D case, if one eigenvalue is much greater than the other then the data is strongly one dimensional; if the two eigenvalues are approximately equal, then the data is two dimensional. The eigenvector corresponding to the largest eigenvalue give the dominant direction of the data, the one which corresponds to the smallest eigenvalue the direction of least data spread, which is in fact the normal to the least-squares best-fit plane (in N-D space) fitted to the data (see [Faugeras 93], Section 10.7). This relationship with plane fitting means that whenever a data set is described by the results of PCA, then the data set has been implicitly modelled as a plane.

The 2D Algorithm

The edge detector developed uses both the eigenvalues and eigenvectors produced when performing PCA on local neighbourhoods in the data. If the local eigensystem has eigenvalues λ_1, λ_2 ($\lambda_1 > \lambda_2$) and corresponding eigenvectors $\mathbf{v_1}, \mathbf{v_2}$, then the pixel at the centre of each neighbourhood (i.e. the centre of the local eigensystem) is assigned two items of data; $\frac{\lambda_1}{\lambda_2}$, which is a measure of the dimensionality of the neighbourhood, and is treated as the 'intensity' value for the detector, and $\mathbf{v_2}$, the local normal vector, which is used in the non-maximal suppression phase.

a discussion of this).

After the PCA is performed, pixels with eigenvalue ratios below a certain threshold are removed from the image, as these represent areas where the data is not one dimensional and could be noise. Once this filtering has been performed, all remaining "intensity" values are smoothed by convolving with a 2D Gaussian kernel, which links neighbouring (non-zero) pixels, creating ridges in the image. The normals of the pre-smoothing image are now propagated into the 'new' pixels created by the convolution, so each set pixel in the image now has an intensity value and a normal vector.

At this stage non-maximal suppression of the image is performed, as in the Canny filter but with the pixel normals providing the suppression directions rather than an image gradient; the result of this operation is a thin ridge in intensity space. This thinned intensity edge is now treated as a binary image, and a single morphological closure is applied (one dilation followed by an erosion) which acts to fill in any gaps in the reconstructed edge (i.e. the ridge). The algorithm is summarised below:

```
// initialise PCA output image
PCAImage=zero

// calculate PCA
for every pixel in image (zero or non-zero)
  if neighbourhood non-empty
  calculate local eigensystem, centred on
      centre of local non-zeros (local_i,local_j)
  put eigenvalue ratio and normal vector into
      PCAImage(local_i,local_j)

Output PCAImage
```

```
convolve PCAImage with 2D Gaussian kernel

// non-maximal suppression on smoothed output
for each pixel (i,j) which is non-zero
  if (value (i,j) < value(neighbours in normal directions))
    set pixel(i,j) to zero

// morphological closure performed on non-max suppressed
// image

// dilate
for each pixel(i,j) which is non-zero
  set to non-zero any of its 8 neighbours currently zero

//erode
for each pixel(i,j) with any of its 8 neighbours zero
  set pixel(i,j) to zero
```

Results on Simulated Data

The diagrams presented here (Figures 4.6 and 4.7) show four major stages of the algorithm. Figure 4.6 shows the initial outline and sampled version of the outline; the regular sample points were placed on each pixel, with noise added to move the sample result to a random, nearby position within the sample bound shown in Figure 4.1. This noise on the sample output position was normally distributed with a mean of zero pixels and standard deviation of half a pixel. Finally, 0.5% 'salt and pepper' noise was added to model the completely spurious outlier points.

In Figure 4.7, the final two steps are shown, in (a) the output of the non-maximal suppression and (b) the algorithm output image after morphological

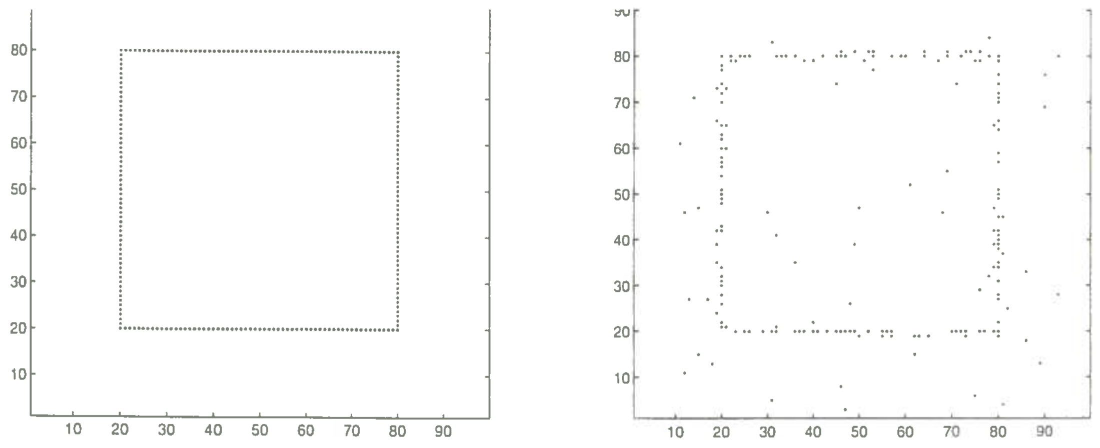
Figure 4.6: (a) Original outline (b) sample

closure. The threshold for the $\frac{\lambda_1}{\lambda_2}$ ratio used in the rejection of 2D regions was 20, and the neighbourhood in which the PCA is calculated is 11×11. Experimentation showed that changing the threshold for the ratio has little effect, and study of the eigenvalues produced showed why; there were very many ratios of $\frac{1}{0}$ (set to a value of 50 for implementation purposes), where the PCA classified the data as intrinsically one dimensional, and these values dominated the smoothing and hence the reconstruction. Figure 4.8 shows a final image where the threshold was set to zero; there is very little difference between this and that shown in Figure 4.7 (b), just a few pixels at the corners.

The PCA neighbourhood is more critical, and lowering it too far results in an over-sensitivity to local noise which ultimately causes a more fragmented result (Figure 4.9). The reason for this is that the eigensystem is over-influenced by local noise, causing spurious normal directions to be calculated which lead to breaks in the ridge during the non-maximal suppression.

A qualitative observation that should be made about the results is that the reconstructed image is neither perfectly smooth nor perfectly thin; however, the outliers have been removed and there are few breaks in the data, which are the

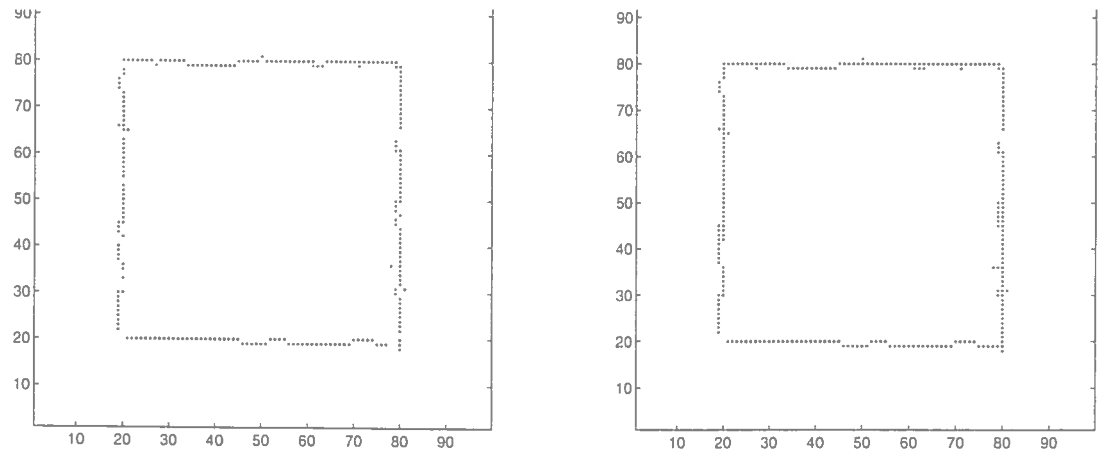
Figure 4.7: (a) After non-maximal suppression (b) after closure

4.2 Into The Third Dimension

4.2.1 Adapting the 2D Methods

The data structure used in the three dimensional implementation is a "Voxmap", i.e. a quantised cuberille in 3D space (see Appendix C for details). The edge length of each cube (or 'voxel') is determined on the basis of expected sensor noise and data density (*c.f.* Section 2.3.2), and as such was set to be $6mm$, this being c. ± 3 standard deviations of the sensor noise. Data points from the striper are placed into the appropriate voxels, and the resulting voxmap treated as a 3D binary image; the operations described for 3D below were then carried out at the voxel level, i.e. PCA is performed on the indices of occupied voxels rather than the data itself. The final algorithm output is the mean value of the data in each remaining voxel in the thinned image.

The effect of performing the algorithm at the voxel level rather than directly on the data is to exacerbate the quantisation errors introduced by using a voxmap in the first place. On the other hand, the use of voxels whose edge length is equal to the random noise offers a crude smoothing of local noise which may reduce the

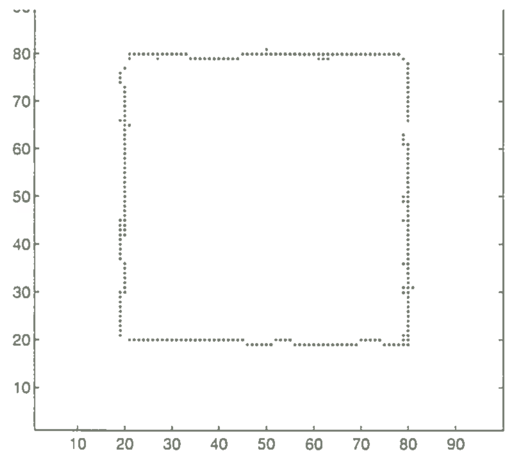

Figure 4.8: With a zero $\frac{\lambda_1}{\lambda_2}$ threshold

points, so operating on the voxels will produce faster results, which is another aim of this work.

In theory, the algorithms for outlier removal and thinning described above extend trivially into three dimensions. The structuring element used by the morphological operator simply becomes a *Nbhd* × *Nbhd* × *Nbhd* (*Nbhd* being the neighbourhood size) cube in which the central voxel is one and all of the others zero. The PCA process will now produce a 3×3 covariance matrix, and hence three eigenvalues $\lambda_1, \lambda_2, \lambda_3$ (with $\lambda_1 > \lambda_2 > \lambda_3$) and three eigenvectors $(\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3})$. The criteria for determining dimensionality has changed slightly due to the extra eigenvalue, and is summarised in the Table 4.1.

Eigenvalues	Intrinsic dimensionality
$\lambda_1 \gg \lambda_2, \lambda_1 \gg \lambda_3$	1
$\lambda_1 \approx \lambda_2, \lambda_2 > \lambda_3$	2
$\lambda_1 \approx \lambda_2 \approx \lambda_3$	3

Table 4.1: Eigenvalue sizes and intrinsic dimensionality

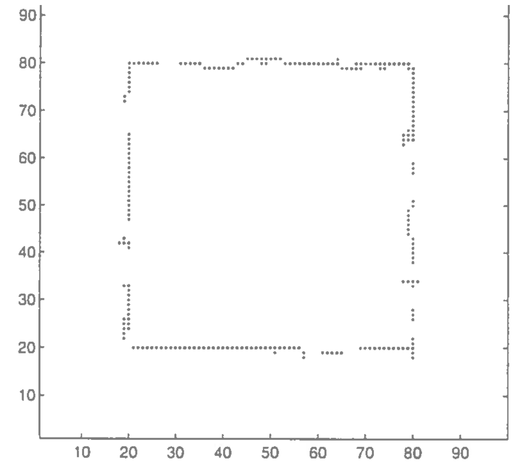
Figure 4.9: Using a 5×5 PCA neighbourhood

Problems with the Algorithm

Although the theory holds well, in practice the use of the eigenvalues to determine the intrinsic dimensionality of the data was not straightforward, and lead to the failure of the algorithm. Ideally, all regions of smooth data should be intrinsically two dimensional, with $(\lambda_1 \approx \lambda_2) > \lambda_3$; on the test data shown after outlier removal (in a $3 \times 3 \times 3$ neighbourhood) in Figure 4.10, which is taken from a horizontal (i.e. z constant) plane, observation of the eigenvalues gave no clear indication of how the relative sizes of the eigenvalues should be judged in order to produce the intensity image. Many of the points from the centre of the plane had $\lambda_1 \gg \lambda_2, \lambda_1 \gg \lambda_3$, suggesting that the data was locally one dimensional. The eigenvectors, by contrast, were well behaved and those associated with the planar 'cloud' largely pointed in (or close to) the $(0, 0, 1)$ direction. The nature of the outliers as seen in Figure 4.10 underlines the importance of the dimensionality estimator in this algorithm – the small cluster of points above the plane must be identified as noise, otherwise they will be passed on to the next stage, where they will cause deformation of the reconstructed surface.

This ambiguity of the intrinsic dimensionality test makes the algorithm prac-

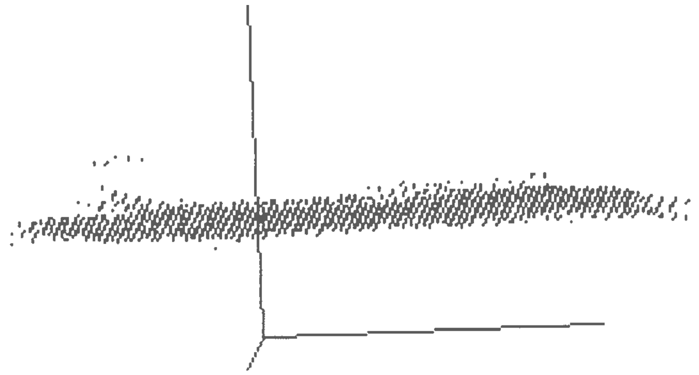
Figure 4.10: Planar data

tically useless; with no way of implementing this algorithm for the real 3D data, another must be found, and such an approach is described below.

4.2.2 A Second Approach

This new approach improves the outlier removal routine and also provides an effective replacement for the PCA-based smoothing algorithm which proved ineffective on real data.

Improving Outlier Removal

As seen above, the outlier removal routine does not filter out all such points; those left remaining are small clusters of noise points, which the simple structuring element devised in 2D does not match. Two steps have been added to improve the performance of the outlier routine – a second hit-or-miss operation with a more complex structuring element, and the introduction of removing outliers at several different scales.

The new hit or miss operator is designed to remove small clusters of noise turn are ringed by zeroes. Any set pixel which matches the central pixel or one of the 'don't cares' will be removed. In 3D, there is a central voxel, 26-surrounded by 'don't cares' which are enveloped by zeroes.

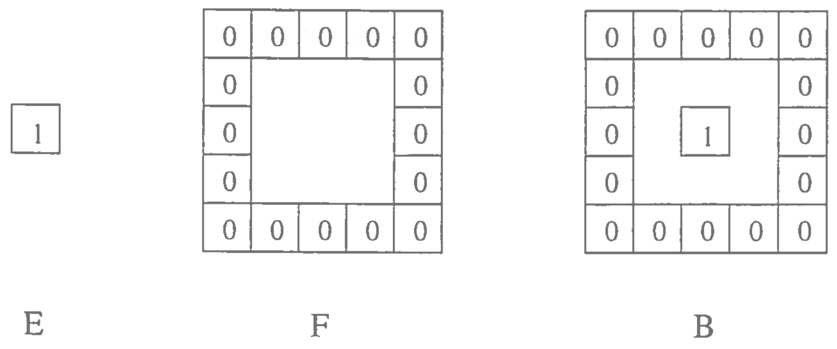

Figure 4.11: Structuring element pair for outlier removal

The second strategy is to perform multi-scale outlier removal, i.e. putting the data into a coarse voxmap, removing the outliers and then making the voxmap finer and repeating. Finally, the operating resolution (here $6mm$) is reached and the rest of the algorithm continues. In theory, outlier removal puts a lower limit on the size of object that may be scanned in, in that if an object fits fully inside a voxel, then it will be erased by the outlier filter. In practice the coarsest voxmap used was one with a voxel side of $20mm$, so theoretically no object with dimensions smaller than this may be scanned in (if such an object lies centred in a voxel – otherwise quantisation will spread points from the object into two or more voxels). However, it should be noted that such an object is not floating in space – the surface that it lies on will also be partially scanned, so the small object is unlikely to be the only feature in the scene.

Figure 4.12 shows the effects of using the original outlier removal filter with the 'cluster' filter at 3 scales of voxmap ($20mm, 10mm, 6mm$) on the same data seen in Figure 4.10. It is easy to see that the second multi-stage filter is more effective, and to quantify this, the new routine leaves 60 fewer voxels in the

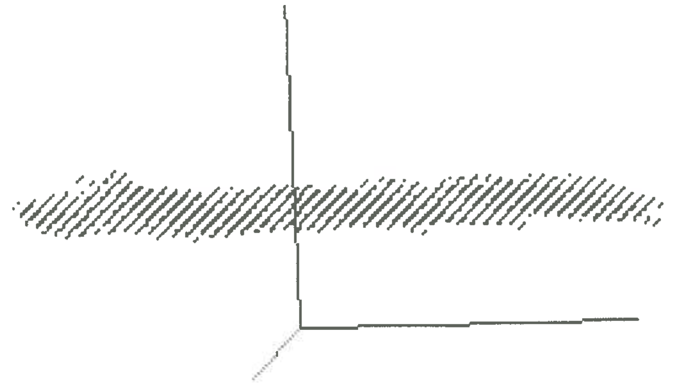

Figure 4.12: Multi-scale and cluster outlier removal

containing data prior to outlier removal was 2, 521.

A Maximum Likelihood Approach to Smoothing

During the data acquisition stage, parts of the surface of the object are often traversed several times, either to specifically gather more data from that area, or en route to another area. The result of this is that some voxels will have many data points inside them, and this information can and should be exploited – *intuitively*, the more points that lie in any voxel, the more likely it is that the surface passes through that voxel. The maximum likelihood estimator ([Press *et al.* 92], Section 15.1) allows this intuition to be quantified, and this can be put to use in smoothing the range data.

To introduce the concept of maximum-likelihood estimation, consider a one dimensional measurement, such as the weight of an apple. If an experiment is conducted where the apple (actual unknown weight W) is weighed many times and the measurements m_i noted, a distribution can be produced (Figure 4.13), which will have its mean at W, and a certain standard deviation σ. This distrivery low probabilities $P(m_i|W)$, then the conclusion drawn would be that the model W is not very *likely* to be correct, and if $P(m_i|W)$ is quite high, then it would be said that the model is *likely* to be right.

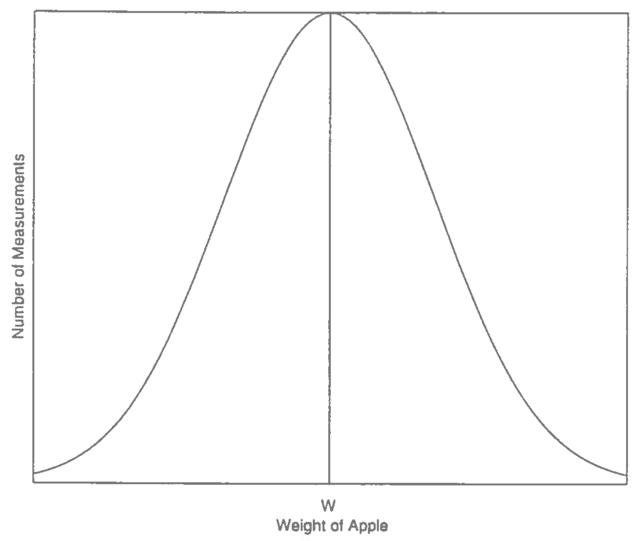

Figure 4.13: Normally distributed measurements of weight

This intuitive link between the probability of a measurement given a model, and the model's correctness, can now be used to estimate the model's parameters, hence the name maximum likelihood estimation.

For the set of N measurements m normally distributed about W, the probability of such a set occurring is the combined probability of each measurement occurring, i.e.

$$P \propto \prod_{i=1}^{N} \left\{ exp \left[-\frac{(m_i - W)^2}{2\sigma^2} \right] \right\}$$

The *most likely* model W will maximise this product, or indeed, the logarithm of the product:

$$\sum_{i=1}^{N} \left[-\frac{(m_i - W)^2}{2\sigma^2} \right]$$

So, the most likely model can be found by maximising the sum of a set of quadin each voxel. Each voxel now holds the sum of a set of Gaussians, the approximation to a sum of quadratics. By performing non-maximal suppression on this data set, the voxels with largest smoothed values (i.e. Gaussian sums) are preserved – these are the regions of space which maximise the sum, and hence have the maximum likelihood of lying on the surface. In effect, each voxel is being proposed as a model (with its own distribution centred around it after smoothing), and the most likely models are selected by the non-maximal suppression.

How well does a Gaussian approximate a quadratic? Figure 4.14 shows a Gaussian (mean = 0, σ=1) (solid), the best-fit quadratic to this (dash-dotted), and the difference between the two (dashed), all centred on voxel '0'. The $*$s mark the values of the error at each of voxel – this is where the convolution kernel which is used to calculate the smoothing samples the Gaussian. Only at the edges of the curve (i.e. at ± 2 voxels from the centre) does the error become significant.

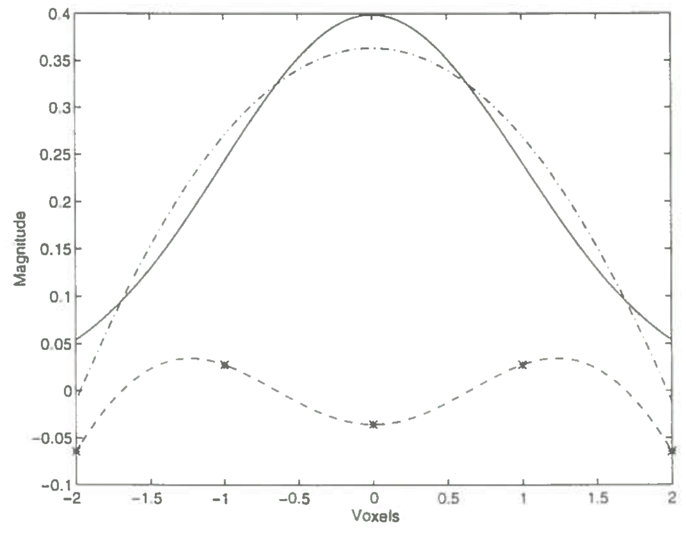

Figure 4.14: Comparison of Gaussian (solid) and quadratic (dash-dot). Error is shown dashed.

This error is tolerable, however, due to the fast calculation time of a 3D to three equivalent 5×1 kernels, which can be convolved with the voxmap much faster than the larger 3D kernel. A quadratic can not be decomposed in such a way, and would be much slower to use.

After non-maximal suppression, the mean value of the data point from each remaining occupied voxel is output to file, this file being the set of data points for entry to the Hoppe surface reconstruction algorithm.

So, the smoothing algorithm simply becomes:

```
// smooth voxmap
convolve voxmap with 1D Gaussian in z direction
convolve result with 1D Gaussian in y direction
convolve result with 1D Gaussian in x direction


calculate surface gradients (see text below)


do non-maximal suppression


output mean data point from each voxel
```

Initially the principal components analysis routine was used to calculate the plane normals along which the suppression was performed. However, there are problems with this in that it is a slow process, and also very sensitive to noisy distribution of the data, which results in the mis-alignment of some normals and therefore erroneous suppression of associated points. A much faster and more reliable method was to use the intensity (I) gradients of the smoothed image. The simple kernel used in the implementation (Figure 4.15) calculates the backward difference gradient of the intensity data and was applied in the x, y and z directions, the resulting gradients along those axes ($\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial z}$) being put into a vector $[\frac{\partial I}{\partial x} \; \frac{\partial I}{\partial y} \; \frac{\partial I}{\partial z}]$, which is then scaled to unit length to give the local data normal. This is a standard peak detection method for 2D grey-scale images and

24.04 seconds to perform the normal calculation; the gradient method took only 0.414 seconds – nearly 60 times faster.

-1	+1

Figure 4.15: The simple gradient calculating kernel

4.3 Results of the Algorithm

Presented here are the results of the algorithm applied to data captured by the scanner. The tests performed were on basic surfaces – planes and simple curves, but it is difficult to judge the performance on more complex shapes prior to the full mesh construction (seen in Chapter 5).

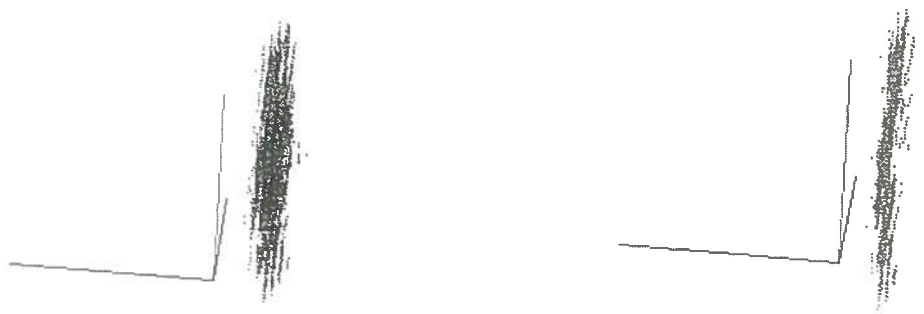

Figure 4.16: (a) Raw data from a plane at $x = 40cm$ (b) thinned data

Figures 4.16,4.17 and 4.18 show raw data from planes at $x = 40cm$, $y = 15cm$ and $z = -23cm$ respectively. In Figures 4.17 and 4.18, the algorithm has performed quite well, producing this planes with no outliers. The situation is slightly different in Figure 4.16, where the data has not been thinned as much.

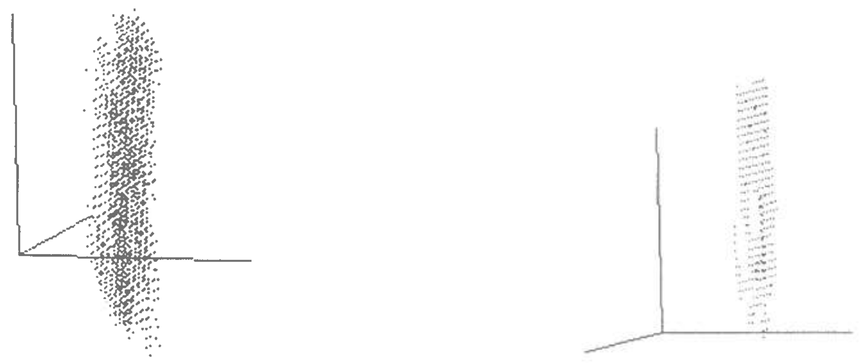

Figure 4.17: (a) Raw data from a plane at $y = 15cm$ (b) thinned data

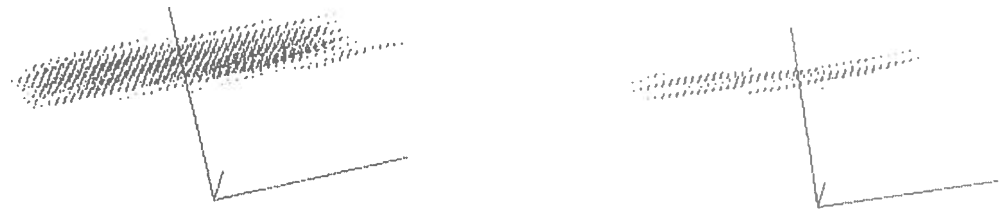

Figure 4.18: (a) Raw data from a plane a $z = -23cm$ (b) thinned data

data is generally more dense. With data like this, the quantisation of the voxmap is likely to tell; it is possible that the peak of the ridge (representing the most likely surface) lies on the boundary of most voxels, thus the voxels each side of this will have comparable number of data points in them, leading to a thicker data after non-maximal suppression than in other cases. The reason that the data in Figure 4.16(a) is less spread than data from the other planes could be related to the fact that the sensor remained at a near constant radius from the Flock of Birds z axis throughout data collection, so the effects of the systematic error noted in Chapter 3 were minimised. For the other two planes, the sensor

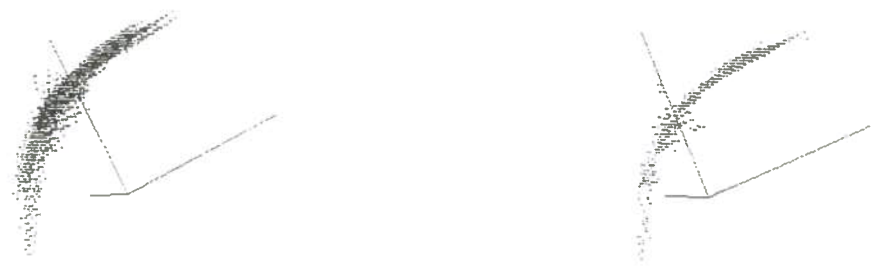
Figure 4.19: (a) Raw data from curved surface (b) after thinning

Figure 4.19 shows range data captured from a curved surface (a wastebasket from the lab). The surface is thinned considerably, but there are still some outlier clusters left, which will lead to irregularities in the results of the surface reconstruction algorithm.

Although the algorithm performs thinning well, the outlier removal is far from perfect, and this causes the Hoppe algorithm problems, as will be seen in Chapter 5. Outliers may be removed by hand, using a program such as `xgobi`, which was the strategy used to improve the output of the original smoothing algorithm (as seen in Chapter 2).

Chapter 5

Surface Reconstruction from the Cleaned Data

In Chapter 4, an algorithm for removing outliers and smoothing the range data from the hand-held striper was presented. The aim of cleaning the data was to prepare it for use by the Hoppe algorithm (see Chapter 2 and [Hoppe *et al.* 92]), the results of which are presented here. The reconstructed surfaces for the range data shown in Chapter 4 (Section 4.3) are shown as well as the geometry of a toy cow, which is used to display the utility of the algorithm and to compare its results to those from the algorithm described in Chapter 2 (Section 2.3.2).

The results are judged qualitatively – with the aim of constructing realistic models of objects, the best test is whether the results look 'good' rather than conform to some mathematical metric, even if a suitable metric could be dreamed up.

5.1 Simple Surfaces

The output of the data cleaning algorithm and the corresponding reconstructed surface for the planes at $x = 40cm$, $y = 15cm$ and $z = -23cm$ are shown in Figures 5.1, 5.2 and 5.3 respectively, where the planes are at different angles to

Figure 5.1: (a) Thinned data from a plane at $x = 40cm$ (b) reconstructed surface

In Figure 5.1, where the cleaned data was still quite 'thick', the resulting reconstructed surface is quite bumpy, although there are no large protuberances present. Considering the spread of the data presented to the Hoppe algorithm, this is not surprising – in attempting to fit a surface to the data, the algorithm will follow the centre of a 'cloud' of points presented to it.

Figure 5.2 displays one of the possible effects of a gap in the input data to the Hoppe algorithm. The reconstructed surface is quite flat, except for the presence of a piece of mesh sticking out into space. This area corresponds to the hole that is present near the edge of the thinned y plane data in (a), but the mesh in (b) cannot be displayed in the same orientation due to the way the viewing software treats the mesh surface normals. This irregularity highlights the necessity to collect dense data for the algorithms to function correctly.

The reconstructed z plane is shown in Figure 5.3, where the rise seen on the left-hand side of the data in (a) can be seen as a bump on the closest right-hand edge of the mesh in (b) – this shows how closely Hoppe's algorithm follows the data. The reconstructed surface is also very smooth, as should be expected from such thin input data with no outliers to perturb it.

To illustrate the thinning algorithm on curved surfaces, Figure 5.4 shows the

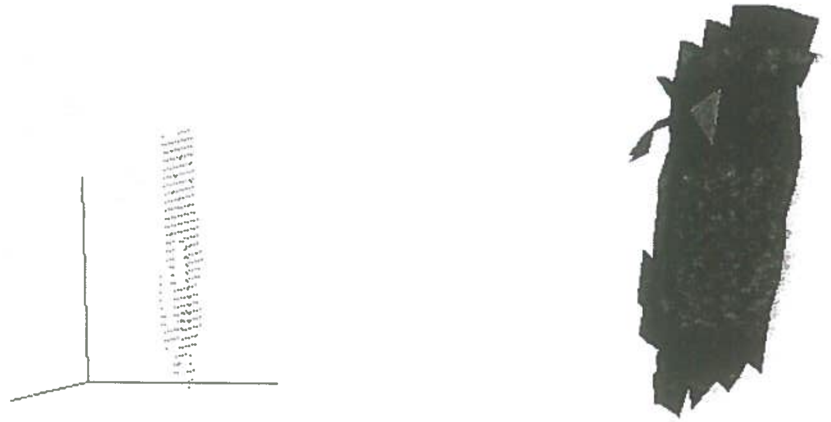

Figure 5.2: (a) Thinned data from a plane at $y = 15cm$ (b) reconstructed surface

data scanned from the side of the wastebasket and the surface constructed from this data. The performance is similar to that described for the flat surfaces above – the surface is fairly smooth, with a small dent caused by the group of outliers which can be seen in (a) just below the surface near the vertical axis.

Having demonstrated the suitability of the thinning algorithm for preprocessing, and the expected results from the Hoppe algorithm on some simple cases, a more complex example was used.

5.2 Reconstructing a "Real" Object

The real test object in question is a small toy cow, pictured in Figure 5.5. This toy was one of the test objects in the original system, so comparison can be made with the results of the previous algorithms.

The raw range data and thinned data are shown in Figure 5.6 (a) and (b) respectively. The cleaning algorithm has not worked particularly well here – there are still many outlier points and also some ridges protruding from the main surface; these are most likely caused by the smoothing of the "intensity" values

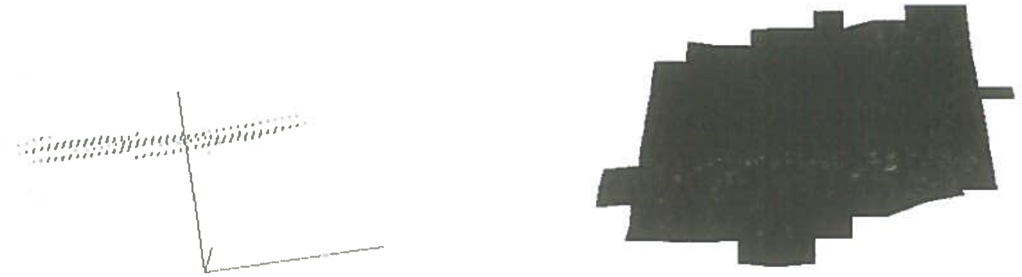

Figure 5.3: (a) Thinned data from a plane at $z = -23cm$ (b) reconstructed surface

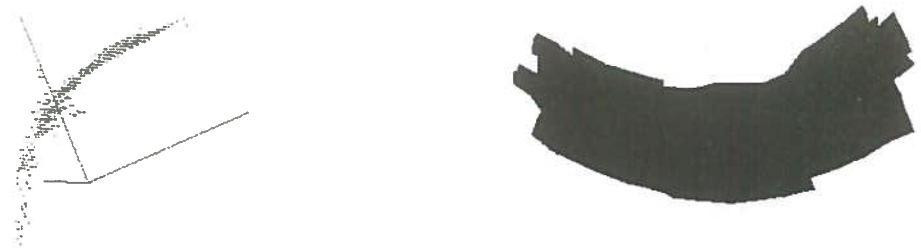

Figure 5.4: (a) Thinned data from the curved bin (b) reconstructed surface

the surface. As with the more primitive surfaces, the non-maximal suppression has performed well thinning the data.

The results of running the Hoppe algorithm on the thinned range data are given in Figure 5.7 (b); for the purposes of comparison, Figure 5.7 (a) shows the algorithm run on the raw range data. Unfortunately, the colour pasting software was not working properly at the time of writing, and the models are shown without texture mapping; as a guide to the reader, the cow's nose is facing to the left of the page. Although the reconstruction in (b) looks quite poor, few outliers are required to produce this 'messy' response from the Hoppe algorithm, which

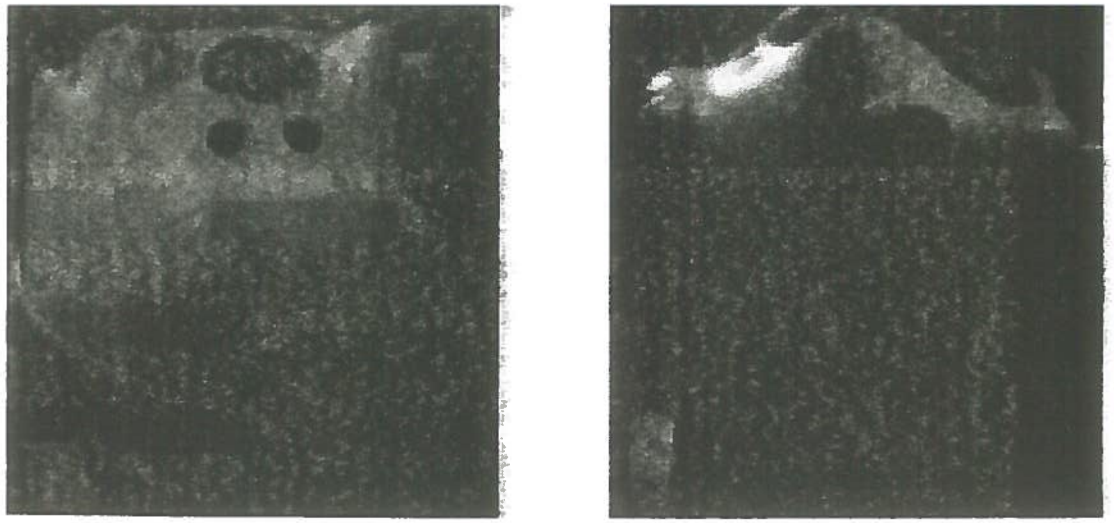

Figure 5.5: Two views of the toy cow

was designed to work on data with only Gaussian distributed noise. However, the mesh in (b) is on the whole much smoother than that in (a), so that thinning and limited outlier removal do improve the situation, despite not leading to a perfect model. The meshes of Figure 5.7 highlight the failings of the algorithm to remove the outlier points, and the unwanted effects of sometimes linking these outliers with the surface itself at the morphological closure stage.

To provide a mark of performance, the model resulting from the thinning algorithm presented in this report is compared with that used previously (Section 2.3.2). In the original system, some hand-editing of the data was performed to remove outlier points which caused similar reconstructions to those in Figure 5.7 (b). For this reason, the data from the new thinning algorithm was also hand-edited, the two resulting meshes being shown in Figure 5.8.

The surface produced by the old algorithm (Figure 5.8 (a)) has been clipped, cutting off the object the toy was standing on during scanning; this has not been done on the surface from the newer algorithm (Figure 5.8). In both images, the cow is again facing left. Both surfaces are quite smooth, and both have lost some

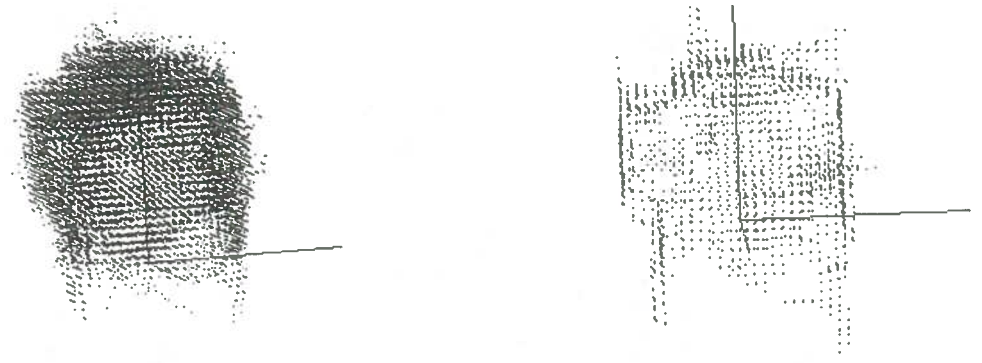
Figure 5.6: (a) The raw cow range data (b) range data after thinning

both horns can be seen in (a), and only one in (b).

5.3 Conclusions

The data cleaning algorithm is far from perfect, although it has a very similar noise reduction capability to the algorithm used in the original prototype system – for best results, both need a final hand-editing stage to remove the last outliers. A major advantage that the new algorithm has over the old is that it is much faster; on a Sun Sparc 10/51, the original algorithm took approximately one hour plus hand-editing time, on the same machine the new algorithm runs in just over 56 seconds before hand-editing, so a speed increase of a factor of 60.

On the simple surfaces, the new algorithm performed quite well, although this is due to cleaner data being presented to it in the first place than in the case of the cow model. It can be concluded that in the presence of only a little noise (certainly few outliers), the algorithm works well, but, like its predecessor, human intervention is still necessary before its output can be successfully reconstructed by the Hoppe algorithm. It is obvious that although the speed increase is welcome, a more useful result would be improved performance in outlier re-

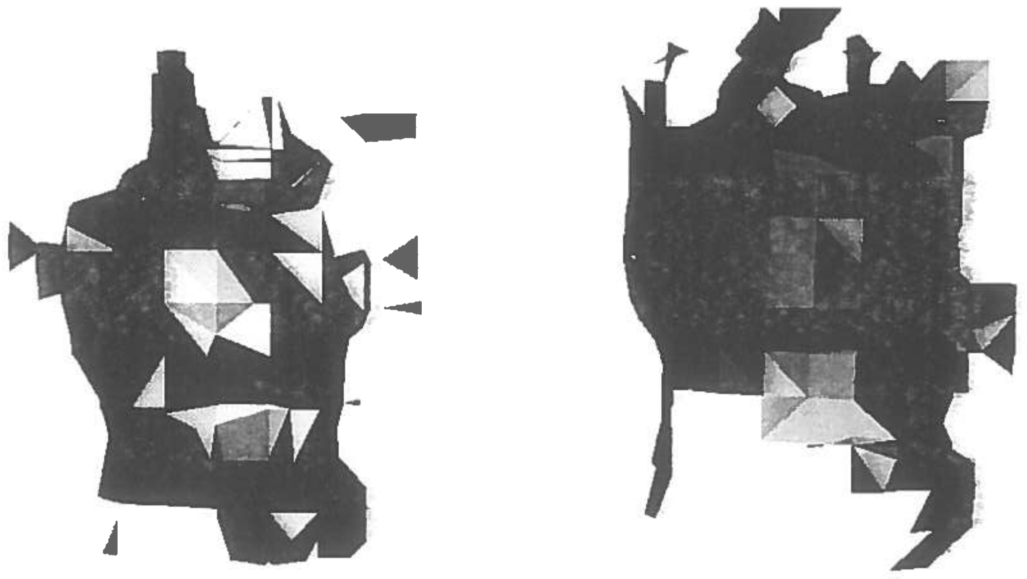
Figure 5.7: (a) The surface from the raw range data (b) surface from the thinned data

voxmap data structure (Appendix C) providing a suitable flexible framework for the development of fast algorithms.

Figure 5.8: (a) The surface from the new algorithm (b) the old

Chapter 6

Conclusions and Further Work

The aim of this project was to improve the performance of a hand-held laser striper, specifically by addressing the problem of noise reduction both at data collection time and by pre-processing the range data for the Hoppe surface reconstruction algorithm. Although not entirely successful in these aims, some results of value have been produced which show a way forward for some problems and illustrate how certain approaches are not suitable for others. Conclusions are drawn on the work done and recommendations made for further work regarding the two areas this dissertation has concentrated on; the Flock of Birds electromagnetic position sensor and noise reduction strategies for cleaning the range data.

6.1 The Flock of Birds Sensor

Chapter 3 details the work carried out on the analysis of the Flock of Birds electromagnetic position sensor, where much of the noise present in the system originates; random positional errors (of $\pm 6mm$) and angular noise (of $\pm 1.3^o$) cannot be calibrated out, so averaging repeated readings is the only immediately available way to reduce this. The Flock of Birds base unit can be used to track several remote units at any one time; an obvious way to combat random errors

Another aim was to determine whether a systematic error was present in the system, and analysis of the sensor revealed that such error *does* exist, and it takes on two forms. In a central region, a cylinder of c. 15cm radius around the base unit's z axis, the readings are vastly different from those outside of this volume, and quite unpredictable (see Figure 3.7). Outside of this cylinder, a different type of error is present, and it was initially thought that this could be corrected using a quadratic model. Further investigation showed that such a simple correction formula made matters worse, causing the stripe data to spread more. Unfortunately a more complex model could not be found – the sensor manufacturers have been contacted, and it is possible that once the physics of the sensor is understood better that an analytical solution will be possible.

If an analytical solution does have to be ruled out, the other option is to build up an interpolation table; however, for this to work well, the workspace will have to be quite densely sampled, because if a correction formula can not be found for the whole workspace, interpolating between points far apart will be just as unreliable.

Study of the sensor has proved useful; it has been discovered that using the sensor close to the base unit is likely to produce unreliable results due to the high magnitude random variations in position reading in this area. Another, smoother, systematic error exists further from the base unit and although efforts to correct for this have not been successful, the problem is now known about and can be approached in the future.

6.2 Data Cleaning

The major source of deformations in the final reconstructed surfaces is the presence of outlier points in the range data, which were not removed effectively by the original algorithm (which was also very slow). Experiments performed in 2D on synthetic data suggested that the use of morphological thinning to remove the data and remove remaining clusters of outlier points. However, when put into practice in three dimensions on real range data, the method proved unreliable due to the effects of noise and data density on the eigenvalues of local neighbourhoods which were used to perform the dimensionality assessment. Due to this unreliability, the method was rejected. The 3D experiments required the development of a suitable data structure, a voxmap being chosen, which provides rapid access to the data and allows local information to be stored in each voxel.

An alternative approach, approximating a maximum likelihood estimator was then implemented which used the number of range points in each voxel to represent the likelihood of that voxel lying on the scanned object's surface. A multi-scale approach to outlier removal was also adopted, which proved more effective than single resolution thinning.

Results of using the maximum likelihood thinning algorithm showed little improvement on the clustering algorithm used in the original prototype system in terms of the quality of appearance of the final surface model, although there were vast speed increases (approximately 60-fold) using the new method. The problem of effective automatic outlier removal remains to be solved, and if the voxmap data structure (one of the reasons for the speed increase) is to be used, methods which go to the 'sub-voxel' level are going to be needed. In the PCA based algorithm, the local eigensystems were calculated in terms of the centres of occupied voxels – on reflection this quantisation will lead to inaccuracies in the directions of the eigenvectors and sizes of the eigenvalues, and could have been the cause of the algorithm's failure; calculating the eigensystems on the basis of points inside each voxel may prove more reliable, although this has its problems. Strictly speaking, the eigenvalues give the relative sizes of the *spread* of the data in each of the eigenvector directions; only if the data is isotropically sampled, i.e. there is a uniform sample density in every direction, will the relative sizes of the eigenvalues give a true reflection of the dimensionality of the data.

There is one potentially valuable source of information which has yet to be each point can be tagged with a stripe number, the structure could be used to good effect. For example, if some stripe points are identified as outliers, then all data from that stripe could be rejected, on the assumption that the stripe has been spuriously positioned due to some random fluctuation in the global position reading. Stripe connectivity could be useful for estimating the local data density too, possibly allowing the scaling of eigenvalues from PCA to give more reliable intrinsic dimensionality estimation.

It should also be noted that the use of PCA will lead to a significantly slower algorithm due to the computational load involved.

6.3 Summary

6.3.1 Results

Global position sensor: Much more is now known about the performance of the Flock of Birds global position sensor; two types of systematic error have been identified and the random errors are better quantified. A model for correcting the systematic errors was proposed but found lacking.

Outlier removal: A multi-scale morphological outlier removal algorithm has been implemented, which works well on removing distant outliers, but is less effective at deleting points near to the surface. Outliers are still the greatest problem in the system, their presence largely due to the noisy position and orientation measurements from the Flock of Birds Sensor.

Data thinning: Maximum likelihood estimation has been used as a basis for extracting the most densely occupied voxels and a thin surface connecting them. The algorithm works well on data with few outliers, but will link nearby outlier points in with the main surface.

Speed increase: The pre-processing of the range data for the Hoppe algorithm

6.3.2 Recommendations for Further Work

Systematic global position errors: The performance of the Flock of Birds sensor is now better understood, and there are two possible paths to follow; either building a look-up table to finely calibrate the workspace and correct the systematic errors, or to investigate the physics of the sensor further to build a more complex model.

Random global position errors: The averaging of several consecutive sensor readings could be made to reduce the effects of random sensor errors. A second remote unit could be added to the hand-held striper to provide two measurements instantaneously and thus lower the position uncertainty.

Outlier removal: This can be achieved by reducing the noise on the initial data measurements from the Flock of Birds sensor or by further data processing. The use of PCA could again be investigated, but on a sub-voxel level rather than at the coarser resolution used in Chapter 4. The stripe-connectivity could be used to counter some of the problems of estimating the intrinsic dimensionality of anisotropically sampled data.

Data Thinning: The current algorithm works well without the presence of outliers, but if an outlier removal algorithm also constructs a surface, then such an algorithm would be preferable.

Real-time display of data: A problem not addressed in this dissertation is that of improving the real-time display of the system as data is collected from a surface. Having used the system, it is difficult to judge whether enough data has been gathered from an object – the use of simple techniques such as 'z-buffering' the display would be useful, and some real-time indication of local data density even more so.

Bibliography

[Asc95] Ascension Technology Corporation, POB 527, Burlington, Vermont 05402 (802) 860-6440. *The Flock of Birds Position and Orientation Measurement System Installation and Operation Guide*, January 1995.

[Canny 86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), November 1986.

[Dougherty 92] Edward R. Dougherty. *An Introduction to Morphological Image Processing*. SPIE Optical Engineering Press, 1992.

[Eggert *et al.* 96] David W. Eggert, Andrew W. Fitzgibbon, and Robert B. Fisher. Simultaneous registration of multiple range views for use in reverse engineering. In *Proc. Int. Conf. on Pat. Recog.*, August 1996.

[Faugeras 93] O. Faugeras. *Three-Dimensional Computer Vision, A Geometric Viewpoint*. MIT Press, Cambridge MA., 1993.

[Faugeras *et al.* 83] O.D. Faugeras, F. Germain, G. Kryze, J.D. Boissonant, M. Herbert, J.Ponce, E. Pauchon, and N. Ayache. Towards a flexible vision system. In A. Pugh, editor, *Robot Vision*, pages 129–142. Springer Verlag, 1983.

ance recovery under point light illumination. In *BMVC '96*, September 1996.

[Fisher 96] R. B. Fisher. M.Sc. machine vision notes. Internal teaching documentation, Dept AI, University of Edinburgh, 1996.

[Fisher *et al.* 96] R.B. Fisher, A. Fitzgibbon, A. Gionis, M. Wright, and D. Eggert. A hand-held optical surface scanner for environmental modelling and virtual reality. In *Proc. Virtual Reality World 96*, February 1996.

[Fitgibbon 96] A.W. Fitgibbon. Geometry calibration of hand-held laser striper. Private Communication, 1996.

[Gionis 96] A.P. Gionis. Camera calibration procedure. Private Communication, 1996.

[Hoppe 94] H. Hoppe. *Surface Reconstruction from Unorganized Points*. Unpublished PhD thesis, Dept. of Computer Science and Engineering, University of Washington, 1994.

[Hoppe *et al.* 92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Steutzle. Surface reconstruction from unorganized points. In *SIGGRAPH 92*, pages 71–78, 1992.

[Lorenson & Cline 87] William E. Lorenson and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[Naidu & Fisher 90] D.K. Naidu and R.B. Fisher. Range sensor hardware description. Imagine project software paper no. 23, Dept Artificial Intelligence, University of Edinburgh, November terling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2^{nd} edition, 1992.

[Ross 96] Peter Ross. M.Sc. connectionist computing notes. Internal teaching documentation, Dept AI, University of Edinburgh, 1996.

[Shirai & Suwa 71] Y Shirai and M Suwa. Recognition of polyhedra with a range finder. In *Proc 2nd Int. Joint Conf. on AI*, pages 80–87, August 1971.

[Wyvill *et al.* 86] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structures for *soft* objects. *The Visual Computer*, 2:227–234, 1986.

Appendix A

Calibration Procedures

There are two calibration procedures; the camera calibration which provides the world ray to image pixel correspondence and the hand-held unit geometry calibration which fixes the transformation between the Flock of Birds sensor remote unit and the camera/laser plane orientations, as well as aligning the base unit reference frame with the global co-ordinate system. The procedures described below were not developed by the author.

A.1 Camera Calibration

The basis of this procedure [Gionis 96] is to capture two images which contain reference points at known global co-ordinates. For each image, a correspondence is found between the image and world co-ordinates, in the form of a set of vectors $[i\ j\ x\ y\ z]$, where $[i\ j]$ is the image point and $[x\ y\ z]$ the world point, i.e. a position along the ray passing through pixel $[i\ j]$. With a second set of correspondences, a second point on the ray through each $[i\ j]$ can be found, and therefore the ray itself.

The correspondences will only be known for a few of the pixels in the image, and the `newstripe` program reads in the $[i\ j\ x\ y\ z]$ vectors and fits a 4^{th} order interpolation function to them, so the whole image plane is now calibrated. This calibration pattern (Figure A.1) is used, with a typical pair of calibration images (one taken with the pattern horizontal, the other vertical) shown in Figure A.2. The distance between disc centres in the pattern is 2.0cm, with the grey disc used as the reference point for finding the global positions of the other discs, measured from a set of axes marked on the base of the hand-held unit.

Each image is captured using the command

```
% rframe-dasm | xv
```

saving the results as a `.pgm` file. This file is then processed in the UNIX shell using a set of HIPS functions as follows:

```
% pgmtohips filename | thresh -t 120 | accrete | erode |
   | addframe 1 | cclabel | xv
```

The threshold value of 120 may need to be altered to produce the best results (to view, pipe the output of `thresh` to `xv`). The images produced by this process are the disc images, with each disc having a different (constant) grey level – these are saved in the grey-scale HIPS format.

The `?separate` (`?` is `v` or `h` depending on whether the image was the vertical or horizontal plane) program is run on the processed images, which produces a list of the image co-ordinates (in pixels) of the disc centres (these files are `hor.centers` and `vert.centers`). Each image in turn is loaded into MATLAB along with the appropriate list of disc centres, and a set of scripts are used to produce a list of $[x\ y\ z]$ values for each disc centre.

The scripts to run are called `?setup.m`, where `?` is `v` or `h`, depending on whether the vertical or horizontal image is being used. This loads in the image, and then allows the user to label the discs on the edge of the image by clicking with the left mouse button (clicking the middle button on a point labels that point and stops the labelling process), the labelling making it easier for the user to keep track of which image discs correspond to which world discs. An example

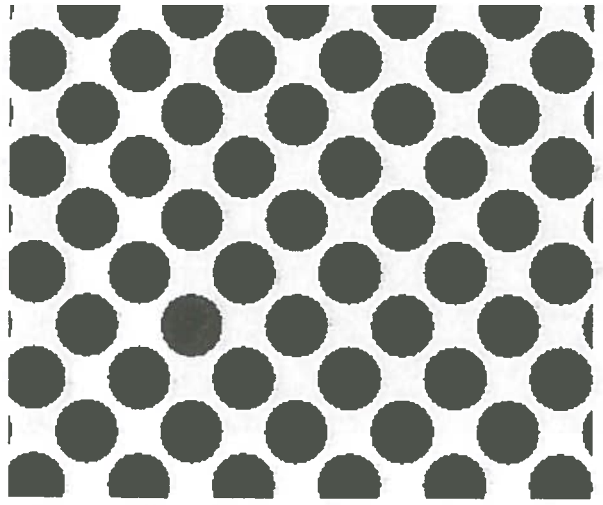
Figure A.1: The camera calibration pattern

The next stage involves entering the $[x\ z]$ co-ordinates of the first labelled point (in the vertical case $[x\ y]$), and then clicking the left mouse button on each disc in that column; again to finish a column, make the last click with the middle button. Figure A.4 illustrates this process halfway through the first column. The script takes the position of the mouse (marked by a $\circ$) and finds the nearest pixel to this in the list of centres (marked by a $*$) and assigns this centre $[i\ j]$ pairing the $[x\ z]$ co-ordinate (again, $[x\ y]$ vertically) it has calculated from the initial position of that column and the $2cm$ increment between disc centres. This process is repeated for each column. When this phase is complete, the $[i\ j\ x\ z]$ matrix has a column of constant $[y]$ added equal to the y value of the horizontal plane ($[x]$ added to $[i\ j\ y\ z]$ in the vertical case).

The resulting matrix is then saved in ASCII format as `hor.a` (or `vert.a`) and a copy is placed in the `calib/` directory of the directory from where the `newstripe` program is run, so that it may locate the files and fit the interpolation function.

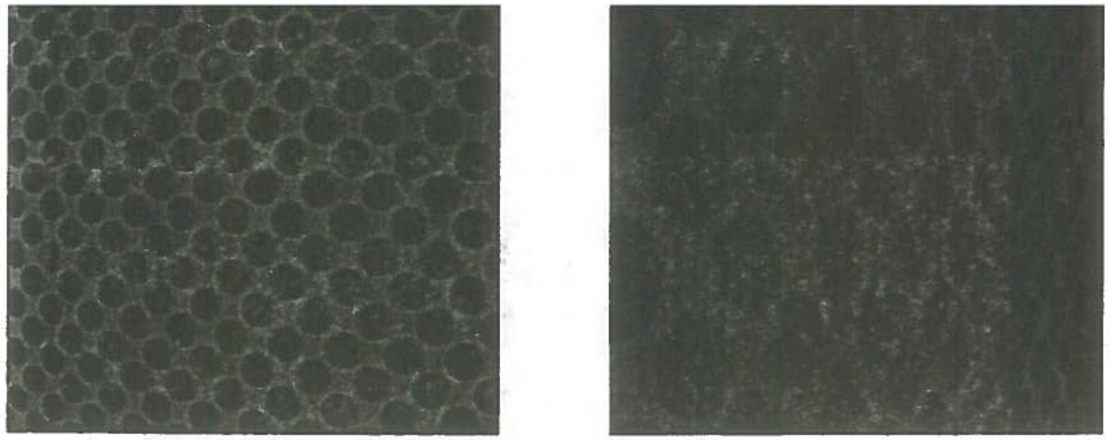
Figure A.2: (l) Horizontal plane image (r) vertical plane image

A.2 Geometry Calibration

This simple process [Fitgibbon 96], illustrated in Figure A.5, is used to calculate the transformation T_{r-c} (3×3 rotation matrix and 3×1 translation vector) between the camera and the Flock of Birds remote unit, and aligns the Flock of Birds sensor base unit's axes with a global co-ordinate system (T_{base}). The procedure is to collect stripe data off four known planes (whose co-ordinates are measured from an arbitrary but known global origin, close to the Flock of Birds sensor base unit) and then calculate the residual of each data point from its plane.

A transformation is the fitted to minimise these residuals, using a sophisticated version of a Levenberg-Marquardt function optimisation technique ([Press *et al.* 92] Section 15.5). The parameters of the planes are set in the array `realplanes` in the file `calibrate.cc`, and are vectors whose length is the orthogonal distance of the plane from the origin, and whose direction is the plane normal. A wooden cube was used for this calibration process in all the experiments presented in this dissertation. The data collection is now carried out by running the program

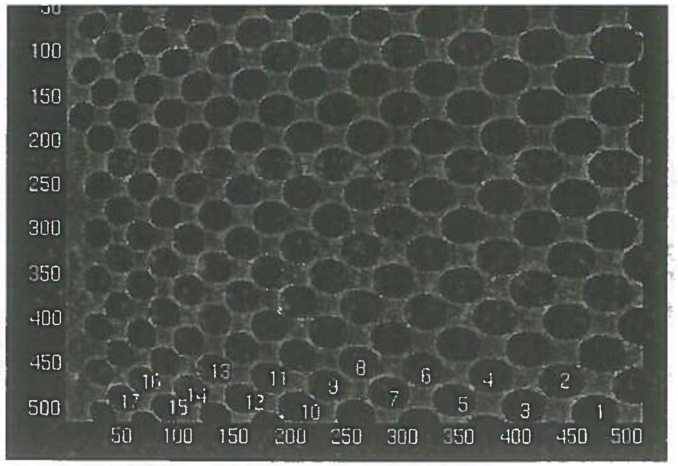

Figure A.3: Labelling the image

and following the on-screen instructions. To fit the transformations, the program

```
% newstripe -cf
```

is run, and this will take several minutes to complete. After this process, enough information will be known to accurately compute the global position x_{world} of any range point in the camera frame, x_{camera}, by using the formula

$$x_{world} = T_{base} \times T_{FoB} \times T_{r-c} \times x_{camera}$$

where T_{FoB} is the transformation between world co-ordinates and the Flock of Birds base frame.

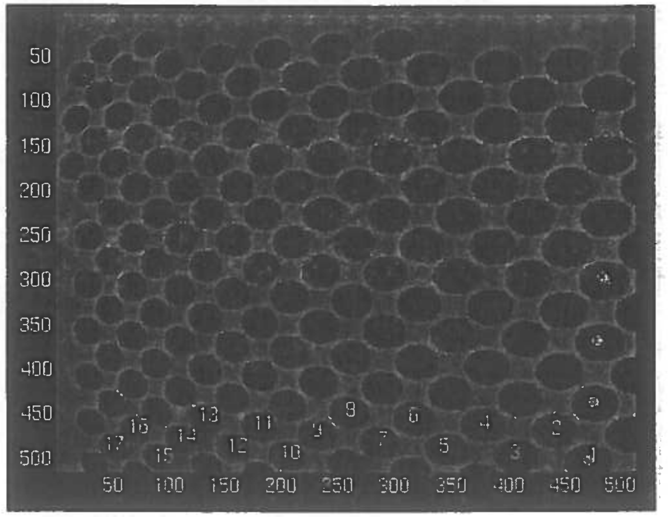

Figure A.4: Making the correspondences

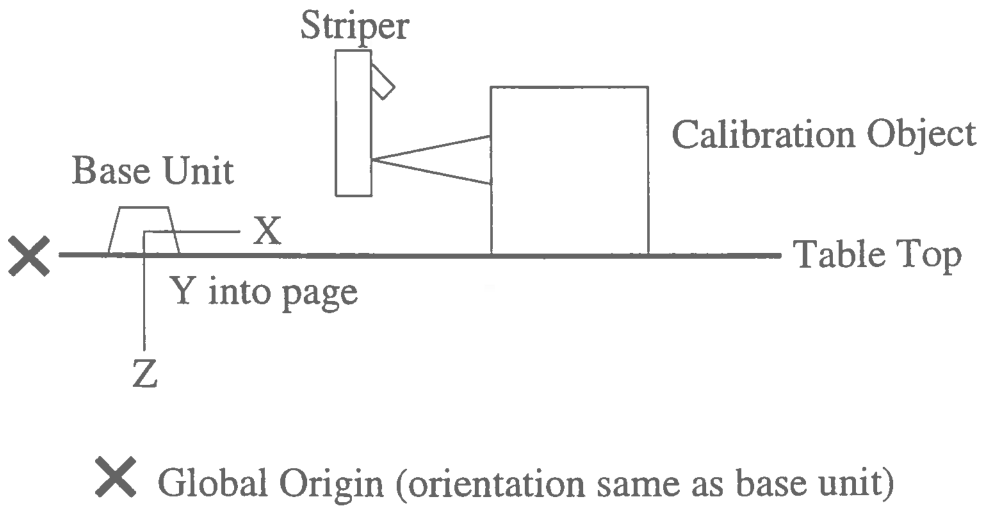

Figure A.5: Geometry calibration

Appendix B

Experimental Results

X Position (cm)	Standard Deviation (mm)
10	0.043
20	0.044
30	0.024
40	0.058
50	0.054
60	0.085
70	0.076
80	0.146
90	0.213

Table B.1: Flock of Birds static position accuracy

X Position (cm)	Standard Deviation (degrees)
15	0.3034
25	0.4042
35	0.3680
45	0.3436
55	0.4223
65	0.3402
75	0.3580

Table B.2: Flock of Birds static angular accuracy

Appendix C

The `VoxMap` Data Structure

The `VoxMap` structure (illustrated in Figure C.1) offers a speedy way to reference the 3D data points collected by the striper, and was inspired by structures from [Wyvill *et al.* 86]. Its basis is a three dimensional array of pointers, which either point to `NULL` if a voxel is empty, or to a `Voxel` data structure if there is some data in the voxel. The `Voxel` structure contains three integers; `intensity`, used to hold the number of elements in the voxel, `temp` which is used as a temporary variable in the smoothing convolution and `smoothed` which is used to store the result of the smoothing. The `Voxel` also holds a pointer to a doubly linked list `data_points` which holds the 3D co-ordinate of each point within the voxel, and a pointer to `normal`, a 3D vector to hold the local plane normal.

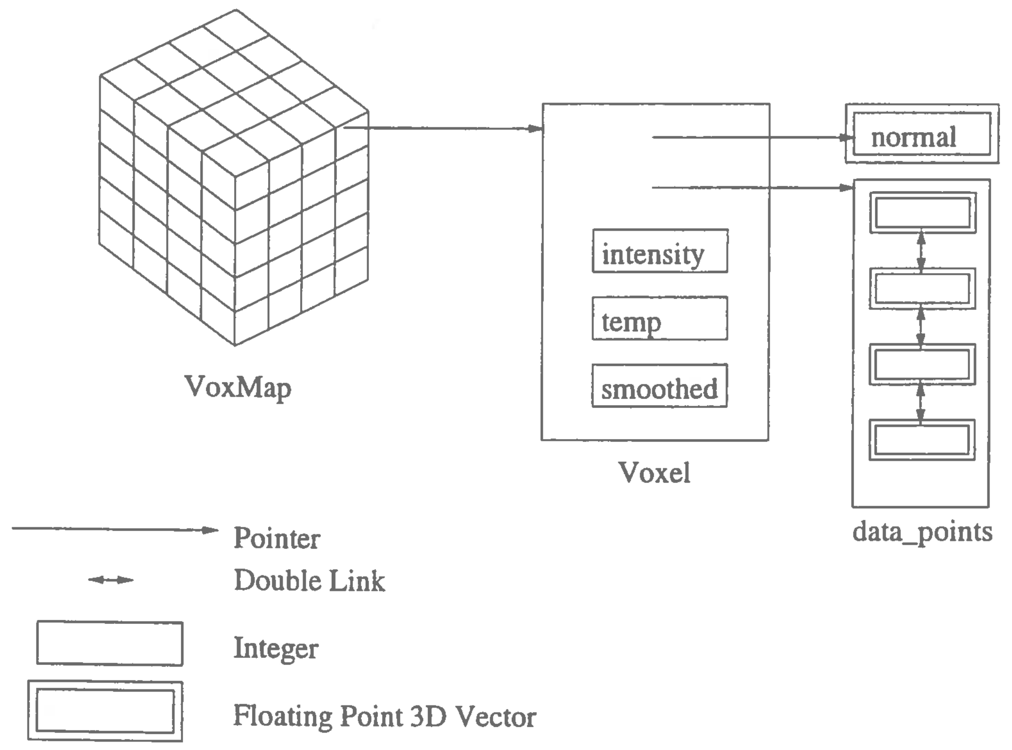

Figure C.1: The VoxMap data structure