



UNIVERSITY OF  
CAMBRIDGE



DEPARTMENT OF  
ENGINEERING

# Volume measurement and surface visualisation in sequential freehand 3D ultrasound

Graham Michael Treece

Christ's College

November 2000

A dissertation submitted to the University of Cambridge  
for the degree of Doctor of Philosophy.

# Summary

Three-dimensional (3D) acquisition and visualisation techniques are increasingly being incorporated into commercial ultrasound scanners. The diagnostic benefit of such techniques is not yet convincing, compared to the added inconvenience of using them. Although it is straightforward to use special probes to acquire 3D data, these are more restrictive than conventional 2D probes. The only 3D technique which retains the scanning flexibility and wide area of application of 2D ultrasound, is *freehand* 3D ultrasound, where a 2D probe is moved manually and its location sensed remotely. However, it is hard to generate a regular volume of data from the resulting non-parallel ultrasound images.

Probably the largest body of evidence justifying the use of 3D ultrasound relates to the accurate measurement of volume. However, volume measurement with 3D ultrasound requires segmentation (i.e. outlining of the area of interest), which is a time consuming, manual task. Both this problem, and that of creating a regular volume of data, are eased by the use of *sequential* freehand 3D ultrasound, a recent technique amplified in this thesis, where all the processing is performed on the original ultrasound images. Thus a regular array of data is not required, and segmentation is performed on images whose features and artifacts are recognisable to the clinician.

In this thesis, novel volume measurement and surface visualisation algorithms are developed for sequential freehand 3D ultrasound. A particular emphasis is placed on limiting the number of segmented cross-sections required for a given measurement accuracy. Inherent accuracy is demonstrated by simulation to be within  $\pm 2\%$ , from 10 or fewer cross-sections. *In vivo* precision, including registration and segmentation errors, is shown to be within  $\pm 7\%$ , even on complicated objects such as the liver or a foetus, from similar numbers of cross-sections. The volume can be updated in real-time as each image is segmented, and surfaces can be interpolated and visualised within a few seconds. Such visualisation can reveal errors in the segmentation which are otherwise hard to see.

In addition, a framework for multiple-sweep data is developed, which allows volume measurement and surface visualisation of anatomy that can only be scanned by several sweeps of the ultrasound probe. Accurate volume measurements can therefore be made of *all* areas observable by conventional ultrasound. These measurements can be accomplished within approximately 5 minutes of examining the patient.

The surface visualisation algorithms can also produce high quality renderings of data from a wide range of sources in medical imaging and other fields. The interpolation of cross-sections is an improvement on *shape-based interpolation*, and the triangular mesh created from the data is of a much higher quality than that using *marching cubes*. An extension of the surface interpolation algorithm can also be used to gradually change, or ‘morph’ one 3D surface to another, a technique commonly used in the film industry.

**Keywords** freehand 3D ultrasound, volume measurement, surface from cross-sections, isosurface triangulation, volume metamorphosis.

# Acknowledgements

This thesis has benefited greatly from many discussions with colleagues in the Speech Vision and Robotics group at the Department of Engineering. For these discussions, and for their friendship, I would like to thank especially: Robert Rohling, Jonathan Carr, Gavin Smith, Patrick Gosling, Silke Witt and Tom Drummond. Thanks also to David Tricker, for much needed encouragement and long lunch breaks.

I am also indebted to Diana Galletly, Claire O'Brien, Richard Prager and Andrew Gee for being willing to be scanned; and to Loïs and baby Prager for not moving *too* much during the process. Much of the quality of the ultrasound data in this thesis is due to Laurence Berman's expertise in scanning. Without funding from the EPSRC and a Newton Trust award from the Department of Engineering, this work would not have been possible.

Thanks also to Richard Prager and Andrew Gee for a level of support throughout which was well beyond the call of duty.

Finally, thanks to my wife, Sarah, for everything.

# Declaration

This dissertation is the result of my own original work and does not include anything done in collaboration with others. It has neither been submitted in whole nor in part for a degree at any other university. It contains 100 figures and approximately 58,000 words including figures, appendices and references. The following publications were derived from this work:

## Journal articles

G. M. Treece, R. W. Prager, A. H. Gee, and L. Berman. Fast surface and volume estimation from non-parallel cross-sections, for freehand 3-D ultrasound. *Medical Image Analysis*, 3(2):141–173, 1999.

G. M. Treece, R. W. Prager, and A. H. Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Computers and Graphics*, 23(4):583–598, 1999.

G. M. Treece, R. W. Prager, A. H. Gee, and L. Berman. Surface interpolation from sparse cross-sections using region correspondence. *IEEE Transactions on Medical Imaging*, 19(11):1106–1114, 2000.

G. M. Treece, R. W. Prager, A. H. Gee, and L. Berman. 3D ultrasound measurement of large organ volume. *Medical Image Analysis*, 5(1):41–54, 2001.

G. M. Treece, R. W. Prager and A. H. Gee. Volume-based three-dimensional metamorphosis using automatic region correspondence. Accepted for *The Visual Computer*.

## Conference presentations

G. M. Treece, R. W. Prager, A. H. Gee, and L. Berman. Volume measurement in sequential freehand 3-D ultrasound. In *Proceedings of the 16th International Conference on Information Processing in Medical Imaging*, pages 70–83, Visegrad, Hungary, June 1999. Springer.

G. M. Treece, R. W. Prager, A. H. Gee, and L. Berman. Surface interpolation from sparse cross-sections using region correspondence. In *Proceedings of Medical Image Understanding and Analysis*, pages 17–20, Oxford, July 1999.

G. M. Treece, R. W. Prager, A. H. Gee, and L. Berman. Volume measurement of large organs with 3D ultrasound. In *Medical Imaging 2000 (Ultrasonic Imaging and Signal Processing)*, volume 3982 of *Proceedings of SPIE*, pages 2–13, San Diego, California, February 2000.

G. M. Treece, R. W. Prager, A. H. Gee, and L. Berman. Volume measurement of large organs with 3D ultrasound. In *Proceedings of Medical Image Understanding and Analysis*, pages 133–136, London, July 2000.



# Contents

<b>Glossary</b>	<b>viii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.1.1 What is ultrasound? . . . . .	1
1.1.2 Why 3D ultrasound? . . . . .	2
1.1.3 Why volume measurement and surface visualisation? . . . . .	3
1.1.4 Why freehand 3D ultrasound? . . . . .	6
1.1.5 Summary of motivation . . . . .	10
1.2 Challenges of using freehand 3D ultrasound . . . . .	10
1.2.1 Data registration . . . . .	12
1.2.2 Data interpolation . . . . .	12
1.2.3 Data segmentation . . . . .	13
1.2.4 Surface visualisation . . . . .	14
1.2.5 Clinical constraints . . . . .	15
1.3 Sequential freehand 3D ultrasound: a partial response . . . . .	15
1.4 Original contribution . . . . .	16
1.4.1 Volume measurement in a sequential framework . . . . .	16
1.4.2 Surface visualisation in a sequential framework . . . . .	17
1.4.3 Application to multiple-sweep data . . . . .	17
1.5 Beyond ultrasound: application to other fields . . . . .	18
1.5.1 Medical imaging . . . . .	18
1.5.2 Computer graphics . . . . .	18
1.6 Outline of thesis . . . . .	19
<b>2 Volume measurement</b>	<b>20</b>
2.1 Current methods of ultrasound volume measurement . . . . .	20
2.1.1 Volume from length measurements . . . . .	20
2.1.2 Volume from area measurements . . . . .	20
2.1.3 Volume from surfaces . . . . .	21
2.2 A new approach: cubic planimetry . . . . .	22
2.2.1 Overview . . . . .	22
2.2.2 Dealing with non-parallel cross-sections . . . . .	22
2.2.3 A 2D representation of the problem . . . . .	23
2.2.4 Cubic interpolation of the 2D representation . . . . .	25
2.3 Simulated scanning results . . . . .	29
2.3.1 Test setup . . . . .	29

2.3.2	Volume measurement accuracy . . . . .	30
2.3.3	Registration and unsteady hand effects . . . . .	35
2.4	<i>In vivo</i> ultrasound results . . . . .	37
2.4.1	Kidney . . . . .	37
2.4.2	Bladder . . . . .	40
2.4.3	Foetus . . . . .	41
<b>3</b>	<b>Surface interpolation and visualisation</b>	<b>44</b>
3.1	Current methods of surface interpolation . . . . .	44
3.1.1	Surface from cross-sections . . . . .	44
3.1.2	Shape-based interpolation . . . . .	45
3.1.3	Surface from non-parallel cross-sections . . . . .	49
3.1.4	Surface from scattered points . . . . .	50
3.1.5	Morphing . . . . .	50
3.2	A new approach: disc-guided interpolation . . . . .	51
3.2.1	Overview . . . . .	51
3.2.2	Calculation of the distance transform . . . . .	51
3.2.3	Maximal-disc representation . . . . .	56
3.2.4	Calculation of region correspondence . . . . .	58
3.2.5	From region correspondence to point correspondence . . . . .	61
3.2.6	Interpolation using point correspondence . . . . .	62
3.3	Current methods of surface visualisation . . . . .	66
3.3.1	Marching cubes and its derivatives . . . . .	67
3.3.2	Mesh simplification . . . . .	68
3.3.3	Related applications . . . . .	70
3.4	A new approach: regularised marching tetrahedra . . . . .	71
3.4.1	Overview . . . . .	71
3.4.2	Choice of sampling grid . . . . .	72
3.4.3	Topology preservation . . . . .	74
3.4.4	Clustering and triangulation . . . . .	76
3.4.5	Adapting to local curvature . . . . .	78
3.4.6	Implementation . . . . .	80
3.5	Results: surface interpolation . . . . .	80
3.5.1	Simulated scanning results . . . . .	81
3.5.2	<i>In vivo</i> ultrasound results . . . . .	82
3.5.3	Other applications . . . . .	82
3.6	Results: surface visualisation . . . . .	86
3.6.1	Simulated results . . . . .	90
3.6.2	<i>In vivo</i> ultrasound results . . . . .	90
3.6.3	Other applications . . . . .	93
<b>4</b>	<b>Application to multiple-sweep data</b>	<b>97</b>
4.1	Current methods for combining data from multiple sweeps . . . . .	97
4.2	A new approach: partitioning multiple sweeps . . . . .	98
4.2.1	Overview . . . . .	98
4.2.2	Partitioning of data using dividing planes . . . . .	101
4.2.3	Segmentation with dividing planes . . . . .	104

4.2.4	Application to volume measurement	104
4.2.5	Application to surface visualisation	106
4.3	Simulated scanning results	106
4.4	<i>In vivo</i> ultrasound results	109
4.4.1	Liver	109
4.4.2	Foetus	110
<b>5</b>	<b>3D volume morphing: an extension of surface interpolation</b>	<b>114</b>
5.1	Current methods of 3D morphing	114
5.1.1	Comparison to image morphing	114
5.1.2	Comparison to functional surface interpolation	114
5.1.3	Volume-based as opposed to polygon-based or implicit morphing	115
5.2	A new approach: sphere-guided interpolation	116
5.2.1	Overview	117
5.2.2	Conversion to volume-based representation	118
5.2.3	Distance transformation	120
5.2.4	Sphere representation	121
5.2.5	Interpolation using sphere correspondence	122
5.2.6	Manual guidance of correspondence	124
5.2.7	User interface design	125
5.3	Results	128
5.3.1	Pawn to queen	128
5.3.2	Tricycle to sphere	130
5.3.3	Dragon to creature	130
5.3.4	Pear to mushroom	130
<b>6</b>	<b>Summary and conclusions</b>	<b>135</b>
6.1	3D ultrasound	135
6.1.1	Volume measurement	135
6.1.2	Surface visualisation	136
6.2	Application to other fields	138
6.2.1	Surface interpolation and visualisation	138
6.2.2	3D volume morphing	138
6.3	Conclusions and further work	139
<b>A</b>	<b>Test setup</b>	<b>141</b>
A.1	Generating simulated scans	141
A.2	Acquisition and processing of <i>in vivo</i> data	142
<b>B</b>	<b>Algorithms used in volume measurement</b>	<b>145</b>
B.1	Area from parametric cubic splines	145
B.2	Volume from geometric surface	146
<b>C</b>	<b>Geometric algorithms</b>	<b>148</b>
C.1	Discrete representation of a cross-section	148
C.2	Topological checks for clustering	150
C.3	Intersection of a polygon with a set of planes	153
C.4	Clipping a surface to a set of planes	154

---

<b>D Non-ultrasound software implementations</b>	<b>156</b>
D.1 IsoSurf — surface visualisation for parallel data . . . . .	156
D.2 EqnSurf — visualisation of surfaces of equations . . . . .	156
D.3 VolMorph — volume morphing of surfaces . . . . .	157
<b>E Description of CDROM</b>	<b>158</b>
<b>Bibliography</b>	<b>159</b>

# Glossary

<b>2D/3D/4D</b>	two-/three-/four-dimensional.
<b>atherosclerosis</b>	hardening of the arteries.
<b>B-scan</b>	a cross-sectional ultrasound image within the plane of the ultrasound beam.
<b>catheterisation</b>	insertion of a tube into the bladder, for measurement of urine volume.
<b>chamfer code</b>	a discrete, propagated estimate of distance.
<b>choroidal melanoma</b>	a malignant tumour of the eye.
<b>cleft lip</b>	(or hairlip) a congenital deformity of the upper lip.
<b>compounding</b>	the averaging of data, usually acquired from different directions, to reduce noise.
<b>cross-dissolving</b>	the averaging of the grey-levels of two images, to generate intermediate images.
<b>CT</b>	Computed Tomography, a technique for acquiring 3D cross-sectional data, using X-rays.
<b>distance transform</b>	a discrete image whose values have been replaced by distances to an object within that image.
<b>dysplasia</b>	abnormal development of skin, bone or other tissue.
<b>echocardiography</b>	ultrasound of the heart; regarded as a separate discipline to ultrasound of other areas.
<b>ejection fraction</b>	the ratio of the blood ejected from the left ventricle of the heart, to that remaining.
<b>endometrial polyp</b>	a benign tumour of the lining of the uterus.
<b>fibroid</b>	a benign fibrous tumour.
<b>fontanelle</b>	a gap between the skull bones in a young infant.
<b>freehand</b>	unrestricted manipulation by hand.
<b>Gouraud shading</b>	a computer graphics technique for shading lit polygonal surfaces smoothly.
<b>grey-scale</b>	any data whose values represent a range from white to black.
<b>haemodynamic</b>	related to blood flow.

---

<b>haemorrhage</b>	loss of blood.
<b>hypo/hyperplasia</b>	under-/over-development of tissue.
<b>hysteroscopy</b>	investigation of the inside of the uterus.
<b>iliac crests</b>	part of the pelvis.
<b>implicit surface</b>	a surface defined as the zero value of a 3D function.
<i>in vitro</i>	conducted on a specimen; not <i>in vivo</i> .
<i>in vivo</i>	conducted on a living organism.
<b>insonification</b>	broadcast of ultrasound over a region.
<b>intravascular</b>	inside a blood vessel.
<b>isosurface</b>	a surface of constant value of a function.
<b>IUD</b>	Intra-Uterine Device, a contraceptive device.
<b>lattice</b>	a regular arrangement, used in this context to determine where a continuous function should be sampled.
<b>LED</b>	Light Emitting Diode.
<b>lumen</b>	the space within a blood vessel, or other fluid filled cavity.
<b>mammography</b>	X-ray examination of the breast.
<b>manifold</b>	any surface which can be flattened out to a plane without local distortion.
<b>MC</b>	Marching Cubes, a technique for triangulating the surface of a regular array of 3D data.
<b>mesh simplification</b>	any technique for reducing the number of polygons in a mesh.
<b>morphing</b>	the metamorphosis of one object to another.
<b>MRI</b>	Magnetic Resonance Imaging, a technique for acquiring 3D cross-sectional data, using magnetic fields.
<b>MSD</b>	Maximal Set of Discs.
<b>MSMD</b>	Minimal Set of Maximal Discs.
<b>MT</b>	Marching Tetrahedra, a technique similar to MC, but based on tetrahedral, rather than cubic, lattices.
<b>myointimal layer</b>	part of the arterial wall.
<b>pathology</b>	the presence of abnormalities associated with disease.
<b>PET</b>	Positron Emission Tomography, a technique for determining the location of function, e.g. blood flow.
<b>pixel</b>	each element of a discrete image.
<b>planimetry</b>	the measurement of volume from planar area.
<b>pulmonary</b>	relating to the lungs.
<b>raster scan</b>	row by row processing of an image.
<b>ray casting</b>	a computer graphics technique for rendering volume-based data, where a light ray from each screen pixel is cast through the volume.

---

<b>real-time</b>	any processing which is performed as the data is generated.
<b>registration</b>	relative alignment of sets of data.
<b>rendering</b>	the representation of an object on a computer screen.
<b>reslice</b>	a cross-section through a 3D data set.
<b>saphenous vein</b>	a superficial vein in the leg.
<b>scan conversion</b>	the sampling of a geometric representation to a regular grid by a raster scan.
<b>segmentation</b>	the process of specifying a region of interest within a data set.
<b>speckle</b>	noise in ultrasound images due to random diffuse reflection from many small scatterers.
<b>splatting</b>	a computer graphics technique for rendering volume-based data, where each voxel is projected onto the screen in turn.
<b>spline</b>	a piecewise polynomial function used to represent a part of a curve in computer graphics.
<b>sweep</b>	one motion of the ultrasound probe across the skin surface.
<b>transvaginal</b>	through the vagina.
<b>triangulation</b>	the construction of a triangular mesh from a set of points, or data values.
<b>ultrasound</b>	very high frequency sound waves, generally measured in MHz, used for examining internal organs.
<b>vertex clustering</b>	the merging of vertices in a surface mesh, to reduce the number of primitives.
<b>urethrogram</b>	X-ray of the urethra, after addition of a radio-opaque fluid.
<b>ventricle</b>	either a chamber of the heart, or a fluid-filled cavity in the brain.
<b>volumetry</b>	the measure of volume by counting voxels.
<b>voxel</b>	(from <b>volume pixel</b> ) each element in a rectilinear 3D array.
<b>warping</b>	controlled distortion of data, usually by moving the relative location of each data value.

# Chapter 1

## Introduction

### 1.1 Motivation

In the last three decades, a great many researchers have attempted to produce systems which will allow the construction and visualisation of three-dimensional (3D) images from ultrasound data [130]. There is general agreement that this development represents a positive step forward in medical imaging, and clinical applications have been suggested in many different areas. However, although manufacturers are now beginning to introduce some 3D features to their flagship ultrasound machines<sup>1</sup>, it is clear that 3D ultrasound has not yet gained widespread clinical acceptance. There are many algorithms for providing new visual and quantitative information from such data, but the effort required to generate such information often exceeds the benefit it would provide. More importantly, such algorithms can mask clinical information, or introduce artifacts: new information which is not justified by the measured data.

The emphasis in this work is not to provide new types of information from 3D ultrasound data, but to improve the integrity of such information, the ease with which it can be produced, and to expand the areas to which it can be applied. These are of particular importance in a clinical setting. Hopefully, in doing so, the advantages of using 3D ultrasound will begin to sufficiently outweigh the disadvantages, such that it can become a useful clinical tool.

#### 1.1.1 What is ultrasound?

Ultrasound images, or B-scans<sup>2</sup> are essentially a measure of acoustic response to an impulse at a particular frequency<sup>3</sup>. A simplified outline of this process is shown in Figure 1.1. In reality, all stages of the process are considerably more complex; for a more detailed description, see [103]. Ultrasound images cannot be associated with any particular tissue parameter; the magnitude of the reflected signal is affected both by the compressibility, and *changes* in the density, of the material. Hence the acoustic response at any location is affected by tissue above as well as at that location — a scan of the same area from another direction will therefore generate a different image. This, and many other effects (reviewed extensively in [158]) can make ultrasound images

---

<sup>1</sup>For instance Kretz Technik, <http://www.kretz.co.at>, ATL, <http://www.atl.com> and Siemens Ultrasound, <http://www.siemens-ultrasound.com>

<sup>2</sup>B-scans are one of several images which can be produced using ultrasound. The ‘B’ is historical and distinguishes this mode of image from ‘A’, ‘C’ or ‘M’ mode. However, B-scans are commonly used, and this thesis is concerned exclusively with this type of ultrasound image.

<sup>3</sup>The choice of frequency represents a trade-off between spatial resolution and maximum depth. ‘High’ frequencies around 15MHz have very good resolution, but can only scan to a depth of a few centimetres. Lower frequencies around 3MHz, with poorer resolution, can scan to depths of 25cm or above.



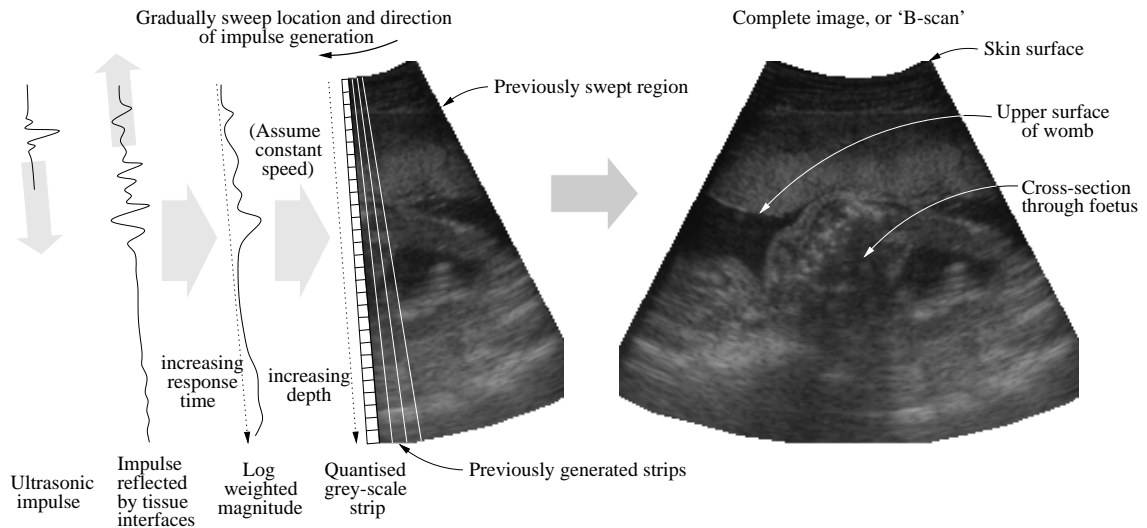


Figure 1.1: **Ultrasound B-scan generation.** Each B-scan is composed of a set of strips acquired sequentially over time and space. Each strip represents the magnitude of the response from an ultrasound impulse generated in that location and steered in that direction. A constant speed of sound is assumed in order to relate this response to depth within the patient. Since the magnitude of the response decays exponentially with depth, it must also be log-weighted before it can be quantised and represented as a grey-scale image strip.

very hard to interpret. Nevertheless, it is one of the most common forms of medical imaging in use today.

### 1.1.2 Why 3D ultrasound?

A 3D ultrasound system, as its name suggests, is one in which the location of the ultrasound signal is known in three dimensions. This allows a whole volume data set to be generated, rather than just the conventional planar data set, or image.

3D ultrasound is the least widespread amongst a variety of medical imaging modalities, e.g. Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET) or conventional B-mode and Doppler 2D ultrasound. Each has its own distinctives and is appropriate for particular situations. The main advantages which have been put forward for 3D ultrasound are:

- Ultrasound is a real-time imaging modality, and 3D ultrasound also has the potential for displaying information in near real time. This is currently limited more by processing than acoustics, which is the ultimate limit. High rates of acquisition (between 10-60 images per second) [152] can allow imaging within a single breath hold, greatly increasing modelling accuracy in organs which move with inspiration and expiration. This also allows the study of organ motion itself; much of the development of 3D ultrasound has been targeted towards such studies of the heart. Shorter examination times are also of benefit to both clinician and patient [51, 57].
- Extension of ultrasound to 3D provides new images which would otherwise be impossible to visualise, and previously could only be imagined by the clinician building up a mental picture from 2D information [131]. This may make the modality more accessible to those

less experienced in analysing ultrasound images. The acquisition of an entire volume allows the clinician to view 2D ultrasound images in planes which would not normally be possible due to the physical restrictions of the scanning process, for instance parallel to the skin surface. It also allows for surface-rendered or volume-rendered displays — different ways of looking at the data that can sometimes reveal pathology which is otherwise hard to see [50]. These are also very accessible images which are often appreciated by patients, particularly in the field of obstetrics [173].

- The reconstruction of 3D ultrasound by computer brings greater standardisation and repeatability to conventional examinations [152], which are otherwise quite subjective [57]. It may provide a better point of reference for discussing diagnosis than a conventional 2D hard copy. It would also provide a better means of documentation of the examination for clinical record, or for remote diagnosis.
- 3D data can be used to provide much more accurate assessments of volumes due to the use of non-geometric methods rather than the conventional ellipsoid-based formulae [1].
- 3D ultrasound offers the potential for improving the quality of a conventional 2D image, by compounding images acquired from different perspectives [18, 159].
- There are some practical reasons for using ultrasound rather than CT, MRI or PET. Existing ultrasound machines are less expensive both to buy and to run, and require less specialised facilities [131]. They can also be upgraded to 3D surprisingly cheaply [152]. Unlike CT, ultrasound scanning does not involve the use of ionising radiation, and the examination is less traumatic for the patient than MRI.
- Finally, in some cases 3D ultrasound may be able to replace other, more invasive, procedures [152].

The main disadvantage of 3D ultrasound is the same as for conventional 2D ultrasound, in that the image quality (though not necessarily resolution) is much worse than it is in CT or MRI. In addition, it is not possible to examine through air or bone interfaces, since these interfaces completely reflect the ultrasound signal. This restricts the anatomical areas in which ultrasound can be used. The decrease in signal amplitude with depth also limits the usefulness of the modality for examining larger patients.

### 1.1.3 Why volume measurement and surface visualisation?

There are many clinical areas to which 3D ultrasound has been applied [50, 57, 152]:

**Heart** Much of the work involving 3D (and 4D) ultrasound has been concerned with echocardiography. For reviews of this wealth of literature see, for example [125, 165]. In part, this has been driven by the desire to understand and visualise the structure of the heart and how it changes throughout the cardiac cycle. Most clinical applications relate to the measurement of ventricular volume, from which ejection fraction can be derived, for instance in heart disease [8, 136, 165]. Measurements of both ventricular volume and mass by 3D ultrasound have been shown to be accurate even for hearts of abnormal size and geometry [71]. Increased left ventricular mass has been established as a predictor of increased cardiac morbidity and mortality independent of age and blood pressure [71]. Accurate

measurements of atrial diameter can also be made [117]. In addition, 3D visualisation may reduce invasive monitoring for managing critically ill patients during surgery [119].

**Carotid Artery** Many researchers have suggested that 3D visualisation of the carotid artery can help in the accurate definition of the size and features of atherosclerotic plaques [57, 162]. For instance, it can generate simultaneous displays of both longitudinal and circumferential distribution of the plaque, which is impossible with 2D images. However, Rankin [152] argues that clinical applications for the popular 3D reconstruction of the carotid bifurcation [6] have yet to be defined.

**Vascular Anatomy** Serial surveillance of saphenous vein bypass grafts is of benefit for the detection of haemodynamically significant myointimal hyperplastic lesions [84, 95]. Geometric measurements of luminal change with 3D ultrasound may allow detection prior to the development of haemodynamic disturbances.

**Brain** 3D ultrasound has been proposed for the localisation of tumours and arteriovenous malformations during surgery of the brain, in order to provide accurate guidance for surgeons [152].

**Cerebral ventricles** Although it is not normally possible to examine the brain using ultrasound (except during surgery, as above), new-born babies have skulls which are not completely formed, and the brain can hence be examined through the anterior fontanelle. Volume measurement of the ventricles using 3D ultrasound has been achieved using this technique [96, 127].

**Eye** Most of the work on the eye has concentrated on the evaluation of choroidal melanomas. The 3D display shows the tumour in relation to the globe and optic nerve, which can be significant for surgical planning, and allows volume measurement, which is important as tumour size is significantly correlated to survival rate [91]. Accurate volume measurement of melanomas has also been used for follow up of therapeutic response [152]. 3D ultrasound display has been suggested to be better than conventional 2D for the evaluation of vitreous haemorrhages and retinal detachment [51].

**Breast** Some early work by Lalouche [104] indicates that the hypoechoic areas, which are difficult to distinguish in 2D, organise into a visible ductular system in 3D. There is also a higher accuracy of diagnosis of benign cystic disease with 3D ultrasound than with X-ray mammography [57, 58].

**Foetus** Like the heart, the foetus has had much attention in this area. There have been several reviews of foetal applications of 3D ultrasound [121, 156, 173]. Much of this work has been concerned with the detection of foetal abnormalities [10], for instance cleft lip or other malformations of the face [107]. The visualisation of the foetal spine and thorax has also been achieved; important in evaluating skeletal dysplasia, abnormalities leading to a small thorax and subsequent pulmonary hypoplasia, and neural tube defects [132]. Volume measurements of the foetal lumbar spine have been reported [167]. Foetal volume and weight have been estimated and used to evaluate intrauterine growth retardation [152]. Transvaginal 3D ultrasound has been used to investigate the volume and shape of embryos and early foetuses [26]. Attempts at foetal echocardiography [49] and foetal liver [105] and lung [144] volume measurement using 3D ultrasound have also been reported.

**Placenta** Placental volume around mid-pregnancy has been measured using 3D ultrasound, to investigate its relationship to birth weight and susceptibility to heart disease in later life [87]. Another study of placental volume showed that second-trimester measurements alone were not sufficient to predict small-for-gestational-age infants [75]. 3D power doppler ultrasound has been used to examine the placental vasculature [147].

**Uterus** Reconstruction of transvaginal images of the uterine cavity allows accurate visualisation prior to surgical treatment of fibroids and endometrial polyps [14]. 3D transvaginal ultrasound has also been used to control intrauterine device (IUD) insertion [30]. In this case, abnormal IUD insertions were identified as accurately and precisely as would have been done by hysteroscopy.

**Liver** Reconstruction of the liver with 3D ultrasound has been severely limited, due to its large size, and sheltered position behind the lower ribs. It moves considerably with respiration and the left lobe also moves with cardiac motion [57]. Nevertheless, liver volume assessment is a standard part of clinical examination, and a more accurate measure would undoubtedly be useful.

**Gall bladder** 2D ultrasound is the current standard for measuring gall bladder volume, in order to investigate dysfunction, however 3D ultrasound has been found to be more accurate [79].

**Prostate** There have been a great many successful attempts to measure the volume of the prostate gland using 3D ultrasound [19, 129]. Accurate assessment of prostate size at regular intervals is essential to clinical studies of drug therapy for reducing prostate volume [2]. Tumour volume is of great prognostic significance in prostatic cancer, and its measurement may be more accurate with this technique [58]. A precise estimate of the enlargement due to benign prostatic hyperplasia helps to determine the appropriate therapy [174]. 3D ultrasound has also been used in the examination of baboon prostates in order to better understand the pathogenic mechanisms of prostate cancer. Rather than using “invasive surgery or terminal experiments to excise the prostate”, this method is minimally traumatic [92].

**Kidney** The size and discrete anatomy of the kidney make 3D imaging of this organ attractive. Possible clinical uses include intrarenal neoplasia and renal transplants, where accurate assessment of renal volume and its change over time may be of help in defining rejection [152]. Volume assessment by 3D ultrasound has been compared favourably to that performed by MRI [69].

**Bladder** Incomplete voiding is an important sign of many urologic disorders. This is conventionally measured by catheterisation, but ultrasound estimates of the volume are potentially more accurate and reliable [118].

**Stomach** 3D ultrasound has been used to assess intragastric distribution and gastric emptying, by repeated volume measurement after eating a specified meal [67].

**Urethra** 3D images of the urethra generated from ultrasound scans provide more information than conventional voiding urethrograms [133].

It is apparent from reviewing this list of potential applications for 3D ultrasound that volume measurement is a recurrent theme. This is supported by a more extensive review of the subject

given in [68]. This emphasis is partly due to the prevalence of volume measurements as clinical indicators, and partly due to the very approximate techniques which are currently used to measure volume using conventional ultrasound (reviewed in Chapter 2). 3D ultrasound clearly has much to offer in this area.

There is also something to be gained from being able to show the surfaces of objects in 3D. Although it is not necessary to find the surface of an object in order to calculate its volume, rendering the surface reveals the shape and arrangement of an object which cannot be seen from simply looking at cross-sections. Surface display is also a common feature in the list above. In fact this is probably the type of display which is most frequently conjured up in the imagination by the use of the term ‘3D’.

Volume measurement and surface visualisation can be helpfully regarded as a natural pair; both place very similar requirements on the clinician to define the object of interest. As explained in Section 1.2.3, this manual definition is generally the most laborious task in the whole process. Having completed this task, it is important to make full use of the information generated, and this is best done by both techniques combined. In addition, volume measurement without surface visualisation presents quantitative data to the clinician, but without an indication of how accurate this data is likely to be. A rendered surface from the same source can act as a reassurance that the volume measure is sensible.

#### 1.1.4 Why freehand 3D ultrasound?

*Freehand* scanning is one of many different techniques for generating 3D ultrasound data, and there are several reviews covering the variety of systems which have been successfully used for this purpose [57, 130, 165]. Such systems can be broadly classified into two main areas; *acquisition*, which covers the measurement of data value and spatial location, and *display*, which covers the techniques used to present this data to the user. Most systems have an additional *reconstruction* step where the acquired data is pre-formatted before display techniques are applied. However, as explained in Section 1.3, this step is not a necessity, and for the purpose of this thesis is considered to be part of the display processing. Freehand 3D ultrasound describes a particular acquisition technique. Most display techniques are equally applicable, not only to all ultrasound acquisition techniques, but also to most other medical imaging modalities.

Acquisition techniques can themselves be classified either by scan pattern or positioning equipment. As can be seen from Figure 1.2, most positioning equipment can be used with a variety of scan patterns. Freehand scanning is classified separately since, although it can be used to give approximate linear, fan or rotational scans, the relative plane orientation will not be as precise. All of the acquisition techniques, with the exception of volume scanning, are designed as additions to conventional ultrasound machines, and as a result are focused on calculating the location in space of entire B-scans rather than individual ultrasound responses. It is an underlying assumption of all such techniques that the 2D localisation within each B-scan is already correct. Inaccuracies in this assumption can be considered a further cause of ‘noise’ in each image. The majority of systems use the video output of the ultrasound machine to acquire the ultrasound data to an external PC, introducing further noise to the B-scans; an exception which used raw ultrasound data is presented in [22].

The following sections summarise the reported systems, grouped by the positioning equipment used in each case.

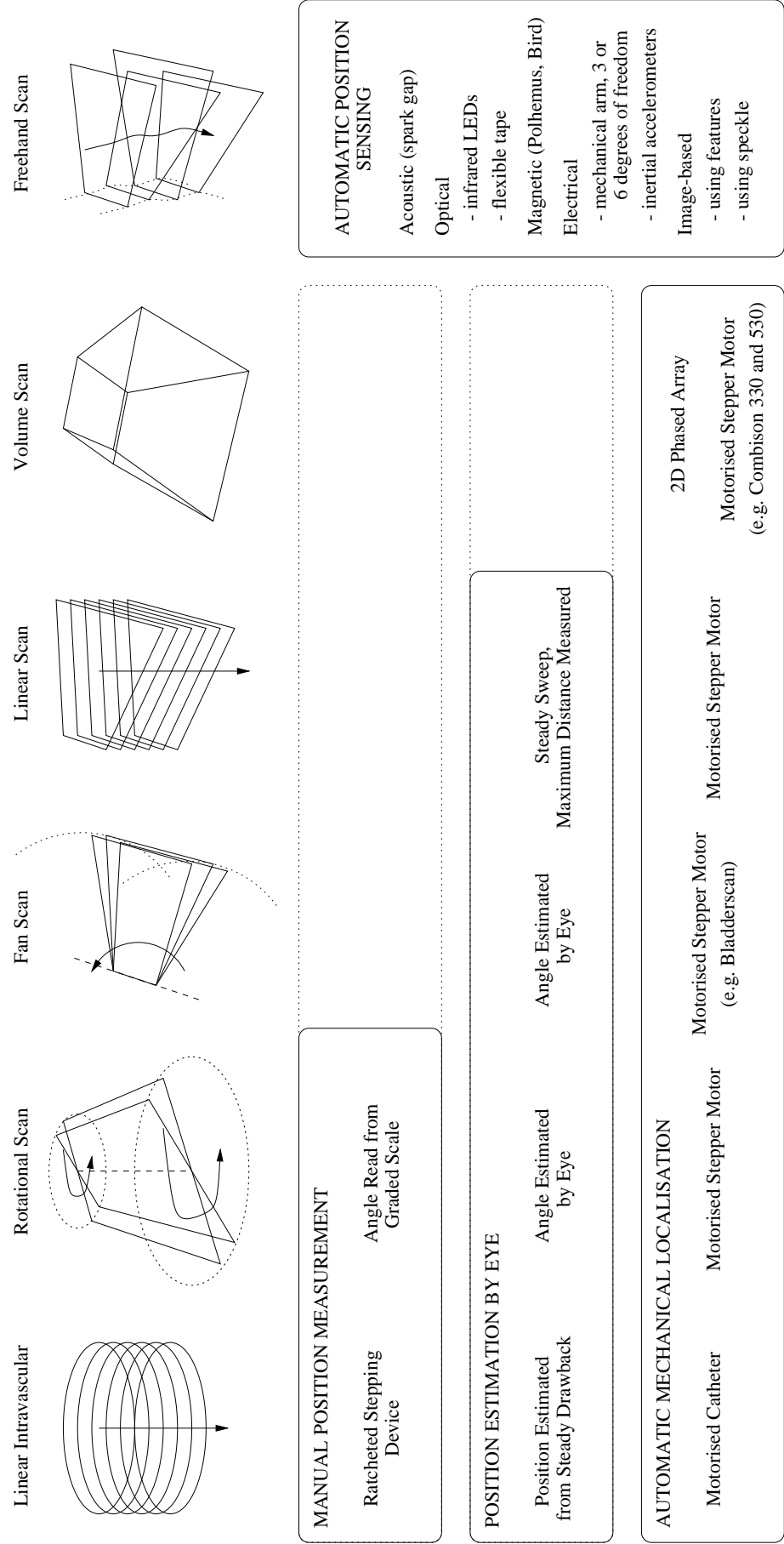


Figure 1.2: **3D ultrasound systems.** A variety of scan patterns can be achieved with different position measurement equipment.

## Position Estimation by Eye

The simplest systems rely on estimation of the position of the individual ultrasound images by eye. This requires extreme care when scanning, as it is imperative that the movement of the transducer is both smooth and constant. Any jitter introduced at this stage is carried through directly to the reconstruction. Examples of linear sweeps include [14] and [162], where in both cases the slice separation was assumed to be uniform. Rotational scans have also been performed using uniform sweeps [87], or by estimating the orientation of each image separately [59, 153]. In one case, 3D surfaces were reconstructed from only two images, manually positioned so that they were approximately orthogonal to each other [47]. While these methods are simple, they clearly leave much to be desired if accurate reconstructions are needed.

## Manual Position Measurement

This is a more reliable technique, but is only possible if each image is acquired individually (therefore giving time to measure its position). Examples include a catheter mounted on a ratcheted device, giving a measurement of drawback [129, 174], and rotational scanning with the angle measured from a scale attached to a fixed mechanical arm and the transducer [46]. Clearly this method precludes fast acquisition, which is one of the potential benefits of 3D ultrasound.

## Automatic Position Sensing

An alternative to measuring the position is to sense it remotely by some means. This is known as *freehand* scanning, since it potentially allows complete freedom of movement of the transducer, and hence results in images which are arbitrarily oriented in space. There are five reported ways of achieving this, namely *acoustical*, *optical*, *magnetic*, *electrical* and *image-based*.

**Acoustical** The acoustical method works by attaching an emitter to the transducer and picking up these signals with three remote microphones, positioned in different orientations (not necessarily orthogonal). An acoustic spark gap is generally used for the emitter as it can produce high frequencies (hence giving good resolution) and is a much smaller source than, for instance, a loudspeaker. The system was originally devised by Moritz [125] and is still used by some groups [7, 71, 101].

**Optical** The optical ('stereotactic') method works in a similar manner, except the emitter is replaced by at least three infrared LEDs [176]. A restriction of this technique is that there must be a line of sight at all times between the LEDs and the optical sensor. However, it is probably the most accurate method.

A new technique involving optical fibres mounted on flexible tape has also recently been investigated [140]. The tape is fixed at one end to a stationary reference and at the other to the ultrasound probe, and the curvature along the tape (measured using the optical fibres) is integrated to give the location and orientation of the probe. Although the technique holds promise, the accuracy currently limits it to lengths of tape which are too short to be used in practice.

**Magnetic** The most popular approach to remote sensing is to use a magnetic field. In this case the receiver is mounted on the ultrasound probe, and the transmitter, which generates the magnetic field, is remote [18, 22, 49, 52, 84, 89, 120, 132, 133, 145]. There are two



main types of sensor; one that uses an AC (alternating current), and one a DC (direct current). Each is affected to some extent by the presence of metallic (AC sensors) or ferromagnetic (DC sensors) materials in the vicinity [24]. However, they allow complete freedom of movement, and do not require a line of sight.

**Electrical** Several systems have used a mechanical arm with either one [119], three [138] or six [57] degrees of freedom. Potentiometers are mounted at the arm joints, and the electrical signals from these can be used to calculate the position. With one degree of freedom, (used for intravascular scanning of the heart) the scanning pattern is similar to a fan, but the angle between each scan need not be constant.

The *relative* position of each B-scan could also be approximated by measuring the movement of the probe using inertial accelerometers, although I am not aware of any published work in this area.

**Image-based** It is also possible to use the ultrasound images themselves to estimate the movement of the probe, and hence the relative position of each image. In-plane movement can be estimated by tracking image features, and used to generate much larger B-scans than is otherwise possible [183]. However, out-of-plane movement in such systems cannot be detected, and results in positioning errors.

It has recently been demonstrated that the statistics of speckle (the structured noise present in ultrasound images) can be used to provide an estimate of out-of-plane movement [179]. In combination with using image features, this can provide all six degrees of freedom. The method has the advantage of not requiring any additional positioning equipment. However, the positioning accuracy has not yet been fully established, and there is considerable processing overhead, which presents a challenge for real-time acquisition.

### Automatic Mechanical Localisation

A completely different approach to measuring position is to determine it in advance by mechanical localisation. This is generally achieved by attaching the transducer to a stepper motor, for linear [92, 104, 117, 169], rotational [51, 91, 136], fan [20, 69] or intravascular linear [102] scanning. This kind of approach has also been used in integrated transducers, for instance Kretz Technik's<sup>4</sup> Combison 330 [1, 2] and 530 [30, 107, 121, 173], which scans an entire pyramidal volume. Diagnostic Ultrasound Corporation also produce an integrated scanner and volume calculation device, Bladderscan<sup>5</sup>, specifically for urological use [118]. All these devices impose limitations on the maximum swept volume, since this is constrained by the mechanics.

In a similar category, two-dimensional phased arrays are also being investigated [112]. These may represent the future of 3D ultrasound; however before they become practical a number of problems must be overcome — for instance low yields due to the large number of small elements [57], and problems associated with the number of signals between the transducer and the ultrasound machine. Like mechanically swept probes, these still constrain the maximum swept volume.

<sup>4</sup><http://www.kretztechnik.com/>

<sup>5</sup><http://www.diagnosticultrasound.com/prodBS.html>



### Summary of acquisition techniques

Both the freehand and mechanical (or phased array) acquisition techniques have clear advantages. The mechanical systems produce a regular data set which simplifies later processing and ensures uniform coverage of the scanned volume. On the other hand, it is impossible to scan larger volumes than the mechanics allow for (the coordinate reference frame is based on the stepper motor housing, which must not be moved during acquisition). This limitation cannot simply be overcome by increasing the size of the system. Larger systems are difficult to use in practice, since the transducer must keep a good contact with the surface at all times, but excess pressure moves the underlying anatomy, generating mis-registered data.

The freehand systems have certain advantages over the mechanical systems, in that they do not constrain the acquisition volume, and they allow the clinician the freedom to scan in planes which are suitable for the area of interest. In many cases, particularly near the ribs, the most appropriate scanning plane is not orthogonal to the skin surface. However, the lack of regularity in the data set complicates the processing, and this potentially increases the time from scanning to display. It is also much easier to produce 4D data sets using the mechanical technique.

Perhaps the main advantage of the freehand acquisition technique is that it preserves the nature of ultrasound examination. Ultrasound is a more tactile modality than CT or MRI; the clinician is in complete control of the scan and can see the images as they are acquired. This is equally the case for freehand 3D ultrasound. 2D images remain the most diagnostically useful aspect of ultrasound imaging, and it is therefore important that any advances in the technology do not detract from the clinician's ability to produce these images.

#### 1.1.5 Summary of motivation

The subject of this thesis is 'volume measurement and surface visualisation in sequential freehand 3D ultrasound'. *Ultrasound* because it is readily available and in widespread use. *3D* because of its many advantages, and since, although there has already been much work in this area, it is not yet in frequent clinical use. *Volume measurement* because this is both the most compelling motivation for 3D ultrasound, and probably the area where the current clinical practice can be most improved. *Surface visualisation* because it provides information which is otherwise impossible to see, and it is also a natural pair with volume measurement, since it is based on the same initial data. *Freehand* ultrasound as it places the least constraints on both the size of volume which can be scanned, and the manner in which it is scanned.

*Sequential* freehand 3D ultrasound is a term, coined in this thesis, which describes an existing response to the challenges of the freehand acquisition technique. The initial stated aims of improving the integrity, the ease of production, and in addition expanding the application areas for 3D ultrasound data, are all contributory factors to this response. The *sequential* approach forms the basis of the volume measurement and surface visualisation algorithms outlined in Chapters 2 and 3, and has also been used for other forms of visualisation [65, 145]. The characteristics of the approach are described in Section 1.3; in order to appreciate these, it is first necessary to look at some of the challenges which freehand 3D ultrasound poses.

## 1.2 Challenges of using freehand 3D ultrasound

As already mentioned, freehand 3D ultrasound brings with it the challenge of working with data acquired in irregularly spaced and oriented planes. This problem, related to *interpolation*

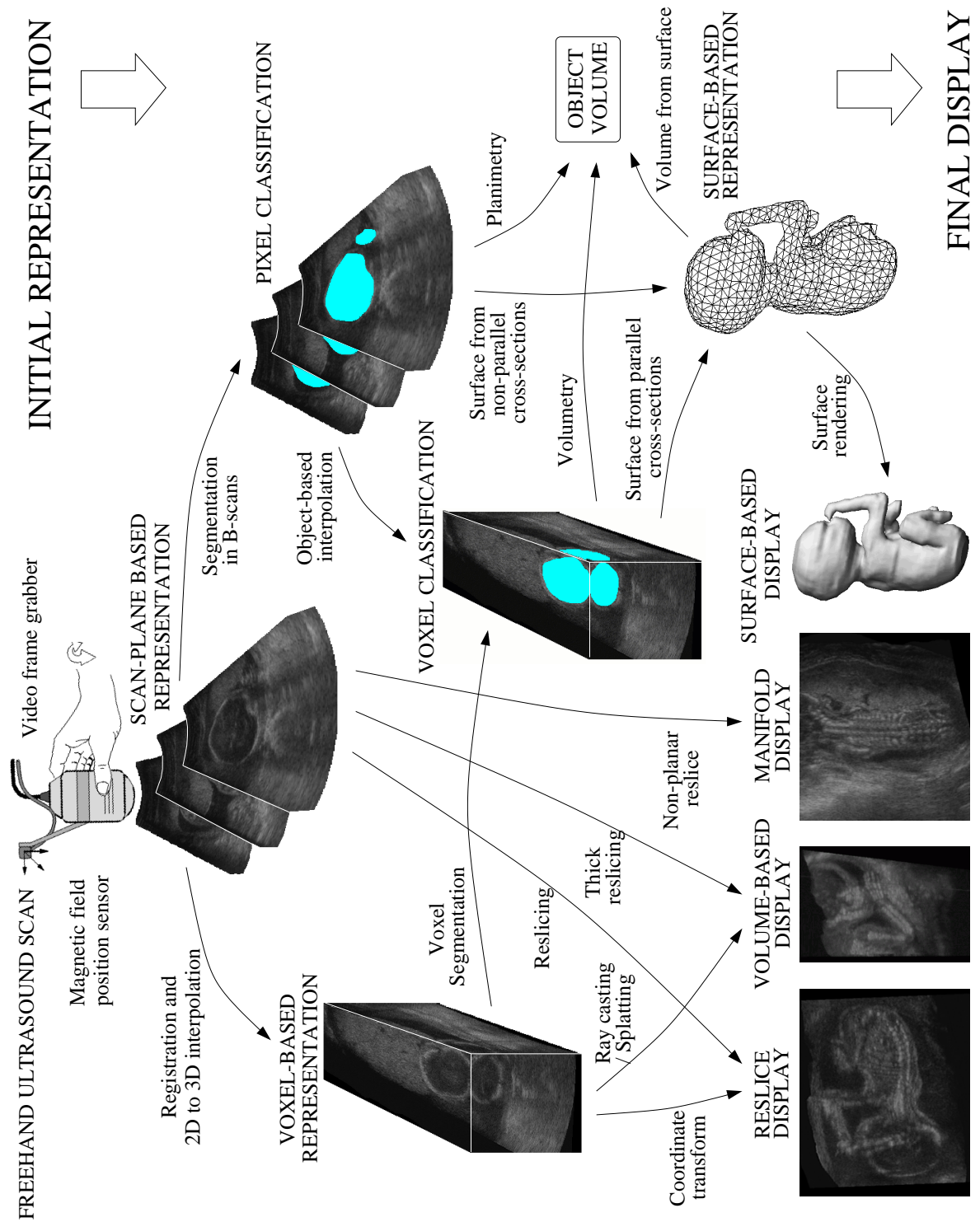


Figure 1.3: **Processing freehand 3D ultrasound data.** Major processing steps are shown from the initial representation to final display, with particular emphasis on volume measurement and surface visualisation.

and *registration*, adds to that of *segmentation* posed by the many artifacts present in ultrasound data. In addition, this modality shares the common problems of *surface visualisation* and working under *clinical constraints* which are associated with all 3D medical imaging modalities. Figure 1.3 places each of the processing tasks in the context of a path from the initial representation (constrained by the acquisition technique) to the final display or quantitative result (desired by the clinician). Each transition generally represents a whole area of research, with many underlying problems and solutions. Those which are relevant to the tasks of volume measurement and surface visualisation are reviewed here.

### 1.2.1 Data registration

The first challenge in freehand 3D ultrasound is to calculate exactly where the data is in space. This is not necessarily just a function of the device used for spatially locating the B-scans; movement can also occur *within* each B-scan, due to the pressure of the probe on the patient's skin, even if the patient themselves is lying perfectly still. In most cases, this second source of mis-registration can be addressed by good scanning practice, i.e. the patient remains still and the clinician keeps the probe contact at a constant pressure. However, there are two situations where good scanning practice alone is not sufficient: *spatial compounding* and *multiple-sweep scanning*.

Spatial compounding is the process of averaging multiple B-scans together to reduce the noise (speckle) present in the images [6, 55, 159]. This is only possible if the images are acquired from different viewpoints<sup>6</sup>; multiple images from the same viewpoint generate highly correlated speckle. However, if these images are not well registered, the averaging process blurs the data in addition to reducing speckle. Intensive image-based registration techniques are necessary as a first step before compounding the images [160].

Multiple-sweep scanning is the process of combining data from several sweeps of the ultrasound probe (i.e. examining the patient at one location, moving the probe, then examining again at another) in order to increase the volume over which data is acquired. This is necessary for examining anything with a larger cross-section than the width of an individual B-scan [3, 80]. Here the problem is not that increased registration accuracy is required, but that the registration errors in this case are much worse. This is a direct result of using multiple-sweep data; both the magnitude and the location of probe pressure change when the probe is moved in this manner. Combining such data can only be achieved by using image-based registration, and warping (stretching) the data [3].

Even if registration is not required within the B-scan, good spatial location is still a challenge; the ultrasound resolution within the scan plane is often greater than the precision of the spatial locator. For accurate 3D ultrasound investigations, these spatial locators first require careful calibration [146].

### 1.2.2 Data interpolation

It is not possible to display irregularly spaced data on a regular grid, such as a computer screen, without first interpolating this data. In fact the majority of freehand 3D ultrasound systems interpolate the data to a regular array before performing *any* other operations. Indeed the current reviews assume that this is the only approach to the problem [130, 165]. This array is often called a *voxel* array to indicate that each element represents a volume (as opposed to

<sup>6</sup>Multiple images acquired using different frequencies would suffice, but in general this is not a practical option.

the *pixels* which make up an image). It is generally of equal resolution in all three dimensions, since this simplifies subsequent processing. For this reason, interpolation is required even if the original scans are already parallel, since the scan spacing is usually not the same as the in-plane pixel resolution.

There are many ways to perform this interpolation. The techniques employed are often very simple, as it would otherwise take a long time to process the typically large amount of data<sup>7</sup>. Generally, a linear method involving a limited number of data points is used [20, 162, 169]. Gilja used trilinear interpolation of the eight neighbouring values from a regular rotational scan [69]. Lalouche used cubic interpolation, but from parallel scans [104].

Interpolation methods for freehand scans include nearest-neighbour bin-filling [176], which is probably the simplest technique — pixel coordinates are transformed to 3D space, and rounded to the nearest voxel, which is then assigned the original pixel value. Larger ‘bins’ (i.e. more than one voxel) can be assigned from each pixel in order to reduce any empty space between scans. More sophisticated is closest-points interpolation [176], where each voxel bounded by two scan planes is assigned a distance weighted value based on the closest pixel in each of the two scan planes. An inverse-distance weighting scheme can also be used, in which the inverse-distance weighted value of all the surrounding pixels in a bounding sphere of some chosen dimension [18], is assigned to each voxel. Alternatively, a 3D convolution of the 2D slice with a 3D reconstruction kernel can create the 3D volume incrementally. Both spherical [138] and ellipsoidal [120] Gaussian kernels have been used; the latter is more representative of the actual sampling resolution. A more accurate method has been suggested in which the data is smoothly approximated by using radial basis functions [161]; however the small gain in reconstruction accuracy is matched by a significantly increased processing overhead.

Voxel-based representations are relatively easy to manipulate, but they also have disadvantages. As already mentioned, considerable time is needed to create them (a minimum of several minutes for the simplest schemes). They also require a lot of memory space, for example a  $256 \times 256 \times 256$  voxel representation with 8-bit grey-scale data requires 16Mbytes. This is in addition to the memory required to store the original scans. Both the time and memory constraints frequently mean that in practice the original resolution of the data is compromised in the production of the voxel array. Also, the interpolation, however it is done, creates new data values for *all* the voxels, which can introduce artifacts, especially if (as is the case with most practical systems) the interpolation is a simple linear one [82]. These artifacts can affect the accuracy of segmentation, or lead to image misrepresentation.

### 1.2.3 Data segmentation

Segmentation is the process of identifying, within the data, a particular region of interest. This is a prerequisite for displaying surfaces<sup>8</sup> or for accurately measuring organ volume. In this context, segmentation is generally performed by constructing cross-sections of an organ, either in a B-scan, or in slices through a voxel array. However, it can also be performed directly in 3D.

<sup>7</sup>Using a dedicated processor rather than a PC can speed up this time such that approximately 10 B-scans can be processed per second [52]. For typical scans of 20 seconds duration at 25 frames per second, the production of a  $128 \times 128 \times 128$  voxel array could therefore be performed in just under one minute. However, this voxel array was only one quarter of the resolution of the original B-scans.

<sup>8</sup>As explained in [180], this is the case even if volume-based techniques are used to render the surface, since regions apart from the surface may need to be removed (for instance the womb, in the case of a foetus). In addition, many volume rendering techniques are themselves a type of segmentation algorithm.

The design of automatic segmentation algorithms is probably the most challenging problem in 3D ultrasound, and indeed in medical imaging and computer vision in general. This is equally the case for voxel-based and scan-plane based data. The problem is accentuated with ultrasound because of the many artifacts present in the images, compared with those of other imaging modalities. Automatic approaches are only feasible if strong assumptions are made about the data; in practice this means such algorithms are restricted to the data (and usually the organ) for which they were designed. Semi-automatic [17, 56, 97] methods can be applied more widely, but are still difficult to use in many circumstances. Many researchers recognise this difficulty and opt instead for manual segmentation (usually by outlining the feature on a computer screen). In a recent paper, Gopal [71] even goes to the extent of suggesting that

... manual boundary tracing is the most accurate method for identifying the surfaces of the ventricle for *all* imaging modalities,

and that

automated methods ... have not yet reached a stage suitable for adaptation to 3D echocardiography.

Manual segmentation is not, however, an ideal solution, since it is very time consuming. It may also have higher inter- and intra-observer variability than automatic techniques, although this has not always been the case in practice [1].

#### 1.2.4 Surface visualisation

The visualisation of surfaces is particularly challenging in freehand 3D ultrasound. Gradient operators can be used in ray casting voxel-based data, but the resulting surfaces are dominated by speckle. Both filtering of the data and removal of extraneous regions are required to produce a useful surface rendering. It is also extremely difficult to directly render the original, irregular data.

The alternative is to construct geometric surfaces from segmented cross-sections. This is a complex problem even when the cross-sections are parallel, as is the case if the segmentation is generated from a voxel array. The topology of each cross-section can be very different, both from the actual object represented by the cross-section, and from other cross-sections of this object. It is not always obvious which parts of the cross-sections correspond to each other, nor what to do if the topology changes. If the segmentation is generated from the original non-parallel B-scans, then the problem becomes similar to that of scattered data interpolation, which is even less well constrained.

Once a geometric surface has been generated, it must still be rendered. Fortunately, most graphics display hardware is optimised for rendering the geometric primitives (usually triangles) which make up such surfaces. However, the shading technique is usually very simple<sup>9</sup>, and can highlight thin or badly proportioned triangles. Higher quality renderings require higher quality surfaces, consisting of regular primitives with good aspect ratios. In addition, surfaces from medical data are often composed of a large number of primitives, and therefore take a proportionately longer time to render. If interaction with the surface is desired, the number of primitives in the surface must also be minimised.

<sup>9</sup>Gouraud shading is usually implemented as it is the fastest realistic technique [60].

### 1.2.5 Clinical constraints

The importance of maintaining the integrity of the original data in subsequent processing has already been established as a prime concern in any clinical context. This context also places other constraints on algorithms designed for freehand 3D ultrasound data; namely speed and simplicity.

Speed is important for any clinical tool — the clinician has limited time to spend with the patient, and this time should not be wasted waiting for algorithms to complete. In practice, where processing of medical data takes more than a few minutes, it becomes an off-line task to be performed after the patient has left. This is the case for much processing of CT and MRI data. Ultrasound, however, is a real-time modality where diagnosis is generally made during the examination of the patient. The most significant advantage of this approach is the ability to re-examine the patient appropriately, to verify conclusions reached from the initial data. It would be a step backwards to design an algorithm that forced ultrasound analysis to become an off-line process. This is particularly important in freehand 3D ultrasound, where acquiring a good data set is dependent on both scanning technique and a still patient; the latter is not an easy requirement to satisfy for foetal scans! The clinician must be able to review the data immediately after acquisition, in order to check that these requirements were met, and acquire further data if not.

The need for simplicity applies both to the underlying algorithms and the tools into which they are built. Clearly, it is good for the clinician to be able to use the tools with relative ease. More importantly, though, it must be clear what the data represents. It must also be possible to compare and contrast this data with similar data from other patients. This task becomes more complicated with increasing dependency of the display algorithms on user parameters.

## 1.3 Sequential freehand 3D ultrasound: a partial response

Recently, an alternative approach to freehand 3D ultrasound has been proposed, which removes the need for a voxel array [145]. In *sequential* freehand 3D ultrasound, the data is visualised and analysed directly from the original B-scans. This is represented schematically in Figure 1.3 — the sequential approach takes the right hand and more direct paths between the scan-plane representation and the display, bypassing the voxel-based representation completely. The main motivation for this approach is the desire to be as true to the original data as possible. In particular:

- When reslicing (displaying data on planes other than those in the original scan), the data is resampled only once, from the B-scan pixels to the slice pixels. The conventional approach requires two resampling stages — from the B-scan pixels to the voxel array, then from the voxel array to the slice pixels. Since resampling usually involves data approximation, more accurate visualisation is possible by avoiding one such process. For volume measurement and surface visualisation, the need for data interpolation is removed entirely.
- Reslicing can be performed at the full resolution of the B-scans, without the significant memory overhead of a high resolution voxel array.
- Visualisation and data analysis can be performed in real time<sup>10</sup>, as the sequential algorithms reference each B-scan only once, in the order of acquisition.

---

<sup>10</sup>This is only the case for sequential algorithms which do not require manual user interaction.



- Segmentation is performed on the B-scans, rather than on parallel slices through the voxel array. The B-scans are high resolution and exhibit no interpolation artifacts, making them easier to interpret than slices through the voxel array. They should also make more sense to the clinician, since these same images were displayed as the data was acquired. In addition, many of the artifacts in ultrasound images are directional, and although they can be understood in the original B-scans, become impossible to detect in slices with other orientations [35].

The advantage of integrity to the original data, however, brings with it the disadvantage of working with data from planes that are not parallel. In the voxel array approach, this is only an issue for the initial step of interpolation to this array; in the sequential framework, all algorithms have to be able to handle non-parallel image data. Nevertheless, any-plane slicing (reslicing), thick reslicing, non-planar slicing and panoramic imaging algorithms have all been successfully designed for the sequential framework [65, 145], as shown in Figure 1.3.

## 1.4 Original contribution

It is the subject of this thesis to design volume measurement and surface visualisation algorithms for this sequential framework, and to extend this framework to multiple-sweep data, so that these algorithms can also be applied to larger, or awkwardly shaped, organs. In doing so, both the challenges presented in Section 1.2, and the additional problems generated by the sequential framework itself, are addressed. Several of the responses to these challenges are common to all of the algorithms presented in this thesis:

- All of the algorithms have been designed with processing speed in mind, as this is such an important feature for clinical use. Potential solutions which would clearly take too long to calculate (i.e. more than a few minutes) are rejected on this basis.
- Everything presented in this thesis has been implemented in software. Volume measurement, surface visualisation, and the extension of both of these to multiple-sweep data, have all been added to Stradx [145], which is a tool for acquiring and processing sequential freehand 3D ultrasound data. This was written in conjunction with Richard Prager and Andrew Gee; an overview of the system is contained in Appendix A.2. Implementing all the algorithms in this way ensures that they are all practically realisable. It also allows others to use them: the software is freely available on the internet<sup>11</sup>, and is in use by other groups.
- The volume measurement and surface visualisation algorithms presented in this thesis have also been designed to operate successfully on sparse segmented data. This minimises the number of cross-sections that need to be manually segmented in order to provide accurate quantitative and visual results, which is of crucial importance, since manual segmentation remains the most time consuming task in the whole process.

### 1.4.1 Volume measurement in a sequential framework

*Cubic planimetry* is a new volume measurement technique that can estimate volume accurately, and quickly, from a small number of non-parallel segmented cross-sections. It is an extension of

<sup>11</sup><http://svr-www.eng.cam.ac.uk/~rwp/stradx/>

a technique proposed by Watanabe [182] for volume estimation using vector areas and centroids of serial cross-sections. The trapezoidal interpolation in this algorithm is the major source of inaccuracy. This causes volume overestimation for conical sections, and underestimation for spheres and ellipsoids (noted in practice [118]).

It was suggested in the original paper, and in [19], that an extension to cubic interpolation may be beneficial. This has been achieved in this thesis. In doing so, the original ability of the algorithm to handle highly non-parallel cross-sections has been preserved.

### 1.4.2 Surface visualisation in a sequential framework

The surface visualisation algorithm has been designed as a two-step process, where the cross-sections are first interpolated, then the surface of the interpolated data is triangulated. Framing the surface generation as an object interpolation problem allows a more fluid treatment of varying topologies. It also breaks the link between the position of the initial segmented cross-sections and the size and shape of the eventual geometric surface primitives, and hence the triangulation algorithm can be designed to produce high quality triangles.

*Disc-guided interpolation* is a new algorithm which addresses the first step of this process — that of surface interpolation. This algorithm is a significant extension of *shape-based interpolation*, first suggested by Raya and Udupa [154] and later adapted by Herman [82]. Shape-based interpolation is a good basis for surface interpolation, since it is both fast, and able to handle arbitrary topology. The important advances to this algorithm are:

- Shape-based interpolation is applied for the first time to arbitrarily oriented planes, so that it can be used on sequential freehand 3D ultrasound data.
- The technique is extended to improve the handling of complex shapes, by allowing the direction of interpolation to vary within the scan planes. The variation in direction is itself guided by the shape of the object cross-sections, as defined by a set of maximal discs [33]. This enables surfaces to be correctly interpolated from a small number of cross-sections.

*Regularised marching tetrahedra* generates a triangulated surface of the interpolated data, with very regular triangles that result in high quality display using standard graphics hardware. This new algorithm is the result of combining *marching tetrahedra* [38] with vertex clustering schemes [5, 155], to generate isosurfaces which are topologically consistent with the data, and contain a number of triangles appropriate to the sampling resolution (typically 70% fewer than marching tetrahedra), with significantly improved aspect ratios. The scheme is also adapted such that the tetrahedral grid upon which it is based can be locally aligned to the non-parallel planes that contain the interpolated surface data.

### 1.4.3 Application to multiple-sweep data

As has already been mentioned, image-based registration is a requirement if data from multiple sweeps is to be combined, which is necessary for the creation of a voxel array. However, the sequential technique operates on the original B-scans, and can examine each sweep one at a time, without using such an array. A new technique is presented which, rather than combining the *data* from each sweep, leaves it separate, such that subsequent operations (for instance reslicing, volume measurement and surface visualisation) are performed on each sweep, and the *results* of



these operations combined in such a way as to give a correct overall result. Combination at this stage is much faster than image-based registration of the original data.

Both volume measurement and surface visualisation are applied to multiple-sweep data using this framework. As far as I am aware, there is only one other published freehand 3D ultrasound system which uses data from multiple sweeps in order to calculate organ volume, without constructing a voxel array [80]. In this system, points at the edge of the organ are selected from the original B-scans, and a piecewise smooth subdivision surface is fitted to these points, subject to some ‘smoothness’ criterion. In contrast, rather than smoothing out inconsistencies in the data, the approach presented in this thesis is true to the aim of maintaining the integrity of the original data, in that mis-registration errors are clearly identifiable, but do not detract from the ‘real’ data. The sweep separation approach also allows the generation of multiple-sweep reslices, which is otherwise impossible.

## 1.5 Beyond ultrasound: application to other fields

There are few fields other than freehand 3D ultrasound in which data is measured in non-parallel planes. However, algorithms which are designed for such data can also be used in situations where the planes *are* parallel. Many of the advantages of the algorithms discussed in the previous section also carry through to this situation. In addition, *disc-guided interpolation* can be extended into four dimensions, such that surfaces are interpolated to form 3D morphing sequences<sup>12</sup>, rather than cross-sections interpolated to form surfaces. This leads to applications in a variety of other fields, including *medical imaging*, more generally, and *computer graphics*, both of which are considered in this thesis.

In the same way as with freehand 3D ultrasound, each of these applications has been implemented in software which is freely available on the internet<sup>13</sup>, and described in Appendix D.

### 1.5.1 Medical imaging

There are several other medical imaging modalities which can generate 3D data sets, for instance CT, MRI and PET. The visualisation of surfaces is just as important from these data sets as from freehand 3D ultrasound, and many of the challenges remain the same. For instance, the distance between image planes in CT is generally greater than the in-plane resolution, hence interpolation of cross-sections is required, even if the segmentation is performed by a simple thresholding operation on the entire data set.

In addition, the desire to generate surface triangulations containing a small number of regular triangles is equally strong for other types of medical data. Thresholding medical data can lead to highly intricate surfaces which are very difficult to display unless the triangulation is generated with a great deal of care. Examples of surfaces from CT and MRI data, which have been generated using disc-guided interpolation and regularised marching tetrahedra, are included in Chapter 3.

### 1.5.2 Computer graphics

Surface visualisation from a 3D data set can be useful even if the original data is not in this format. For instance, surfaces of mathematical functions, or implicit surfaces (i.e. surfaces

<sup>12</sup>i.e. the gradual change, or metamorphosis, of one surface into another

<sup>13</sup><http://svr-www.eng.cam.ac.uk/~gmt11/software/software.html>


defined as the zero level set of a sum of functions), can be displayed by sampling the functions on a regular grid, and using regularised marching tetrahedra to triangulate this sampled data. Examples of such surfaces are contained in Chapter 3.

The extension of disc-guided interpolation to interpolate surfaces, thus generating 3D morphing sequences, can be used to produce the visual effect of one surface ‘becoming’ another. It can also be useful for interpolating time-varying surfaces (for instance a sequence of models of a beating heart) to give better time resolution. 3D morphing has recently been used in place of image morphing, since it allows changes to the viewing location and lighting parameters during the morph sequence. The extension of disc-guided interpolation to sphere-guided interpolation is presented in Chapter 5, and found to be a useful tool for this purpose.

## 1.6 Outline of thesis

The remainder of this thesis is split into three major areas. In Chapters 2 and 3, novel algorithms are presented for volume measurement and surface visualisation from single-sweep freehand 3D ultrasound data. In Chapter 4, a framework is outlined for processing multiple-sweep data, and it is shown how both volume measurement and surface visualisation can be achieved through this framework. Chapter 5 demonstrates how the cross-section interpolation algorithm presented in Chapter 3 can be extended to interpolate surfaces, generating morphing sequences.

Each of Chapters 2 to 5 contain reviews of work related to these areas, a description of the algorithms themselves, and results of these algorithms applied to freehand 3D ultrasound, as well as other data. Finally, in Chapter 6, the performance of each of these algorithms is compared with the aims and objectives outlined in this chapter, and conclusions drawn.

Several of the figures in Chapters 3 and 5 have movie sequences associated with them. Where this is the case, a  symbol has been placed next to the figure legend. Clicking on these symbols in the PDF<sup>14</sup> version of this thesis (supplied on the CDROM described in Appendix E) will display a movie in QuickTime<sup>15</sup> format, provided a suitable plug-in is available. Alternatively the movies can be viewed directly from the CDROM, as described in Appendix E.

---

<sup>14</sup>Portable Document Format ©2000 Adobe Systems Incorporated

<sup>15</sup>©2000 Apple Computer Inc.

## Chapter 2

# Volume measurement

### 2.1 Current methods of ultrasound volume measurement

Organ volumes can be estimated using ultrasound from measurements of length or area, or from surface reconstructions. The first of these is the only method that is applicable to conventional 2D ultrasound; the remainder can only be used in conjunction with 3D ultrasound systems.

#### 2.1.1 Volume from length measurements

Volume measurements using conventional 2D ultrasound are in frequent clinical use today. These are achieved by approximating the organ of interest as a simple mathematical shape (generally an ellipsoid), and measuring the length of the major and minor axes in appropriately selected 2D ultrasound images. A correction is then applied to the result, dependent on the organ, the age and sex of the patient, and possibly additional factors. There are many formulations for the resulting equations [23, 87, 174].

Ellipsoid formulae are easy to use, but they have an inherent disadvantage in that they make geometrical assumptions about the shape of a given organ. This leads to errors in volume measurement, often exceeding 20%. These errors are greater where the organ under examination differs from the expected shape upon which the formulae were based. Unfortunately, these situations are often those in which the volume is of particular clinical importance.

#### 2.1.2 Volume from area measurements

An alternative approach that becomes possible with 3D ultrasound is *planimetry*, where the volume is calculated from a sequence of cross-sectional areas. The most common implementation of this technique is *step-section planimetry*, where it is assumed that the cross-sections are parallel. There are numerous reports which indicate that step-section planimetry is much more accurate than ellipsoid or other geometrical formulae [1, 129, 149, 175]. The only exception to this is [174], where step-section planimetry is compared with sixteen equations for measuring prostatic volume and it is found that using  $\frac{\pi}{6}(\text{transverse diameter})^2(\text{anteroposterior diameter})$  is marginally more accurate than planimetry; however, this result is not applied to an independent data set.

Watanabe developed a planimetric method for calculating volumes from cross-sections that were not parallel [182], and could even be overlapping. This has rarely been used in practice, although it has been implemented for measuring the volume of the prostate [19].

### 2.1.3 Volume from surfaces

If the surface of an object has been reconstructed, the volume can be calculated from this surface representation. There are several ways of doing this:

**Volumetry** If the object of interest has been sampled to a regular voxel array, the volume can be calculated simply by summing the number of voxels inside the object, and multiplying by the volume of one voxel. This is by far the most common technique in use with 3D ultrasound systems — even when the acquisition is performed using freehand scanning; in which case volumetry is applied to the reconstructed voxel array.

**Tetrahedral Volume** If the whole object has been reconstructed as a solid formed of tetrahedra, the volume can be calculated from the sum of the volumes of these tetrahedra, the volume of a tetrahedron being  $\frac{1}{6}\vec{pa} \cdot \vec{pb} \times \vec{pc}$ , where the points  $p$ ,  $a$ ,  $b$  and  $c$  are its vertices. Alternatively, the polyhedral approximation formula developed by Cook [45] can be used. This is based on the above equation, but formulated in terms of the points making up each of the object cross-sections. Although this appears to allow volume calculation from cross-sections without triangulation, in fact a simple triangulation is assumed in the algorithm which will only be correct for simple shapes. This technique is used for instance in [7, 71, 101].

**Cylindrical/Pyramidal Volume** Tetrahedra are not the only simple mathematical shapes that can be used to estimate volume. If the scanning pattern is rotational, parts of cross-sections can be connected with the mid-point of the rotation to form pyramidal or cylindrical part sections, from which the volume can be calculated. This technique has been used for the eye [91]. Moritz also applied this technique to freehand scans by re-sampling these scans in a rotational scanning pattern, and then calculating the volume from the new cross-sections [125].

**Volume from Surface** Hughes has suggested two ways of measuring the volume directly from a triangulated surface, without forming tetrahedra. ‘Ray Tracing’ involves projecting rays from a 2D grid through the object, and calculating the intersection points of these rays with the object. The volume can then be deduced from the length of each ray inside the object, and the granularity of the 2D grid [89]. Alternatively, a discrete version of Gauss’ theorem can be adapted to calculate the volume component for each individual triangle, such that the sum of these components is equivalent to the object volume [90]. An implementation of this algorithm is contained in Appendix B.2.

Comparisons of some of the various volume measurement techniques [90], and many *in vitro* measurements using 3D ultrasound systems [21, 53, 86] have been performed. It is clear from these that all of the 3D ultrasound methods are accurate to approximately  $\pm 5\%$ , and are to be preferred over the ellipsoid or similar equations [42, 71]. The most robust method for working with non-parallel, sequential cross-sections is the planimetric method proposed by Watanabe [182], and this forms the basis of the algorithm presented in the next section.

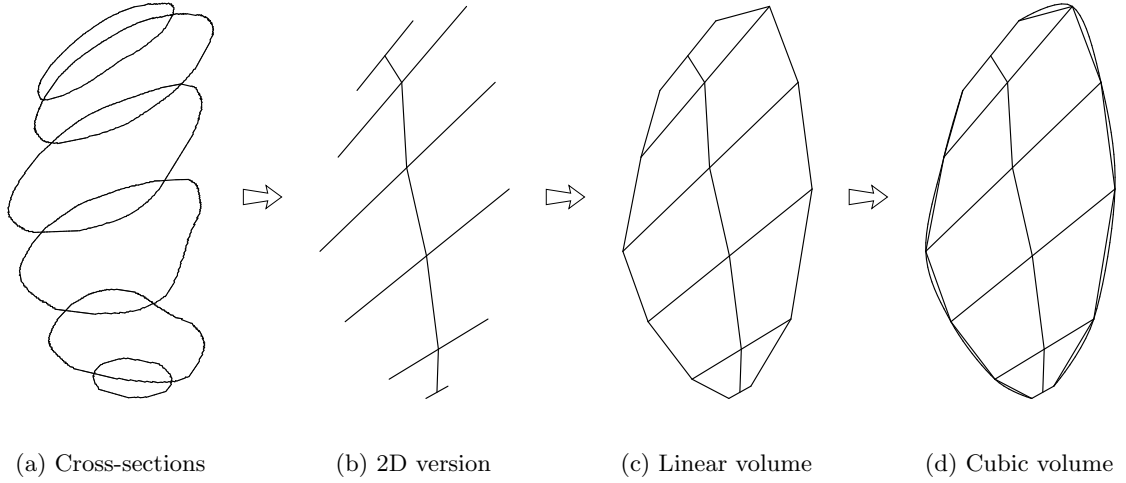


Figure 2.1: **Overview of volume measurement process.** The cross-sections in (a) are converted to a 2D representation in (b), whose total enclosed area is equal to the volume of the cross-sections as given by equation (2.2), and shown in (c). A more accurate volume is given by calculating the area inside a cubic interpolation of the 2D representation, as in (d).

## 2.2 A new approach: cubic planimetry

### 2.2.1 Overview

An overview of the algorithm described in the following sections is given in Figure 2.1. In this illustration, a kidney has been scanned, and six cross-sections segmented in the original non-parallel B-scans. Rather than use the planimetric method directly on these cross-sections, they are reduced to a 2D representation, where the area of each cross-section is represented by the length of each line, and the orientation of the lines is related to the orientation of the cross-sections. In this representation, the enclosed area is equivalent to the volume of the real object. Using a cubic (rather than a linear) interpolant improves the accuracy with which the enclosed area (and hence volume) can be estimated.

### 2.2.2 Dealing with non-parallel cross-sections

The equation for the volume  $v$  of any object defined from non-parallel sequential cross-sections, is given by [182]:

$$v = \left| \int_L \vec{s} \cdot d\vec{\omega} \right| \quad (2.1)$$

where  $\vec{\omega}$  is the position vector of the centroid of the object cross-section, whose vector area (i.e. the vector normal to the plane with magnitude equal to the cross-sectional area) is given by  $\vec{s}$ .  $L$  is the path of  $\vec{\omega}$  as the object is scanned. Equation (2.1) gives the correct volume even if the cross-sections overlap. The only restriction is that the planes containing the first and last cross-sections must be oriented in the same sense with respect to the object, i.e. the object is in front of the first plane and behind the last plane, or *vice versa*. In practice, rotational scanning is the only common scanning pattern which violates this condition.

Equation (2.1) can be implemented discretely by approximating the integral using the trapezoidal rule between each pair of cross-sections, which gives:

$$v = \left| \sum_{i=2}^N \frac{1}{2} (\vec{s}_i + \vec{s}_{i-1}) \cdot (\vec{\omega}_i - \vec{\omega}_{i-1}) \right| \quad (2.2)$$

where the  $N$  cross-sections have vector areas  $\vec{s}_1, \dots, \vec{s}_N$  and centroids  $\vec{\omega}_1, \dots, \vec{\omega}_N$ . This approximation is equivalent to assuming that the surface area projected onto a plane normal to the path of the centroids,  $L$ , varies linearly from one slice to the next. This is true for objects whose cross-sectional area does not vary, and also for paraboloids, for which equation (2.2) is the exact solution of equation (2.1). However, objects which are either more concave or more convex than a paraboloid will not be correctly approximated by this equation. For example, the volume of a cone will be overestimated, and that of a sphere or an ellipsoid will be underestimated. This error increases as the number of cross-sections is reduced.

Equation (2.2) can easily be implemented on a computer once the cross-sections have been generated. In practice, this first segmentation step is by far the most time consuming, typically taking around half a minute for each cross-section, assuming it is outlined manually. Once this has been done, the calculation of volume can be achieved in less than one second.

Using cubic rather than trapezoidal interpolation would increase the accuracy of the volume estimate and eliminate the bias towards paraboloids or prisms. It has been argued [19] that the small increase in accuracy this would provide does not justify the additional complexity. However, two points can be made in defence of this approach. Firstly, the additional complexity is completely transparent to the user — once the algorithm has been implemented, the user performs precisely the same operations (i.e. segmenting the cross-sections) in both cases. Secondly, the reduction in the number of cross-sections required for an accurate volume estimation is very welcome, as segmentation is the time consuming step in the process.

In order to estimate equation (2.1) using cubic interpolation, two important decisions must be made. The first is the selection of the points or vectors to interpolate. The second is the selection of the cubic representation for the interpolation.

### 2.2.3 A 2D representation of the problem

Interestingly, the whole problem can be reduced to finding the area of a carefully constructed 2D graph, that represents a combination of the original 3D object with the scanning pattern. The equivalence between the 3D and 2D representations is shown in Figure 2.2.

The area enclosed within the dashed and solid lines in the 2D representation is equivalent to the volume that would be calculated by Watanabe's trapezoidal equation from the 3D representation. This can be easily proved by considering the 2D representation to have a nominal thickness of one unit, and then applying equation (2.2) to calculate the area:

$$\begin{aligned} A &= \left| \sum_{i=2}^N \frac{1}{2} (a_i \hat{n}_i + a_{i-1} \hat{n}_{i-1}) \cdot (\vec{c}_{i-1}) \right| \\ &= \left| \sum_{i=2}^N \frac{|c_{i-1}|}{2} (a_i \hat{n}_i \cdot \hat{c}_{i-1} + a_{i-1} \hat{n}_{i-1} \cdot \hat{c}_{i-1}) \right| \\ &= \left| \sum_{i=2}^N \frac{|c_{i-1}|}{2} (a_i \cos \beta_i + a_{i-1} \cos \alpha_i) \right| \end{aligned} \quad (2.3)$$

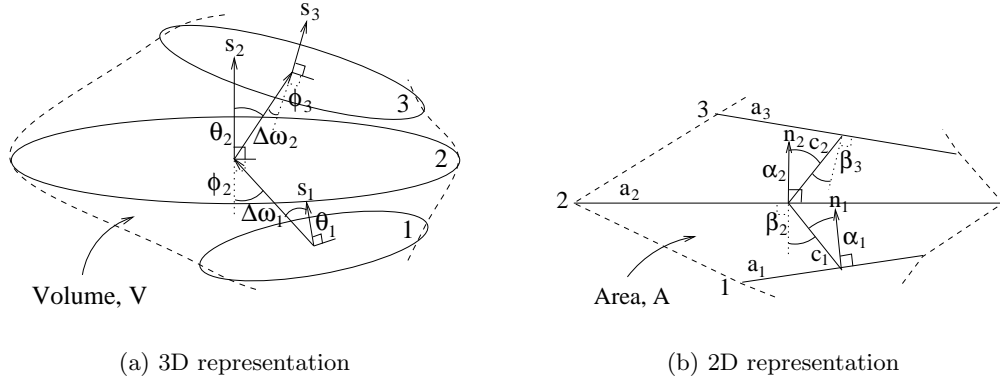


Figure 2.2: **3D and 2D representation equivalence.** i) The length of a 2D line,  $a$ , is equivalent to the area of the cross-section,  $|\vec{s}|$ . ii) The length of the line,  $c$ , is equal to the magnitude of the vector,  $|\Delta\vec{\omega}|$ . iii) The angle,  $\alpha$ , between the line  $c$  joining the centres of each line  $a$  and the normal to those lines is equal to the angle,  $\theta$ , between the vector area  $\vec{s}$  and the vector  $\Delta\vec{\omega}$  joining the centroids of each area. iv) Similarly, the angle,  $\beta$ , is equal to the angle,  $\phi$ .

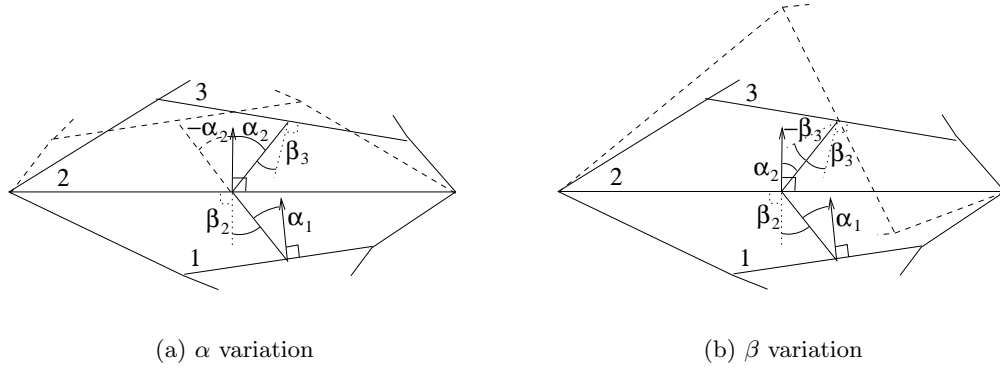


Figure 2.3: **Choice of angles for 2D representation.**

Equation (2.2) can be similarly re-written as:

$$v = \left| \sum_{i=2}^N \frac{|\Delta\vec{\omega}_{i-1}|}{2} (|\vec{s}_i| \cos \phi_i + |\vec{s}_{i-1}| \cos \theta_i) \right| \quad (2.4)$$

If the variables in equations (2.3) and (2.4) are equated for all values of  $i$ , then  $A \equiv v$ . There is, however, significant redundancy in this conversion. Firstly, only the multiple of the lengths of the lines  $a_i$  and  $c_i$  is used, and hence an arbitrary scale factor can be multiplied into one, so long as it also divides the other. This has the effect of stretching or shrinking the 2D graph, but has no bearing on the volume calculation. Secondly, only the cosine of the angles  $\alpha_i$  and  $\beta_i$  are used, hence they can be arbitrarily positive or negative. The effect of this choice is demonstrated in Figure 2.3.

Although the choice of angles has no effect on the volume calculated by the trapezoidal method, it does affect that calculated using cubic interpolation of the 2D representation, as the

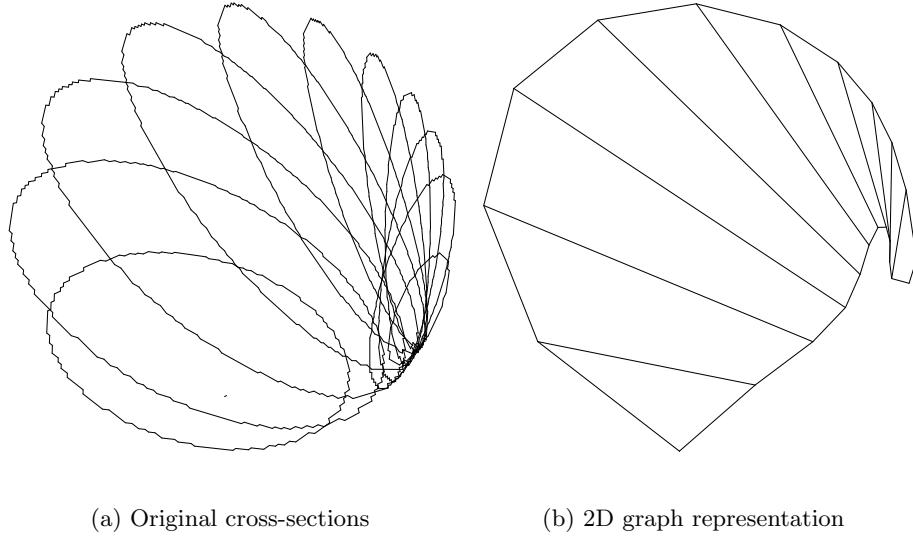


Figure 2.4: **Scanned sphere in 3D and 2D representations.**

latter involves the use of information from several sequential slices. An additional heuristic rule is required to ensure that the angles  $\alpha_i$  and  $\beta_i$  are chosen correctly.

For each choice of angle  $\alpha_i$  between the line joining centroids  $c_i$  and the area representation  $a_i$ , the angle which  $c_i$  makes with  $c_{i-1}$  is also calculated. The value of  $\alpha_i$  is then chosen for which this calculated angle is closest to the 3D version (i.e. the angle which  $\Delta\omega_i$  makes with  $\Delta\omega_{i-1}$ ). A similar rule is employed for the angle  $\beta_i$ , using the area normals rather than the lines joining the centroids as the reference.

The result of this entire process is shown for a sphere in Figure 2.4. The sphere was scanned with a pattern which varied in position, azimuth, elevation and roll, using the simulation software described in Appendix A.1. The resulting 2D graph retains some of the shape of the sphere but also reflects the way in which it was scanned.

### 2.2.4 Cubic interpolation of the 2D representation

If, instead of joining the end points of the lines  $a_i$  in Figure 2.2(b) with straight lines, a set of smooth curves is fitted between them, then the area enclosed by these curves should represent the volume of the original object more accurately. The curves must at least be cubic, since continuity is required in at least the first derivative (i.e. the curves are  $C_1$  continuous). They must also be defined parametrically, since they may have multiple values in both  $x$  and  $y$  directions.

The smoothest possible curve could be obtained by fitting an appropriate function through all the end-points simultaneously. However, this sort of global optimisation is in general costly to compute, which violates one of the motivations for improving the volume calculation: that the increase in processing time is negligible. A less optimal but faster solution can be found by using parametric cubic splines.

In depth descriptions of the various forms of splines can be found, for example, in [60]. In summary, they are curves which in 2D are represented parametrically by sets of coefficients:



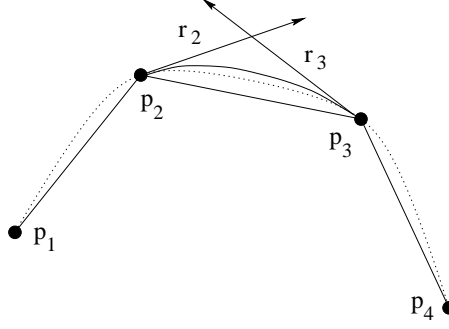


Figure 2.5: Selection of points for cubic splines.

$$\begin{aligned}
 \begin{bmatrix} x(t) & y(t) \end{bmatrix} &= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} x_3 & y_3 \\ x_2 & y_2 \\ x_1 & y_1 \\ x_0 & y_0 \end{bmatrix} \\
 &\equiv \mathbf{TC} \\
 &\equiv \mathbf{TMP}
 \end{aligned} \tag{2.5}$$

where  $\mathbf{C}$  is a matrix of coefficients, and  $\mathbf{T}$  is the parameter matrix. The matrix  $\mathbf{C}$  can be generated from a list of control points  $\mathbf{P}$  and the spline transformation matrix  $\mathbf{M}$ . The choice of the control point and transformation matrices determines how the cubic spline will be defined. In particular, a spline is required that *interpolates* the control points, i.e. the resulting curve passes through the points which are used to define it. One of the simplest cubic spline forms possessing this property is the Hermite, in which case:

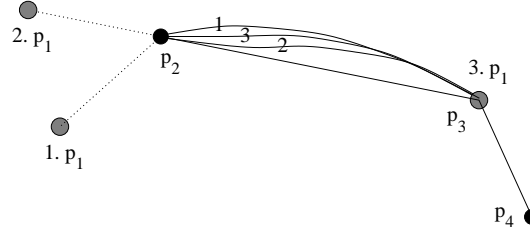
$$\mathbf{P} = \begin{bmatrix} \vec{p}_2 \\ \vec{p}_3 \\ \vec{r}_2 \\ \vec{r}_3 \end{bmatrix}, \quad \text{and} \quad \mathbf{M} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.6}$$

where  $\vec{p}_2$  and  $\vec{p}_3$  are the position vectors of the points  $p_2$  and  $p_3$ , and the gradient vectors  $\vec{r}_2$  and  $\vec{r}_3$  are defined in Figure 2.5. The choice of gradient vectors is not obvious in this case, as we only have a set of connected points. An alternative cubic spline form invented by Catmull and Rom [37] overcomes this by using the four position vectors  $\vec{p}_1$  to  $\vec{p}_4$  directly. In this case, the point and transformation matrices are:

$$\mathbf{P} = \begin{bmatrix} \vec{p}_1 \\ \vec{p}_2 \\ \vec{p}_3 \\ \vec{p}_4 \end{bmatrix}, \quad \text{and} \quad \mathbf{M} = \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \tag{2.7}$$

With this form of cubic spline, the control points  $p_1$  to  $p_4$  are the points to be interpolated, and the curve is fitted between  $p_2$  and  $p_3$ , as in Figures 2.5 and 2.6. This is equivalent to using the Hermite form, with the gradient vectors defined as:

$$\vec{r}_2 = \frac{1}{2} (\vec{p}_3 - \vec{p}_1), \quad \text{and} \quad \vec{r}_3 = \frac{1}{2} (\vec{p}_4 - \vec{p}_2), \tag{2.8}$$

Figure 2.6: **Handling of end points for cubic splines.**

The first and last curve segments are necessarily a special case, as only three points can be used to fit the curve. There are a variety of ways of handling this, that can all be implemented by inventing the additional missing point  $p_1$ , as in Figure 2.6. There are three main options for the position of this point:

1. If  $p_1$  is chosen such that the point set  $p_1, \dots, p_4$  has reflectional symmetry about the mid-point of  $p_2p_3$ , then the final curve will also be symmetrical, i.e. the gradients at each end will be the mirror image of each other.
2. If  $p_1$  lies along the extension of the line  $p_3p_2$ , then the gradient at the last point will be equal to that of the line  $p_3p_2$ .
3. If  $p_1$  is identical to  $p_3$ , this has the effect of placing a null gradient at the end point. The resulting curve has a rate of change of gradient at the end point of zero.

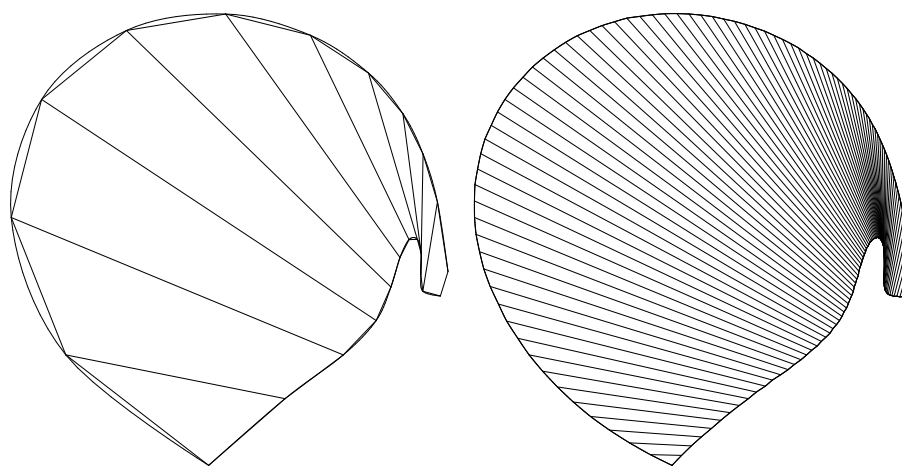
The last solution is the simplest and most natural, since there is no knowledge of the gradient at  $p_2$ , and it is therefore employed in this technique. The resulting curve, for the same situation as Figure 2.4, is shown in Figure 2.7, together with the actual object curve which results from scanning in smaller steps.

Once the curves joining the end points of the lines  $a_i$  have been defined, the enclosed area can be calculated directly from the parametric coefficients of each curve. This calculation is based on the application of equation (2.1), and is given in Appendix B.1.

Rather than fitting curves to the end points of the lines  $a_i$ , they can be fitted directly to the vector normals to these lines  $\hat{n}_i$  and the centroid difference vectors  $\vec{c}_i$ . The result of this, overlaid on the former interpolation technique, is shown in Figure 2.8(a). In fact, the resulting curves can be deduced from each other, and the volume calculation is identical for each method.

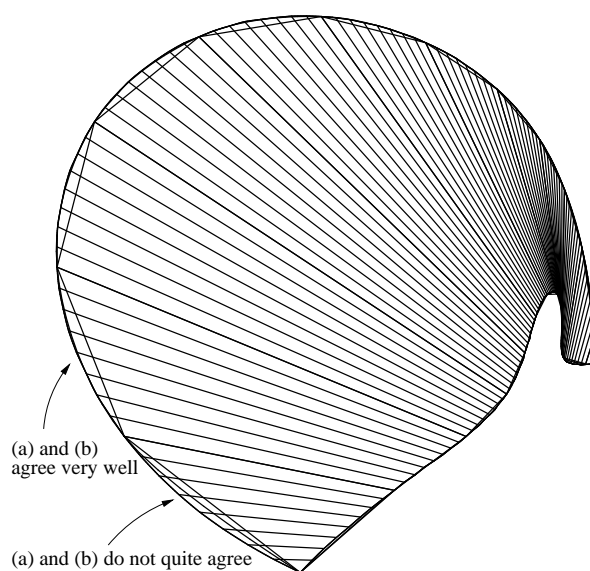
Alternatively, the cubic splines can be fitted directly to the original 3D vector areas  $\vec{s}_i$ , *in situ*. In this case the 2D representation is not required. This 3D representation is hard to visualise, but is shown in Figure 2.8(b) superimposed on the original 3D cross-sections. The vector areas  $\vec{s}_i$  emanate from the centres of the original cross-sections, and the two curves join the end points of these vectors.

In this case, the curve parameters are now defined in 3D; hence the calculation of enclosed area in Appendix B.1 requires an additional  $z(t)$  coordinate. The resulting volume is slightly different from that obtained using the 2D graph method, although the implementation is simpler since the translation from 3D to 2D representation is no longer required. However, the problem is less well conditioned, as there are more parameters to estimate than in the 2D case. The accuracy of the various techniques is investigated in Section 2.3.



(a) Cubic and trapezoidal combined

(b) Greater number of cross-sections



(c) Previous figures superimposed

Figure 2.7: **Cubic and trapezoidal interpolations for a scanned sphere.** Cubic interpolation of a few cross-sections in (a) is compared with trapezoidal interpolation of a greater number of cross-sections in (b). The slight difference between these measures can be seen in (c).

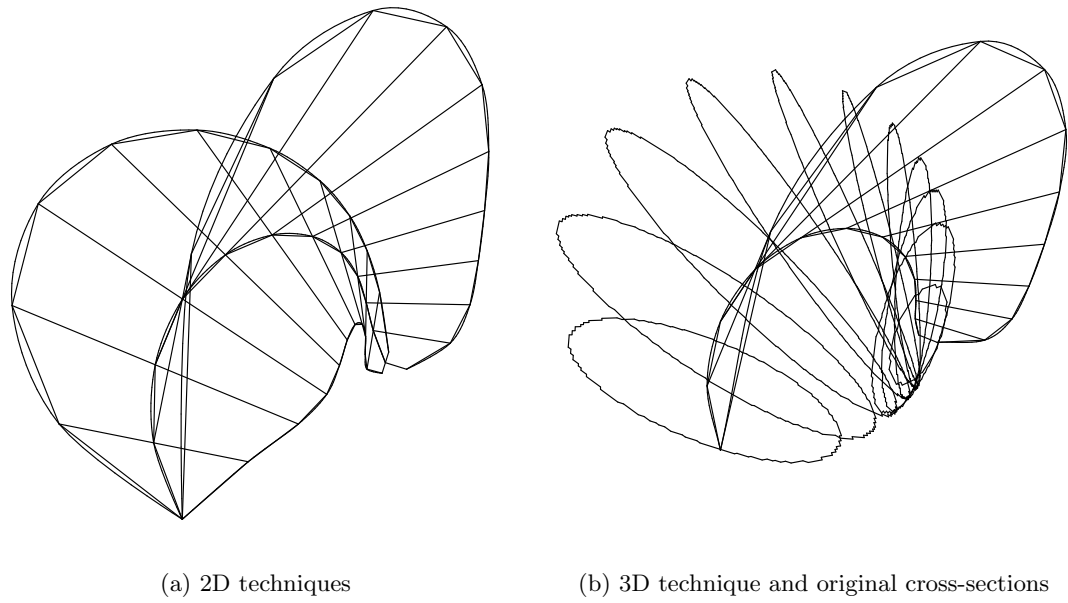


Figure 2.8: **Alternative cubic interpolation techniques applied to a sphere.**

## 2.3 Simulated scanning results

### 2.3.1 Test setup

In order to verify the volume measurement technique presented in this chapter, and the surface visualisation technique presented in Chapter 3, simulation software was developed to allow geometrical objects of known volume and shape to be ‘scanned’. This software is described in Appendix A.1.

Testing the algorithms in this way allowed more control of the sources of error than would be possible with *in vitro* experiments. In particular, segmentation error, which is common to all ultrasound volume measurement techniques, could be eliminated entirely. Both the effects of the clinician moving the probe unsteadily, and errors in the position measurement, could also be controlled.

The location of the first and last scan planes also has a significant effect on the volume measurement, for any sequential technique. This is because any part of the object which lies beyond these planes is not included in the volume measurement or surface estimation process. Hence, in the following experiments, the first and last planes were always positioned at the extremities of the object.

Six different techniques were tested: the three volume measurement techniques discussed in this chapter, and three surface reconstruction techniques discussed in Chapter 3. Volume measurements from surface reconstructions are presented for comparative purposes. These are generally worse than the planimetric volume measures described in this chapter, since the surfaces are reconstructed using linear, rather than cubic, interpolation. Surface reconstructions from the cross-sections are also shown here — these are discussed in Section 3.5.1.

**1. Linear planimetry** This is the original method of volume measurement using vector areas and centroids presented in [182].

- 2. 2D cubic planimetry** The volume measurement technique presented in this chapter using the 2D representation.
- 3. 3D cubic planimetry** The volume measurement technique presented in this chapter which uses cubic interpolation directly in 3D.
- 4. Shape-based interpolation** The surface interpolation method on which the technique presented in Chapter 3 is based [82, 154].
- 5. Centroid-guided interpolation** An extension to the above in [83], and similar to [114].
- 6. Disc-guided interpolation** The surface interpolation technique presented in Chapter 3.

### 2.3.2 Volume measurement accuracy

Each test object (outlined in Table 2.1) was scanned with linear, fan or freehand sweeps. The number of scans in each sweep varied from 4 to 20, to show the change in volume measurement as the number of segmented cross-sections increased. The results of this are shown in Figures 2.9 to 2.12. The solid horizontal lines on the graphs show the actual volume of the object, and a margin of  $\pm 1\%$  around this value. Object contours and surface reconstructions in Figures 2.9 to 2.12 are shown with the smallest number of scans for which the cubic planimetry volume was within this margin. Table 2.1 contains this number of scans for each object, and the percentage error for 20 scans.

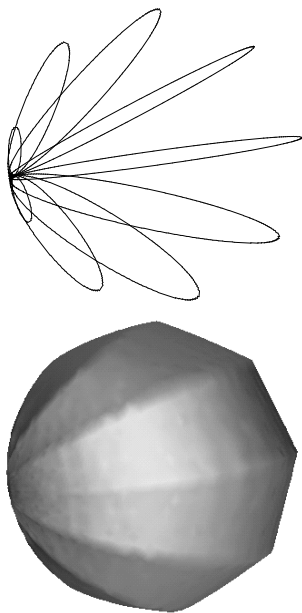
The size of each pixel in the scan planes was 0.012cm. Objects were scaled to just fit inside a cube of  $2\text{cm} \times 2\text{cm} \times 2\text{cm}$ , hence length measurements had an error of approximately  $\pm 0.3\%$ . Surface reconstructions, using disc-guided interpolation, were performed at half this resolution, giving volume errors of  $3 \times 2 \times 0.3 = \pm 1.8\%$ . Planimetry is inherently less dependent on resolution, being based on area measurements, leading to volume errors of  $\pm 0.6\%$ .

The time taken for each measurement was approximately 20ms for cubic planimetry methods, and approximately 7s for surface reconstructions, the latter being relatively independent of the number of scans.

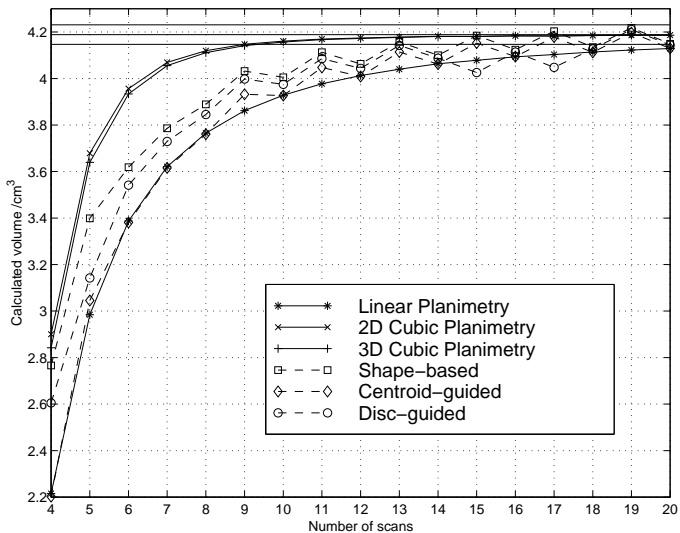
Cubic planimetry gave the most accurate volume measurements with fewer segmented cross-sections, in all cases. Table 2.1 shows that typically only ten cross-sections were required to give an accuracy (due to the volume measurement technique alone) of better than  $\pm 1\%$ . This was the case for both linear and complex scanning patterns, and simple or complex objects; even the sharp-edged cube in Figure 2.9.

The results for 2D cubic planimetry were generally very similar to those for 3D cubic planimetry. The 2D technique was perhaps slightly better, due mostly to the more natural fitting of cubic splines to the end points than was the case with the 3D technique.

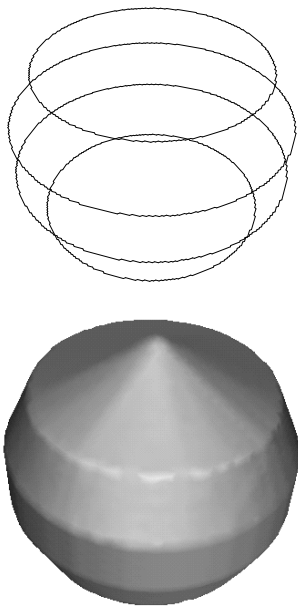
For parallel or nearly parallel scanning patterns, linear planimetry gave more accurate volume measurements than any of the surface reconstruction methods, for example the cone in Figure 2.11(a) and (b) and “baseball glove” shape in Figure 2.11(c) and (d). However, as the scanning patterns increased in complexity, this method became less accurate, and the surface reconstruction methods tended to be more accurate. See, for example, the sphere in Figure 2.9(b) (the apparent oscillations in this and Figure 2.12(e) are in fact due to how the surface is intersected for odd and even numbers of scans). Linear planimetry assumes a linear change in area between scan planes, whereas the surface reconstruction methods fit an approximately linear surface — which of these gives the better volume measurement is therefore a function of the relative accuracy of these assumptions.



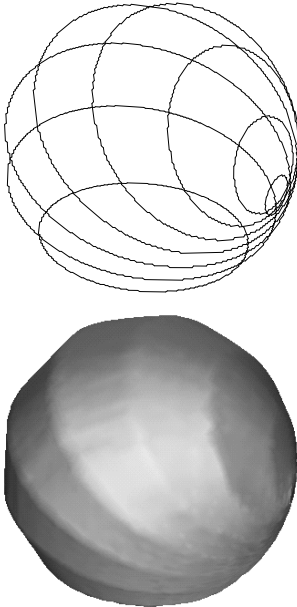
(a) Sphere, fan scan



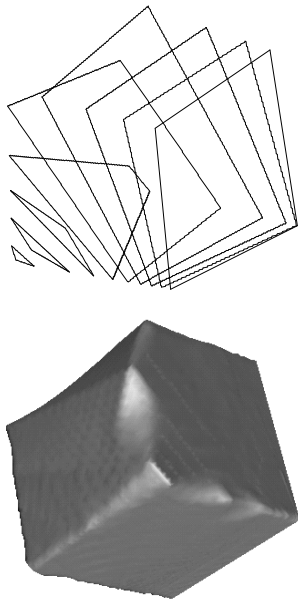
(b) Sphere, fan scan



(c) Sphere, linear scan



(d) Sphere, free scan



(e) Cube, free scan

Figure 2.9: Simulated scan of a sphere and cube.

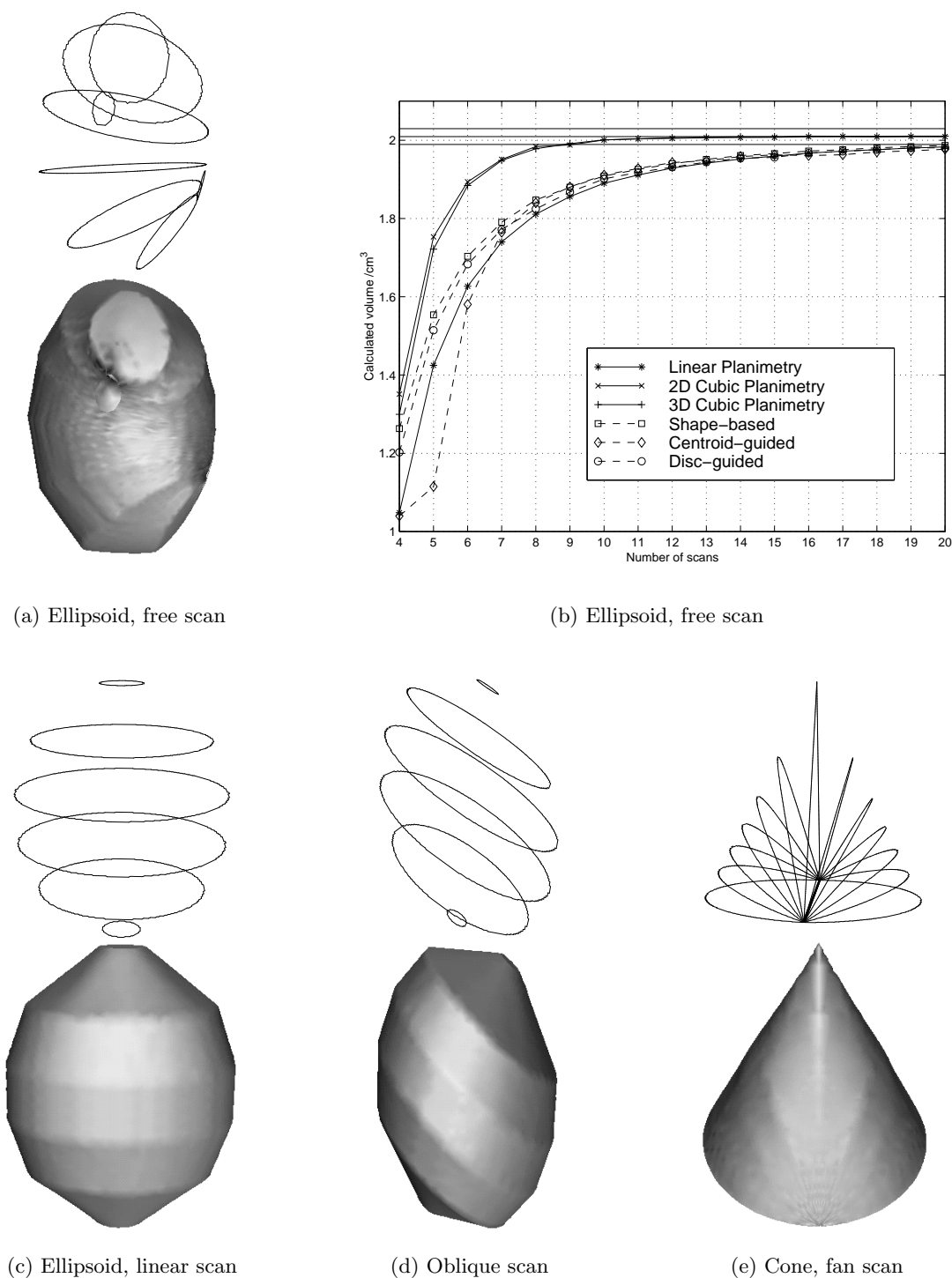
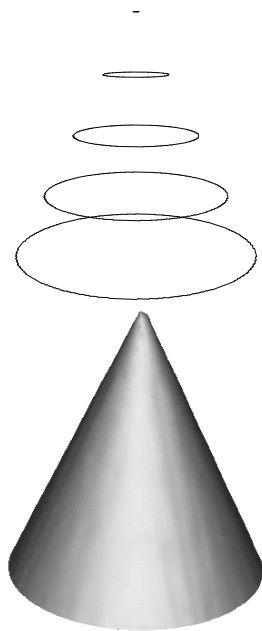
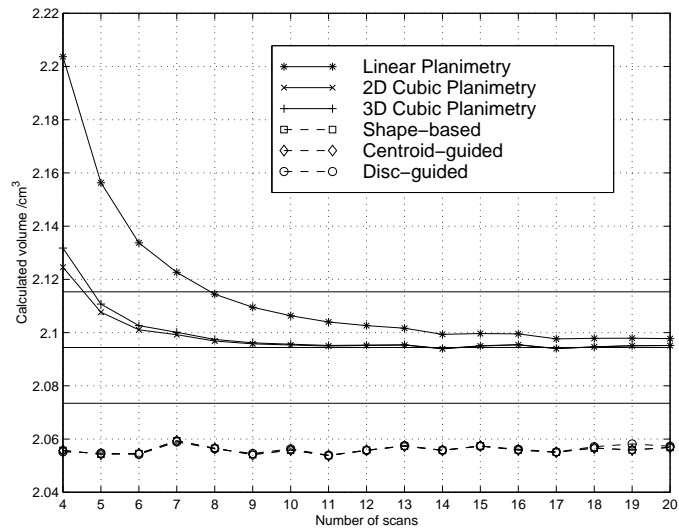


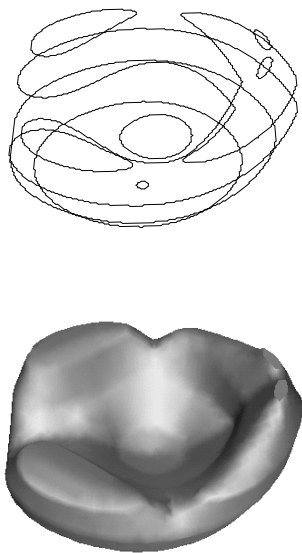
Figure 2.10: Simulated scan of an ellipsoid and cone.



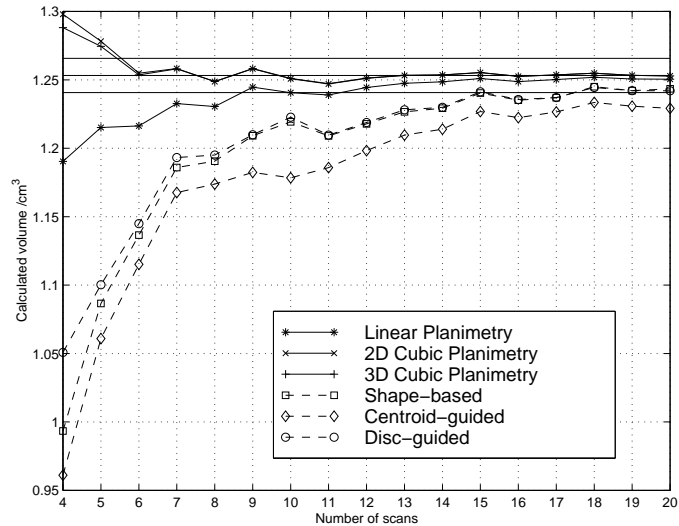
(a) Cone, linear scan



(b) Cone, linear scan



(c) Glove, linear scan



(d) Glove, linear scan

Figure 2.11: Simulated scan of a cone and “baseball glove” shape.



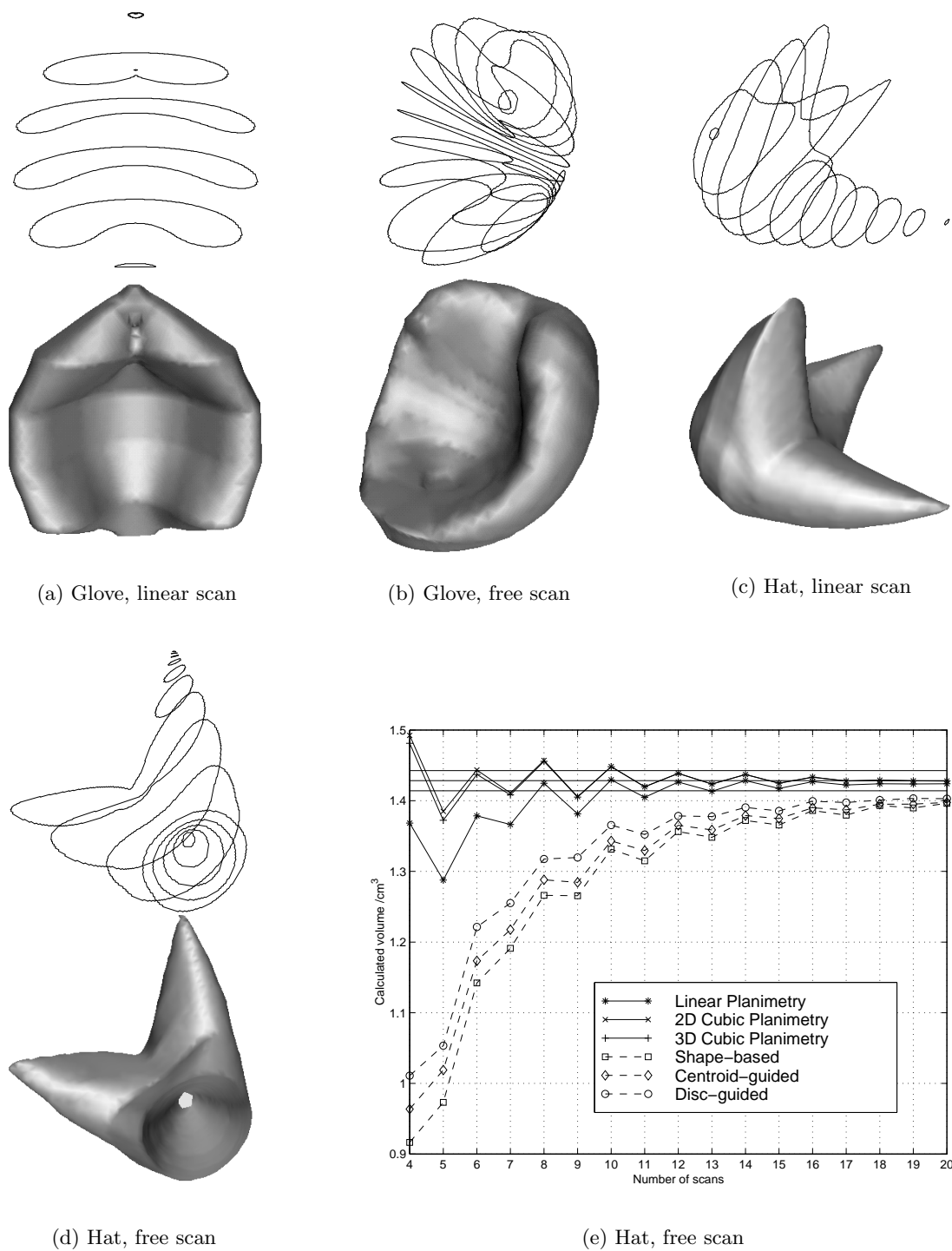


Figure 2.12: Simulated scan of a “baseball glove” and “jester’s hat” shape.

Table 2.1: **Simulation results: all objects.** **1:** Linear planimetry, **2:** 2D cubic planimetry, **3:** 3D cubic planimetry, **4:** Shape-based, **5:** Centroid-guided, **6:** Disc-guided. Missing values imply  $> 20$  scans. The values for the cone fan scan were the result of the symmetry of the situation — odd numbers of sweeps gave more accurate volumes than even.

Object	Scan	Planes for $< 1\%$ error						% error at 20 planes					
		1	2	3	4	5	6	1	2	3	4	5	6
Sphere	linear	11	6	6	20	20	20	-0.2	+0.1	+0.1	-0.8	-0.8	-0.8
	fan	-	9	10	19	19	19	-1.4	-0.1	-0.1	-0.9	-1.4	-1.1
	free	20	9	9	15	16	16	-0.9	-0.1	-0.1	+0.0	-0.1	-0.1
Ellipsoid	linear	11	6	6	15	15	15	-0.3	-0.0	-0.0	-0.5	-0.5	-0.5
	oblique	11	6	6	17	17	17	-0.2	+0.1	+0.1	-0.6	-0.6	-0.6
	free	-	9	9	-	-	-	-1.3	-0.0	+0.1	-1.1	-1.6	-1.4
Cone	linear	8	5	5	-	-	-	+0.2	+0.0	+0.0	-1.8	-1.8	-1.8
	fan	-	9	9	17	17	17	-1.8	-1.3	-1.3	-1.8	-1.6	-1.7
Cube	free	12	8	8	14	14	12	-0.5	-0.0	-0.0	-0.5	-0.5	-0.3
Glove	linear	12	6	6	15	15	15	-0.3	+0.1	+0.1	-1.2	-1.0	-0.9
	linear	9	6	6	15	-	15	-0.2	-0.0	-0.0	-0.8	-1.9	-0.9
	free	-	16	15	-	-	-	-2.4	-0.6	-0.3	-1.5	-1.5	-1.5
Hat	linear	10	9	9	-	-	-	-0.3	-0.0	-0.0	-2.2	-2.2	-1.8
	free	-	14	14	-	-	18	-1.9	-0.0	-0.0	-1.4	-1.9	+0.1

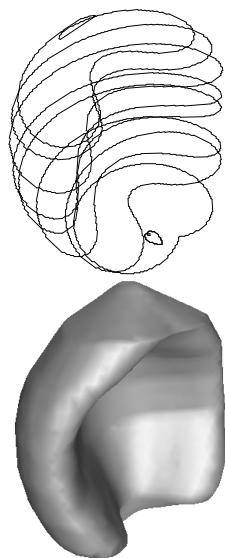
### 2.3.3 Registration and unsteady hand effects

In Figures 2.9 to 2.12, the scan planes were evenly spaced and their location and orientation known precisely. In practice, this is rarely the case, even with good calibration. In order to examine the effects of these errors, the “baseball glove” shape was scanned with ten linear scans, in the same pattern as in Figure 2.12(a). Random errors were applied to the location of the scans, first before scanning the object (unsteady hand simulation), then after scanning the object (mis-registration simulation). Errors were added to all three location and all three orientation parameters simultaneously. Five scales of errors were used in each case, corresponding to the numbers on the x-axes of the graphs in Figures 2.13 and 2.14, as follows:

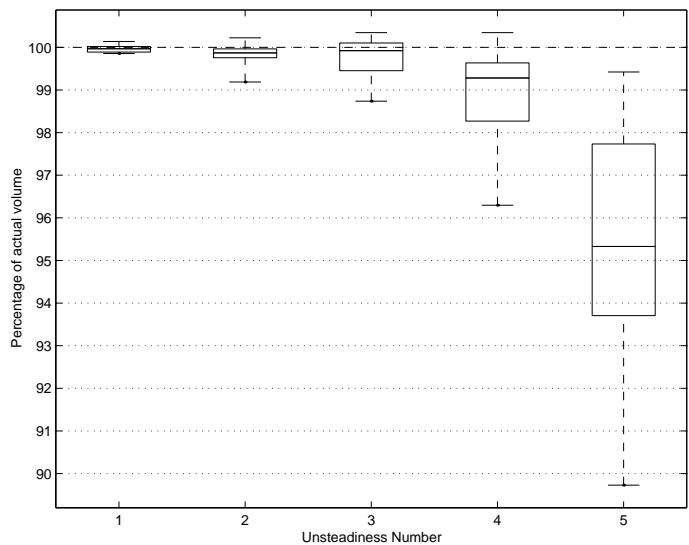
1.  $\pm 0.02\text{cm}$  in location,  $\pm 1^\circ$  in orientation.
2.  $\pm 0.05\text{cm}$  in location,  $\pm 2.5^\circ$  in orientation.
3.  $\pm 0.1\text{cm}$  in location,  $\pm 5^\circ$  in orientation.
4.  $\pm 0.2\text{cm}$  in location,  $\pm 10^\circ$  in orientation.
5.  $\pm 0.4\text{cm}$  in location,  $\pm 20^\circ$  in orientation.

The object dimensions were approximately  $2\text{cm} \times 2\text{cm} \times 2\text{cm}$ , hence these represent a wide range of errors. 2D cubic planimetry volumes were measured for 20 different samples at each scale of error. The graphs show the quartiles (rectangular boxes) and minimum and maximum values (dotted vertical lines) for each of these distributions.

The effect of the clinician holding the probe with an unsteady hand is very small for all of the methods examined — volume errors remain at about  $\pm 1\%$  for unsteadiness of up to  $5\%$  of the object size. This slight reduction in both volume and accuracy, seen in Figure 2.13, is the

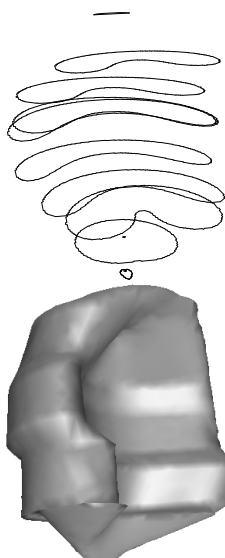


(a) Sections and surface

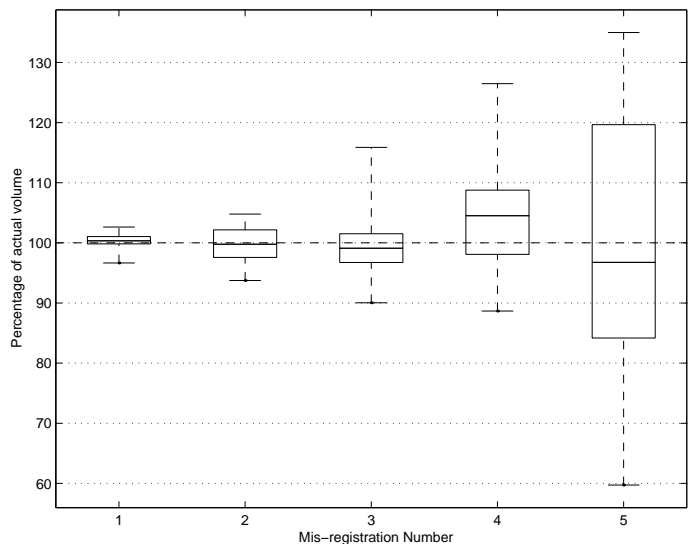


(b) Distribution of cubic planimetry volumes

Figure 2.13: **Simulated unsteady sweeps of a “baseball glove” shape.** Rectangles show quartile values, dotted lines show minima and maxima.



(a) Sections and surface



(b) Distribution of cubic planimetry volumes

Figure 2.14: **Simulated mis-registered sweeps of a “baseball glove” shape.** Rectangles show quartile values, dotted lines show minima and maxima.

result of the larger gaps between object cross-sections rather than the irregularity of the scan positions.

As expected, mis-registration of the scan planes (i.e. errors in the position and orientation information) is a more significant source of error. In this case, volume errors are about  $\pm 10\%$  for mis-registration errors of up to 5% of the object size. In practice, the magnetic position sensor in use with the freehand 3D ultrasound system described in Appendix A.2, is accurate to typically  $\pm 1\text{mm}$  [146]. This implies a worst case mis-registration of approximately 2% for a small organ<sup>1</sup>. At this level of mis-registration, the volume measurement error shown in Figure 2.14 is at about the same scale as the mis-registration error.

## 2.4 *In vivo* ultrasound results

*In vivo* measurements have the disadvantage of including both registration and segmentation errors, but do therefore represent the volume measurement accuracy that is achievable in practice. The ultrasound scanner and the system used to acquire freehand 3D ultrasound data are both described in Appendix A.2.

It was not possible to compare ultrasound volume measurements with those from another imaging modality, for instance CT, as there was insufficient access to a CT scanner. However, volume measurement *precision* could be assessed by repeated observations. The only practical way of assessing *in vivo* volume measurement *accuracy* was to compare bladder volume with measurements of urine output.

Three areas were examined: the kidney, bladder and a foetus. Typical B-scans for all of these are shown in Figure 2.15, along with the segmentation. The scan plane pixel size for these scans was 0.035cm, hence for typical volumes of 150ml the volume errors due to resolution were approximately  $\pm 1\%$ .

### 2.4.1 Kidney

Four scans were performed of the same kidney, two using a transverse and two a longitudinal scanning pattern. Each of these was then segmented with approximately 25 cross-sections, spaced reasonably evenly, and ensuring that the organ extremities were sufficiently covered. Each set of segmented cross-sections was then gradually reduced, closest ones being removed first, and volume measurements made at each step.

The actual volume of the kidney is not known. However, the volume measurements should agree both for the varying number of scans within a sweep, and across each of the sweeps. Graphs of the volume measurements for each of the six techniques are shown in Figures 2.16 and 2.17, for transverse and longitudinal scans respectively; surface reconstructions are shown for ten scans. Table 2.2 shows the volume measurements for ten cross-sections.

*In vivo* volume measurement precision was achieved to  $\pm 6\text{ml}$ ,  $\pm 5\%$  for the kidney. This was the case even if only six cross-sections were used to measure the volume; the difference between the scanning patterns was more significant than the number of cross-sections used. This variation of volume with scanning pattern was probably a result of difficulties in segmenting the kidneys in each case.

<sup>1</sup>This does not take into account the unknown error due to organ movement, which also causes mis-registration.

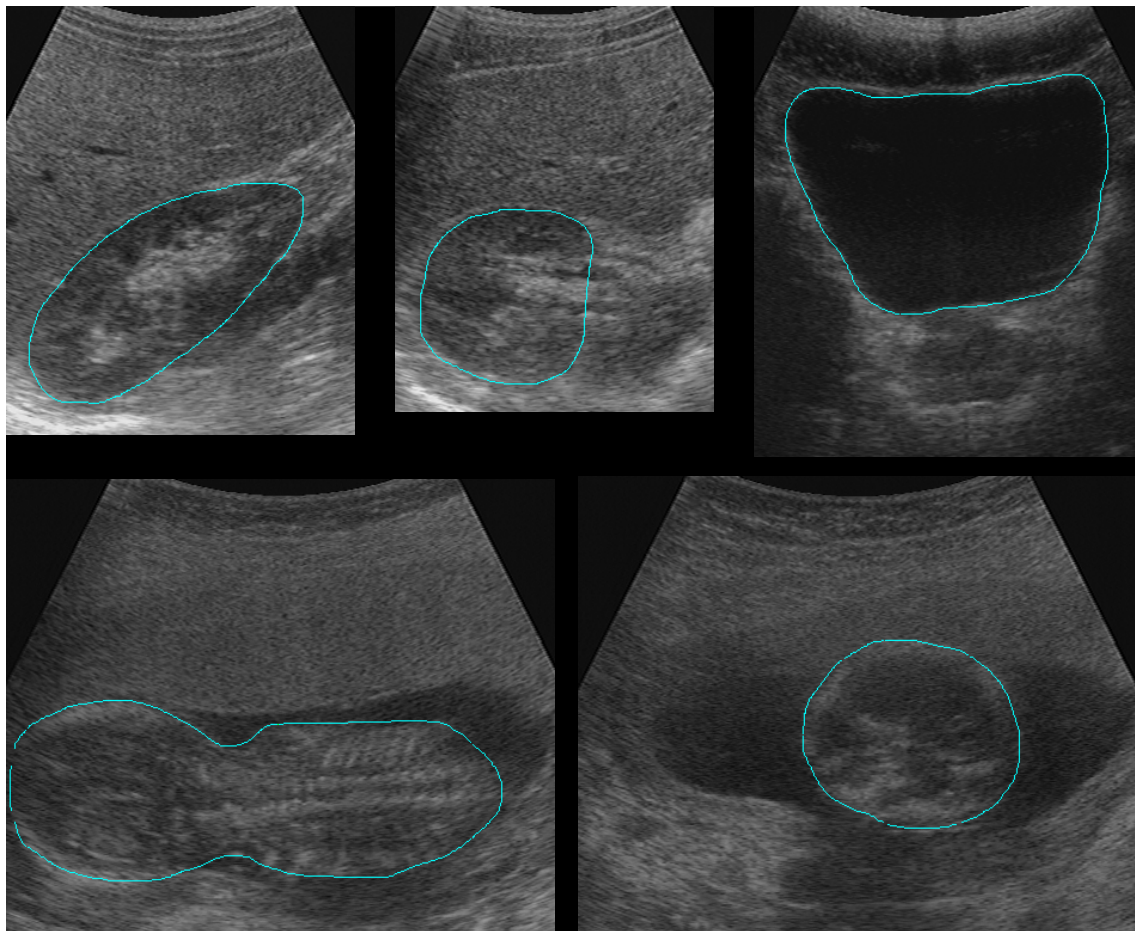


Figure 2.15: **Typical *in vivo* ultrasound B-scans.** The top row shows (from left to right) a longitudinal scan of a kidney, a transverse scan of a kidney and a scan of a bladder and prostate. The bottom row shows a longitudinal and transverse scan of a foetus at 16 weeks. Manual segmentation of the anatomy is also shown.

Table 2.2: **Human kidney volume: all scans** Volumes were measured using ten cross-sections of the transverse and longitudinal scans. **1:** Linear planimetry, **2:** 2D cubic planimetry, **3:** 3D cubic planimetry, **4:** Shape-based, **5:** Centroid-guided, **6:** Disc-guided.

Scan	Volume measurement in ml						
		1	2	3	4	5	6
Transverse	1	127.3	128.4	128.3	125.7	126.2	126.2
	2	123.7	124.6	124.7	121.3	122.2	121.9
Longitudinal	1	126.9	125.7	125.4	122.0	122.1	122.6
	2	113.7	117.2	117.3	114.4	114.2	114.9

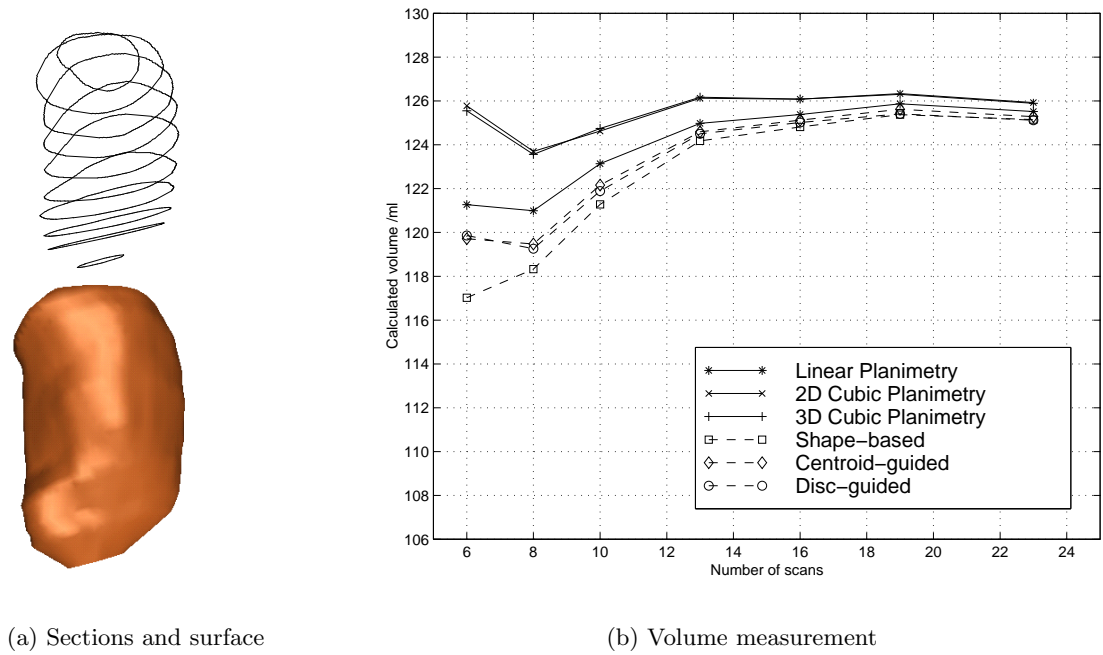


Figure 2.16: *In vivo* transverse scan of a human kidney.

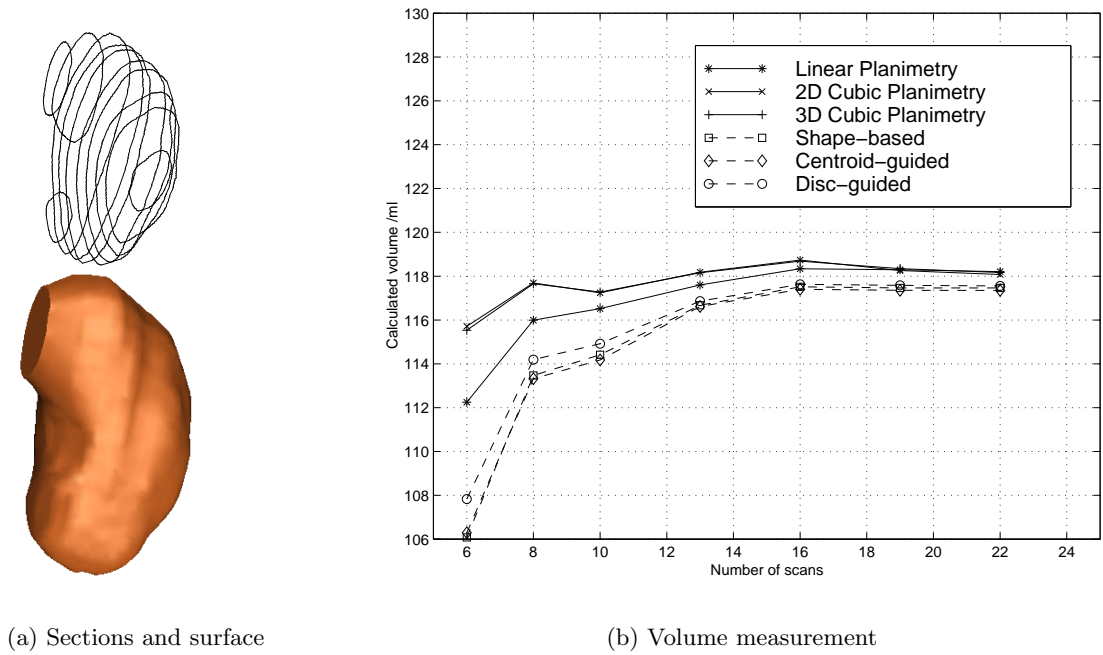


Figure 2.17: *In vivo* longitudinal scan of a human kidney.

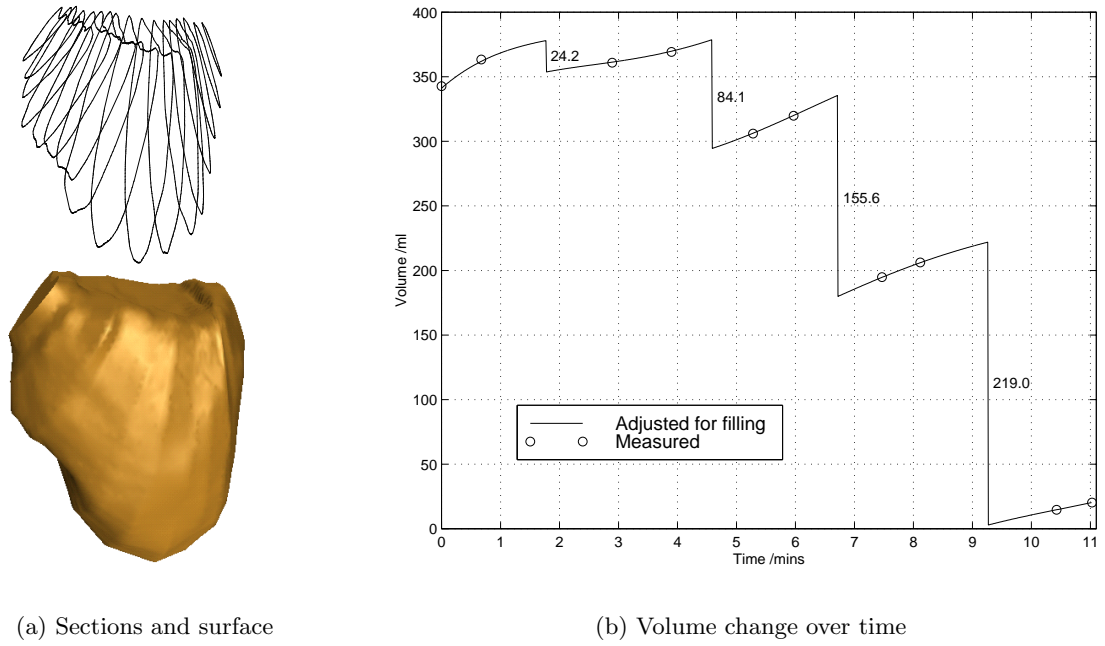


Figure 2.18: ***In vivo* scan of a human bladder.** The bladder was partially voided four times during the examination. Two sets of scans were recorded between each partial voiding. The data is contained in Table 2.3.

### 2.4.2 Bladder

In order to validate *in vivo* volume measurements, the actual volume must be known by an alternative, more accurate, method. This is possible for the bladder, which is a collapsible sack whose volume at any time is equal to the amount of fluid it contains. The output of the bladder can be easily measured from urine volume. The input to the bladder is more difficult to measure, but can be estimated from sequential volume measurements in periods with no urine output. In addition, the bladder wall is very well defined by ultrasound, and is therefore easier to segment than, for instance, the kidney.

Ten scans were performed of an initially full bladder, in pairs, with varying amounts of urine output between each pair. The bladder was completely voided after the eighth scan. The scans were performed in fast sequence, the output being collected for later measurement, in order to

Table 2.3: ***In vivo* bladder volume.** **Fill** is the estimated rate at which the bladder was filling. **Diff** is the calculated difference in volumes, adjusted for bladder filling, and **Void** is the actual measured output. The same data is shown graphically in Figure 2.18(b).

Time, m:s	00:00	00:40	02:54	03:54	05:17	05:58	07:28	08:07	10:26	11:02
Volume, ml	342.7	363.4	360.9	369.2	306.0	319.8	194.8	206.2	14.7	20.3
Fill, ml/m	30.7		8.3		20.0		17.5		9.3	
Diff, ml	24.2		84.1		155.6		219.0			
Void, ml	25		74		156		234			
Error, %	1.6		6.8		0.1		3.2			

limit the amount of bladder filling during the experiment. The output was then measured using a 20ml or 60ml graded syringe (dependent on the volume) to an accuracy of approximately  $\pm 1\text{ml}$ . The stored ultrasound B-scans were then segmented, using 15 to 20 cross-sections per examination: Figure 2.18(a) shows an example of this. Volumes were calculated from these cross-sections using 2D cubic planimetry, as in Table 2.3.

The amount of bladder filling was estimated in three stages. Firstly, the linear rate of filling was calculated, for each pair of scans, from the volume measurements. Secondly, cubic splines were used to interpolate these values and give a continuous bladder filling rate. Thirdly, this function was integrated, to give the estimated amount by which the bladder had filled at any point during the experiment. This information, along with the measured volumes, was then used to estimate the actual bladder volume at any point (for this purpose, the voiding was considered to be instantaneous at the mid-point between pairs of scans). The resulting curve is shown in Figure 2.18(b), along with the estimated urine output calculated from this curve.

The errors in Table 2.3 were calculated for the *bladder volume measurements*, rather than for the urine volume. The urine volume is a complicated function of the bladder volume measurements, due to the adjustments for bladder fill rate, making these errors hard to estimate. Since it is essentially a measure of difference, the urine volume error is assumed to be twice the actual bladder volume error. This leads to a volume measurement accuracy of approximately  $\pm 7\%$ .

A similar study of 3D ultrasound bladder volume measurement was recently performed, using the reverse of this procedure: the bladder was initially empty, and *filled* with known quantities of saline between each measurement [168]. Filling the bladder in this manner requires catheterisation, which is obviously less comfortable than the approach outlined above. Indeed, one patient

...felt a very strong, unpleasant desire to urinate at 210ml and refused to have the bladder filled with 250ml.

The reported measurement accuracy was worse than  $\pm 20\%$ , however the 3D system in use involved very approximate automatic, rather than manual, segmentation. In addition, no adjustment was made for *internal* filling of the bladder during the experiment.

### 2.4.3 Foetus

The foetus presents more of a challenge to ultrasound volume measurement than most other areas of anatomy, since it is a more complicated shape which is harder to segment, and is also capable of extreme movement. This movement causes problems during scanning, as it can lead to artifacts in the 3D data; in practice the acquisition process must be repeated if too much movement occurs. Changes in shape can also lead to significant changes in the topology of the cross-sections, which adds to the segmentation difficulty.

To investigate the precision with which the volume of a foetus can be measured, a foetus of 16 weeks was scanned ten times over the course of 20 minutes<sup>2</sup>. The scan pattern was either longitudinal or transverse, and between 15 and 25 cross-sections were segmented in each case. Surfaces reconstructed from each of these scans are shown in Figure 2.20 — the large variation in pose is clear from this figure.

<sup>2</sup>In practice, roughly three times as many scans were acquired than could be used, due to movement of the foetus.



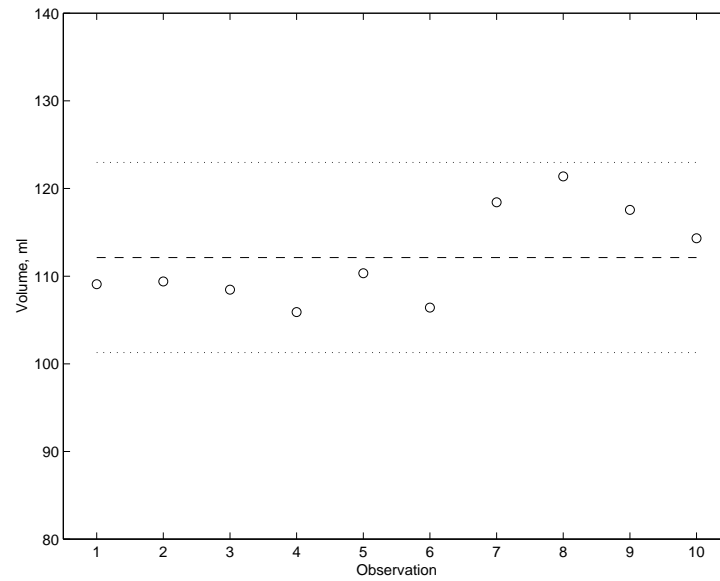


Figure 2.19: ***In vivo* volume measurement of a fetus at 16 weeks.** The dashed line shows the mean volume and the dotted lines the 95% confidence limits.

Figure 2.19 shows the result of using 2D cubic planimetry to calculate the volume from these sets of cross-sections. Taking the transverse scans alone, the 95% confidence limit for the volume measurement precision is  $\pm 7\%$ , or  $\pm 9\%$  over all ten scans. This error is a result of difficulty in segmenting (as with the kidney) and changes in pose of the fetus resulting in a variety of regions outside the fetus being included in the volume measurement.

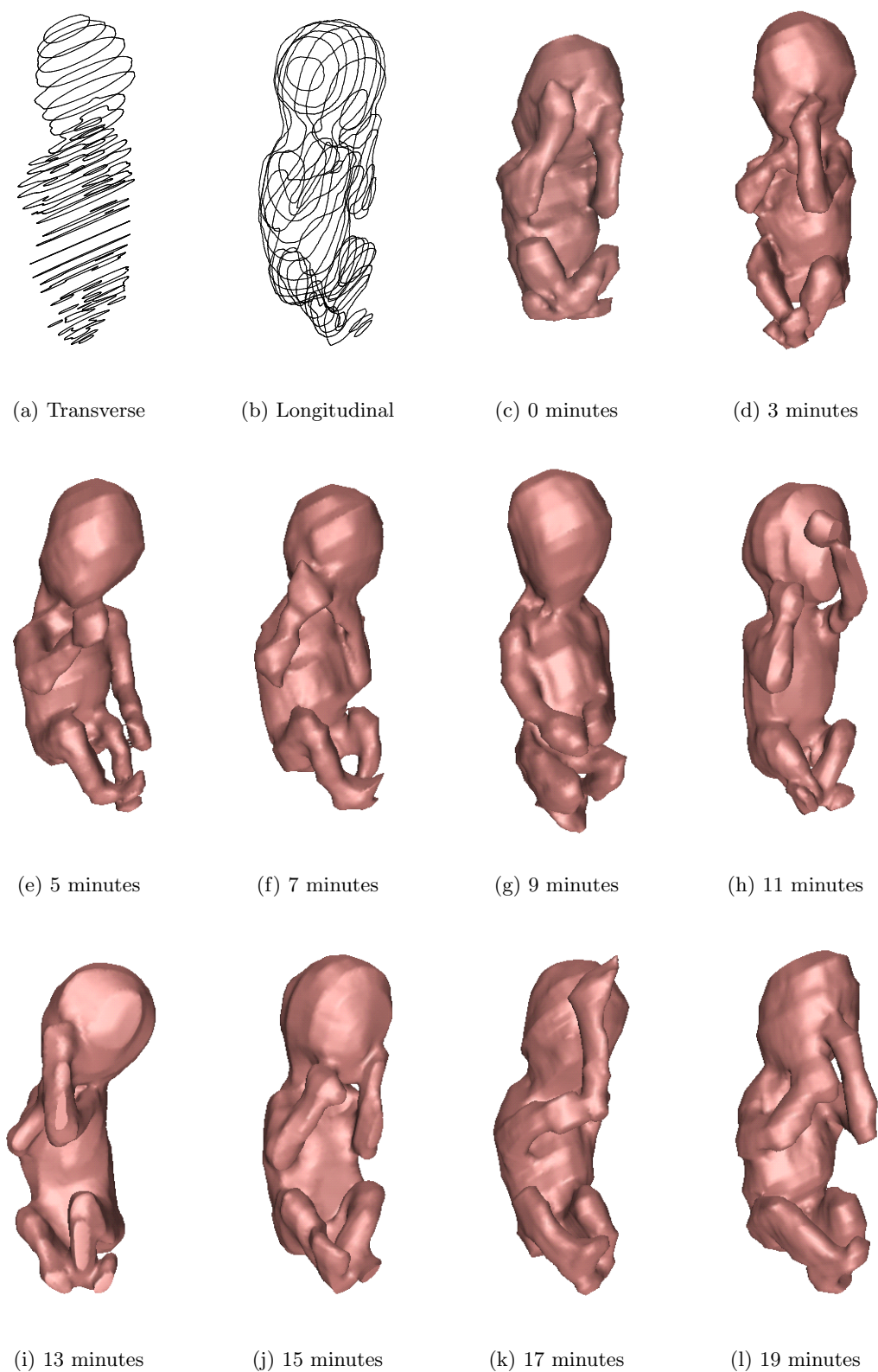


Figure 2.20: *In vivo* scan of a foetus at 16 weeks. Scans (h) to (j) were longitudinal as in (b), the remainder were transverse as in (a).

## Chapter 3

# Surface interpolation and visualisation

### 3.1 Current methods of surface interpolation

Surface interpolation, as explained in Section 1.4.2, is the first step in generating a geometrical representation of the surface, which can be used for interactive display. Surface reconstruction can also be achieved by direct triangulation from cross-sections. Most reconstruction techniques have been developed for parallel cross-sections; even then, estimating the surface position in a robust and feasible manner is a very difficult task. These techniques are briefly reviewed in Section 3.1.1, before concentrating on interpolation (functional) methods in Section 3.1.2, and reviewing the few techniques that have been applied to non-parallel data in Section 3.1.3. Some of the related techniques of scattered data interpolation are finally reviewed in Section 3.1.4.

#### 3.1.1 Surface from cross-sections

There are two distinct approaches for constructing a surface from cross-sections of an object. Both approaches can be used to generate a triangular mesh of the surface, which is useful for rendering using standard graphics hardware. In the first approach, the set of points making up each of these cross-sections are directly triangulated, such that these points become the vertices of the triangular mesh. The alternative is to use the cross-sections to estimate a 3D function that represents some measure of distance from any point to the surface. Once this function has been created, the zero isosurface (also known as the ‘level set’) can be triangulated to reveal the object surface. This approach was originally suggested by Levin for the interpolation of CT data [111].

Direct triangulation of points on the cross-sections is a difficult (though well studied) problem in cases where the cross-sectional shape varies between planes. Additional vertices must be created if the topology changes, for instance a saddle point for a branching structure. It is also impossible to constrain the aspect ratios of the generated triangles, since the vertices are defined by the positions of the cross-sections. This can lead to poor quality surface displays if Gouraud shading (which is the fastest effective shading technique) is used. Much effort is required to detect and correct special cases where the triangulation of complex shapes might otherwise fail [13, 16, 29, 122]. In addition, there have been very few examples in the literature of direct triangulation of non-parallel cross-sections [48, 143].

In contrast, in the functional approach, the same criterion is applied to both simple and

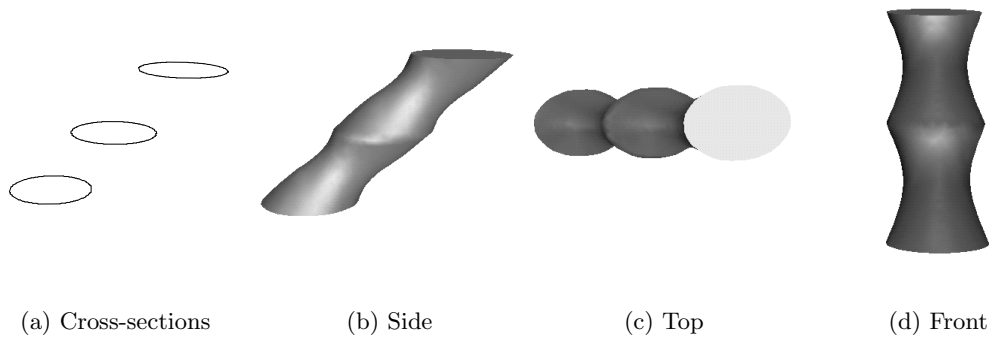


Figure 3.1: **Limitations of shape-based interpolation.** (b), (c) and (d) show shape-based reconstructions from three cross-sections of a simple cylinder, shown in (a).

complex shapes. Once the function has been created, the isosurface can be triangulated quickly by a variety of algorithms, discussed in Section 3.3. In some cases, extracting the isosurface at a non-zero value (i.e. at a constant distance from the actual surface) can also provide useful information [142]. This approach also lends itself more naturally to the interpolation of non-parallel cross-sections, or even scattered data.

### 3.1.2 Shape-based interpolation

Shape-based interpolation, first proposed by Raya and Udupa [154], is probably the simplest form of distance function creation. In this method, the first step is to calculate, for each pixel in the image, the distance from the closest point on the object cross-section. In order to define the inside of the object, this distance is chosen by convention to be positive for points inside the object and negative for points outside. Secondly, these transformed images are interpolated using a linear or cubic function, to generate new transformed images. Finally, the new images can be thresholded at zero, such that any pixels greater than zero are considered to be inside the objects on the new scan planes. A faster method was also proposed, which restricted the distance calculation to pixels that were contained by the object cross-section on only one of the parallel scan planes. The authors suggested that the method could be adapted for near-parallel images, although this was not demonstrated.

Herman *et al.* [82] later introduced a more accurate and efficient algorithm for calculating the distance values, based on a distance transformation introduced by Borgefors [31]. This transformation used a chamfer code, which is a better estimate of Euclidean distance than the city-block distance in the previous method. At much the same time, Montanvert and Usson [124] also arrived at a near identical algorithm for use in 2D granulometry and 3D reconstruction, again by interpolating Borgefors' chamfer code. In this case, the distance maps were interpreted as “fate maps” — for each pixel, the distance value indicated a date, where zero indicated the “birth” of the pixel in the interpolated images. The results were, however, identical to the linear interpolation used by Herman. Another near identical algorithm was presented later by Jones and Chen [94], although here the distances were calculated by a more complex method. Ohashi [137] used morphological erosion and dilation operators to produce similar results, this time applied to pore geometry.

All of the above shape-based methods can handle bifurcations and relatively complex shapes.

However, they can also produce undesirable artifacts when the shape on each contour changes radically, or undergoes significant translation. The effect of translation can be demonstrated by examining the results of shape-based interpolation applied to the reconstruction of a simple cylinder, scanned at an oblique angle, as in Figure 3.1. The side view is nearly correct, however the front view shows the surface of the cylinder as concave. This is because the interpolated object can only exist within the bounds of the original cross-sections, projected in the direction of interpolation, as seen from the top view. This is also demonstrated in Figure 3.2, which contains some (near) parallel cross-sections from a freehand 3D ultrasound scan of part of the system of hepatic ducts. Figures 3.2(a) and (b) show the actual anatomy, whereas Figure 3.2(d) shows the result of applying shape-based interpolation to every other cross-section.

This problem has been partly addressed by moving the cross-sections to align the centroids of each object *before* interpolation (then using the inverse transform on the interpolated cross-sections) [83], as in Figure 3.2(e). This works well for tree-like shapes that have been scanned transverse to the main axis, for instance the top three cross-sections in Figure 3.2(e). In this case, individual objects are treated separately, and the union of the interpolated objects is used to create the final object contours. However, disastrous results can ensue when the shape of each cross-section changes dramatically, even if the original shape is still a tree-like structure, as can be seen from the other cross-sections in Figure 3.2(e). Scaling each cross-section such that the bounding rectangles are the same size, in addition to aligning the object centroids, can help in some cases [116] — but makes the situation worse in the case of Figure 3.2(f).

A similar method has been proposed recently by Liu *et al.* [114]. This interpolation method, *edge-shrinking interpolation*, is a close relative of shape-based interpolation, in that the intermediate cross-sections are still derived from distance transforms calculated using a chamfer code. In this case, the start contour is eroded by a morphological disc of varying size, determined by the distance code at that point in the contour. The results of this operation are, however, very similar to those achieved with a more straightforward linear interpolation. An interpolation direction similar to connecting centroids is used, but based on minimisation of the distance field of one contour, sampled along the other contour. In addition, one contour can be “shrunk” before interpolation in order to fit inside the other contour, thus reducing the effects of radical shape changes. This technique improves the interpolation result slightly, but is much more complex. In addition, the treatment of multiple contours is the same as for Higgins’ method, in that the union of the results is used, generating gradient discontinuities at the junctions.

An example of a gradually varying interpolation direction, applied to parallel slice data, is given by Moshfeghi [126]. Here a simple scheme is used to interpolate magnetic resonance angiography data, where the centroid of each separate contour is first calculated, then the connectivity between centroids on adjacent scans is determined. The vector connecting matched contours is then used to interpolate data within those contours, in the same way as in [83, 114]. However, the data *between* contours is interpolated with a varying direction which is itself a linear interpolation of centroid connection vectors. This works well for multiple simple shapes with clear connectivity, but not for complicated shapes, since only a single interpolation direction is calculated for each contour.

Several other object-based strategies have been proposed for interpolating images as well as cross-sections. *Cores* can be extracted from the grey-level image (without segmentation) and used to interpolate intermediate parallel slices [148]. The idea of varying the direction of interpolation across the image has also been applied to this case [63]. Morphing (particularly known for its use in films such as Terminator II) has been used recently on object cross-sections

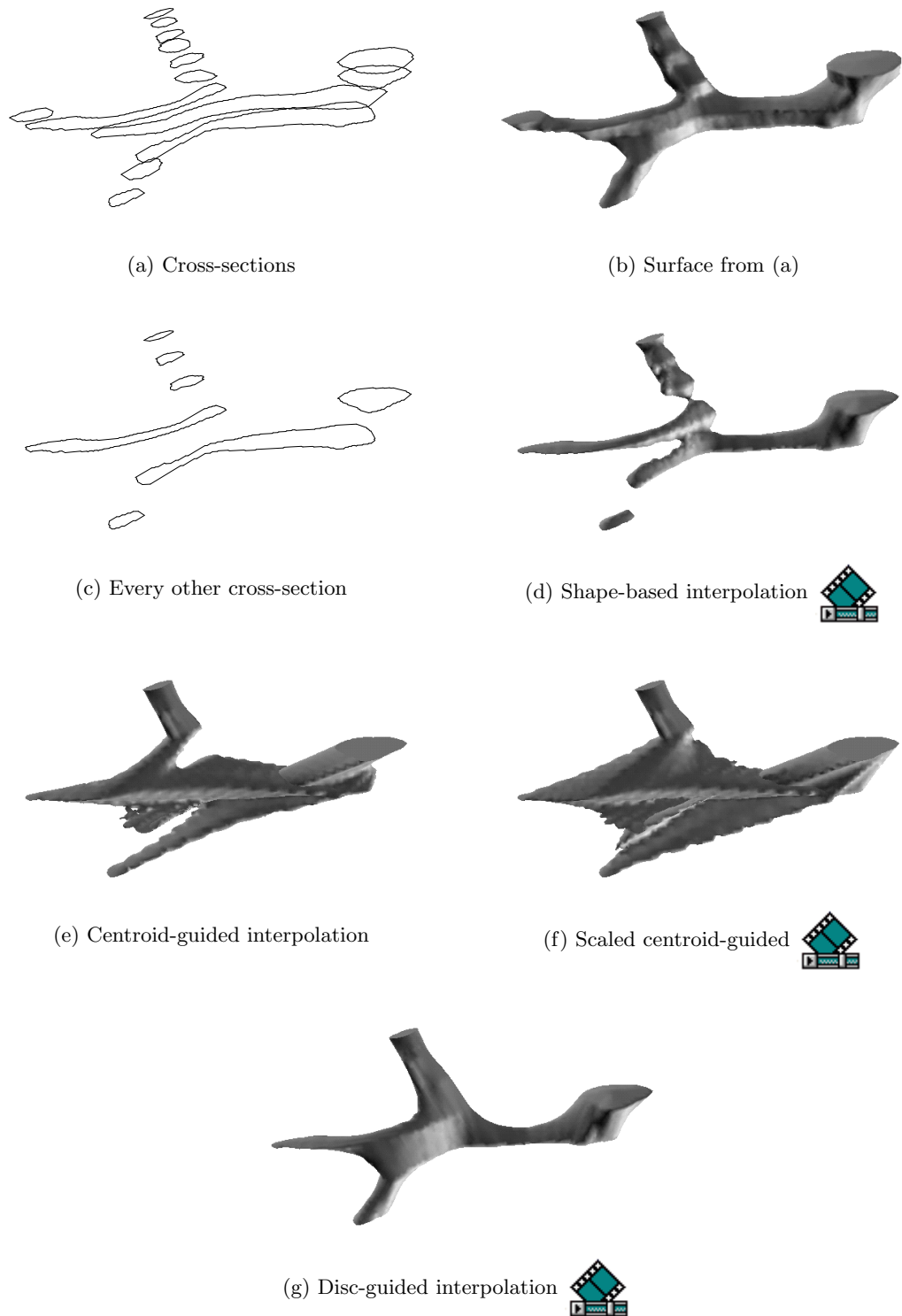


Figure 3.2: **Reconstruction of part of the system of hepatic ducts.** The cross-sections were manually outlined on the original B-scans of a freehand 3D ultrasound scan. (a) and (b) show the original cross-sections, (d) to (h) show the surface interpolated from the selected cross-sections shown in (c).

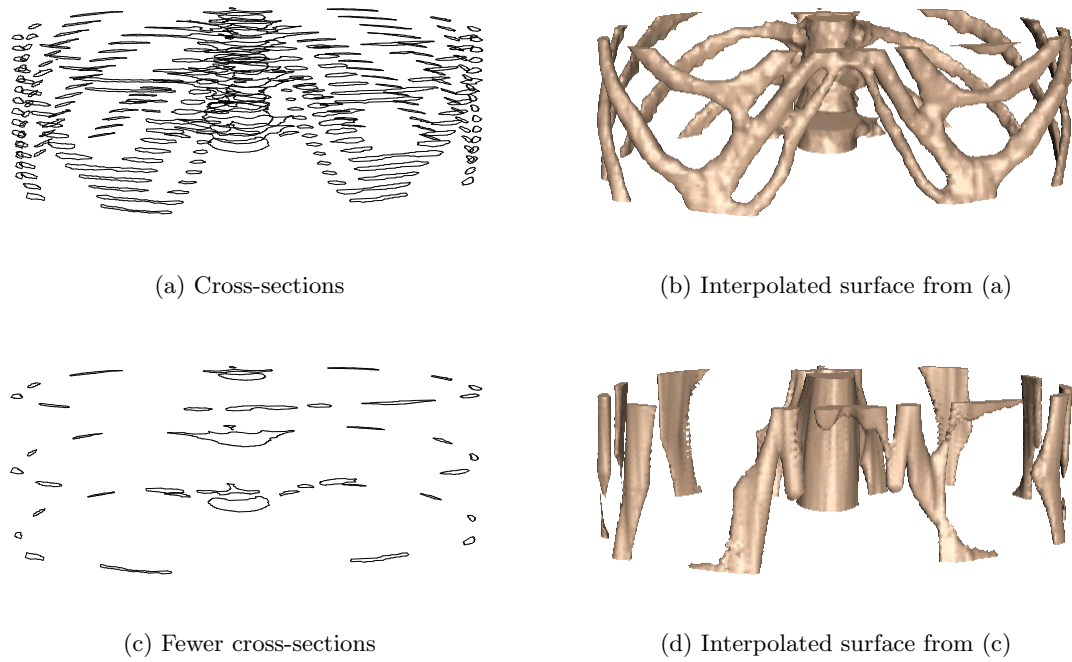


Figure 3.3: **Section of human ribs.** The cross-sections were automatically segmented by thresholding CT data.

to guide the interpolation [171]. Interpolation in the direction of object centroids has also been used to reconstruct an object from ‘staircases’ formed by iteratively connecting points on two cross-sections [184].

Where there is some definition of connectivity in all of the above algorithms, it is based on *whole contours*, but it can be seen from Figures 3.2(e) and (f) that this is not always appropriate: the small contours only correspond to a part of the longer contours. An alternative object-based interpolation approach, *dynamic elastic interpolation*, begins to address this problem [41]. Here, a force field is constructed which iteratively deforms one cross-section towards the other. The initial formulation produced similar results to centroid-guided shape-based interpolation for complex objects. However, this behaviour was improved by imposing an upper threshold on the forces at each point on the contour. This prevents far portions of the cross-section having a detrimental effect on the deformation, but also adds further complexity, and the requirement to estimate an additional parameter.

In *disc-guided interpolation*, presented in this thesis, the interpolation of a distance transform is guided by using correspondence of *regions* of cross-sections. This region correspondence is determined by representing each cross-section by a set of discs that are contained by the object. The result of this algorithm is shown in Figure 3.2(g). A shape interpolation method using a disc representation has also been presented in [151]. However, in this method the ‘union of circles’ representation is interpolated directly, rather than used as a guide for shape-based interpolation. The advantages of shape-based interpolation, in particular treatment of complex topology and faithfulness to the original data, are not therefore apparent in this case.

It is important to recognise that no technique can guarantee to reconstruct the actual anatomy from *any* set of cross-sections. It will always be possible to create a set of cross-sections whose connectivity will be misjudged by a given algorithm. This is equally true for



techniques which use more than two cross-sections to calculate connectivity [15]. For instance, Figure 3.3(a) and (b) show a section of ribs, segmented from a CT scan. If fewer cross-sections are used, as in Figure 3.3(c), the incorrect surface in Figure 3.3(d) is generated, as this is the simplest way to connect the cross-sections. This is a consequence of under-sampling: if the inter-slice spacing is greater than the shortest wavelength present in the data, fine detail will inevitably be missed in the surface reconstruction. However, it is fair to expect a reconstruction algorithm to produce a ‘reasonable’ surface, in the sense that similar contours should be connected and the surface should have the simplest topology given the cross-sections (see [76] for further discussion of this issue) — this is the goal of the algorithm presented in Section 3.2.

### 3.1.3 Surface from non-parallel cross-sections

All of the algorithms in the previous section have been proposed for parallel cross-sections. In order to reconstruct the surface in a sequential freehand 3D ultrasound framework, a method that does not assume parallel planes is required. Such reconstruction methods are few and far between. Most of the following techniques will not in fact handle arbitrarily oriented cross-sections. Usually the scan plane orientation is restricted such that they are nearly parallel (or nearly regular for fan or rotational scanning patterns), or at least do not self-intersect. In addition, there are often restrictions on the shape and number of contours in each plane.

In [84], a surface was triangulated between cross-sections of phantoms of the saphenous vein. They were nearly parallel (a ‘translational sweep’ was used) and all approximately circular, such that the resulting surface was conical or cylindrical. A similar application, again with nearly circular cross-sections, was investigated in [102].

Several researchers [7, 71, 101] make use of an algorithm initially developed by Cook [45]. Two contours are sampled as a set of lines joining boundary points, then the volume calculated from these points by forming tetrahedra with a common central point on each plane. This volume represents that contained by the surface formed by triangulating between each contour boundary point in turn. This will be accurate for fairly similar contours, but not so for larger changes of shape. In addition to this restriction on contour similarity, the planes containing the contours are not allowed to intersect in the region of the contours.

A completely different approach to surface reconstruction by using deformable models is presented in [46], which also contains a review of the application of similar techniques to the heart. Rather than starting with the cross-sections and attempting to extract a surface, this method starts with a model and attempts to fit this to the cross-sections. The fitting process is controlled by appropriate forces applied to the model from the data, and usually also internal forces (e.g. bending stress) to prevent the model from moving too far from its original shape. The technique has several advantages — notably that the cross-sections need not be parallel, nor even closed. In fact, the data used to drive the model deformation can consist of arbitrarily scattered points or line segments, as with the algorithms described in Section 3.1.4. The adoption of a model allows more *a priori* knowledge to be included in the process, but this is a two-edged sword. *A priori* information can greatly assist with difficult reconstruction problems if the object has the expected shape. However, objects which do not have the correct shape tend to be forced into the shape of the model. In a clinical context, such objects are often those which have the most importance.

Another method that can handle overlapping planes is given in [125]. This relies on finding a suitable longitudinal axis, which passes near the centre of each of the cross-sections. Having selected the axis, the original cross-sections are rescanned on to new scan planes, that rotate



about the longitudinal axis. This creates points on the new scan planes where they are crossed by the contours, which can then be joined to produce new contours. Importantly, the new contours no longer overlap (the points are joined in the order of distance along the axis, *not* in the original scan plane order) so that further processing is greatly simplified. The technique works well for simple shapes, but does not handle complex ones (e.g. with bifurcations). A very similar approach is adopted in [119] for fan scanning of the heart, except that the scan planes do not overlap in this case.

Robert [157] recently presented a method which can construct surfaces from only three orthogonal cross-sections. The surface is composed of eight triangular patches, which interpolate each quadrant of the cross-sections. The simplicity of the method makes it fast, and useful for defining a volume of data to be rendered. However, visualisation or volume measurement from the surface itself is only useful for simple objects, which can be defined by only three cross-sections. In addition, a set of three orthogonal ultrasound images can in general only be generated from a voxel array, rather than from a sequence of B-scans, due to restrictions in scanning direction.

### 3.1.4 Surface from scattered points

Algorithms that can create surfaces from scattered point sets can be applied to more regular planar cross-sections, by using the vertices of each cross-section — although in general this increases the complexity of the problem. There are several such algorithms that generate a 3D distance function from these points [12, 25, 85]. These represent very general solutions, but as a result are not always well constrained, and many points are required to ensure a correct reconstruction. This approach is attractive for freehand 3D ultrasound, since the cross-sections are already non-parallel: indeed it has recently been implemented for *in vitro* reconstructions of the left ventricle [109]. However, it was noted that the segmentation of the ultrasound data was not sufficiently dense to constrain the surface, and the method had to be adapted to include a prior model of the surface, which was then deformed to represent the data, rather than constructed from the data.

More sparse data can be interpolated by adding the constraint that the surface curvature is minimised, as well as that the surface passes through the data. This can be achieved by using volume splines to reconstruct the surface from cross-sections. In one method [166], each cross-section is first represented by a 2D carrier function, whose values are calculated at the vertices of the cross-section. A spline function of two variables is used to approximate the function, which can then be linearly interpolated between each cross-section, once again giving a 3D distance function which is then thresholded at zero. Alternatively, a volume spline can be fitted to all the cross-sections simultaneously, generating the 3D distance function directly [178]. Since these methods involve the inversion of an  $N \times N$  matrix, where  $N$  is the number of points on each cross-section for the former method, and on all cross-sections for the latter, the processing time increases significantly with data complexity.

### 3.1.5 Morphing

The idea of constructing a surface between two differing cross-sections is very closely linked with that of ‘morphing’ one 2D shape into another, much used in computer graphics [106]. In fact, distance field interpolation has recently been used for this purpose [44]. The surface interpolation algorithm presented in Section 3.2 can also be applied to morphing 3D data, and this is discussed

in more detail in Chapter 5. Distance transformation has also been used in conjunction with field-morphing in order to interpolate the intensity information in medical images [171]. In this method, shape-based interpolation is used to determine intermediate cross-sections which form the basis of the control points required by the field-morphing process.

## 3.2 A new approach: disc-guided interpolation

### 3.2.1 Overview

The overall strategy of the surface interpolation algorithm is summarised in Figure 3.4. It is built on shape-based interpolation, however the interpolation direction is allowed to vary at each point, dependent on the object correspondence. Since shape-based interpolation already requires a distance transform to be calculated for each cross-section, it is straightforward to extract a set of maximal discs from these transforms, which loosely represent the shape of each object. The correspondence between each of these discs is then calculated, assisted by the original distance transforms. The interpolation direction at any point is calculated as a weighted sum of contributions from each disc, where the weighting is once again derived from the original distance transforms. Thus, the distance function can be found at any point in space between sequential cross-sections, and evaluated on a grid suitable for performing zero isosurface triangulation.

The use of the distance transform to weight both the correspondence and interpolation direction calculations makes up for the loose representation of the object by only a few discs. Centres of discs are a useful representation to guide the shape-based interpolation, since they are independent of the disc radii. Shape-based interpolation is itself already very good at handling changes of scale in similar shapes.

Each step in the interpolation process is discussed in detail in the following sections. The final step, that of isosurface triangulation, is the subject of Sections 3.3 and 3.4.

### 3.2.2 Calculation of the distance transform

The first step in shape-based interpolation is to calculate, for each pixel of interest, the minimum distance from the object cross-section. This can either be performed as a pre-processing step, yielding a distance-transformed image; or for each pixel as required, during the interpolation. Chamfer coding [31] is an efficient way of calculating a distance transform within a bounding rectangle, however additional storage is required for this transformed portion of the image. In addition, it is not necessarily obvious how large the distance-transformed region should be — especially if the planes are not parallel and the direction of interpolation is allowed to vary, as in this case.

For these reasons, a two phase approach has been adopted. Firstly, chamfer coding is performed in a rectangle which just bounds the object cross-section in each segmented B-scan. Secondly, an additional algorithm is presented that can calculate far distance codes using the edge information from the chamfer-coded region. When a distance code is required for a given point, the process handling this operation will either return a value from the transformed region, or calculate a new value if the pixel is outside this region. Most of the points will be contained within the chamfer-coded region, so the interpolation processing time is kept to a minimum. However, the far point algorithm guarantees a correct distance code for any arbitrary point, should that be required.

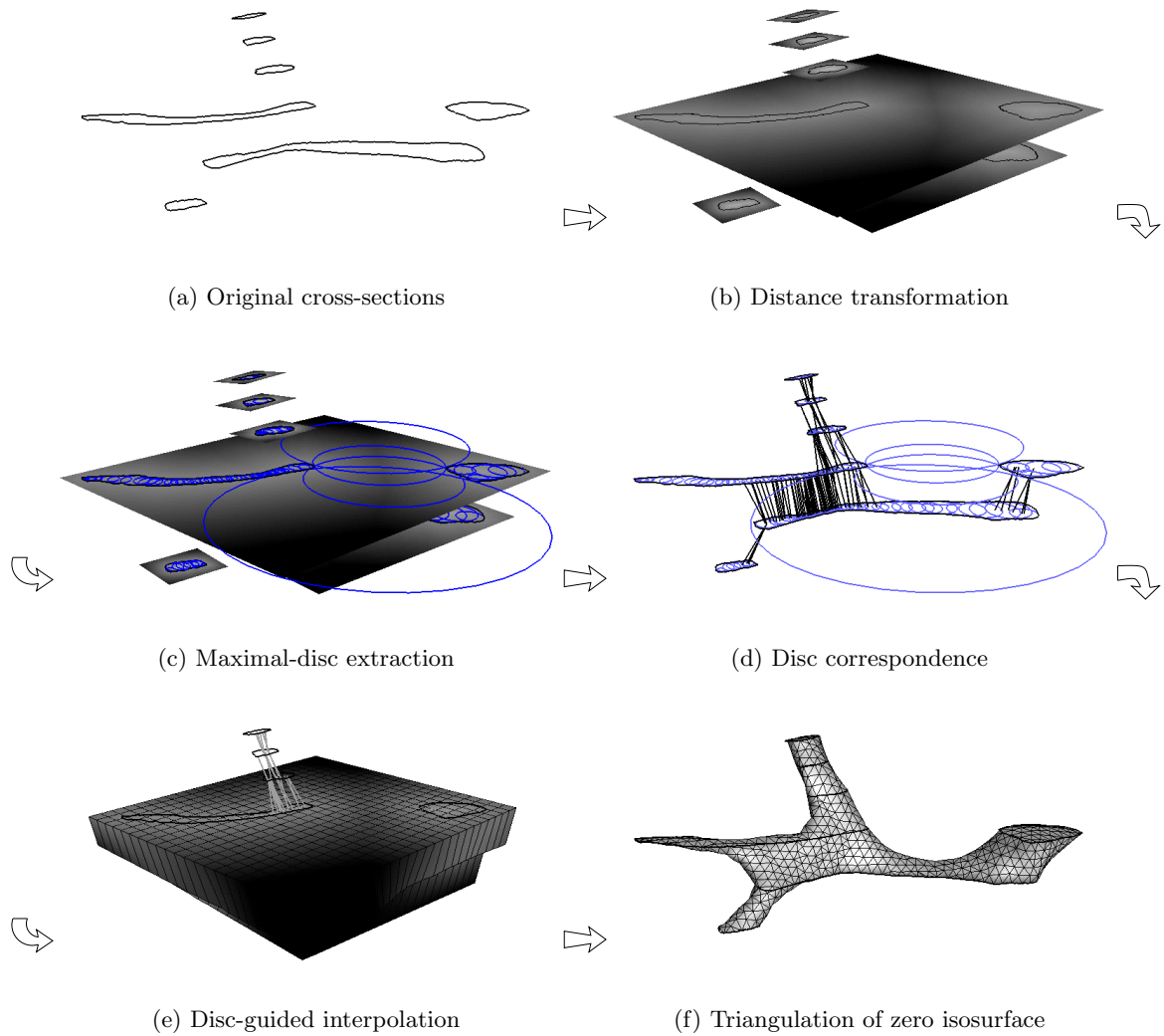
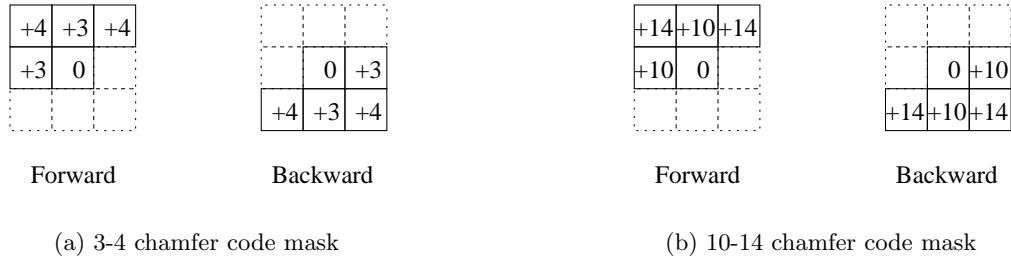


Figure 3.4: **Surface interpolation algorithm.** Distance transformation is first performed on each cross-section (b), then a set of representative discs can be extracted from these transformations (c). Region correspondence is estimated for each disc (d), and this is combined to give a correspondence direction at any point. This provides the interpolation direction for the distance field interpolation (e), which can then be triangulated to reveal the surface (f).

Figure 3.5:  $3 \times 3$  chamfer code masks.

### Near point calculation

The various ways of estimating Euclidean distance from an object are reviewed by Borgefors [32]. It is suggested that any of the reviewed methods (city block, chess-board, octagonal, chamfer  $3 \times 3$  or chamfer  $5 \times 5$ ) may be appropriate in different situations. In practice, the chamfer  $3 \times 3$  code is accurate enough for shape-based interpolation; the added complexity of the chamfer  $5 \times 5$  code is not worth the minimal effect it has on the interpolated surface.

Chamfer coding requires as an input a binary segmented image, i.e. an image in which pixels inside the object of interest are set to one, and those outside set to zero. Cross-sections defined as vectors can be transformed to this representation by scan conversion, described in Appendix C.1. Non-object (zero) pixels are initialised with a large negative value, following which a “mask” is passed first forward (top to bottom, left to right), then backward (bottom to top, right to left), through the image. Each pixel in the image is summed with the respective mask pixel, and the minimum of these sums represents the new value for the centre pixel. Each mask contains the centre pixel, and those pixels which have already been examined in this pass, as shown in Figure 3.5 — pixels without mask values are ignored. Borgefors suggests that the 3-4 chamfer code is a good  $3 \times 3$  mask.

Herman [82] suggested a couple of minor changes to this process. Firstly, the slightly different 10-14 mask is used, rather than the 3-4 mask. The image is then initialised such that non-object pixels are given a very negative number, object pixels a very positive number, and pixels which share an edge with the object contour either  $+5$  or  $-5$  for object and non-object pixels respectively. This ensures that the distance values are measured from the contour itself, rather than the outermost object pixels. Secondly, the masking operation is inverted in both passes for pixels which are outside the object (i.e. negative), in that the mask is *subtracted* from the image and the *largest* value used. This enables the simultaneous calculation of the distance fields both inside and outside the object.

In practice, the edge-detection and initialisation of image pixels can be combined with the first chamfer coding pass, such that pixels are initialised to either  $+5$ ,  $-5$ , or the first pass minimum distance. 16-bit integers are used throughout this thesis to represent chamfer distance codes; this allows the coding of images up to  $2^{15}/14$ , or 2340 pixels width<sup>1</sup>.

An example of a typical cross-section that has been chamfer coded in this way is given in Figure 3.6. This is from the “baseball glove” object (see Appendix A.1), sampled on a  $330 \times 420$  grid, and the distance transformation calculated using both the near and far field algorithms. This figure shows the effect of using this estimate of Euclidean distance — the far field distance

<sup>1</sup>This discretisation introduces a slight error into the surface location, since distance values of 0 are considered to be inside the surface.

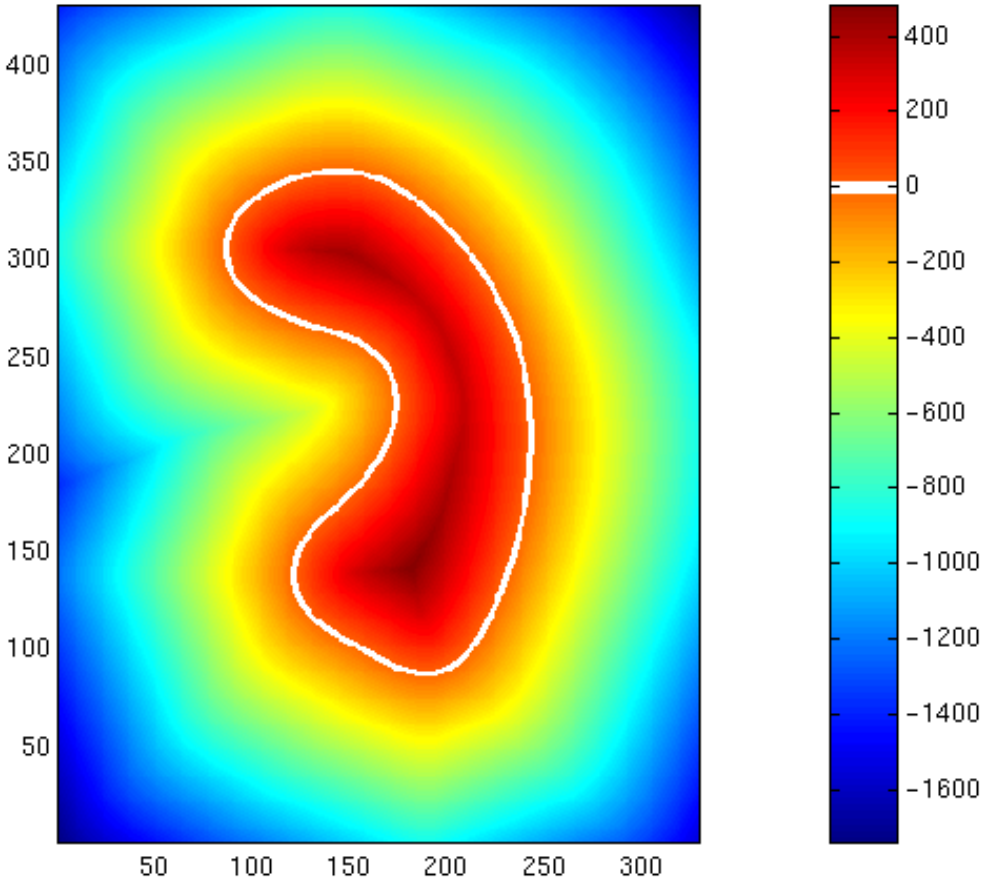


Figure 3.6: 10-14 chamfer-coded object contour.

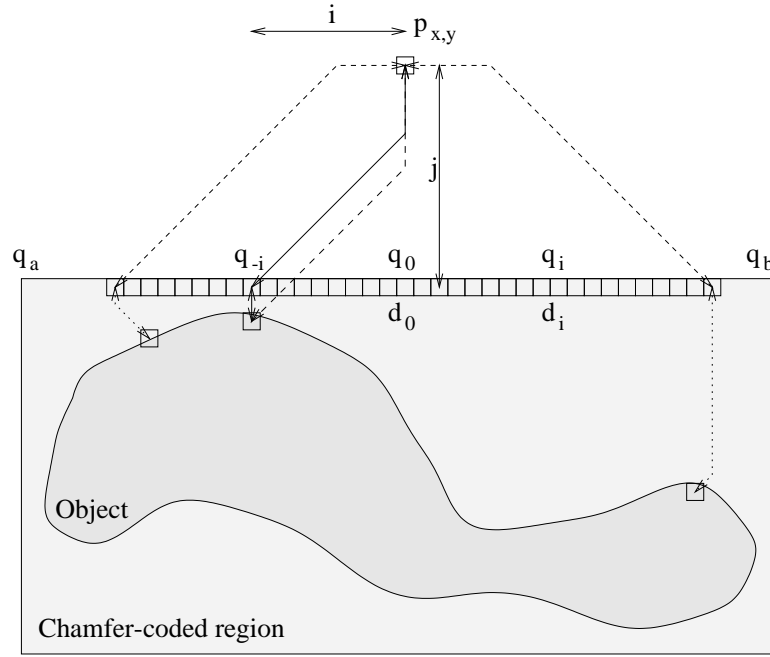


Figure 3.7: **Far point distance calculation from a chamfer-coded rectangle.**

tends towards an eight sided shape similar to an octagon. However, the estimate is good near the contour, which is the region with the greatest effect on the interpolated object surface.

A fast method for calculating a Euclidean distance field, based on city block and chess-board transformations, has also been suggested [39]. This would increase the smoothness of the surface, however the additional processing time is not considered to be worth the minimal additional effect.

### Far point calculation

The simplest way of calculating the distance field for an arbitrary location would be to iterate through the contour points, calculating the distance of each from that location, and retaining the minimum value. However, it is possible to calculate the distance code for arbitrary points more efficiently by making use of the rectangular chamfer-coded region surrounding the object cross-section.

Figure 3.7 is an example of such a cross-section, where the distance code for a far point  $p_{x,y}$  is required. Distances in a chamfer  $3 \times 3$  code can be represented by the minimum Euclidean distance along any path formed entirely from lines parallel to or at  $45^\circ$  to the image edges. Hence the 10-14 chamfer distance from  $p_{x,y}$  to a point  $q_i$  is:

$$\begin{aligned} p_{x,y}q_i &= \max(10(|j| - |i|) + 14|i|, 10(|i| - |j|) + 14|j|) \\ &\equiv \max(10|i| + 4|j|, 10|j| + 4|i|) \end{aligned} \quad (3.1)$$

where  $i$  and  $j$  are the distances in pixels from  $p_{x,y}$  to  $q_i$  in the  $x$  and  $y$  directions respectively. The chamfer distance from the point  $p_{x,y}$  to the nearest object point can be calculated by minimising the sum of the distance to the edge of the chamfer region,  $p_{x,y}q_i$ , and the chamfer code at the edge,  $d_i$ . This property of the chamfer  $3 \times 3$  code is demonstrated by Embrechts [54]. Hence, if  $d_{x,y}$  is the chamfer code at point  $p_{x,y}$ , then:

---

```

i := 0;   dmin := j + d0;   dpart := j;

while dpart < dmin and ( −i ≤ a or i ≤ b ) do

    dpart := max(10i + 4j, 10j + 4i);

    if −i ≤ a then

        dmin := min(dpart + d−i, dmin);

    if i ≤ b then

        dmin := min(dpart + di, dmin);

    i := i + 1;

```

---

Figure 3.8: **Far distance code algorithm.**  $d_{min}$  is the current distance code estimate for the point,  $d_{part}$  is the chamfer distance from the point to the current edge point. Other variables are as previously indicated.

$$d_{x,y} = \min_{a \leq i \leq b} (p_{x,y} q_i + d_i) \quad (3.2)$$

where  $a$  and  $b$  are the distances in the  $x$  direction from the point  $p_{x,y}$  to each corner of the chamfered region. The search space can be further limited by noting that if the distance  $p_{x,y} q_i$  is already greater than the minimum  $d_{x,y}$  found so far, there is no need to search for greater (or smaller if  $i$  is negative) values of  $i$ . The resulting process, for calculation of distance field points directly above the chamfer-coded region, is described in Figure 3.8.

The distance code for points immediately below, to the sides, or outside the corners of the region can be calculated in a similar manner. For the corner regions, the search proceeds from the nearest corner point along each of the region edges connected to that point.

### 3.2.3 Maximal-disc representation

In order to calculate correspondence between parts of each cross-section, a description of the shape of the cross-sections is required, which can be used as a comparison. The distance field itself contains information about the cross-sections, from which it might be hoped some definition of shape could be formed. For instance, the direction of the gradient of the distance field always points towards the closest point on the cross-section, and hence also to the local ‘centre’ of the object. Unfortunately, although the distance field itself is contiguous, the gradient is not — in fact for the 10-14 chamfer it is +10 everywhere except at the ridges and canyons, (i.e. the local maxima and minima), where it is between +10 and 0. In order to generate a smooth surface, the change in interpolation direction must also be smooth, so this gradient can clearly not be used directly — another representation of the object shape is required.

One possibility is the use of the multi-scale medial axis (MMA). This has been used in image registration and it is suggested that it could form the basis for object-based interpolation [61]. However, the MMA is calculated directly from the grey-scale information in the image, rather

than from the segmented cross-sections which are available in sequential freehand 3D ultrasound. Blum [28] proposed an object-based representation which consisted of a set of discs just enclosed by an object (i.e. which touch the boundary at two or more points). The locus of centres of these discs forms the *symmetric axis*, which has also been termed the *skeleton* or *medial axis* of the object<sup>2</sup>.

Rather conveniently, this set of discs can be derived from the maxima of the distance transform which has already been calculated for each object cross-section. Firstly, all possible internal or external discs that touch the cross-section at two or more points (the *maximal set of discs* or MSD) are found. For city-block and chess-board distances, the MSD can be found simply from the local maxima of the distance field. The derivation of the MSD for chamfer  $3 \times 3$  codes is given by Arcelli and di Baja [9]. Essentially, any pixel with distance field value  $p$  is a disc centre if the following equation is satisfied:

$$p > q_{ij} - \begin{cases} 10, & i, j \in (1, 0 \quad -1, 0 \quad 0, 1 \quad 0, -1) \\ 14, & i, j \in (1, 1 \quad 1, -1 \quad -1, 1 \quad -1, -1) \end{cases} \quad (3.3)$$

where  $i, j$  are the pixel coordinates referenced to  $p$ , and  $q_{ij}$  is the distance field value of the pixel at  $i, j$ .

However, it is possible for discs in the MSD to be completely enclosed by the union of several other discs, making them redundant in terms of shape representation. Therefore, this set is reduced to eliminate any such discs, giving finally the *minimal set of maximal discs*, or MSMD. This process is entirely loss-less, and the MSMD can be used to exactly reconstruct the original cross-section. Several methods have been suggested for finding the MSMD. Nilsson and Danielsson [134] consider border coverage and build a “relation table” which relates border pixels to maximal discs, and is reduced by iteration. Borgefors and Nyström [33] suggest a simpler technique where a second image is created, each pixel representing the number of maximal discs that cover it. Redundancy can then be found by checking for discs which have a value greater than one for every contained pixel. These discs are removed and the process iterated.

In this case, a set of discs that completely covers the object is not required — the object border position is precisely defined by the distance field which will be interpolated. Only enough discs are needed to adequately define the shape of the cross-section. Having a smaller set of discs than the MSMD also decreases subsequent processing time. The MSD is therefore reduced iteratively, by keeping the largest disc at each step, and discarding any others that protrude from this disc by less than half of their radii.

In order to include external discs (those outside the cross-sections, rather than contained by them), the troughs and valleys are also considered, using the same criteria as for the internal discs. This allows correspondence of holes, concavities and gaps between objects, in addition to objects themselves. The region over which external discs are gathered is limited by the size of the distance-transformed area<sup>3</sup>. The set of internal and external discs for the distance-transformed region in Figure 3.6 is shown in Figure 3.9.

<sup>2</sup>It is also possible to represent an object with any other shape, for instance an ellipsoid [100]. Using ellipsoids can reduce the number of elements required for a given representational accuracy, but is over complicated for our purpose.

<sup>3</sup>In reality, there are situations where external discs exist outside this area, but this has little practical effect on the results.



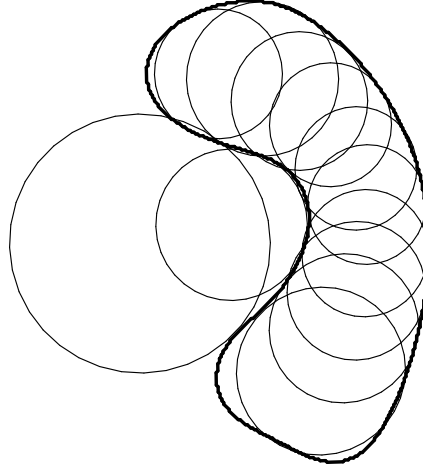


Figure 3.9: **Maximal-disc set used to determine correspondence.** The cross-section is the same as in Figure 3.6. Both internal and external discs are shown.

### 3.2.4 Calculation of region correspondence

In order to find the correspondence between a region on one cross-section and a neighbouring sequential cross-section, correspondence is first found for each of the discs representing the cross-section. This correspondence is a vector calculated for each disc on each cross-section, that points towards the corresponding part of a neighbouring cross-section<sup>4</sup>. This vector is a weighted sum of the relative locations of each of the discs on the neighbouring cross-section from the centre of the disc under consideration:

$$\vec{c}_a = \begin{cases} \frac{1}{\sum_{b \in \text{discs}} \omega_b} \sum_{b \in \text{discs}} \omega_b \vec{l}_{ab} & \text{if } \sum_{b \in \text{discs}} \omega_b > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

where  $\vec{c}_a$  is the correspondence vector for disc  $a$ ,  $\vec{l}_{ab}$  is the vector from the centre of disc  $a$  to disc  $b$ , and  $\omega_b$  is a weighting indicating the likelihood of correspondence of discs  $a$  and  $b$ , which are from cross-sections on neighbouring planes  $A$  and  $B$ . All distances and vectors are calculated in a plane that has the average orientation of both these planes, by projecting the discs from the planes along the normal to this average plane. Such projection allows correspondence to be calculated between cross-sections on non-parallel sequential planes.

The calculation of correspondence likelihood,  $\omega$ , for each pair of discs, is the most important step in this process. It is estimated by comparing, in each plane, the difference in distance transform values at the centre of each disc with the planar distance between the disc centres. This gives an error which, if small compared to the radii of each of the discs, is used as the likelihood estimate for this disc pair. If the error is larger than either of the disc radii, no correspondence is made between these discs, and the likelihood estimate is set to zero:

$$\omega_b = \begin{cases} \frac{1}{(\varepsilon_a^2 + \mu)} + \frac{1}{(\varepsilon_b^2 + \mu)} & \text{if } \varepsilon_a < r_a \text{ and } \varepsilon_b < r_b \\ 0 & \text{otherwise} \end{cases} \quad (3.5)$$

<sup>4</sup>This vector does not, therefore, simply connect a disc from one cross-section to that on another, as is the case in [151]

$$\varepsilon_a = \text{chamfer}(\vec{l}_{ab}) - |r_a - d_b|, \quad \varepsilon_b = \text{chamfer}(\vec{l}_{ba}) - |r_b - d_a| \quad (3.6)$$

$\omega_b$  is itself simply a combination of the ‘coverage errors’ for each disc,  $\varepsilon_a$  and  $\varepsilon_b$ , with a small value  $\mu$  chosen to be the square of the chamfer distance represented by one image pixel, which prevents an infinite weighting.  $\varepsilon_a$  is an estimate of the extent to which the distance field of plane  $A$  at the projected centre of disc  $b$  is determined by disc  $a$ . Conversely,  $\varepsilon_b$  is an estimate of the extent to which the distance field of plane  $B$  at the projected centre of disc  $a$  is determined by disc  $b$ .  $r_a$  and  $r_b$  are the radii of the discs, and  $d_a$  and  $d_b$  are the distance field values at the projection of each disc on the opposite plane. The function  $\text{chamfer}(\vec{l}_{ab})$  is an estimate of  $|\vec{l}_{ab}|$ , measured along a path formed entirely from lines parallel to or at  $45^\circ$  to the image edges. This is the same estimate as used in the distance transformation of the cross-sections, described in Section 3.2.2.

This method of weighting the contributions from each disc has a variety of features which make it attractive:

- The correspondence relationship is not symmetrical, and discs will only correspond with the nearest discs on the neighbouring cross-section if this relationship is reciprocal. This allows regions to be left unconnected, if appropriate.
- It is *not* necessary for regions (or discs) to overlap in order to correspond.
- Small discs will only tend to have a local effect on correspondence, unless the contour they represent is itself small. Larger discs on the same contour will take priority in the far field, since the ‘coverage error’ will be less for the larger disc.

The only limiting assumption is that at least one contour must be connected to one other contour on a neighbouring plane, i.e. at least one object must span the cross-sections.

Figure 3.10 shows an example of the calculation of coverage error, given in equation (3.6), for one disc from a pair of simple cross-sections. Essentially,  $\varepsilon$  is large if there are other discs on the cross-section that are nearer to the centre of the projected disc. This is equally the case for discs from other contours, as in Figure 3.10(a), or from the same contour, as in Figure 3.10(d). The criteria for there to be any correspondence at all is that  $\varepsilon_a < r_a$ , rather than  $\varepsilon_a = 0$ : this allows for the loose representation of the contour by a small number of discs — in practice  $\varepsilon$  will rarely be equal to zero. If there *is* correspondence, then  $\varepsilon$  indicates how strong that correspondence is; so for instance the disc in Figure 3.10(b) has a smaller error than that in Figure 3.10(c). This leads to a correspondence vector which is closer to the disc with the smaller error, as in Figure 3.10(e).

In the previous case, the converse error  $\varepsilon_b$  is nearly the same for each pair of discs, and as a result has little bearing on the correspondence. In general, both errors must be considered before correspondence between a pair of discs is determined. The addition of another contour to plane  $B$  in Figure 3.11 demonstrates this.  $\varepsilon_b$  for discs  $b_1$  and  $a_2$  is now greater than  $r_b$ , so even though  $\varepsilon_a$  has not changed, there is no longer any correspondence between these discs. The effect on the correspondence vector for  $b_1$  is to move it further away from  $b_2$ , thus ‘sharing out’ the larger contour on plane  $A$  more equally between  $b_1$  and  $b_2$ , as in Figure 3.11(b).

The correspondence  $\vec{c}$  is calculated for each internal disc, based on all the other internal discs; and for each external disc, based on all the other external discs, such that internal discs do not correspond to external discs, and *vice versa*. Any discs that have no correspondence at all are ignored in later processing. The result of this operation on the cross-sections of Figure 3.4(a) is shown in Figure 3.4(d) — note that in this case the external discs have no partners on neighbouring planes, and hence do not contribute to the correspondence estimate.

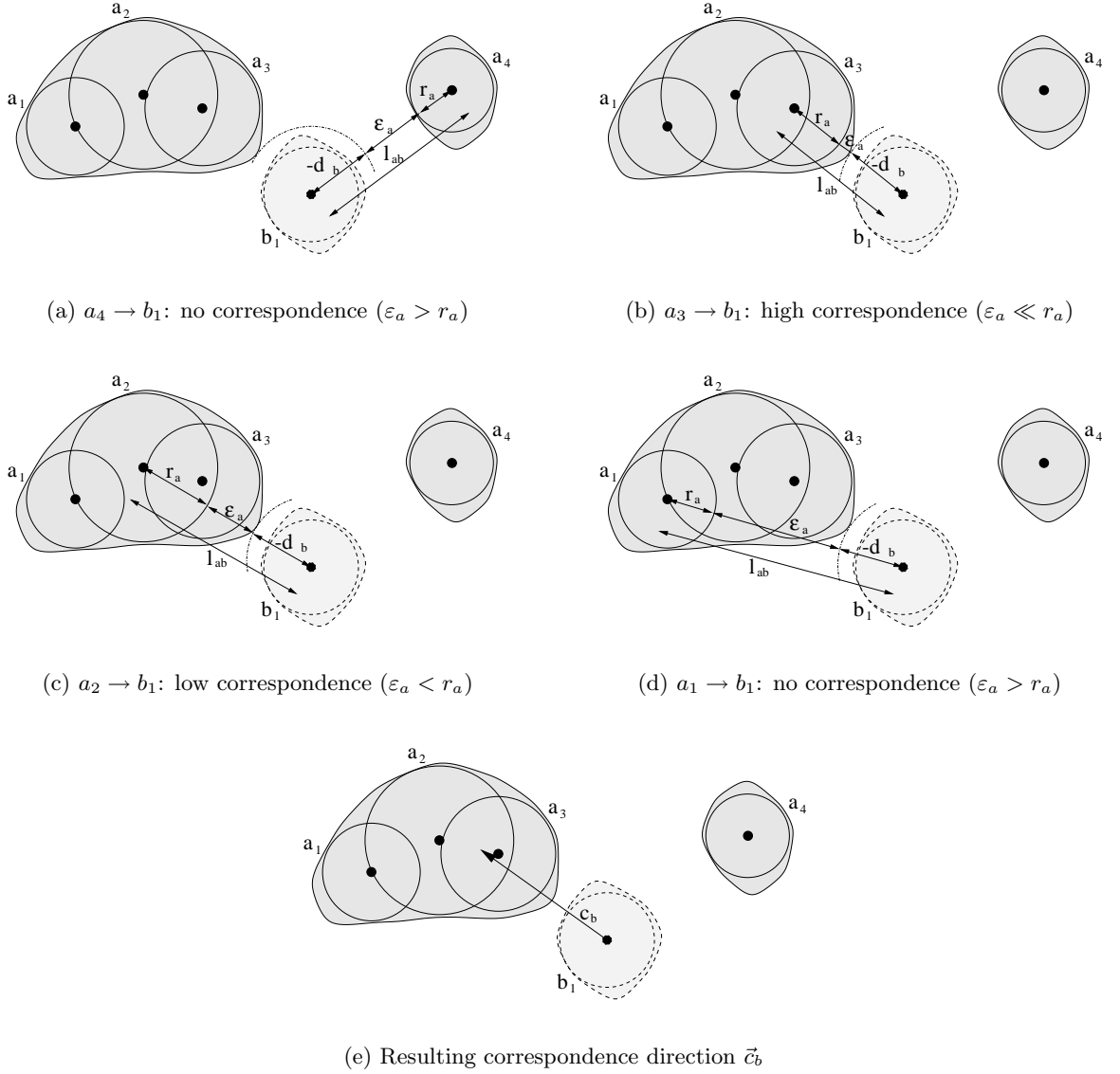


Figure 3.10: **Effect of  $\varepsilon_a$  on correspondence.** Discs  $a_1 \dots a_4$  (shown in dark grey) are from one plane, and disc  $b_1$  (shown light grey and dashed) is projected onto this plane from another. (a) to (d) show the calculation of ‘coverage error’  $\varepsilon_a$  for each pair of discs — the converse error  $\varepsilon_b$  is not shown, as in this case it is nearly the same for all disc pairs. The resulting correspondence vector for disc  $b_1$  is shown in (e).

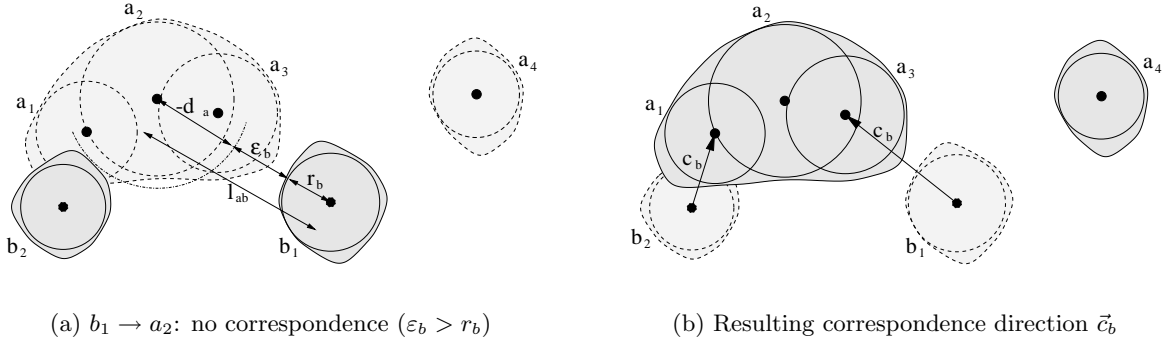


Figure 3.11: **Additional effect of  $\varepsilon_b$  on correspondence.** An additional contour and representative disc  $b_2$  has been added to those in Figure 3.10. This has no effect on the values of  $\varepsilon_a$ , but it changes those for  $\varepsilon_b$ , such that there is no longer any correspondence between disc  $a_2$  and  $b_1$ , as shown in (a). The resulting correspondence vector for discs  $b_1$  and  $b_2$  is shown in (b).

### 3.2.5 From region correspondence to point correspondence

Once the correspondence for each disc has been estimated, the correspondence at any point on each plane can also be estimated from a weighted sum of the disc correspondences on that plane. This is performed in much the same way as for disc correspondence, save that the information from both internal and external discs must now be combined. The point correspondence  $\vec{c}_p$  at any point  $p$  on the plane, is:

$$\vec{c}_p = \begin{cases} \frac{1}{\sum_{a \in \text{int}} \omega_a} \sum_{a \in \text{int}} \omega_a \vec{c}_a & \text{if } \sum_{a \in \text{ext}} \omega_a = 0 \\ \frac{1}{2(l_{\text{int}} + d_p) \left( \sum_{a \in \text{int}} \omega_a + \sum_{a \in \text{ext}} \omega_a \right)} \left[ (l_{\text{int}} + 2d_p) \sum_{a \in \text{int}} \omega_a \vec{c}_a + l_{\text{int}} \sum_{a \in \text{ext}} \omega_a \vec{c}_a \right] & \text{else if } d_p > 0 \\ \frac{1}{2(l_{\text{ext}} - d_p) \left( \sum_{a \in \text{int}} \omega_a + \sum_{a \in \text{ext}} \omega_a \right)} \left[ (l_{\text{ext}} - 2d_p) \sum_{a \in \text{int}} \omega_a \vec{c}_a + l_{\text{ext}} \sum_{a \in \text{ext}} \omega_a \vec{c}_a \right] & \text{otherwise} \end{cases} \quad (3.7)$$

where *int* implies internal discs, *ext* implies external discs, and  $d_p$  is the distance field value at the point  $p$ . The weighting  $\omega_a$  is defined as:

$$\omega_a = \frac{1}{\varepsilon_a^2 + \mu} \quad (3.8)$$

$$\varepsilon_a = \text{chamfer}(\vec{l}_{ap}) - |r_a - d_p| \quad (3.9)$$

This is similar to the definition in equations (3.5) and (3.6), save that the error is now based on the distance field  $d_p$  at the point position  $p$ , rather than at the centre of the projected disc.

There are three separate cases in equation (3.7). In the first and simplest case, if the geometry is such that there are no external discs, or no external discs have any correspondence, then the point correspondence depends solely on the internal discs, as in equation (3.4). Otherwise, the point correspondence is calculated by a combined weighting of internal and external discs, using a slightly different equation for points inside ( $d_p > 0$ ) and outside ( $d_p < 0$ ) the object cross-section. This combined weighting is further affected by the estimate of the distance to

the internal and external disc sets,  $l_{int}$  and  $l_{ext}$ . These estimates are calculated using a similar weighting to the first case of equation (3.7):

$$l_{int} = \frac{1}{\sum_{a \in \text{int}} \omega_a} \sum_{a \in \text{int}} \omega_a l_a, \quad l_{ext} = \frac{1}{\sum_{a \in \text{ext}} \omega_a} \sum_{a \in \text{ext}} \omega_a l_a \quad (3.10)$$

where  $l_a$  is the distance from the point to the centre of each disc.

The effect of these values in equation (3.7) is to ensure that the use of internal or external discs is governed by the distance transform value  $d_p$ ; i.e. how far the current point is inside or outside the object cross-section. Inside the cross-section, the weighting of internal discs increases as the point moves closer to the nearest internal disc. Outside the cross-section, the weighting of external discs increases as the point moves closer to the nearest external disc. At the edge of the cross-section ( $d_p = 0$ ), internal and external disc correspondence is evenly weighted.

The final correspondence for a point *between* neighbouring planes is found from the average of the values of  $\vec{c}_p$  from each plane.  $\vec{c}_p$  is calculated at the intersection with each plane, of the average plane normal through this point. As with the disc correspondence, this vector is calculated in a plane which has the average orientation of the neighbouring planes.

### 3.2.6 Interpolation using point correspondence

The previous sections describe how to create distance transforms of each of the cross-sections, and calculate correspondence for each point between neighbouring cross-sections. How can this information be used to improve the surface interpolated between these cross-sections? There are two main classes of interpolation techniques — those which interpolate data only from the planes that bound the point of interest, and those which interpolate data from all planes within a bounding volume of the point of interest. The latter class includes all of the numerous techniques for interpolating scattered data. In order to interpolate surfaces in the sequential framework, the former class must be used.

This class, *bounding plane* interpolation, includes as a subset those techniques that apply to data from parallel planes. In fact, very few such techniques exist for non-parallel planes, and many of these have been developed with 3D freehand ultrasound specifically in mind, as discussed in Section 3.1.3. Both the *order* and the *direction* of interpolation must be chosen carefully for this task.

The order of interpolation is in fact determined by the type of data to be interpolated: i.e. 2D distance transforms. There are two features of this data which make linear interpolation the only sensible solution. Firstly, the distance transform represents a 2D distance, and is not the same as that which would be calculated by a 3D distance transform on the eventual surface. In particular, this 2D distance is dependent on the orientation of the cross-sections, and not just the object from which the cross-sections are drawn. Hence it will not vary consistently across non-parallel cross-sections. Secondly, the distance transform is only  $C_0$  continuous, so it makes no sense to impose  $C_1$  or higher continuity in the third dimension by using cubic interpolation. The effect of this choice is demonstrated in Figure 3.12 — it can be seen from this figure that, far from improving the result, cubic interpolation actually degrades the surface estimate. Disc-guided interpolation is also shown for comparison.

Figure 3.13(a) to (c) show three possible options for the interpolation direction between two non-parallel planes<sup>5</sup>. In each of these cases, the distance field value at the point  $p$  is calculated by weighting the distance codes with the lengths,  $l_1$  and  $l_2$ , as in equation (3.11):

<sup>5</sup>For data which comes from randomly oriented planes, any given point may be bounded by many of these

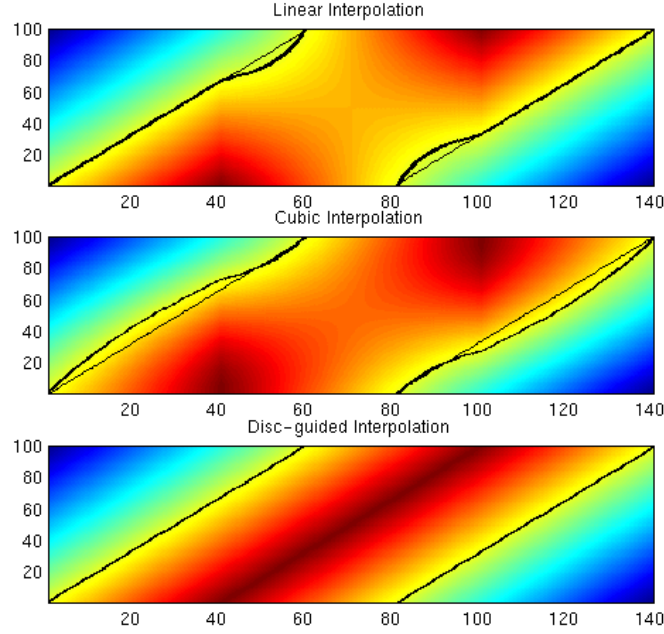


Figure 3.12: **Effect of interpolation order on distance fields.** The graphs show a cross-section through the centre of a cylinder such as in Figure 3.1, containing the interpolated distance field between two cross-sections, one at the bottom and one at the top. The thin black line is the actual cylinder surface, the thick black line the interpolated surface. Cubic interpolation is achieved by using two additional cross-sections surrounding those given in the figure. Disc-guided interpolation is a form of linear interpolation.

$$d_p = \frac{l_2 d_1 + l_1 d_2}{l_1 + l_2} \quad (3.11)$$

where  $d_p$  is the interpolated distance code at the point  $p$ . The effect of using each of these schemes is shown for a cylinder scanned with a fanning action in Figure 3.14. The closest-points method, although simple, produces unacceptable artifacts when the planes are highly non-parallel. This is because the line of interpolation is bent at the interpolated point, causing the surface also to be curved, as in Figure 3.14(b). It is more natural to connect the contours with surfaces that are as straight as possible, hence the line of interpolation must also be straight. Interpolating by using the average normal to both planes corrects this tendency, but suffers from the same problems outlined in Section 3.1.2 for parallel planes.

As already discussed, interpolating in a different direction, for instance in the line joining the centroids of the cross-sections, can produce better results, as in Figure 3.1(d). However, there are restrictions on the set of directions that can be used for interpolation when the planes are not parallel. Figure 3.13(d), for example, demonstrates the use of centroid-guided interpolation on overlapping planes. In this case, the interpolation direction is such that the intersections with both planes are on the same side of the point,  $p$ . If  $d_1$  and  $d_2$  are both positive, and have similar values, the distance code at the point  $p$  will always be positive, irrespective of the magnitude of the planes, hence the interpolating data may be from more than two planes. The techniques described here are appropriate for sequential data, in which case the two interpolating planes are those that bound the point in sequence. Freehand ultrasound data is inherently sequential, and the manner in which it is acquired produces gradually varying, rather than randomly oriented, planes.

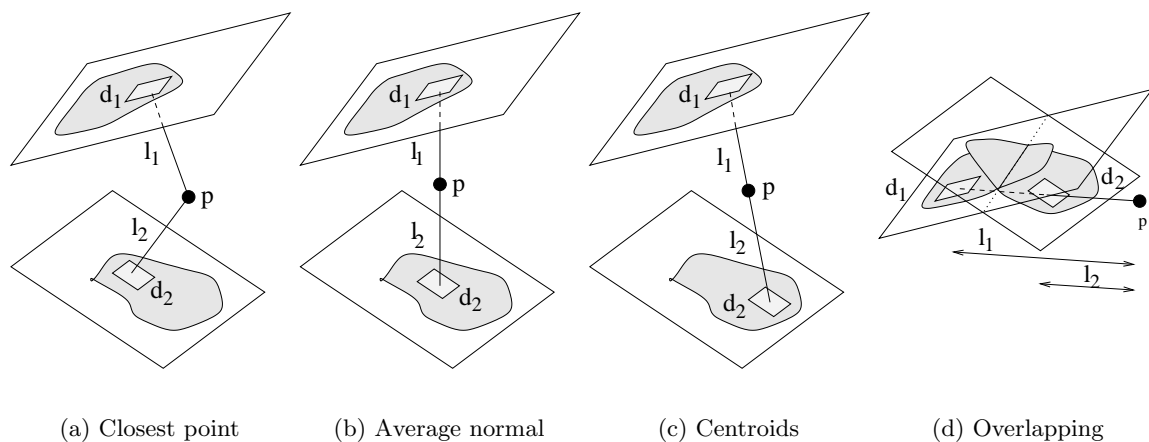


Figure 3.13: **Choice of interpolation direction.**  $d_1$  and  $d_2$  are the sampled distance field values in each plane. The interpolation in (c) is parallel to the line joining the object centroids.

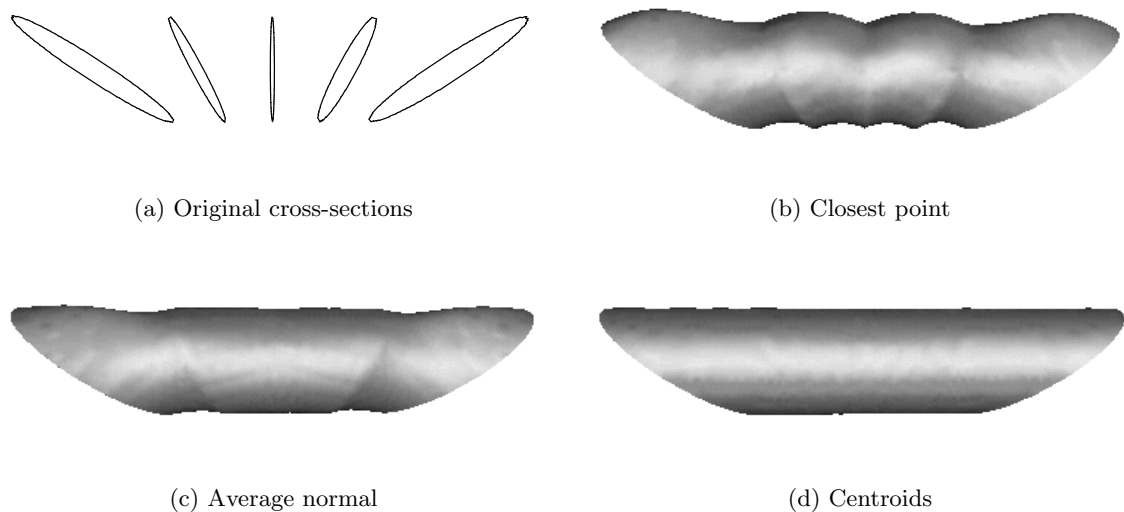


Figure 3.14: **Effect of interpolation direction for non-parallel planes.** Interpolation of a cylinder defined by non-parallel cross-sections is shown, for the three types of interpolation presented in Figure 3.13. The rendered surfaces are the zero crossing points of the interpolated volume.

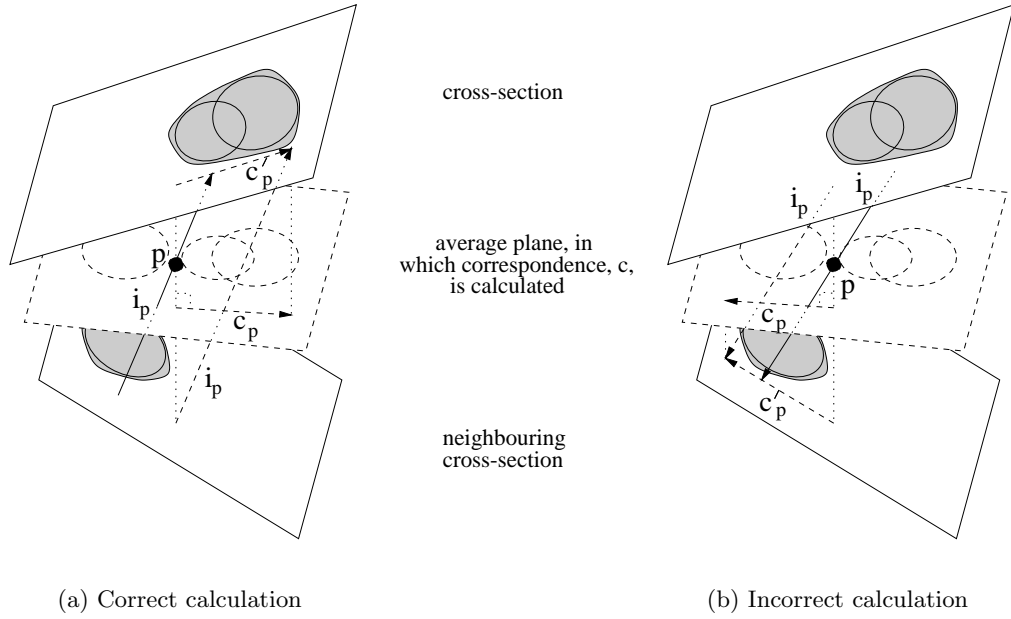


Figure 3.15: **Construction of interpolation direction from correspondence.** To ensure that the ‘spilling out’ artifact illustrated in Figure 3.13(d) is avoided, the correspondence vector  $\vec{c}_p$  is always projected onto the plane that causes its projection  $\vec{c}'_p$  to point away from  $p$ . This ensures that the interpolation direction  $\vec{i}_p$  intersects each plane at either side of the point  $p$ .

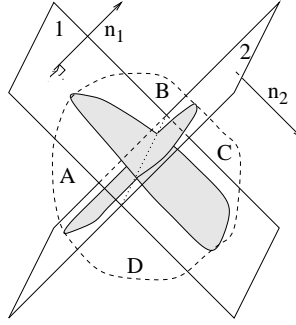
of the lengths  $l_1$  and  $l_2$ . This is the case whether or not  $l_1$  is considered to have the same sign as  $l_2$ . In effect, the object ‘spills out’ in the direction of interpolation, potentially to infinity. The interpolation direction must be chosen such that the intersections with each plane are on opposite sides of the point to be interpolated.

Figure 3.15 demonstrates how the interpolation direction,  $\vec{i}_p$ , used to interpolate a new value for the point, can be derived from the point correspondence vector,  $\vec{c}_p$ . Figure 3.15(a) and (b) are both reasonable solutions which generate similar values of  $\vec{i}_p$ , however only the former guarantees that the ‘spilling out’ effect of Figure 3.13(d) is never apparent, even for overlapping planes.

For each interpolated point in-between two planes, the point correspondence vector,  $\vec{c}_p$ , is calculated at the orthogonal projection of this point onto the average plane. This vector is projected onto one of the neighbouring planes, as described in Figure 3.15, to generate  $\vec{c}'_p$ .  $\vec{c}'_p$  is then added to the vector connecting the two planes through  $p$ , along the average planar normal. Finally, the resulting vector is normalised to give the interpolation direction,  $\vec{i}_p$ . This interpolation direction is used to determine both the distance field values, and lengths, used in the linear interpolation equation (3.11).

There are two additional complications which arise because of the possibility of the scan planes intersecting each other. An extreme version of this situation, with two orthogonal planes, is shown in Figure 3.16. The first task is to decide which regions should be interpolated from these planes. The four regions in the example are all equally ‘enclosed’, but presumably only two of them should be interpolated. Since the scan planes are sequential, it makes sense to interpolate only the area which was swept out during the scan. This can be deduced from the direction of the normals to each plane — regions that are ‘in front’ of one plane and ‘behind’ the other plane are interpolated (corresponding to  $B$  and  $D$  in Figure 3.16).



Figure 3.16: **Object scanned with two orthogonal scan planes.**

Secondly, if the intersection of the scan planes passes through the object, then part of the swept volume between these scan planes must have already been swept out by previous planes. The volume calculation for these regions should be negative, else the total volume will be overestimated. This can once again be resolved by examining the plane normals — if the normals point more from the first towards the second plane, then the volume is positive (region *B*), else it is negative (region *D*), i.e.:

$$\text{sign}(\text{volume}) = (\vec{n}_1 \cdot \vec{n}_{avg}) > 0 \quad (3.12)$$

where  $\vec{n}_{avg}$  is a vector connecting plane 1 to plane 2 within the region, along the average of the plane normals  $\vec{n}_1$  and  $\vec{n}_2$ .

### 3.3 Current methods of surface visualisation

Section 3.2 describes a method of determining the location of a surface reconstructed from cross-sections, at any point in space. However, this in itself does not allow the surface to be visualised. There are two ways to display an isosurface from such interpolated distance field data: volume-based or polygon-based rendering. In the volume-based approach, the sampled data is displayed directly, e.g. by ray casting or splatting [99, 170]. However, this is a complex operation for freehand 3D ultrasound data, which is not sampled to a regular cubic lattice. The alternative polygon-based approach, where a geometric representation of the isosurface is first created from the data, has a variety of advantages:

- There is currently more hardware support for polygon-based rendering than for volume-based rendering. This may equate to faster display, and hence better interactivity with the data, depending on the balance between processor and display hardware speed.
- Sequential interpolated data on non-parallel planes can lead to regions that are swept out more than once, i.e. have multiple data values. This can be handled with polygon-based approaches by allowing the surface to follow the sweep of the planes. This sort of data cannot be rendered directly using volume-based approaches.
- A geometric representation of the surface can also be used to provide an additional volume estimate.

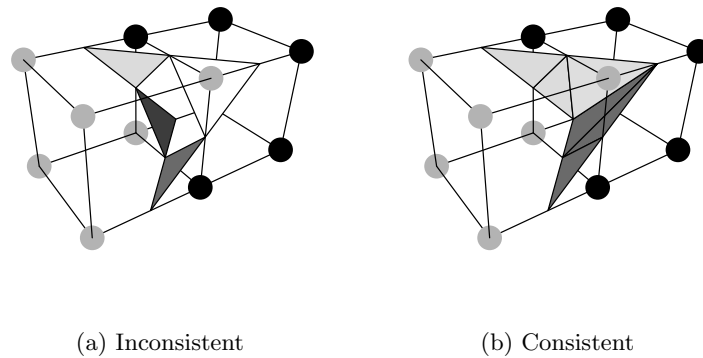


Figure 3.17: **Marching cubes.** (a) The use of complementary symmetry can generate topological inconsistencies. (b) This can be corrected by use of an alternative triangulation for the complementary case. Grey vertices indicate data on one side of the surface, and black on the other.

The current polygon-based methods for extracting an isosurface are reviewed in Section 3.3.1. A class of techniques for post-processing such surfaces to improve the quality of the triangulation is briefly reviewed in Section 3.3.2.

### 3.3.1 Marching cubes and its derivatives

*Marching cubes* (MC) [115] is a simple method for constructing isosurfaces, which was introduced for 3D medical data. The basic principle is to reduce the problem to that of triangulating a single cube, intersected by the isosurface. Surface intersection points are defined along the edges of the cube, by linear interpolation of the sampled data at each corner. These points become the vertices of one to five triangles, to form a polygonised surface patch. The whole isosurface can be triangulated by ‘marching’ this cube through the data and creating surface patches whenever the isosurface passes through the cube. There are  $2^8$  different cases for the triangulation of a cube, based on whether sampled data at the corners is greater or less than the surface threshold; however this can be reduced to only 15 by symmetry. The triangulations for these 15 cases are stored in a look-up table — it is only necessary to compute the symmetry and the case at each cube location.

There are two well known problems with this otherwise very simple and attractive algorithm, connected with the *topology* and the *regularity* of the triangulation.

The original MC algorithm does not produce surfaces that are topologically consistent with the data from which they are derived. This is due to the arbitrary choice of triangulation for a cube with any face containing two diagonally opposite corners on one side of the isosurface, and two which are on the other. If the same triangulation is chosen for both the cubes containing this face, then there will be a hole in the surface, as shown in Figure 3.17. Several methods have been proposed to correct this situation [135, 186]. The simplest of these employs a different cube triangulation for complementary symmetries of ambiguous faces [123], which results in a bias in the topology towards one side of the surface. More complex methods use more sophisticated interpolation to determine the cube triangulation [128], or incorporate data from outside the cube.

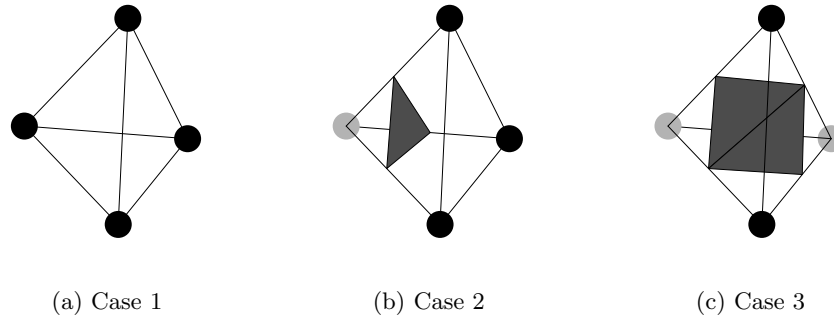


Figure 3.18: **Marching tetrahedra.** The three possible cases, two of which require triangulation, and the resulting triangulations. (c) Case 3 can be rendered as a quadrilateral rather than two triangles, since the surface is guaranteed to be planar.

This ambiguity can be eliminated by tessellating space with tetrahedra rather than cubes. Tetrahedra only have  $2^4$  possible triangulations, which reduce to 3 by symmetry. These are shown in Figure 3.18. A tetrahedral tessellation can be constructed by dividing each cube into five tetrahedra [74, 141]. Unfortunately, this introduces an additional ambiguity, since the symmetry of the cube subdivision has to alternate between cubes, in order to align the faces of the tetrahedra. There are, therefore, two possible tessellations for a given cubic lattice, which can generate opposed topologies. This ambiguity can only be resolved by using cubic interpolation, rather than linear, thereby allowing the isosurface to intersect the tetrahedral edges more than once [188]. However, this is much more complex, and the resulting look up table has 59 cases — many more than the original MC method.

Recently, Chan and Purisma proposed a scheme, referred to here as *marching tetrahedra* (MT), where space was tessellated into tetrahedra based on a body-centred cubic lattice [38]. The lattice can also be formed from a higher resolution cubic lattice. In contrast to the subdivided cube method, the resulting tetrahedra are all identical, and there is no ambiguity in the tessellation. The tetrahedra are also more regular than those formed by subdividing a cube. The resulting triangulation is more uniform, and has fewer of the sharp edges associated with tetrahedral decompositions. Using a body-centred cubic lattice also results in better sampling efficiency compared with the cubic lattice [38]. The main disadvantage of tetrahedral schemes is that they create an even larger number of triangles than MC for a given data set. This aggravates the second problem with MC, that of the *regularity* of the resulting surface triangulation.

### 3.3.2 Mesh simplification

The MC algorithm creates on average three triangles for each cube intersected by the surface; more if different complementary case triangulations are used. In addition, where the cubic lattice just intersects the surface, these triangles can be arbitrarily small or thin. In typical medical applications this often results in more than 500,000 triangles — with substantial associated storage and rendering requirements. There is a large body of literature on methods of post-processing triangulated surfaces to reduce this number of triangles, whilst maintaining the ‘quality’ of the surface. These *mesh simplification* methods have recently been reviewed in [43].

The driving force behind much of the work on mesh simplification is the reduction of storage and rendering times, often with complex virtual reality scenes in mind. However, badly propor-

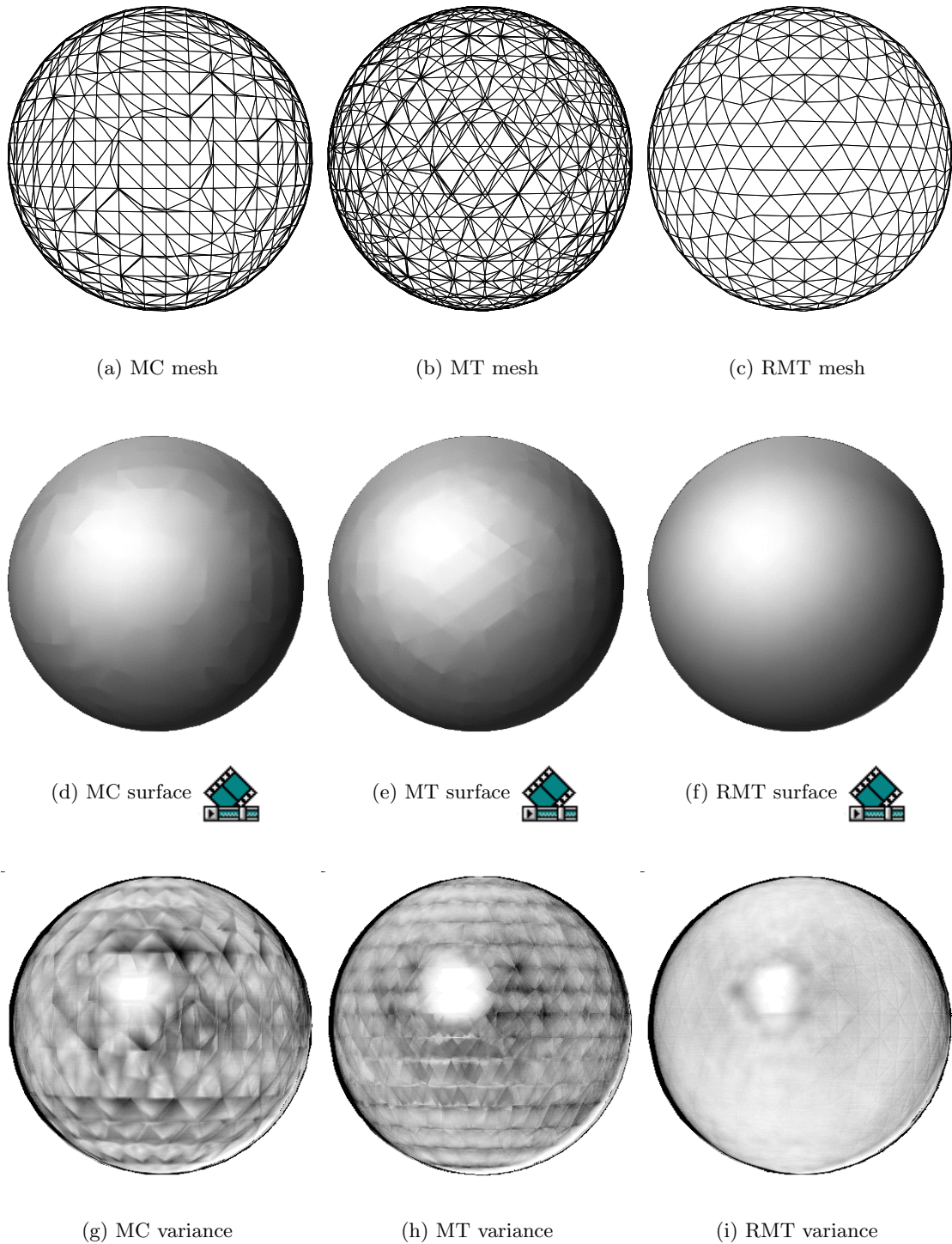


Figure 3.19: **Effect of triangle condition on interpolated shading.** Results are shown for marching cubes (MC), marching tetrahedra (MT) and regularised marching tetrahedra (RMT). (a), (b) and (c) show the triangulated surface, (d), (e) and (f) a smooth shaded rendering of this surface. (g), (h) and (i) show the standard deviation of the grey level of each pixel in a set of 20 renderings with different orientations. White represents a standard deviation of 0 grey levels (out of 255), and black represents 8 or more grey levels.

tioned triangles can also result in poor quality renderings when any form of interpolated shading is used to make the surface appear smooth. This is particularly the case for Gouraud shading, which is the simplest form and has the most hardware support. Specular reflections using this technique highlight the original mesh — very thin triangles tend to appear as sharp edges and very small triangles as sharp points. Figure 3.19 demonstrates this effect for a sphere — (a) and (d) show the isosurface formed using MC and (b) and (e) show that formed using MT. In addition, mesh simplification algorithms do not take advantage of the regular way in which the isosurface is created, since they are mostly designed to operate on arbitrary (although usually closed and oriented) triangle meshes.

Mesh simplification methods are generally iterative, and many are suitable for generating meshes of various resolutions, which can then be used for efficient rendering at different distances [43]. One of the simplest and fastest groups is *vertex clustering*. Here vertices in a region are merged to form one new vertex, and triangles containing more than one of the merged vertices can then be removed [5, 155]. *Edge collapsing*, where an edge is reduced to zero length and two triangles removed, can be seen as a subset of *vertex clustering* in which only two vertices are clustered.

The main differences between these methods are how they are iterated, and what cost function is used to drive the vertex clustering and determine whether clustering is allowed in any one case. *Quadric error metrics* [64] is a fairly sophisticated example, in which an error is calculated for each vertex, based on the sum of squared distances from the planes that originally contained that vertex.

There are only two published instances where the ideas of vertex clustering are applied during, rather than after, isosurface construction. Both of these employ tetrahedral lattices. In [77], vertices that are near to the corners of the tetrahedra (within 5% of the tetrahedra edge length) are ‘snapped’ to the nearest corner location, thus eliminating any triangles with edges shorter than this distance. In [73, 74], *surface perturbation* is allowed as an option, whereby positive values sampled at the tetrahedral vertices are set to zero. This ensures that all new vertices are positioned at the tetrahedral vertices, and the triangulation consists entirely of tetrahedral faces.

### 3.3.3 Related applications

If the original data is sampled sufficiently densely, isosurface triangulation can be used directly to generate a surface from this data. In this case, the values from which the surface intersections are interpolated are the grey scale data values, rather than the interpolated distance values presented in the previous section. This is the context in which marching cubes was originally presented.

Implicit surfaces, i.e. surfaces which can be defined by a mathematical function, can also be displayed by using isosurface extraction techniques. In this case, the function is sampled to a regular grid, from which the isosurface is extracted. This is the technique used to generate the spheres in Figure 3.19. As has been noted in [78], this can also include discrete surfaces as long as it is possible to *implicitise* them with an appropriate function.

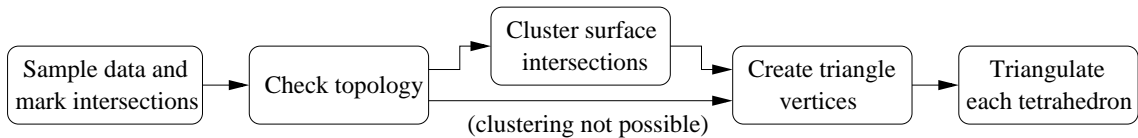


Figure 3.20: **Regularised marching tetrahedra: overview.** The data from which an isosurface is required is first sampled on a tetrahedral grid. Intersections of the isosurface with edges of the tetrahedra are marked at each sample point. Following this, the surface topology at each sample point is examined, and surface intersections are clustered if this can be done without affecting topology, to form the vertices of the triangle mesh. Finally, each tetrahedron is triangulated, using these new vertices, keeping only those triangles which contain three different vertices.

## 3.4 A new approach: regularised marching tetrahedra

### 3.4.1 Overview

The main idea of this approach is to combine the two fastest and most simple algorithms — MT for isosurface extraction, and vertex clustering for mesh simplification — using each to improve the performance of the other. Vertex clustering can be used to produce a regularised triangle set from MT, and basing the clustering around the original tetrahedral lattice allows the preservation of the original topology, which is otherwise difficult to achieve in clustering methods. The result is a triangulated surface consisting of near regular triangles that can be generated in nearly the same time as a surface using MT alone. A sphere generated using this technique, *regularised marching tetrahedra* (RMT) is shown in Figure 3.19(c) and (f).

The RMT algorithm presents the following benefits:

- Simplification using vertex clustering is performed *before* triangulation, taking advantage of the original sampled data. This allows the original topology to be preserved through the clustering process.
- The additional time required for clustering is offset by a reduction in the number of stored points and triangles, resulting in a fast algorithm.
- Triangle condition is dramatically improved, which results in better quality interpolated shading.
- Triangle count is typically reduced by 70% compared to that obtained from MT, and by 40% compared to MC. This reduces the rendering time for the surface, which in turn improves the real-time interaction.
- Since the technique does not require searching of the point or triangle sets, these can be stored quickly and efficiently in simple lists.

An overview of the process is shown in Figure 3.20. The choice of sampling grid and the marking of surface intersections at each sample point is considered in Section 3.4.2. Calculation of the local surface topology at each sample point is discussed in Section 3.4.3. Clustering of surface intersections and triangulation of the tetrahedra to form a surface mesh are described in Section 3.4.4. A more complex approach, taking into account data curvature to position the clustered surface intersections more appropriately, is presented in Section 3.4.5. Finally, some issues of implementation are discussed in Section 3.4.6.



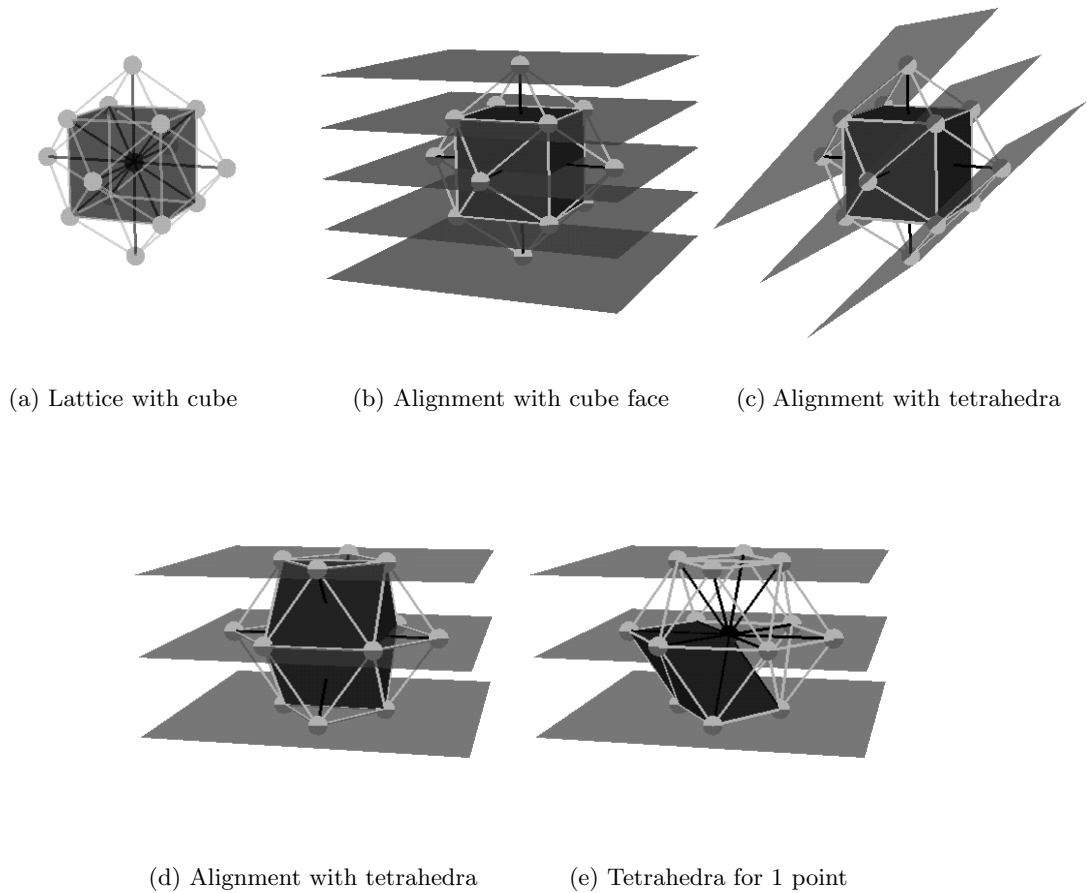


Figure 3.21: **Body-centred cubic sampling.** (a) shows a body-centred cubic lattice. (b) and (c) are two possibilities for defining sample planes through (a). In (b), the distribution of planar points is square, whereas in (c), the tetrahedral faces are aligned with the plane. (d) shows the alignment of (c), rotated such that the planes are horizontal, and (e) the six tetrahedra associated with each point for this alignment.

### 3.4.2 Choice of sampling grid

#### Cubic versus tetrahedral lattices

The first step is to construct a suitable lattice on which to sample the data, in this case the interpolated distance field values. As in [38], a body-centred cubic lattice is used — this gives better sampling resolution than the conventional cubic lattice, and can be subdivided into a set of identical and nearly regular tetrahedra. Each of these tetrahedra have one pair of opposite edges of length 2 units, and the remaining four edges of length  $\sqrt{3}$  units. This lattice is shown in Figure 3.21(a), along with the cube on which it is based (rendered semi-transparent to show the hidden edges). Sample points are connected to their nearest neighbours to form four tetrahedra per cube face.

In [38], the planes containing the sample points are aligned with the faces of the cube, as in Figure 3.21(b). The orientation can also be chosen so that the planes lie along the diagonal of two cube faces, as in Figure 3.21(c). The first orientation leads to sample points spaced on a square grid on each plane, which is more suitable if the original data is arranged in this manner.

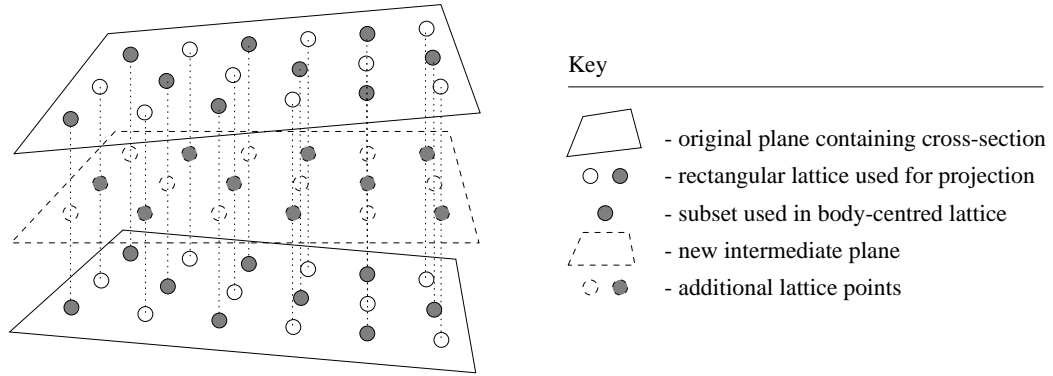


Figure 3.22: **Progression of lattice for non-parallel planes.** A rectangular grid of lattice points is projected onto the next plane along the average normal to the planes. If the planes are too far apart, additional lattice points are defined on intermediate planes, in a similar fashion. The body-centred lattice used for triangulating is formed from a subset of this projected lattice, which alternates with each plane.

However, the tetrahedra *intersect* these planes, such that the space between two planes cannot be exactly tessellated by them. In contrast, the second orientation leads to non-square (though still rectangular) spacing of data points on each plane, but with no tetrahedra intersecting a plane. This orientation simplifies the processing for the first and last planes in a sequence, and is assumed in the following description. The sample points within the plane are spaced 2 units apart in one direction and  $\sqrt{2}$  in the orthogonal direction. This is not a problem for distance functions based on cross-sections of the original data, since this data can be interpolated to any lattice.

### Sampling grid for sequential non-parallel planes

In sequential freehand 3D ultrasound, the data is on non-parallel planes. In this case the lattice of Figure 3.21(c) is projected to the next plane as in Figure 3.22. This means tetrahedra vary in size from regions where the planes are close, to those where they are further apart. It also means that tetrahedra will not be completely regular where the bounding planes are not parallel. However, it ensures that lattices between each pair of planes meet at the in-plane sample points, and hence can be constructed sequentially from such data.

It is even possible to handle data from overlapping planes, as in Figure 3.23, although a few modifications are required. Firstly, the direction of sweep of the sequential planes affects the orientation of the triangles. Unfortunately, this imposes two opposite requirements. For correct surface shading, all triangles should be oriented consistent with outside/inside criteria. For correct volume measurement, when the sweep direction changes, the triangle orientation should flip such that the volume is reduced as a result. If volume calculation is performed after triangulation, an additional flag is required per triangle, indicating the sweep direction of the planes that the triangle lies between.

Secondly, surface intersection points which are on planes at the extremity of a sweep direction, are only allowed to move within that plane. This ensures that the edge of the surface will still be contained within the original plane after vertex clustering. Thirdly, sample points that are close to the intersection of two overlapping planes are moved on to the line of intersection. This ensures that there are no tetrahedra which are themselves intersected by this line, and hence



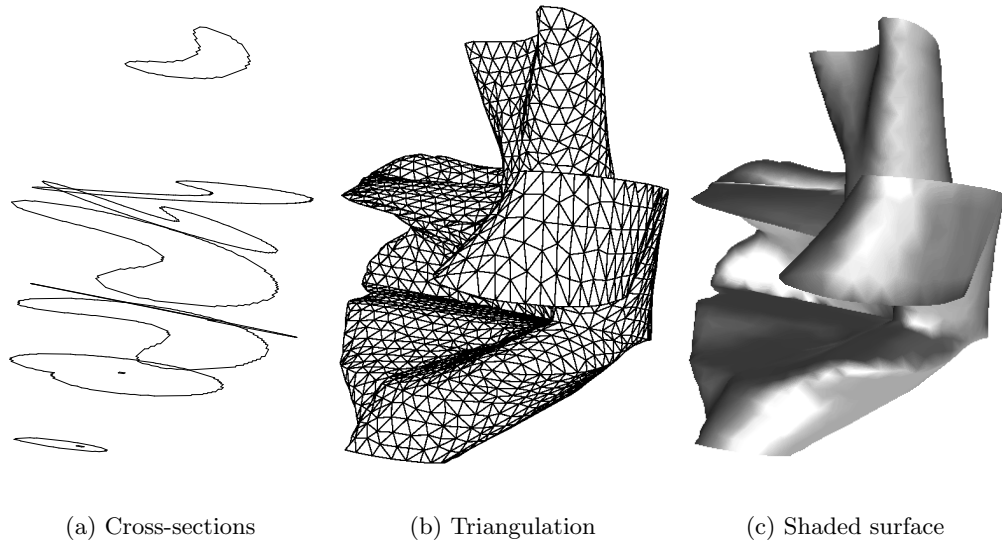


Figure 3.23: **Overlapping cross-sections.** The contours have been highly mis-registered such that the planes in which they are defined overlap each other. The resulting surface is self intersecting, but consistent with the mis-registered contours.

would not be correctly triangulated.

### Marking surface intersections

Each sample point has 14 edges radiating from it, shown in black in Figure 3.21(a) for the central point. Since each edge is associated with 2 sample points, only 7 edges need be examined for each new sample point. All of the 14 edges are assigned a reference in the form of a bit position, as in Figure 3.24. If any edge examined intersects the isosurface (i.e. the sampled distance values at each end point are of opposite sign) then the appropriate reference is marked in the bit field of the *nearest* sample point associated with that edge. Hence, each sample point contains a bit field indicating the existence of any *near* surface intersections, and on which edges these intersections are found. This way of labelling surface intersections greatly simplifies subsequent topology calculations.

### 3.4.3 Topology preservation

The next step is to determine whether the near surface intersections that have been marked at each sample point can be combined into new vertices, whilst preserving the topology. There are a variety of situations where clustering these surface intersections would lead to a change of topology in the resulting surface. The possible cases, shown in Figure 3.25, are:

- (a) **Closed surface** The sample point has a value opposite in sign from all the surrounding points. Clustering the surface intersections to a single vertex would result in the elimination of this surface, so the original surface intersections remain.
- (b) **Hole** There is a single (open) surface, but with a hole in it. Clustering the surface intersections would close up the hole, so once again the original surface intersections remain.

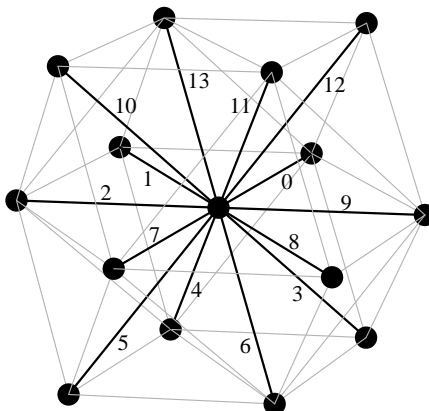


Figure 3.24: **Edge labels for marking intersections.** The 14 edges associated with one sample point are given the labels  $0 \dots 13$ . Edges  $0 \dots 6$  are the 7 new edges introduced when the sample point is traversed for the first time. It is convenient to label the opposite set of edges by adding 7 to each of these labels.

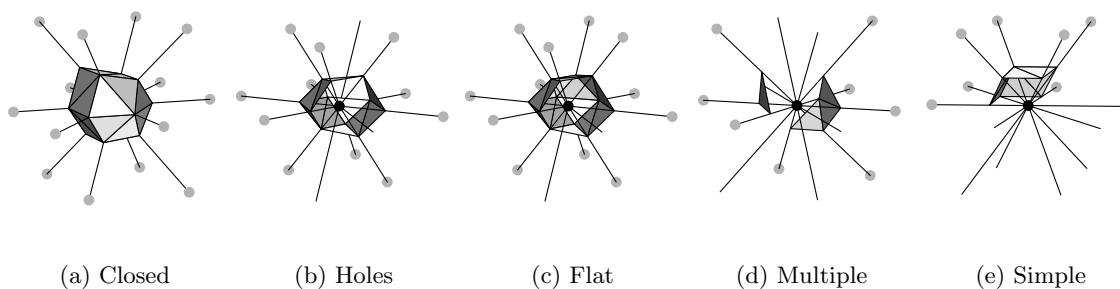


Figure 3.25: **Topology preservation.** (a) to (e) show the topological cases considered during vertex clustering. (a), (b) and (c) cannot be clustered. (d) and (e) can be, with one new vertex per surface. Clustering of the vertices in (c) might lead to non-manifold surfaces.

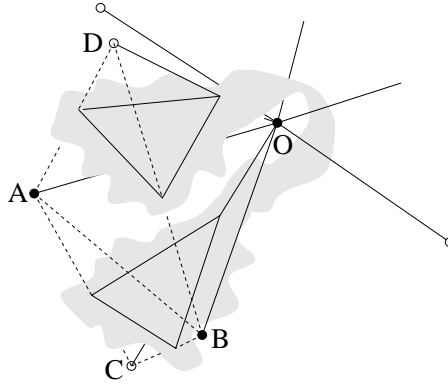


Figure 3.26: **Criterion for a ‘flat surface’.**  $O$  is the current sample point,  $A \dots D$  are some of the neighbouring sample points. The existence of edges  $OA$  and  $OB$  with no near surface intersection, and edges  $OC$  and  $OD$  with near surface intersections, will lead to a collapsed surface if it is connected around  $O$ , and the surface intersections on edges  $AD$  and  $AC$  are both near  $A$ , or those on  $BC$  and  $BD$  are both near  $B$ .

- (c) **Flat surface** There is a single (open) surface with no hole, but the boundary of the surface is such that clustering the surface intersections might result in the closing up of this boundary. Hence the surface would be ‘flattened’, leading to two edges or two triangles being back to back. Again, the original surface intersections remain.
- (d) **Multiple surfaces** The surface intersections form more than one separate surface; each can be clustered into one new vertex.
- (e) **Simple surface** The simple (and most common) case, where there is only one surface, that does not fall into case (c). This can be clustered to a single new vertex with no change in surface topology.

Since the surface intersections are each recorded as one bit of a bit field, the determination of the topological case can be performed simply by using logical operations on these bit fields. The operations for each of the topological cases in Figure 3.25 are described in Appendix C.2.

The criterion for a ‘flat surface’, shown in Figure 3.26, is the most complex case. Each of the 36 outer edges (i.e. those joining the surrounding sample points, for instance  $AB$ ) are examined to see if any fall into the category described in the figure. The surface intersections surrounding such holes cannot all be clustered, as this would result in the closing up of portions of the surface, potentially making it non-manifold.

#### 3.4.4 Clustering and triangulation

Once a collection of surface intersections near to the sample point have passed the topology checks, they can be clustered to form a single new vertex. These vertices are stored in a simple list, and the vertex reference marked in each of the edges containing the surface intersections from which the vertex was formed. Each sample point is surrounded by 24 tetrahedra, and each tetrahedron is associated with 4 sample points. Therefore there are 6 new tetrahedra associated with each sample point, as in Figure 3.21(d). All of the tetrahedra with surface intersections on any edges are triangulated using either case (b) or (c) from Figure 3.18, as appropriate. However,

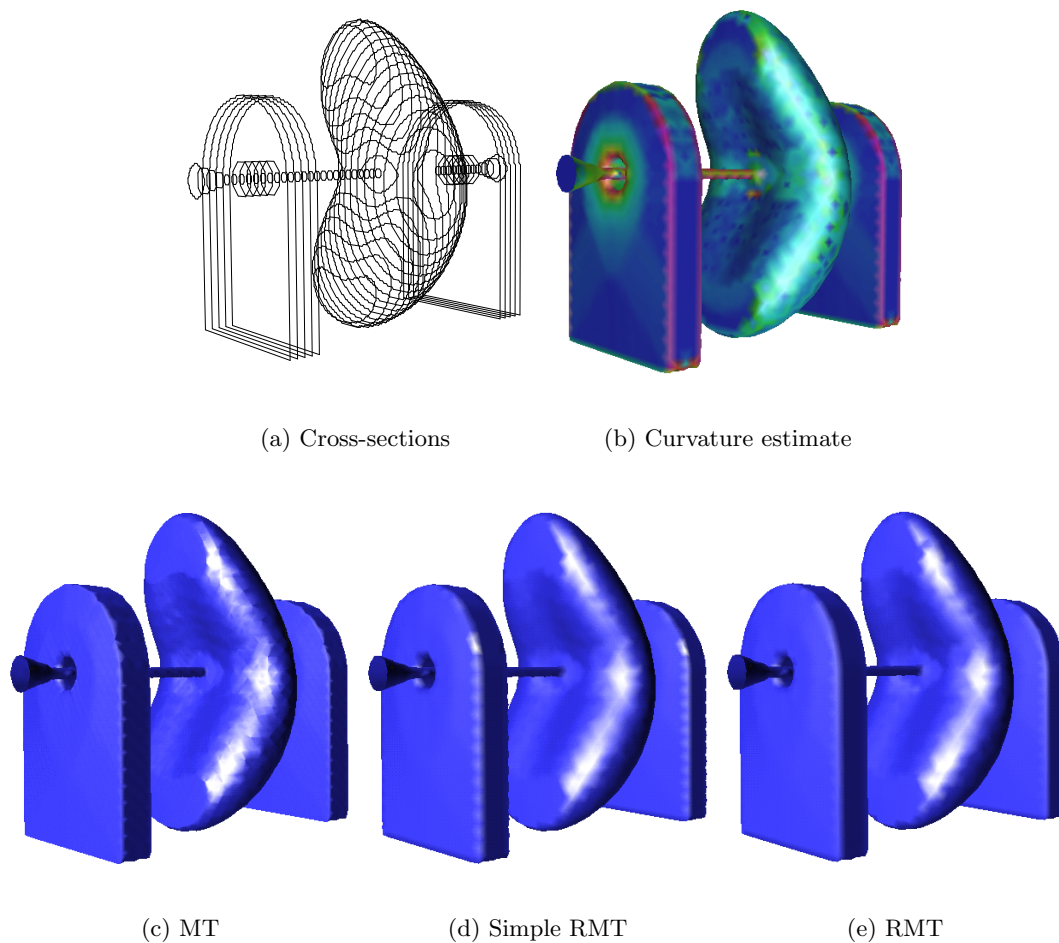


Figure 3.27: **Effect of clustering.** (a) shows the object cross-sections from which a distance function was derived. (c) shows the result of using MT, (d) includes vertex clustering, and (e) also includes weighting of the vertex position using curvature. (b) shows the curvature estimate mapped to colours — blue indicates low curvature through green then red indicating high curvature.

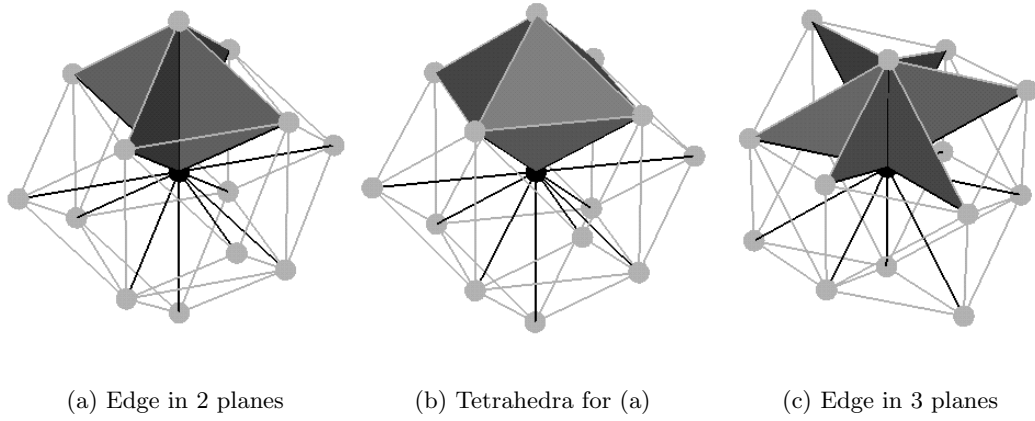


Figure 3.28: **Geometry for gradient and curvature calculation.** An edge can be contained in (a) four or (c) six faces. Calculations of gradient and curvature projections within each of the (a) two or (c) three planes are combined to give the gradient and curvature estimates. (b) shows the region used to estimate values for the central edge in case (a).

any triangles which contain two or more vertex references that are identical are discarded before storage.

The simplest method of calculating the position of the new vertex is to take the average location of each of the surface intersections. Figure 3.27 shows an example of an object whose surface has been reconstructed by calculating a distance function based on the object cross-sections in Figure 3.27(a). The triangulation formed by averaging the surface intersection positions is much more regular than the original MT triangulation, and this results in a smoother rendering. In the majority of cases, all surface intersections near a sample point can be clustered, so the triangulation has approximately one vertex for each sample point near the surface. The evenness of the distribution is a direct result of the evenness of the original sample lattice.

### 3.4.5 Adapting to local curvature

Using the average of the surface intersection locations works well for smooth surfaces, but not so well for sharp corners or regions of high curvature. This can be seen in Figure 3.27(d), where the front vertical edge of the ‘stand’ has a sequence of specular reflections on it, due to the triangulation not following the actual edge of the surface. This can be overcome by using a data curvature estimate to weight the surface intersection locations before averaging.

The previous technique involving tetrahedral lattices [38] used orthogonal estimates for gradient calculations, taking an average of these estimates over the nearest points to the sample point. For this technique, a gradient and curvature estimate is required for each *edge*, since this is where the surface intersections are located. The natural region of influence for this calculation includes those tetrahedra which share this edge. The geometry is shown in Figure 3.28. There are two types of edge, of length 2 and  $\sqrt{3}$  units, the former surrounded by four tetrahedra as in (a), and the latter surrounded by six as in (c). For both, the gradient and curvature are first calculated for each of the planes containing a pair of triangles. Figure 3.29(a) shows a pair of triangles for an edge of length  $\sqrt{3}$  units.

If the sampled values at each of the points  $o$ ,  $a$ ,  $b$  and  $c$  are  $d_o \dots d_c$ , then the angle that the

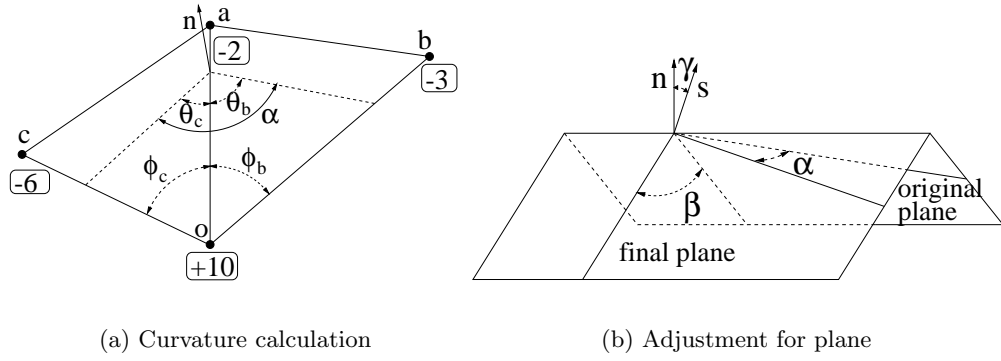


Figure 3.29: **Gradient and curvature calculation in each plane.** (a) Gradients and curvatures for an edge are calculated in each plane using the two triangles neighbouring the edge. (b) Curvatures must then be adjusted for the orientation of the plane with respect to the calculated surface normal.

data surface in each triangle makes with  $oa$  is given by

$$\theta_b = \arctan \left[ \frac{\sin(\phi_b)}{\frac{(d_o - d_b)oa}{(d_o - d_a)ob} - \cos(\phi_b)} \right], \quad \theta_c = \arctan \left[ \frac{\sin(\phi_c)}{\frac{(d_o - d_c)oa}{(d_o - d_a)oc} - \cos(\phi_c)} \right] \quad (3.13)$$

and the curvature, expressed as an angle which is zero for a flat surface, is therefore

$$\alpha = |\theta_b| + |\theta_c| \quad (3.14)$$

This curvature estimate requires further adjustment, since the plane in which it is calculated may be at a shallow angle to the actual surface, as in Figure 3.29(b). If  $\alpha$  is the original estimate, and the actual surface normal makes an angle  $\gamma$  with a plane orthogonal to the original plane and the line  $oa$ , then the adjusted estimate,  $\beta$  is given by

$$\frac{1}{\tan^2\left(\frac{\beta}{2}\right)} = (1 - \cos^2 \gamma) \left( \frac{1}{\sin^2\left(\frac{\alpha}{2}\right)} - 1 \right) \quad (3.15)$$

We are interested in the maximum curvature for each surface intersection, hence the minimum value of  $\left| \tan\left(\frac{\beta}{2}\right) \right|$  in each plane is used as the curvature measure. The location of the new vertex  $p_o$  is then calculated from the weighted locations of each surface intersection  $s_i$  in the set of intersections to be clustered  $S$ , with the following equation:

$$\omega_i = \frac{1}{\left| \tan\left(\frac{\beta}{2}\right) \right|_{min}}, \quad p_o = \frac{1}{\sum_{i \in S} \omega_i} \sum_{i \in S} \omega_i s_i \quad (3.16)$$

Note that an estimate of the actual surface normal for each surface intersection is also required for equation (3.15), in addition to being useful for shading the surface.  $1/\tan \theta$  gives the projection of the normal in each plane orthogonal to the line  $oa$  and scaled for a unit distance of  $oa$ . In the case of Figure 3.28(a), vector addition of these projections in each of the two planes plus a unit vector along  $oa$  gives an estimate of the surface normal. For the case in Figure 3.28(c), the surface normal is given by  $\frac{2}{3}$  of the vector addition of the projections in each plane, plus a unit vector along  $oa$ . This is because there are three planes at  $60^\circ$  to each other<sup>6</sup>.

<sup>6</sup>Lattices constructed from non-parallel planes as in Figure 3.22 are slightly distorted, which introduces a small error into the calculation of the surface normal.

### 3.4.6 Implementation

For the lattice orientation of Figure 3.21(c), the whole process can be performed by storing one plane of information, then progressing this plane through the data. The locations of the sample points on neighbouring planes alternate, so that two planes taken together form a rectangular lattice of sample points with half the spacing of that on an individual plane. One data structure is required for each of these points, with the following information:

- The location and the distance field value of the current sample point.
- The location and the distance field value of the last sample point at this lattice position (i.e. the plane before last).
- An unsigned integer indicating which of the 14 edges radiating from this point contain a near surface intersection.
- A reference for each of the 7 edges associated with this point, indicating the vertex created for this edge, if any.

In the first pass through a plane, the location and value of the current sample point are calculated (and the previous values updated), and any near surface intersections are marked. In the second pass, these surface intersections are clustered, creating new vertices, whose references are marked in each of the edges in which the surface intersections were located. Triangulation can then be performed for the six tetrahedra associated with this point, based on the vertex references. For lattices that are entirely orthogonal (i.e. for non-ultrasound applications), the locations for each point can be inferred from the lattice indices.

The triangles and vertices can be stored in two simple lists, since it is not necessary to search either list at any point in the process. Each triangle contains a (32-bit) reference to each of three vertices. Each vertex contains a location (a 16-bit integer was used for each direction) and a surface normal (again using 16-bit integers). Each vertex is stored once only, but can be referenced by several triangles.

## 3.5 Results: surface interpolation

The surface interpolation method outlined in Section 3.2 was assessed both by reconstructing surfaces of known objects, and by comparison with reconstructions by alternative techniques. It is not possible to visualise the results of these experiments without also using a surface visualisation algorithm, and in all the following figures, the algorithm of Section 3.4 has been used to triangulate the surface of the interpolated data. These triangulations are displayed with Gouraud shading using the Geomview<sup>7</sup> visualisation package. Experiments designed specifically to assess the surface visualisation algorithm itself are outlined in Section 3.6.

Surfaces were generated using shape-based interpolation, and two variants:

**Centroid-guided interpolation** This was a combination of the techniques in [83] and [116], whereby the interpolation was performed for each pair of corresponding contours along the direction of linked centroids. The bounding rectangles for each contour were also scaled independently in the x and y directions such that each were the same size. Correspondence of contours was determined manually (rather than simply by detecting object overlap). The

<sup>7</sup><http://www.geom.umn.edu/software/geomview/>.



Table 3.1: **Processing time for surface visualisation.** This includes all operations after creation of the cross-sections, up to and including display of the triangulated surface. Times were recorded on a Silicon Graphics Indigo 2 workstation; however they were comparable to a 166MHz Pentium MMX running Linux.

Object	Fig.	Modality	Triangles	Processing Time /secs		
				Shape-based	Centroids	Discs
Hepatic ducts	3.2	Ultrasound	1,800	0.7	0.8	1.1
Bladder/Prostate	3.30	Ultrasound	13,000	2.3	2.3	4.0
Foetus	3.31	Ultrasound	39,000	8.1	12.6	12.9
Child's skull	3.32	CT	93,000	13.9	-	147.1
Pelvis	3.33	CT	103,000	31.6	-	71.1
Liver	3.34	MRI	7,000	1.5	1.6	1.7

union of the generated surfaces was used to create the final object surface. The method of edge-shrinking interpolation [114] was not implemented as it is much more complicated, and in practice gives very similar results to the centroid-guided technique. The results will be *slightly* better, due to the more sophisticated alignment of corresponding contours, but the same gross artifacts are present.

**Disc-guided interpolation** The new technique described in Section 3.2.

It should be noted that there are many examples of simple surfaces where the differences between the results of any shape-based technique are negligible. The following data have been selected to highlight the properties of each algorithm over a broad range of surface types, particularly concentrating on surfaces that produce different results with each algorithm.

Processing times are shown in Table 3.1. The software was designed for non-parallel cross-sections, and the processing time for the parallel cases (CT and MRI) would have been significantly less had this not been so. However, since this applies equally to all the surface interpolation algorithms, the times still give a good relative measure of performance.

Unsurprisingly, the more complex disc-based algorithm takes longer than the centroid-based or standard shape-based algorithms. However, the increase in time is generally quite small, since all algorithms require calculation of the distance transform and triangulation of the interpolated data. The additional processing time for the disc-based algorithm increases with the number of discs involved in the correspondence calculation, hence objects with cross-sections which are long and thin (for instance the skull in Figure 3.32) take more time to process than those with fatter cross-sections.

### 3.5.1 Simulated scanning results

All of the surfaces shown in Figures 2.9 to 2.12 of Chapter 2, are reconstructed from cross-sections using disc-guided interpolation. Clearly, they all correctly represent the original objects, even in cases where those objects are branching or concave (Figure 2.12). Since the algorithm makes no assumption about smoothness, it is equally applicable to the reconstruction of sharp objects, for instance the cube of Figure 2.9(e).

The volumes calculated from these surfaces, using the algorithm of Appendix B.2, are also shown in the graphs of Figures 2.9 to 2.10. These volumes are less than the actual volume,



since the interpolation is linear and the objects are mostly convex. However, the disc-guided surface volume is better than the centroid-guided surface volume in all cases, and better than the shape-based surface volume for the more complex shapes in Figures 2.11(d) and Figure 2.12(e).

### 3.5.2 *In vivo* ultrasound results

The same ultrasound scanner and system were used to acquire freehand 3D ultrasound data as in Section 2.4, described in Appendix A.2. Three areas were examined: the hepatic ducts, the bladder, and a foetus.

#### Hepatic ducts

The interpolation of cross-sections through part of the system of hepatic ducts, shown in Figure 3.2, has already been discussed in Section 3.1.2. The disc-guided interpolation technique (Figure 3.2(g)) clearly gives the best results in this instance, even though the cross-sections are fairly simple.

#### Bladder and prostate

The bladder and prostate in Figure 3.30 demonstrate the effect of interpolating two close objects. Although both have very simple shapes, the contours from each overlap, and hence shape-based interpolation incorrectly makes a connection between these objects, as seen in Figure 3.30(b). Centroid-guided interpolation is not affected, since each of the bodies are treated individually (although they would still be connected if object correspondence was based on contour overlap, rather than manual determination). Disc-guided interpolation correctly constructs the surface without any manual interaction, as the external discs ensure the algorithm also ‘connects’ the gap between each of the objects.

#### Foetus

The foetus in Figure 3.31 is a much more complicated shape than the bladder or prostate. Many cross-sections contain only one contour which nevertheless includes the body and limbs. Centroid-guided interpolation performs very badly when connecting such contours with simple cross-sections of the torso — thin ‘webs’ are formed between regions which should not be connected. Shape-based interpolation is much better than centroid-guided interpolation in this case, but still incorrectly connects the upper hand with the head. This connection is not made by disc-guided interpolation, even though the contours overlap.

Further examples of disc-guided interpolation of foetal cross-sections are shown in Figure 2.20 of Chapter 2. These demonstrate the ability of the algorithm to reconstruct sensible surfaces from cross-sections with varying topology.

### 3.5.3 Other applications

#### Computed tomography: child’s skull

The skull in Figure 3.32<sup>8</sup> [181] is a good example of the sort of cross-sections that result from automatically thresholding medical images. Here, a high resolution (0.41mm by 0.41mm by 1mm)

<sup>8</sup>From CHILD.IMO, provided with 3DViewnix v1.1.1 (c) 1993-1996 M I P G University of Pennsylvania, All Rights Reserved.

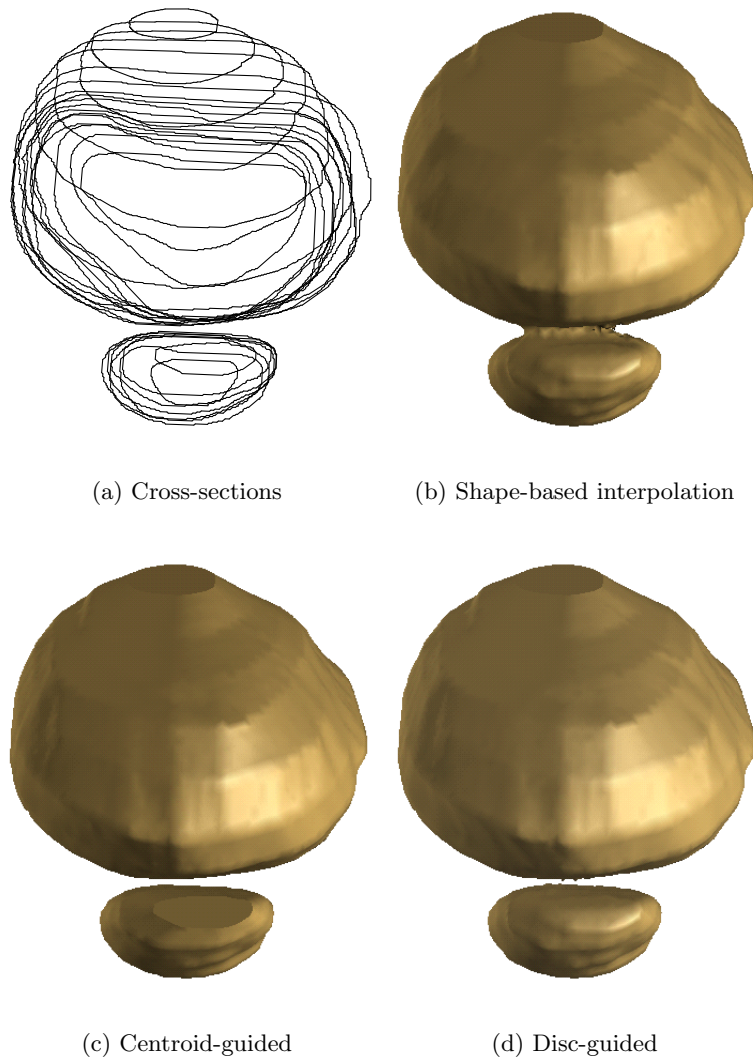


Figure 3.30: **Human bladder and prostate.** The cross-sections were manually outlined in the original B-scans of a freehand 3D ultrasound investigation.

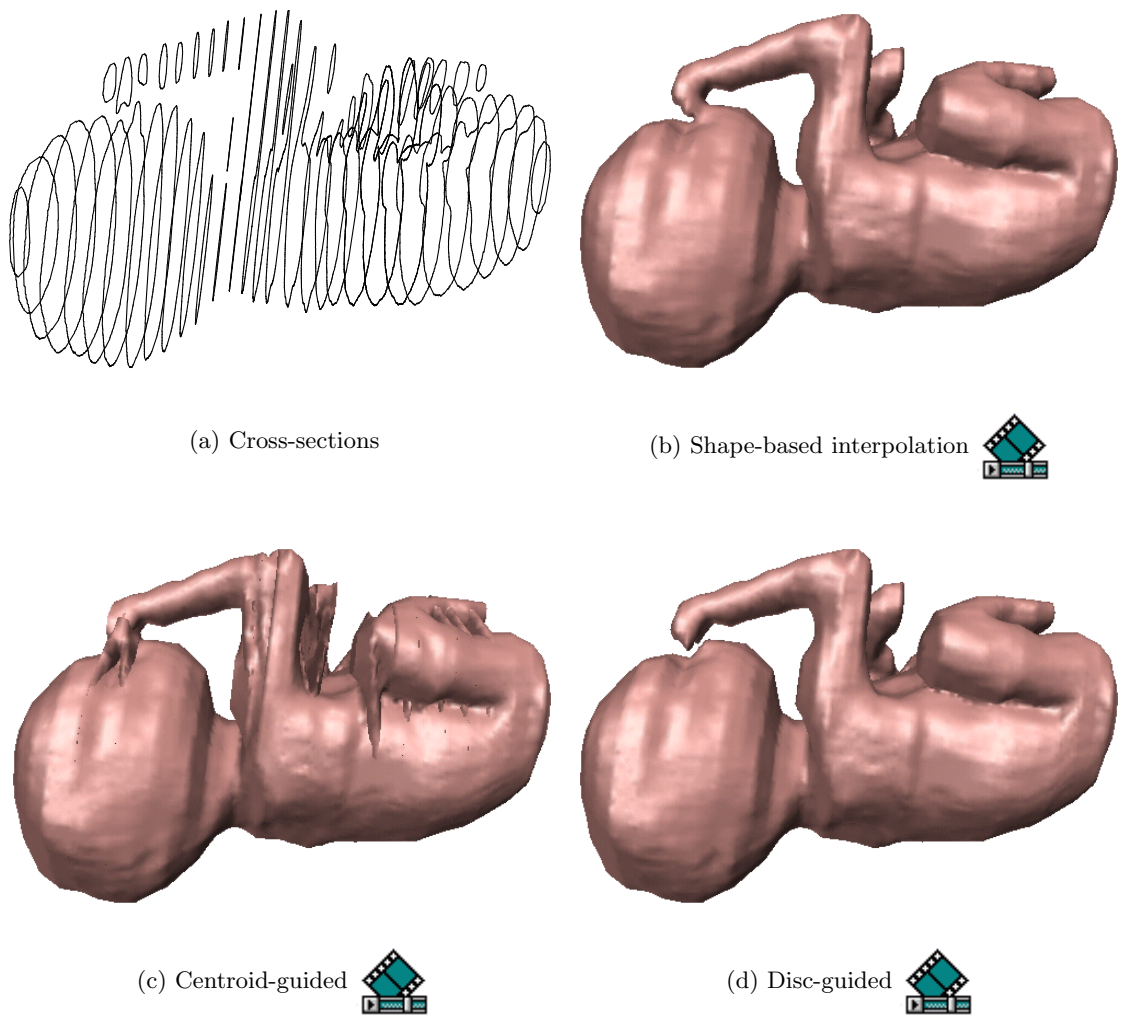


Figure 3.31: **Foetus at week 22.** The cross-sections were manually outlined in the original B-scans of a freehand 3D ultrasound investigation.

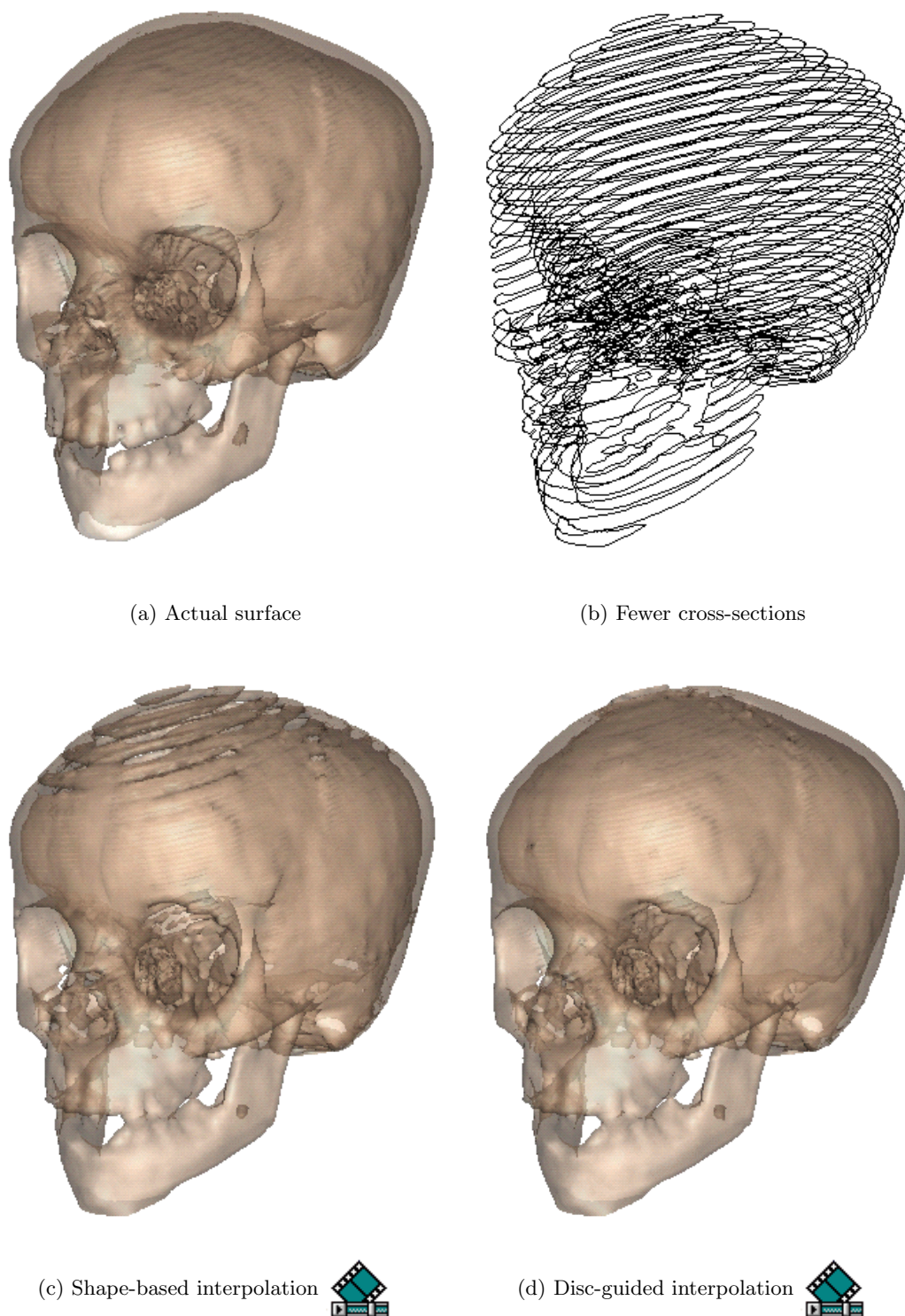


Figure 3.32: **Child's skull.** The cross-sections were automatically segmented by thresholding CT data. (a), (c) and (d) have been rendered with a slight transparency, so that both the inner and outer surfaces can be seen.

Computed Tomography (CT) scan of a child’s skull has been thresholded using the appropriate coefficient for bone. Most cross-sections have many contours, and the shape of each contour is complex. The centroid-guided technique is not appropriate in this case, since there are multiple contours which are neither simple nor tree-like. Liu’s method [114] would have given sensible results if the inner skull contour was separate from the outer, and hence the inside of the skull could be interpolated separately from the skull itself. However, this is not the case on many of the cross-sections.

The cross-sections in the upper part of the skull are far from overlapping, and hence shape-based interpolation reconstructs these as separate rings, rather than one surface. Disc-guided interpolation has no such constraint, so the skull can still be correctly reconstructed despite the lack of overlap.

### Computed tomography: pelvis

The female pelvis in Figure 3.33 is from CT data provided by the Visible Human Project<sup>9</sup>. In this case the original CT resolution was  $0.9375\text{mm} \times 0.9375\text{mm} \times 1\text{mm}$ ; however only a small subset of these planes was used to generate the cross-sections in Figure 3.33(b).

The surface in Figure 3.33(c) reveals the interplay between the interpolation process and the resolution of the interpolated data from which the triangulation is created. The hole in the left hand side of the pelvis is the result of under-sampling of data — the pelvis is very thin in this region. Even though Figure 3.33(d) has been generated at the same resolution, this effect has been reduced by a better interpolation direction. Disc-guided interpolation also improves the smoothness of the surface at the edges, for instance at the upper rims (iliac crests).

### Magnetic resonance imaging: liver

The human liver in Figure 3.34 is from MRI data, also provided by the Visible Human Project, in this case from the male data set. The original data resolution was  $1.875\text{mm} \times 1.875\text{mm} \times 4\text{mm}$ . Five cross-sections were manually segmented from this data. This is enough information to give a reasonable idea of the shape of the liver, and an estimate of its volume. However, it is not enough information for shape-based interpolation to estimate the surface — hence the large invagination in Figure 3.34(b). This sort of shape should be ideally suited to centroid-guided interpolation, however the disc-guided interpolation performs slightly better (particularly noticeable on the sharp upper edge of the liver). This is because the calculation of centroid position is dominated by the bulk of the object, hence small features on large objects do not contribute and may not be reconstructed correctly. Disc-guided interpolation uses local information and can re-adjust the interpolation direction in the region of the local feature.

## 3.6 Results: surface visualisation

The surface visualisation algorithm of Section 3.4, *regularised marching tetrahedra* (RMT) was assessed by a variety of methods, and compared to both *marching cubes* (MC) and *marching tetrahedra* (MT)<sup>10</sup>. The quality of smooth renderings of such surfaces, the number of triangles,

<sup>9</sup>The Visible Human Project is an initiative from the National Library of Medicine in Bethesda, Maryland.

<sup>10</sup>Comparisons were made with MT and with MC at a resolution such that the inter-plane spacing was the same in all cases.



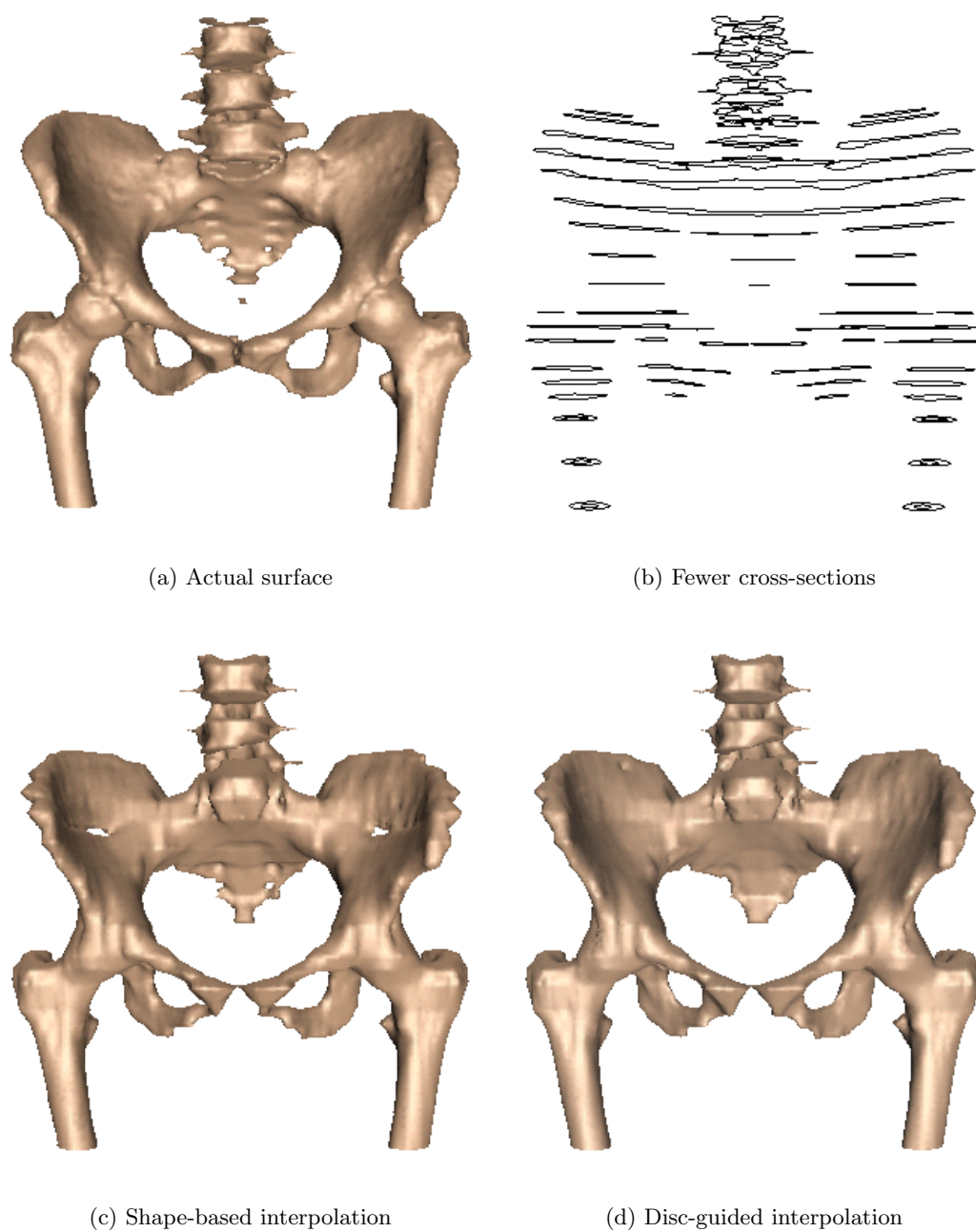


Figure 3.33: **Female pelvis.** The cross-sections were semi-automatically segmented by thresholding CT data, then manually editing the results of this threshold operation.

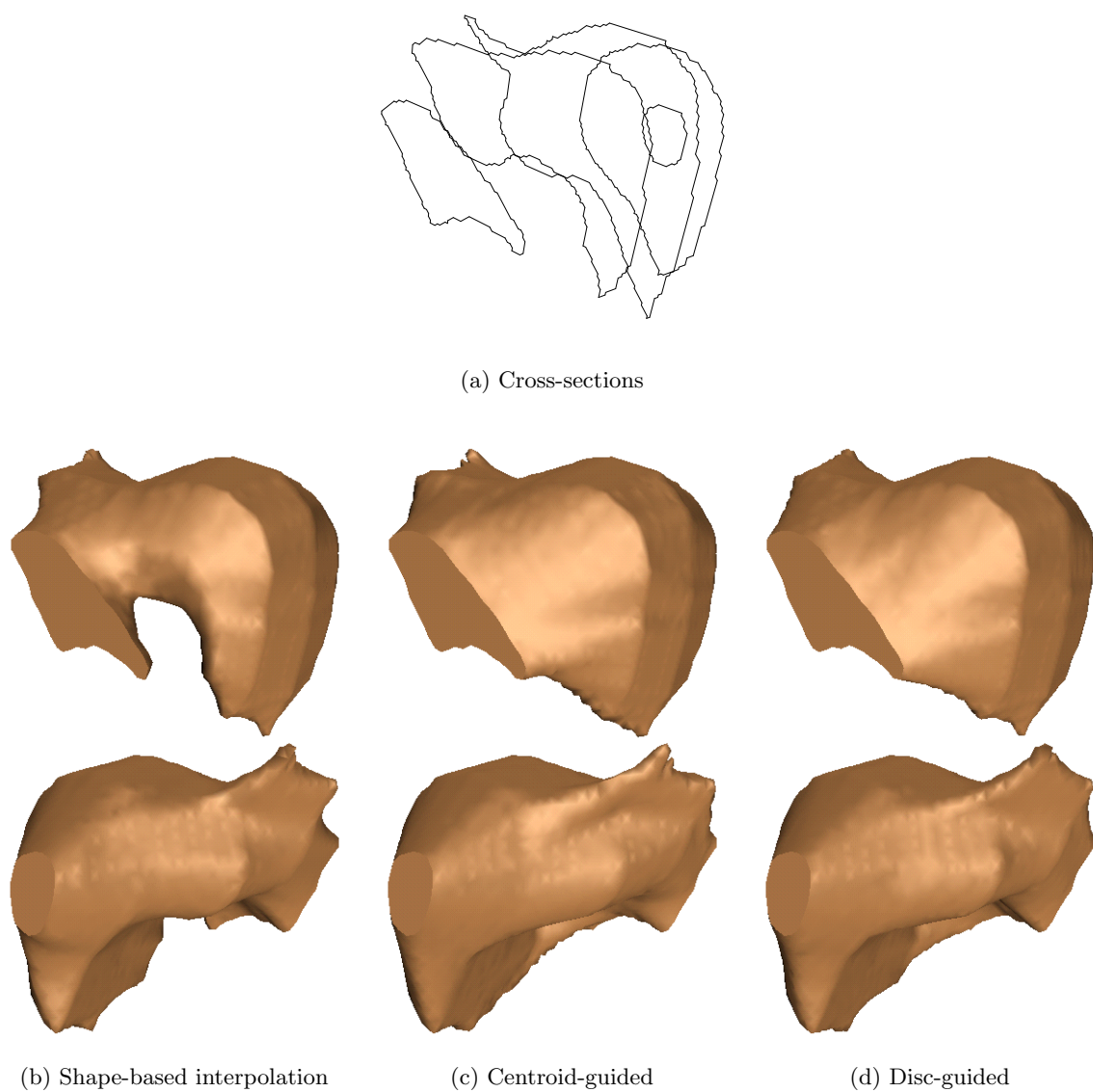


Figure 3.34: **Human liver.** The cross-sections were manually outlined from coronal MRI slices of the abdomen.

Table 3.2: **Number of triangles.** The topological cases indicated for each sample point are **cs**: closed surface, **ms**: multiple surfaces, **mh**: multiple holes, **fs**: flat surface.

Object	Fig.	Vertices			Triangles			Topological Cases			
		MC	MT	RMT	MC	MT	RMT	cs	ms	mh	fs
Sphere	<a href="#">3.19</a>	1,161	2,480	660	2,320	4,956	1,316	0	0	0	0
Example	<a href="#">3.27</a>	8,646	20,924	5,563	17,450	41,796	11,102	0	20	12	0
Peaks	<a href="#">3.39</a>	3,150	8,415	2,199	6,130	16,498	4,238	0	1	0	0
Closed	<a href="#">3.39</a>	4,113	9,368	2,500	8,196	18,744	5,008	0	6	0	0
Skull	<a href="#">3.32</a>	78,103	169,750	50,543	156,528	339,942	101,573	19	974	258	445
Hepatic	<a href="#">3.35</a>	12,909	27,404	7,646	26,087	55,350	14,890	0	60	2	16
Bladder	<a href="#">3.36</a>	15,053	28,524	7,828	30,213	56,848	15,556	0	4	0	0

Table 3.3: **Volume and surface area.** The values shown are not calibrated, except for the skull, bladder and hepatic ducts, in which case the volume is in  $ml$ , and the surface area in  $cm^2$ .

Object	Fig.	Volume			Surface area		
		MT	RMT	%	MT	RMT	%
Sphere	<a href="#">3.19</a>	1.0159	1.0121	99.63	4.8905	4.8793	99.77
Example	<a href="#">3.27</a>	0.7728	0.7716	99.84	8.8393	8.7799	99.33
Peaks	<a href="#">3.39</a>	-	-	-	9.2656	9.2482	99.81
Closed	<a href="#">3.39</a>	0.8196	0.8145	99.38	7.9559	7.8811	99.06
Skull	<a href="#">3.32</a>	362.84	361.95	99.75	1614.8	1602.9	99.26
Bladder	<a href="#">3.36</a>	317.60	317.51	99.97	249.37	249.01	99.86
Hepatic ducts	<a href="#">3.35</a>	37.510	37.248	99.30	173.70	171.42	98.69

the triangle aspect ratios and the effect on volume and surface area measurements were all investigated. Surface visualisation techniques were applied to simulated data, data from ultrasound and CT examinations, and data from mathematical functions.

As can be seen from Table 3.2, the number of both vertices and triangles with RMT were reduced by about 70% compared to MT and 40% compared to MC. This reduction is affected by the complexity of the surface at the sampling resolution. The reduction is slightly less for the child's skull, which has the highest complexity, since no vertex clustering can be performed for the 'closed surface', 'multiple holes' and 'flat surface' topological cases. In all the other examples, and indeed in most practical situations, there are very few occurrences of topological cases that prevent clustering. Indeed, such occurrences are often an indication of the lack of appropriate data filtering.

The effect of the isosurface extraction method on the volume and surface area of the triangulation is compared to that for standard MT in Table 3.3. In all cases, there were approximately 100 sample points spanning the objects in each direction. This leads to at least a 1% error in distance, 2% error in area and 3% error in volume measurement. By comparison, all of the measurements on the objects were within 1.5% of the measurements calculated with MT. The volume measurements were calculated from the triangulation by a method outlined in Appendix B.2.



Table 3.4: **Quality of surface rendering.** The values shown give the mean and maximum for the standard deviation of each pixel in the images of Figure 3.19(g), (h) and (i). Units are grey-levels, for an 8-bit (255 level) image. The values for the sphere are based on all pixels inside the border pixels.

Statistic	Triangulation		
	MC	MT	RMT
Mean std (whole image)	2.52	2.54	1.56
Mean std (sphere)	2.83	2.85	1.21
Maximum std (sphere)	8.82	7.86	3.32
% border pixels	1.47	1.38	1.54

### 3.6.1 Simulated results

The triangulations of Figure 3.19(a), (b) and (c) clearly demonstrate improved regularity when using RMT to visualise a sphere. Smooth renderings of these triangulations in Figure 3.19(d), (e) and (f) indicate that this regularity improves the quality of the rendering; but how can this improvement in quality be assessed?

A sphere has the same shape when viewed from any direction, and so long as the light source is kept fixed with respect to the viewer, surface renderings should also be identical. If the sphere has been rendered by triangulating the zero isosurface of the underlying function  $f(x, y, z) = x^2 + y^2 + z^2 - r$ , as in Figure 3.19, any variations in these renderings are therefore a result of the triangulation quality. This can be quantified by measuring the variation in image grey-scale for each pixel in a set of images rendered from different viewing directions.

Figure 3.19(g), (h) and (i) shows graphical results for this variation. It is immediately obvious that using RMT results in much less variation in the smooth rendered images. In addition, this variation is more uniformly distributed across the image. The average standard deviation for each of the pixels, in Table 3.4, reveals that the mean variation in the RMT image is only 43% of that in the MC image, 38% for the worst case. In practice, the variation is reduced to such an extent that it is difficult to detect whether the RMT sphere is being spun or not, an action which is very apparent with MC or MT.

Figure 3.19(g), (h) and (i) also highlight an additional effect of representing a sphere with a triangulated surface. Although the sphere shading can be interpolated in order to smooth across the triangles, the *shape* is still defined by the actual triangulated surface. Hence the black rings around each of the figures, which represent the variation in the surface position (i.e. for some viewing directions, the surface exists at a pixel, and for others it does not). This effect increases as the number of triangles used to make up the surface is reduced — Table 3.4 shows that MT, which has the greatest number of triangles, has the fewest border pixels. However, the slight increase in border pixels with RMT is dramatically less than the reduction of triangles compared to MC or MT.

### 3.6.2 *In vivo* ultrasound results

#### Bladder and hepatic ducts

The hepatic ducts in Figure 3.35 and bladder in Figure 3.36 were both reconstructed in the same way as in Section 2.4, using software described in Appendix A.2. The triangulations in both

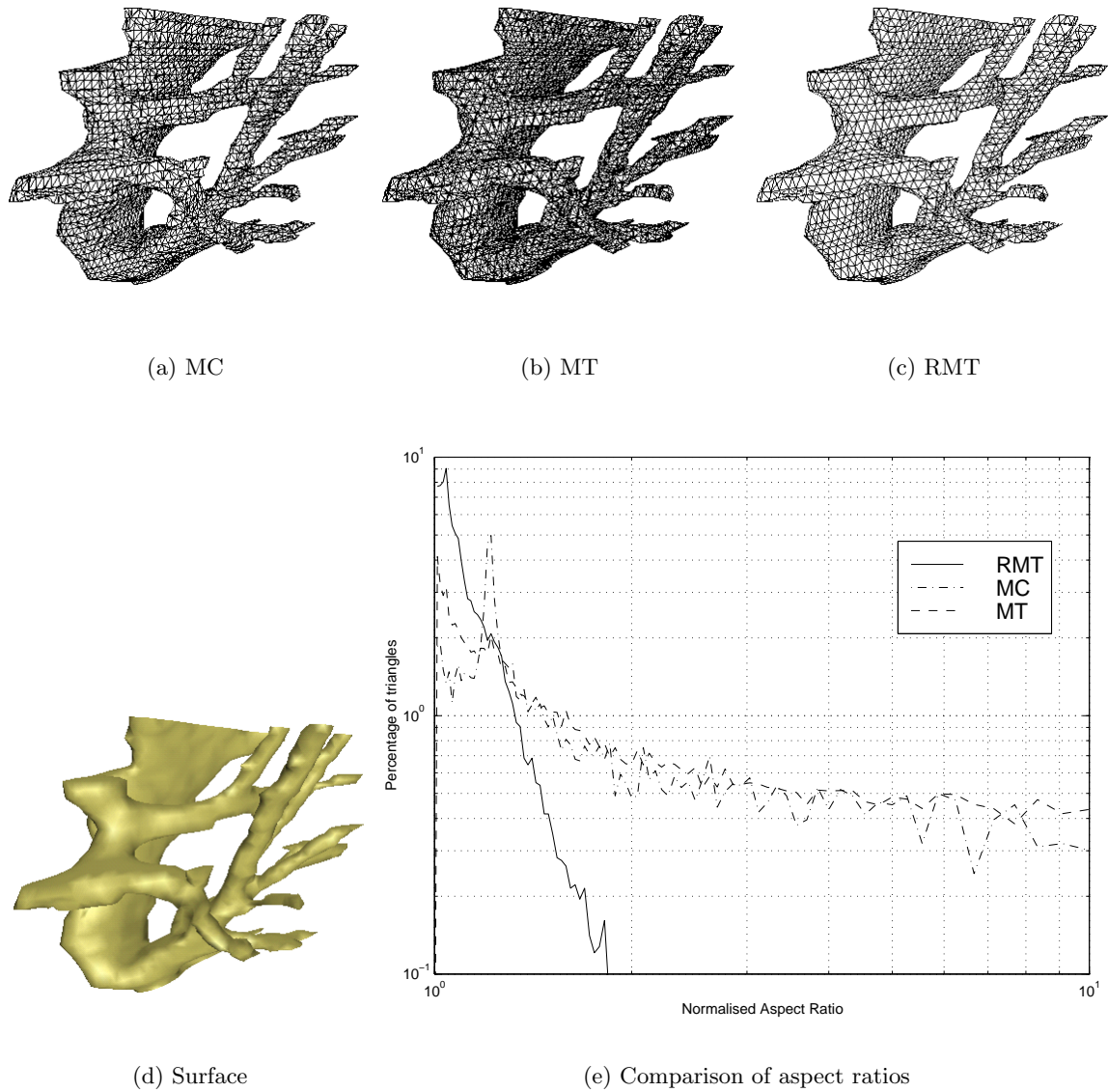


Figure 3.35: **System of hepatic ducts.** (a), (b) and (c) show the triangulated surface after disc-guided interpolation and MC, MT and RMT isosurface extraction, respectively. (d) shows a smooth rendering of (c). (e) The aspect ratio is defined by the ratio of the radius of the circumscribed circle, to the radius of the inscribed circle. This is normalised by the aspect ratio for an equilateral triangle.

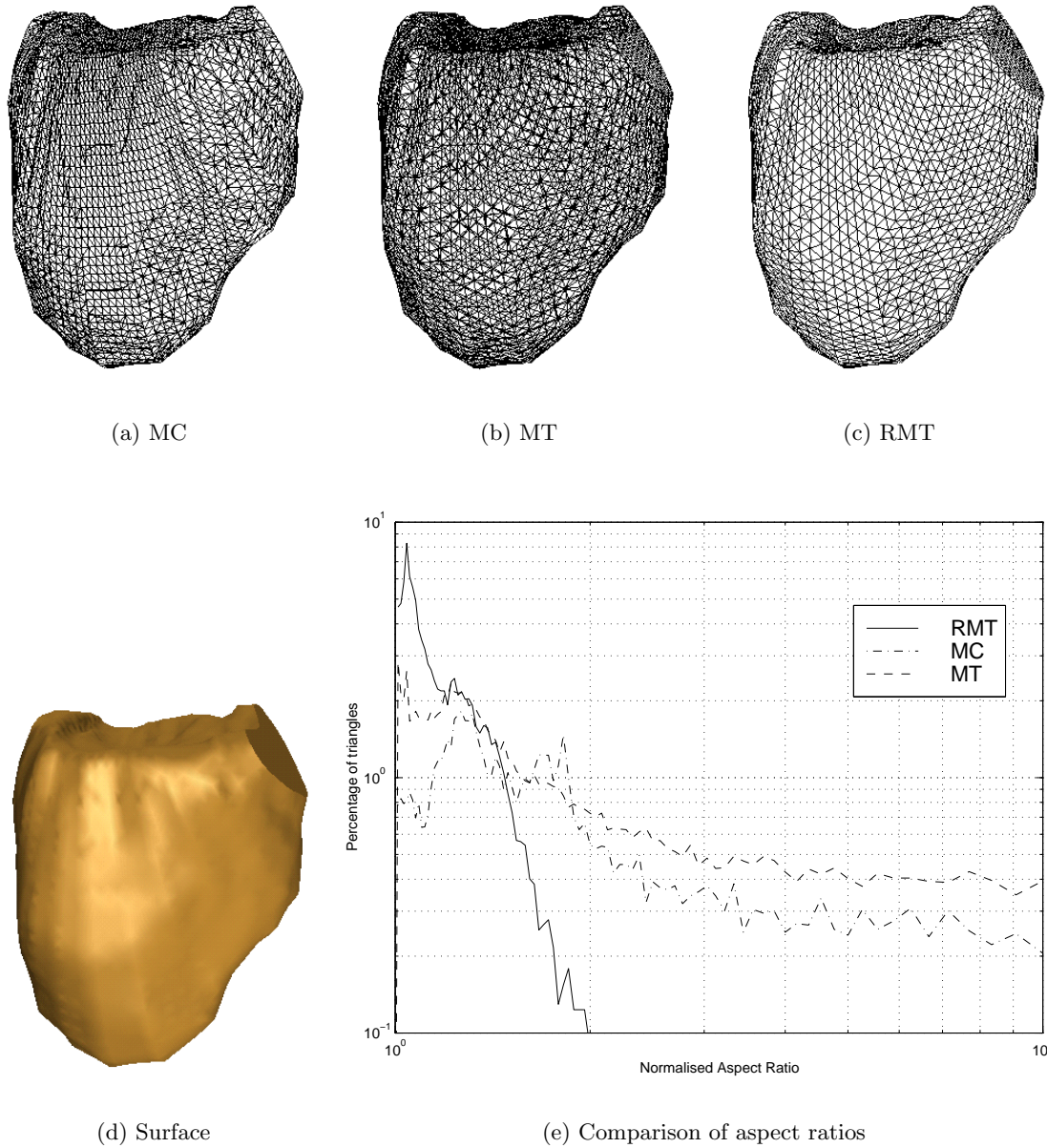


Figure 3.36: **Human bladder.** (a), (b) and (c) show the triangulated surface after disc-guided interpolation and MC, MT and RMT isosurface extraction, respectively. (d) shows a smooth rendering of (c). (e) The aspect ratio is defined by the ratio of the radius of the circumscribed circle, to the radius of the inscribed circle. This is normalised by the aspect ratio for an equilateral triangle.

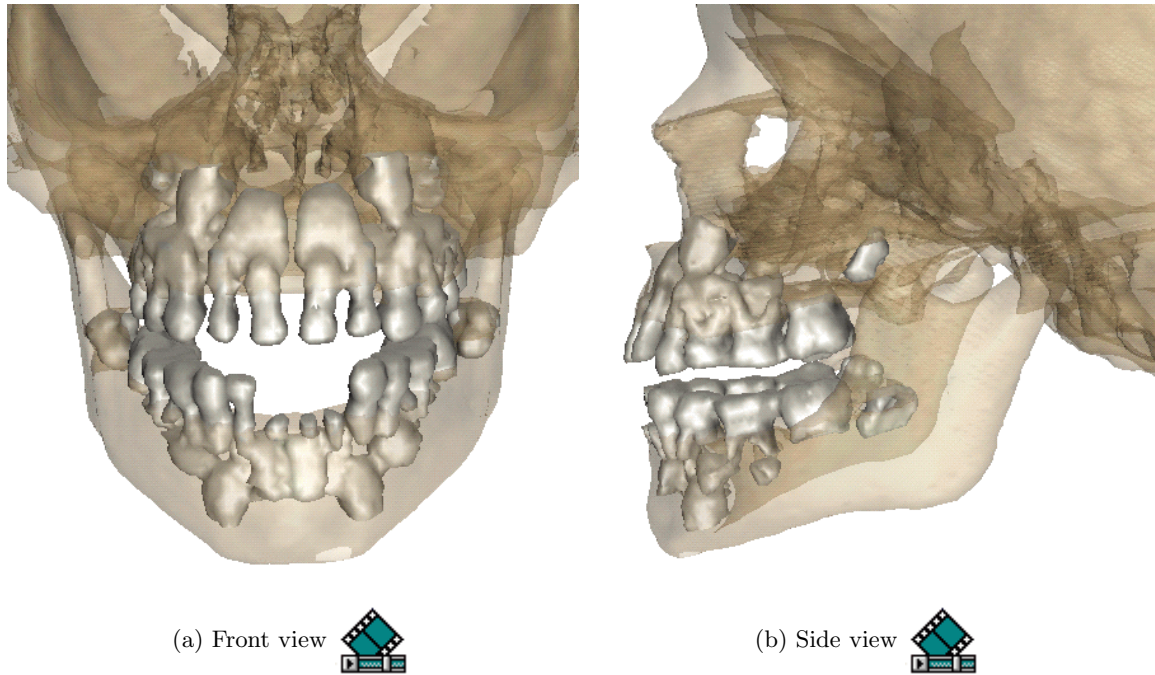


Figure 3.37: **Teeth and skull.** The cross-sections were automatically segmented by thresholding CT data. The teeth are rendered white and opaque, and the skull transparent, so that details, such as the second set of adult teeth, are clearly visible.

these figures demonstrate the improved regularity of the RMT technique over MT and MC.

This regularity can be assessed by considering the aspect ratios of the triangles making up the surface, as shown in the graphs of Figures 3.35(e) and 3.36(e). In these graphs, the triangles for MC, MT and RMT have been sorted in order of increasing aspect ratio, and normalised by the total number of triangles. The first peak on the graph represents near equilateral triangles, and the second smaller peak near right-angled isosceles triangles. The former type of triangle is generated when the four tetrahedra around a 2 unit edge are intersected by the surface, and the latter when the six tetrahedra around a  $\sqrt{3}$  unit edge are intersected. In both cases, 99.9% of the triangles in RMT had aspect ratios of better than 2. MC and MT, on the other hand, generated many triangles with aspect ratios of worse than 10.

### 3.6.3 Other applications

#### Computed Tomography: child's skull and teeth

Figure 3.37 shows an example of RMT applied to thresholded CT data: this is from the same data set as in Figure 3.32. 3D data sets contain more information than can be easily visualised in 2D. Colour and transparency can help to display surfaces more clearly, and show relative positions which would otherwise be impossible to see.

Equally important is the ability to reconstruct surfaces within the same data set at varying resolutions. The teeth in Figure 3.37 are the area of interest, and these have been reconstructed at a high resolution. The skull, however, is only there to show where the teeth are located, and can be reconstructed at a lower resolution. This allows the optimum number of triangles to be used, which improves the rendering speed and hence the user interaction with the display.

### Computed Tomography: pelvis and thigh muscles

Figure 3.38 is another example of using colour, transparency and varying resolution. This is from the same data set as in Figure 3.33. In this case the body, skeleton and thigh muscles have been reconstructed separately; the body at a lower resolution than the other areas. Changes in the transparency of each surface, and the gradual removal of the surfaces, can both be used to show relative location.

The surfaces in Figure 3.37 and Figure 3.38 were generated using software which implements RMT, and is described in Appendix D.1.

### Visualisation of implicit surfaces

The triangulations of two implicit isosurfaces are shown in Figure 3.39. Figure 3.39(a) and (b) are from the well known ‘peaks’ function provided with Matlab v5.0<sup>11</sup>:

$$\begin{aligned} f(x, y, z) &= (3 - 3x)^2 e^{-x^2 - (y+1)^2} \\ &\quad - 10 \left( \frac{x}{5} - x^3 - y^5 \right) e^{-x^2 - y^2} \\ &\quad - \frac{1}{3} e^{-(x+1)^2 - y^2} - z \end{aligned} \quad (3.17)$$

Figure 3.39(c) and (d) are from a closed surface used in a recent paper [78]:

$$\begin{aligned} f(x, y, z) &= \left( 1 - \left( \frac{x}{6} \right)^2 - \left( \frac{y}{3.5} \right)^2 \right) \\ &\quad \left( (x - 3.9)^2 + y^2 - 1.44 \right) \left( x^2 + y^2 - 1.44 \right) \\ &\quad \left( (x + 3.9)^2 + y^2 - 1.44 \right) - z^2 \end{aligned} \quad (3.18)$$

Any such function can be visualised by evaluating it on the tetrahedral lattice, and using RMT to triangulate a surface at a particular threshold. The surfaces shown are for  $f(x, y, z) = 0$ . Display of surfaces by this method has a significant advantage over the usual method of setting up a grid in the x-y plane, whose z-values at each location are defined by the function. In the latter method, it is not possible to visualise surfaces with multiple values in any direction, whereas RMT is completely independent of topology and has no such restriction.

The surfaces in Figure 3.39 were generated using software which implements RMT to visualise surfaces of arbitrary functions, described in Appendix D.2.

<sup>11</sup>©1984-96 The MathWorks, Inc., <http://www.mathworks.com>.



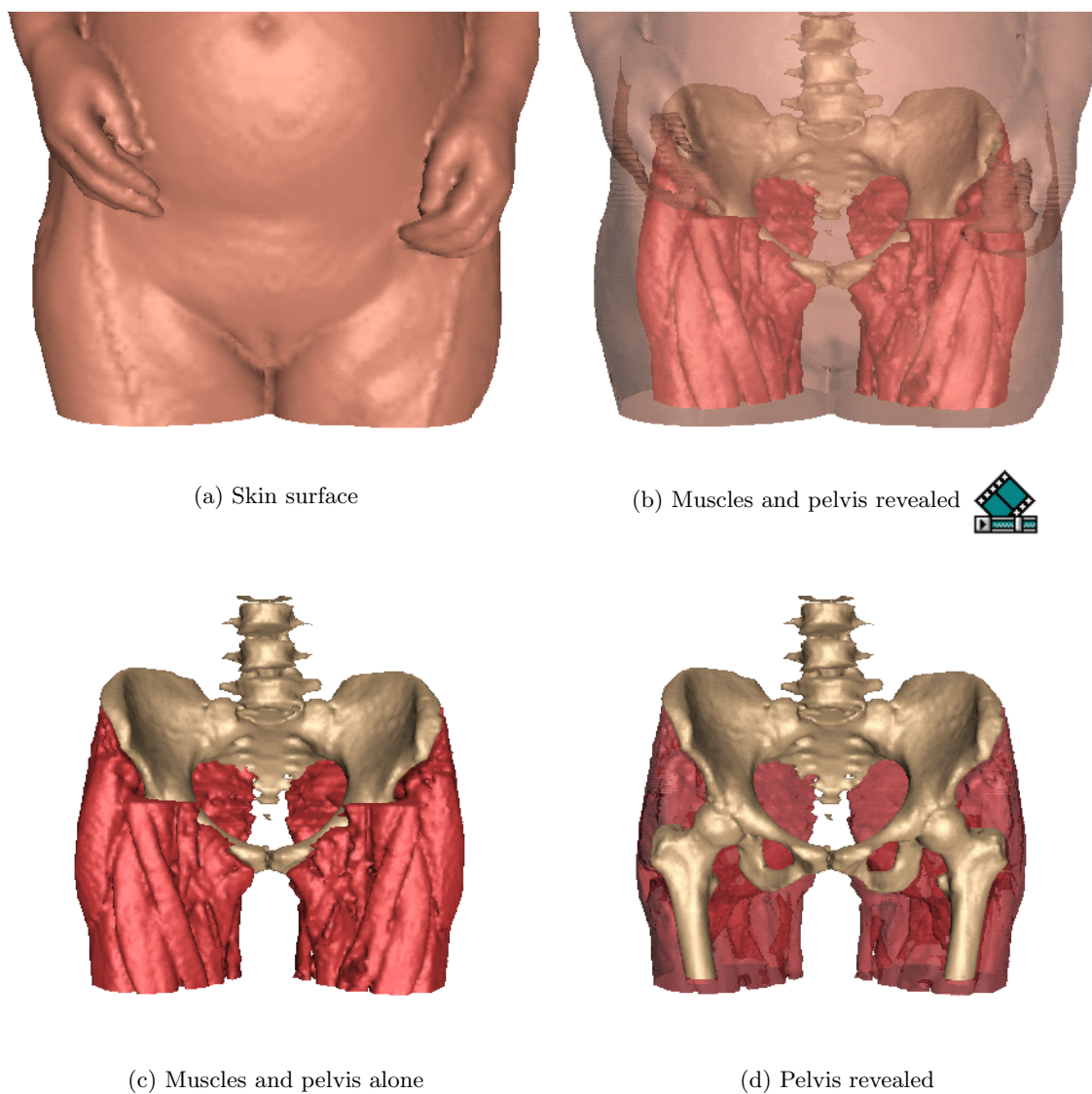
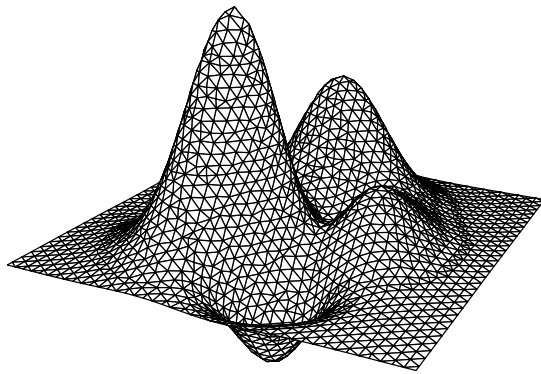
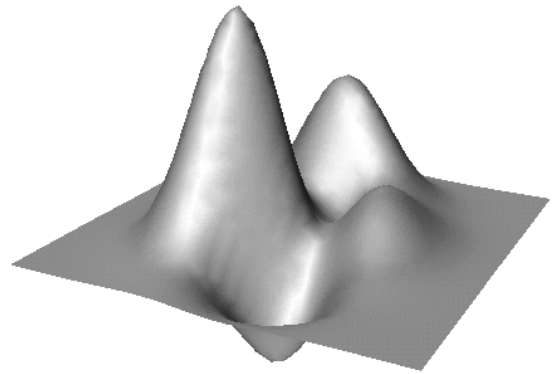


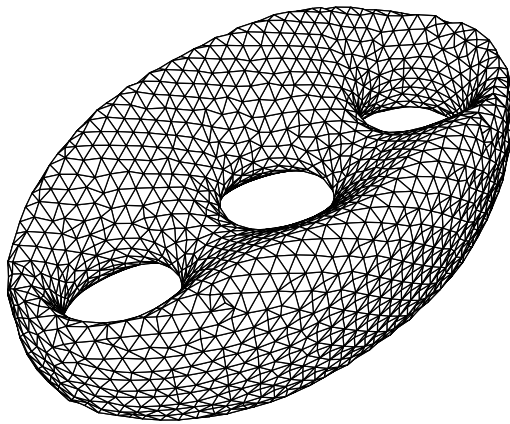
Figure 3.38: **Pelvis and thigh muscles.** The cross-sections were automatically segmented by thresholding CT data. Rendering each surface in a different colour, and changing the transparency properties of each surface, allows the relative location of different structures to be seen.



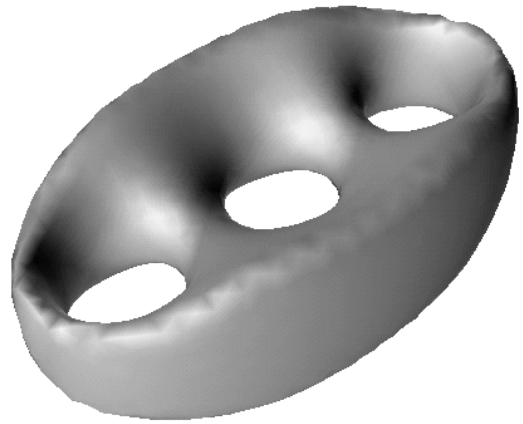
(a) Triangulation of equation (3.17)



(b) Surface of equation (3.17)



(c) Triangulation of equation (3.18)



(d) Surface of equation (3.18)

Figure 3.39: **Isosurfaces of implicit functions.** Both surfaces have been generated using RMT.

## Chapter 4

# Application to multiple-sweep data

### 4.1 Current methods for combining data from multiple sweeps

As explained in Section 1.2, despite the many advantages 3D ultrasound offers, it has seldom been used to examine anatomy which is larger than the width of a single B-scan, since this would require multiple sweeps to cover the entire organ.

There are several such cases where the anatomy is too large, or the shape is too awkward, or the direction of view is too restricted, to scan the entire volume in a single sweep. For instance the foetus beyond mid-term is difficult to scan in this way. Tumours, too, can grow to be larger than will fit into a single B-scan, or if they are located beneath the lower ribs (e.g. in the liver) can be impossible to scan in one motion of the probe. The liver itself is also in this category, though there are few situations where an accurate volume measurement of the liver is clinically relevant<sup>1</sup>. However, in a recent CT study of liver volume, it was suggested that an accurate measure of volume compared to patients' weight and height correlated well with the severity of chronic liver disease [113]. In any case, the liver provides an excellent test case for using ultrasound in this context, since it is large, has a relatively complex shape, and its position beneath the lower ribs constrains the direction of insonification.

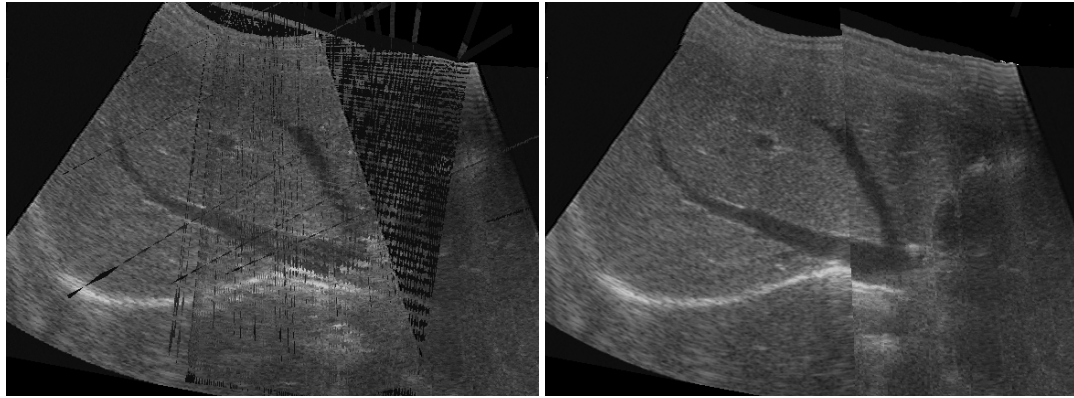
There have been a few attempts in the past to use a primitive form of 3D ultrasound, where the position of each B-scan is recorded manually, to measure liver volume [11, 36, 62, 153]. The earliest of these used a freehand scanning technique [153], although the accuracy of the volume measurements was limited by the large number of hand calculations and inaccurate B-scan localisation. The remainder used parallel transverse slices, with which it is difficult to examine the entire liver. In none of these cases was it possible to include the entire liver cross-section in all of the B-scans — approximations to the shape of missing sections were required. It is only recently that freehand scanning techniques have again been applied to liver volume measurement [80] and foetal liver volumes [163].

As discussed in Section 1.2.1, mis-registration is more apparent when using multiple-sweep data — it is much worse between data sweeps than within them. This is primarily a result of movement of the organ under examination with respect to the position reference frame, due to varying probe pressure from one sweep to the next. Interpolating such data to a regular voxel array, using one of the standard techniques discussed in Section 1.2.2, produces undesirable results. Figure 4.1(a), for instance, shows two sweeps from a liver examination which have been interpolated using the *voxel nearest neighbour* [161] interpolation. The poor image quality is

---

<sup>1</sup>Assessment by palpation of the *approximate* liver volume, by contrast, is a standard part of an abdominal examination.





(a) Nearest neighbour interpolation

(b) Partitioning using dividing planes

Figure 4.1: **Viewing multiple-sweep data.** Performing simple nearest neighbour interpolation on multiple sweeps generates results as in (a). (b) is the result of using additional planes to separate the sweeps.

due to a combination of treating the black regions around each B-scan as ‘real’ data, and mis-registration of the data itself. This mis-registration has been studied in the context of spatial compounding (a technique to reduce the noise in ultrasound images) [6, 159, 160]. The data from each sweep must be registered before it can be compounded, which is a time consuming and often poorly constrained problem.

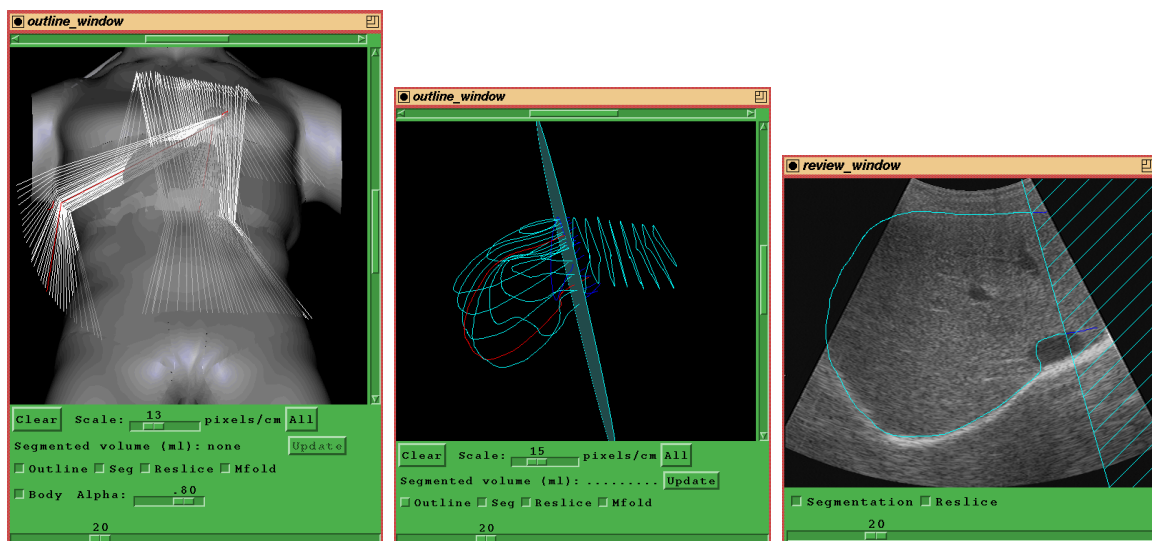
Combining data from multiple ultrasound sweeps has been successfully achieved for use in an ultrasound simulator [3]. Creating a good data set for simulation is, however, a somewhat different problem to that of examining a patient. The techniques of image warping and blending used to register the data are both time consuming and inappropriate in an unsupervised clinical context, since the warping algorithm can make changes to the data which are not justified by the physics of the acquisition. The simplest way to combine data from multiple sweeps in a clinical context, and possibly most appropriate, is to use only one sweep in any one part of the interpolated volume, as in Figure 4.1(b). The data from each sweep can now be clearly seen. This partitioning approach can also be used to allow volume measurement and surface visualisation from segmentations of the original B-scans, in a sequential framework.

## 4.2 A new approach: partitioning multiple sweeps

### 4.2.1 Overview

The entire process of volume measurement and surface visualisation from multiple-sweep data is outlined in Figure 4.4. Once the ultrasound data has been acquired, the orientation of the B-scans can be displayed in an ‘outline’ window, as in Figures 4.2(a) and 4.3(a)<sup>2</sup>. The clinician can then review the scans in a ‘review’ window, as in Figures 4.2(c) and 4.3(c). B-scans which are in-between the valid sweeps, for instance where the probe has been lifted from the skin surface temporarily, are then marked as invalid. Each sweep of data is thus defined as a contiguous set

<sup>2</sup>These scans have been registered to a computer generated manikin in order to show their location and orientation [177].

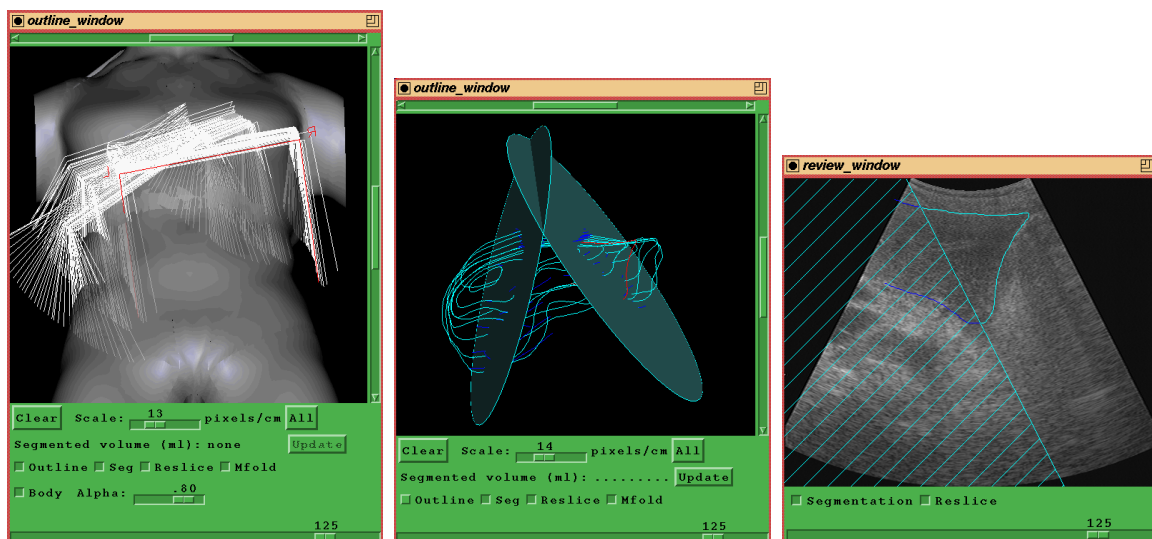


(a) Orientation of B-scans

(b) Segmentation

(c) Typical B-scan

Figure 4.2: **Liver examination using two sweeps.** The viewing direction in (b) is the same as in (a). The position of the dividing plane shown in (b) is based on the sweep orientation. Cross-hatching in the B-scan, as in (c), indicates areas that are better covered by a B-scan from another sweep. One longitudinal and one horizontal sweep is used.



(a) Orientation of B-scans

(b) Segmentation

(c) Typical B-scan

Figure 4.3: **Liver examination using three sweeps.** The figures and orientation are as in Figure 4.2. In this case, the liver is covered by three overlapping longitudinal sweeps.

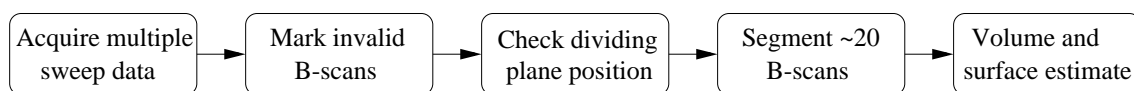


Figure 4.4: **Volume measurement and surface visualisation from multiple sweeps.** The multiple-sweep data is acquired and invalid B-scans (i.e. those between sweeps) marked. Dividing planes (which separate the sweep data) are calculated automatically, but can be manually adjusted if necessary. Cross-sections are then outlined in the original B-scans — only a handful are required to give an accurate volume estimate.

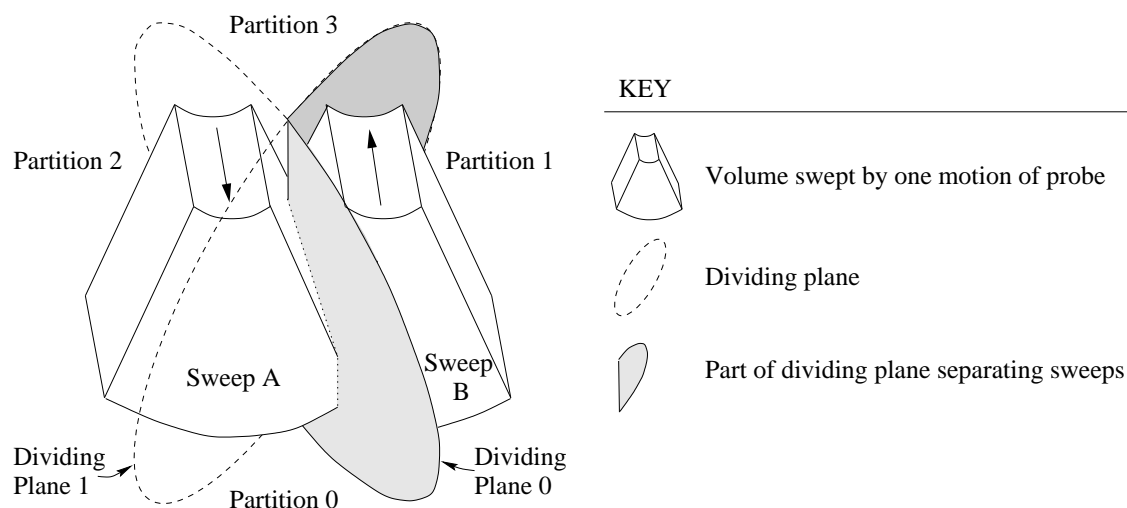


Figure 4.5: **Definition of terms for multiple sweeps.** Sweep *A* and *B* are separated by two *dividing planes* 0 and 1. These planes form four *partitions* of space 0...3. In this case sweep *A* is used in partitions 0, 2 and 3, sweep *B* is used in partition 1. Dividing planes are rendered opaque only where they separate data from different sweeps.

of B-scans separated by these invalid scans.

Once the sweeps have been defined, a set of ‘dividing planes’ can be automatically generated to partition space, such that only one sweep of data is used in each partition (see Figure 4.5 for a definition of these terms). Dividing planes are displayed as discs in the ‘outline’ window, Figures 4.2(b) and 4.3(b), and their location and number can also be edited manually if necessary.

The clinician can now manually segment the area of interest in the ‘review’ window. Any region not in a partition of space for which its sweep is being used, is cross-hatched. Hence the clinician can clearly see which data is relevant, and which data is better covered by a B-scan from another sweep. Segments must either be closed or end in cross-hatched regions, which is easily ensured by automatically closing a contour if either the start or end point is outside a cross-hatched region. Typical B-scans with cross-hatching and open contours are shown in Figures 4.2(c) and 4.3(c), together with the result of a complete segmentation (roughly 20 cross-sections) in Figures 4.2(b) and 4.3(b).

Volume measurement is performed from these cross-sections *in each partition of space*, where, if necessary, incomplete cross-sections are closed by the dividing planes defining the partition. The total volume is the sum of the absolute volumes in each partition. Surface reconstruction is also performed in each partition of space, and each surface is clipped to the dividing planes

which make up that partition. The total surface (which is not guaranteed to be closed) is the union of these surfaces. Volume can also be estimated from each surface, provided that it is only clipped by a maximum of two dividing planes (which is usually the case in practice).

Each step in this process is outlined in more detail in Sections 4.2.2 to 4.2.5. It is fundamental to this system that the apparent complexity is hidden from the clinician behind a carefully designed user interface. The important features of this interface are also outlined in the following sections.

## 4.2.2 Partitioning of data using dividing planes

### Automatic generation of dividing planes

The only restriction on the scanning pattern for each of the sweeps is that the sweep must pass into and out of the object under investigation, and the side of the scan plane which first enters the object must also be the first side to leave it. This means that the freehand nature of the acquisition process during each sweep is preserved, in that the B-scans can have any orientation and spacing, and can even be overlapping. Multiple sweeps are recorded in one sequence, and the number of sweeps determined after recording, by marking B-scans in-between each valid sweep as invalid. This has the advantage of enabling recording during a single breath hold. The sweeps could equally be separated by pausing the acquisition process between each of them, making the marking of invalid B-scans unnecessary, although it is less likely that the examination could be completed in a single breath hold with this method.

Initially, one dividing plane is placed between each pair of sweeps (redundant planes are removed later), up to a maximum of 32. This limit allows each partition of space to be labelled with an integer, where bit  $i$  defines the side of dividing plane  $i$  in which that partition exists<sup>3</sup>. In most practical situations only one to three planes are required to separate the data — the worst case processing time is exponential in the number of planes.

The position of each dividing plane is based on the corners of the extreme B-scans in each sweep, and the centre of the sweep: for sweep  $A$  in Figure 4.6(a), these are the points  $a_0 \dots a_7$  and  $g_a$  respectively. The vector from the centre of one sweep to the other,  $\vec{v}$ , is compared with the orientation of each of the average planes through  $\{a_0, a_1, a_2, a_3\}$ ,  $\{a_0, a_4, a_5, a_1\}$ ,  $\{a_1, a_5, a_6, a_2\}$ ,  $\{a_5, a_4, a_7, a_6\}$ ,  $\{a_3, a_2, a_6, a_7\}$  and  $\{a_4, a_0, a_3, a_7\}$ . The plane from each sweep, for which the dot product of its normal with  $\vec{v}$  is greatest, is selected for defining the new dividing plane. In the case of Figure 4.6(b), the planes defined by  $\{b_1, b_5, b_6, b_2\}$  and  $\{a_5, a_4, a_7, a_6\}$  are used. The dividing plane is then defined by  $\vec{p} \cdot \hat{n} - o = 0$ , where  $\vec{p}$  is any point on the plane, and  $o$  and  $\hat{n}$  are defined by equations (4.1):

$$\begin{aligned} \hat{n}_a &= \text{norm}((\vec{a}_4 - \vec{a}_5 - \vec{a}_6 + \vec{a}_7) \times (\vec{a}_4 + \vec{a}_5 - \vec{a}_6 - \vec{a}_7)) \\ \hat{n}_b &= \text{norm}((\vec{b}_1 - \vec{b}_5 - \vec{b}_6 + \vec{b}_2) \times (\vec{b}_1 + \vec{b}_5 - \vec{b}_6 - \vec{b}_2)) \\ \hat{n} &= \text{norm}(\hat{n}_a + \hat{n}_b) \\ o &= \frac{1}{8} (\vec{a}_4 + \vec{a}_5 + \vec{a}_6 + \vec{a}_7 + \vec{b}_1 + \vec{b}_5 + \vec{b}_6 + \vec{b}_2) \cdot \hat{n} \end{aligned} \quad (4.1)$$

where norm indicates vector normalisation.  $\hat{n}_a$  and  $\hat{n}_b$  are the average planar normals for the selected planes from each sweep.

<sup>3</sup>There is an inherent redundancy in this representation, since for more than three planes, it is not possible to partition space into  $2^n$  partitions, where  $n$  is the number of planes. However, this simple representation improves the calculation speed for the usual case of only a few planes.

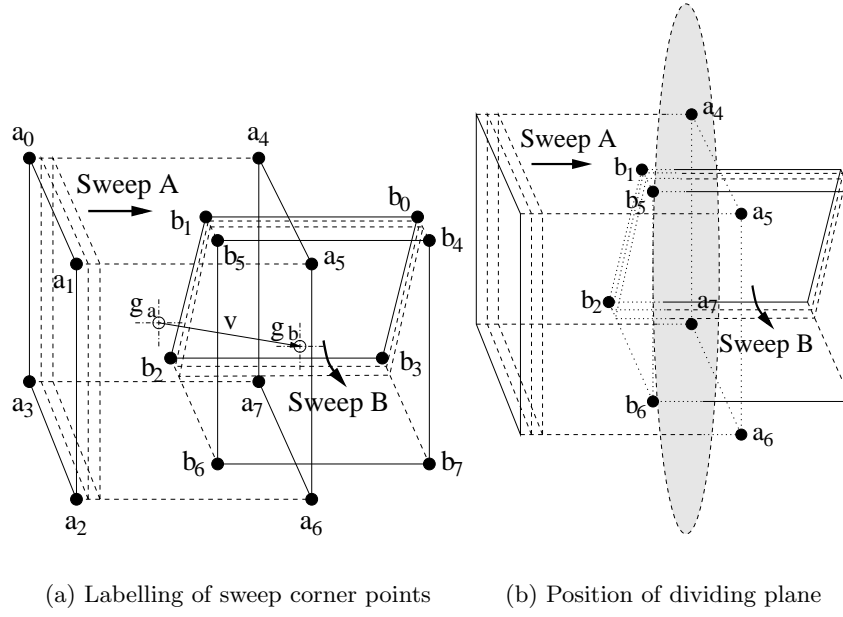


Figure 4.6: **Calculation of dividing plane position.** Sweep A is linear, and B is a fan sweep. (a) The corner points of the extreme scans and the average centre point for each sweep are calculated. (b) The dividing plane is positioned at the average location of the two sweep sides which are most nearly oriented perpendicular to the vector  $\vec{v}$  joining the sweep centres.

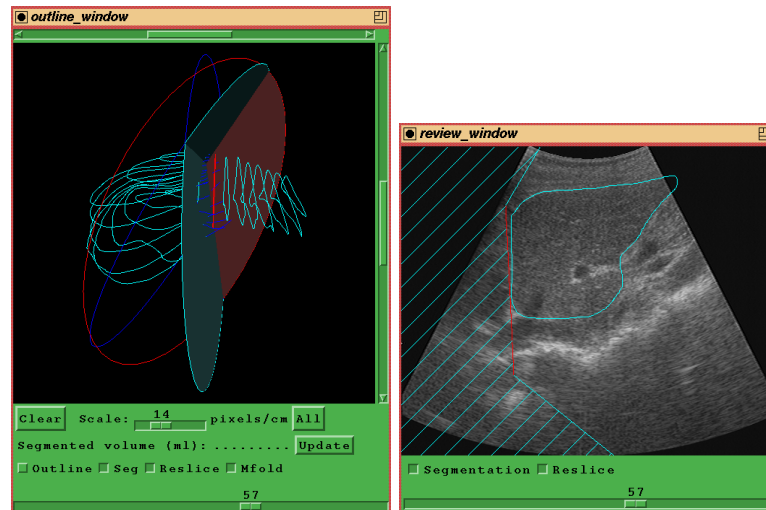
Once all the planes have been defined, the next step is to decide which sweep should be used in which partition of space. This is done by considering for each partition (of which there are up to  $2^n$ , where  $n$  is the number of planes), which sweep has its centre farthest inside (or least outside) that partition. This can be done by calculating  $l_{s,p}$ :

$$l_{s,p} = \frac{1}{N_s} \sum_{i=1}^{N_s} \min_{1 \leq j \leq D} (d_{ij}) \quad (4.2)$$

where  $s$  is the sweep,  $p$  is the partition,  $N_s$  is the number of B-scans in sweep  $s$  and  $D$  is the number of dividing planes.  $d_{ij}$  is the perpendicular distance of the centre of B-scan  $i$  to dividing plane  $j$ , where the sign is positive if the centre is the same side of that plane as the partition. The sweep with the largest value of  $l_{s,p}$  is the one used for partition  $p$ .

Redundant planes can be found by considering, for each partition, whether the same sweep is used in that partition as in the one which is the opposite side of the plane. If this is the case for *all* partitions, then the plane does not separate any sweeps and can be removed.

Once dividing planes have been defined, they can be used to re-slice the data (i.e. view it on arbitrary planes). The reslice algorithm is as explained in [145], save that for each point in the reslice, its partition is calculated, and only B-scans from the sweep used in that partition are considered for interpolation to the reslice plane. An example of using one dividing plane has already been demonstrated in Figure 4.1(b) — the mis-registration of the data can still be seen, but no longer detracts from the useful information in the B-scans.



(a) Segmentation and planes

(b) B-scan review window

Figure 4.7: **Editing dividing planes.** The data is from the same examination as in Figure 4.2, although a more complex arrangement of dividing planes has been used in this case. Planes being edited are marked in red in both ‘outline’ and ‘review’ windows, and both windows are updated as the plane is moved.

### Display and manual editing of dividing planes

There are a couple of situations where dividing planes may not be generated in the most appropriate positions. If the geometry of each sweep is complex, it will not be well represented by the B-scans at the extremities of the sweep, and the dividing planes may not partition space correctly. In addition, if the ultrasound data does not completely cover the recorded image for each B-scan, and there is not much overlap between each of the sweeps, the dividing planes may not be at the optimum position to separate the actual ultrasound data in each image. For this reason, dividing planes can also be edited in the ‘outline’ window. Figure 4.7 shows an example of a more complex dividing plane arrangement from the same data as in Figure 4.2. There are two important features in the display of these planes, that simplify the editing process.

Firstly, dividing planes are inherently infinite — since it is not possible to show the orientation of an infinite plane, all planes are clipped to a sphere, centred at the mid-point of the data and large enough to contain all of it. Clipping all the planes to the same sphere ensures that the displayed discs will meet at their edges, as in Figure 4.7(a). Secondly, again shown in this figure, dividing planes are only rendered opaque *where they separate data from two different sweeps*: in other areas only the outer edge of the disc representing that plane is rendered. Hence the number of displayed surfaces is dependent only on the number of sweeps, and not on the number of dividing planes — in this figure there is only one surface, since there are only two sweeps. Rendering is achieved by clipping the disc representing each plane to each partition, resulting in a set of convex polygons, using the algorithm in Appendix C.3.

A particular plane can be edited by selecting it with a right mouse click (whereupon it is drawn in red), then moved by clicking and dragging (one button rotates, the other translates). Planes can also be inserted and deleted. As the plane is being moved, the sweep used in any



given partition is recalculated using equation (4.2), and the opaque shading of the planes is updated. This means that as the plane is moved it is immediately obvious which part is being used to separate data (i.e. the opaque part of the plane), and whether the plane is being used at all — if not it will be shown as a red circle with no opaque part.

In addition, as the plane is being moved, the position of its intersection with the current B-scan in the ‘review’ window is also updated, i.e. the red line and cross-hatching in Figure 4.7(b). This is useful for showing whether the plane intersects the actual B-scan data, rather than just the rectangular recorded image.

### 4.2.3 Segmentation with dividing planes

Segmentation is performed in the ‘review’ window. Any part of the review window that is in a partition for which data from another sweep is being used, is cross-hatched. This means that it is clear which part of the B-scan is better covered by data from another sweep, as in Figure 4.7(b). Which regions to cross-hatch can be determined by the same algorithm used to calculate the opaque regions of the dividing plane discs, outlined in Appendix C.3. For the latter case, this algorithm is initialised with a polygon representing the dividing plane disc. For the former, it can be initialised with a rectangle representing the B-scan under review.

Contours can be drawn in one of two modes: either by holding the mouse button down and dragging, or by marking points individually with a mouse button click. Once the last point has been marked (or the mouse button has been released, if using the former mode), the contour is joined, *unless* both the start and end points lie within the cross-hatched region, in which case the contour is left open. This ensures that contours are either themselves closed, or closed by the dividing planes — which fulfils a requirement of the volume measurement and surface reconstruction algorithms in Sections 4.2.4 and 4.2.5. In both the ‘outline’ and ‘review’ windows, parts of the cross-sections which lie in partitions for which the sweep is being used are coloured cyan, and the remainder dark blue. Hence, only cyan parts of the cross-section are used in the volume and surface calculations.

### 4.2.4 Application to volume measurement

In order to use cubic planimetry, introduced in Chapter 2, to calculate volumes with partitioned multiple-sweep data, the volume must be calculated in each partition of space, then summed to give the total. The effect this summation will have on the volume estimate can be assessed by considering a set of cross-sections cut by a single dividing plane, as in Figure 4.8. It is assumed that the dividing plane passes through *all* of the cross-sections — situations where this is not the case are discussed later. Note that the vector areas  $\vec{s}$  and centroids  $\vec{\omega}$  in each partition are related to the true values as follows:

$$\vec{s}_i = \vec{s}_{ai} + \vec{s}_{bi}, \quad \hat{s}_i = \hat{s}_{ai} = \hat{s}_{bi} \quad (4.3)$$

$$|\vec{s}_i| \vec{\omega}_i = |\vec{s}_{ai}| \vec{\omega}_{ai} + |\vec{s}_{bi}| \vec{\omega}_{bi} \quad (4.4)$$

The volume of the object, calculated using linear planimetry<sup>4</sup>, is:

$$v = \left| \frac{1}{2} \sum_{i=2}^4 (\vec{s}_i + \vec{s}_{i-1}) \cdot (\vec{\omega}_i - \vec{\omega}_{i-1}) \right| \quad (4.5)$$

<sup>4</sup>The linear version is examined here, rather than the cubic version, in order to make the maths more tractable.

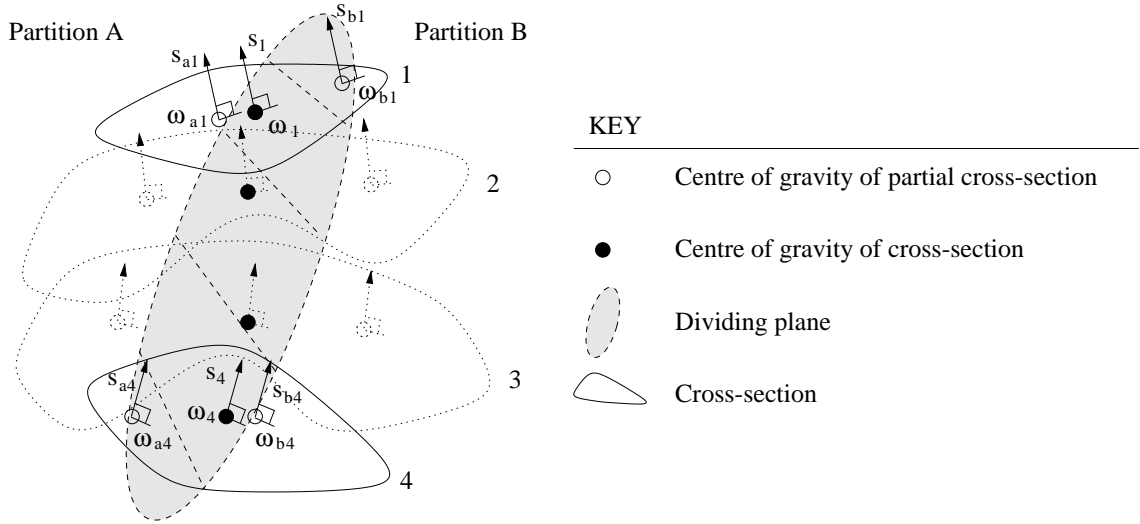


Figure 4.8: **Calculation of volume in each partition.** The dividing plane splits the cross-sections  $1 \dots 4$  into partitions *A* and *B*. The object volume can be calculated from equation (4.5) with  $\vec{\omega}_1 \dots \vec{\omega}_4$  and  $\vec{s}_1 \dots \vec{s}_4$ , or from the sum of the similar values with subscript *a* and *b*, in partitions *A* and *B* respectively.

where  $v$  is the volume of the four cross-sections in Figure 4.8, which have vector areas  $\vec{s}_1 \dots \vec{s}_4$  and centroids  $\vec{\omega}_1 \dots \vec{\omega}_4$ .

If, instead of using the areas and centroids of the whole cross-sections, the volume is calculated for each of the two partitions, and summed, this gives:

$$v = \left| \frac{1}{2} \sum_{i=2}^4 \{ (\vec{s}_{ai} + \vec{s}_{ai-1}) \cdot (\vec{\omega}_{ai} - \vec{\omega}_{ai-1}) + (\vec{s}_{bi} + \vec{s}_{bi-1}) \cdot (\vec{\omega}_{bi} - \vec{\omega}_{bi-1}) \} \right| \quad (4.6)$$

If we define  $l_{ai}$  as the ratio of partition *A* to the whole cross-sectional area,  $|\vec{s}_{ai}| / |\vec{s}_i|$ , and  $\alpha_{ai}$  as the vector distance between the centroid of partition *A* and the whole cross-section centroid,  $\vec{\omega}_i - \vec{\omega}_{ai}$ , for any cross-section  $i$ , this leads to the substitutions:

$$\begin{aligned} \vec{s}_{ai} &= l_{ai} \vec{s}_i \\ \vec{s}_{bi} &= (1 - l_{ai}) \vec{s}_i \\ \vec{\omega}_{ai} &= \vec{\omega}_i - \alpha_{ai} \\ \vec{\omega}_{bi} &= \vec{\omega}_i + \frac{l_{ai}}{1 - l_{ai}} \alpha_{ai} \end{aligned} \quad (4.7)$$

Substituting equations (4.7) into equation (4.6) and rearranging leads to:

$$v = \left| \frac{1}{2} \sum_{i=2}^4 \left\{ (\vec{s}_i + \vec{s}_{i-1}) \cdot (\vec{\omega}_i - \vec{\omega}_{i-1}) + (l_{ai} - l_{ai-1}) \left[ \frac{\vec{s}_i \cdot \alpha_{ai-1}}{1 - l_{ai-1}} + \frac{\vec{s}_{i-1} \cdot \alpha_{ai}}{1 - l_{ai}} \right] \right\} \right| \quad (4.8)$$

The first half of equation (4.8) is the same as equation (4.5), i.e. the volume calculated without using the dividing plane. The second part therefore represents the error in the volume estimate introduced by splitting the cross-sections into two parts and performing the volume calculation on each part. There are two interesting points to note from this expression.

Firstly, the error is dependent on terms like  $\vec{s}_i \cdot \alpha_{ai-1}$ . Since  $\vec{s}_i$  is by definition a vector normal to the plane  $i$ , and  $\alpha_{ai-1}$  is by definition a vector lying within the plane  $i - 1$ , if  $i$  and



$i - 1$  are parallel, the error will be zero. This is no surprise, since equation (4.5) reduces to  $area \times slice\ thickness$  for the case of parallel B-scans; and we would not expect this formula to be affected by summing volumes over several sections of the object in this way.

Secondly, the second term in equation (4.8) scales with  $(l_{ai} - l_{ai-1})$ , which can be interpreted as the difference in the area ratio into which the cross-sections on sequential B-scans are cut by the dividing plane. Hence, if the dividing plane cuts all the cross-sections with the same area ratio, the error will once again be zero. This scenario will tend to occur for dividing planes that are orthogonal to the B-scans.

The result of this is that equation (4.6) gives volumes close to that of equation (4.5) in all situations except where the B-scans are highly non-parallel *and* the dividing plane passes through these B-scans at an acute angle — which happens rarely in practice. This observation is consistent with the results of the simulated scanning experiments in Section 4.4.

In practice, the main error introduced by using dividing planes is that of *partial voluming*, i.e. where a part of the object is missed from the volume calculation entirely. In Figure 4.8, where the dividing plane cuts through all the cross-sections, this is not a problem. However, it is often the case that the dividing plane does not intersect all the cross-sections. As the volume calculation for each partition only includes cross-sections which exist in that partition, small parts of the object will be left out of the volume calculation. This is particularly true for dividing planes which have a shallow incidence with the B-scans.

#### 4.2.5 Application to surface visualisation

The surface interpolation and visualisation method of Chapter 3 needs some adjustment for multiple-sweep data. The cross-sections in this case are not generally closed, but a closed shape is required in order to calculate the signed distance field used in this algorithm. Cross-sections which are not closed are therefore joined at their end-points, which lie entirely outside the partition in which the data is used. Then, after the surface is interpolated and triangulated, triangles lying outside the relevant partition are removed, and those which intersect the partition are clipped to the dividing planes surrounding the partition. The clipping requires some care, since it is possible for triangles to intersect more than one dividing plane, in which case they must be clipped to all such planes — this is done by a recursive algorithm described in Appendix C.4.

The surface is extracted for each partition, using in each case the sweep most appropriate to that partition, as defined by equation (4.2). This results in a set of surfaces which together make up the entire object. Although the surfaces are not in general closed, edges *are* guaranteed to lie on dividing planes. Examples of such surfaces are included in Figure 4.9 for simulated scans and Figure 4.12 for scans of a human liver. As with the reslice shown in Figure 4.1, errors due to mis-registration are clearly apparent, but do not detract from the remainder of the data.

It is also possible to calculate a secondary estimate of volume from this set of surfaces, providing each surface intersects no more than two dividing planes. This algorithm is described in Appendix B.2.

### 4.3 Simulated scanning results

In order to verify the volume measurement and surface reconstruction techniques, without introducing errors common to all systems due to registration and segmentation, several objects were ‘scanned’ in simulation, using the tool described in Appendix A.1. Surface reconstructions from multiple-sweep scans of these objects are shown in Figure 4.9. Each object was precisely

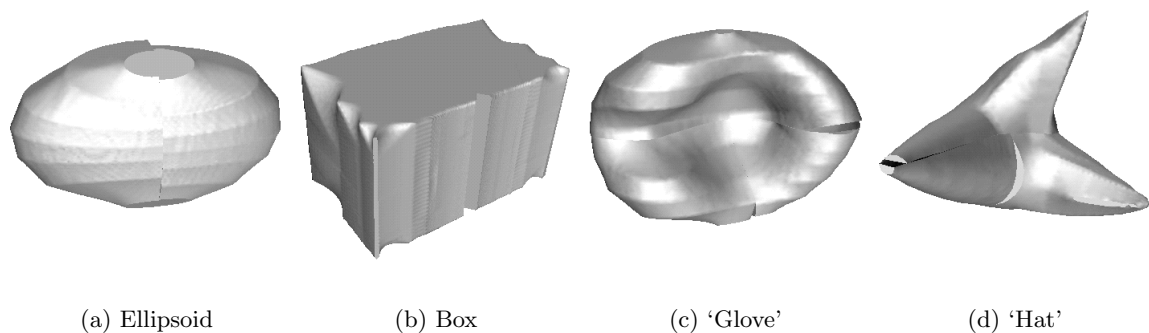


Figure 4.9: **Reconstructed surfaces from multiple sweeps.** Surface reconstructions are shown of some of the objects in Appendix A.1. Multiple sweeps were used as in Figure 4.10(b), (c), (g) and (h) respectively. Difference in B-scan location in each of the sweeps leads to small gaps between the surfaces from each sweep, despite perfect registration of the B-scans.

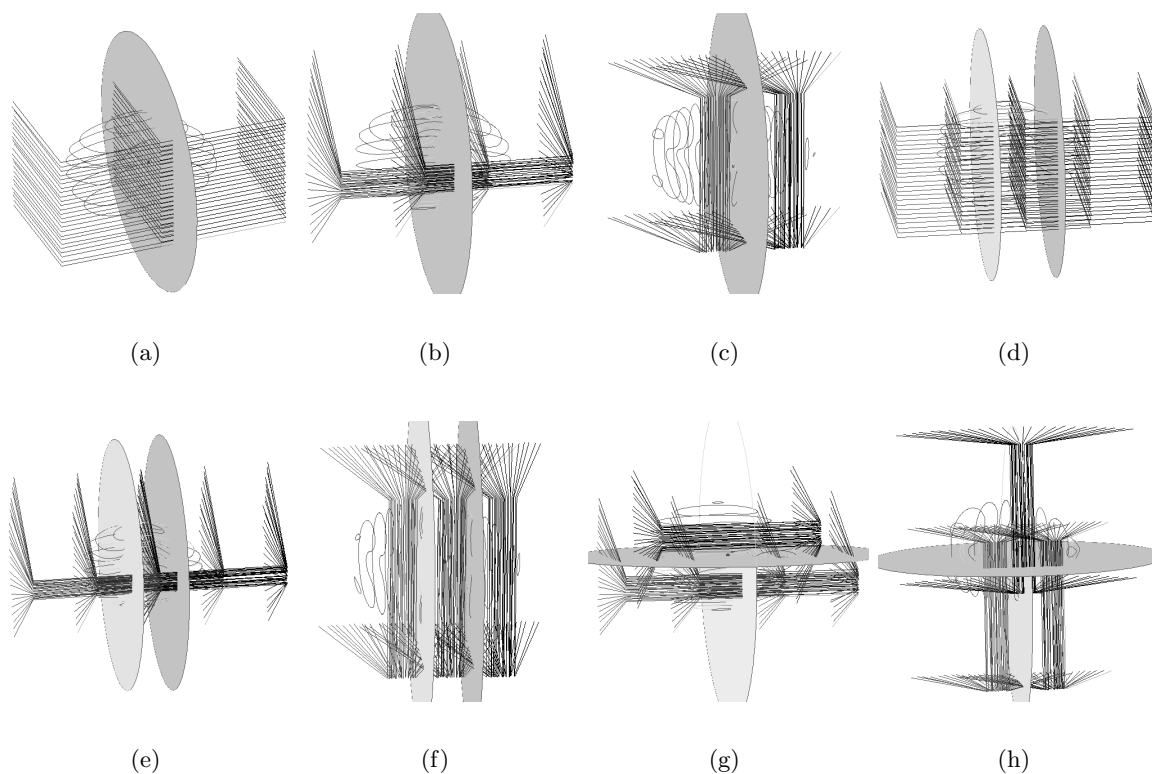


Figure 4.10: **Simulated multiple sweeps.** (a) to (h) are the sweep patterns used to test the volume measurement accuracy on simulated objects. Each B-scan is drawn as a 'goal post', where the 'crossbar' is at the top of the B-scan. All are shown with a segmentation from the 'glove' object. Dividing planes are calculated automatically from the sweep position and orientation.

Table 4.1: **Simulation results.** The table shows the total number of cross-sections, using sweep configurations (a) to (h), required for the cubic planimetry volume to be within  $\pm 2\%$  of the actual volume. The accuracy of the volume, as calculated from linear planimetry and the surface interpolated from these cross-sections, is also shown for comparison.

	Sweep:	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
<b>No. of scans for <math>\pm 2\%</math> accuracy</b>	Ellipsoid	<12	13	24	<16	17	21	22	<17
	Box	<12	16	15	<18	19	<18	19	<19
	Concave	<12	<12	18	17	17	18	16	18
	Branching	16	13	16	17	22	22	18	19
<b>Accuracy of linear planimetry volume at this point, <math>\pm\%</math></b>	Ellipsoid	>4.8	5.9	3.8	>5.8	7.3	6.9	5.4	>7.2
	Box	>1.5	3.5	7.7	>1.5	5.6	>8.8	4.4	>5.7
	Concave	>5.7	>7.7	4.1	6.4	7.9	6.7	7.4	6.8
	Branching	1.0	5.2	5.2	3.0	6.5	5.3	5.3	5.9
<b>Accuracy of volume from surface at this point, <math>\pm\%</math></b>	Ellipsoid	>7.0	7.7	1.5	>9.3	9.3	1.8	4.0	>9.0
	Box	>1.6	4.0	2.0	>1.5	4.4	>0.6	0.5	>4.0
	Concave	>12.0	>11.3	2.5	9.0	10.7	4.7	6.2	5.8
	Branching	7.8	12.6	4.7	10.8	10.5	2.5	7.0	7.7

segmented by thresholding the simulated B-scans, using a range of cross-sections between 4 and 23 *per sweep*, in each of the eight sweep patterns described in Figure 4.10. These sweep patterns were chosen to be representative of actual clinical situations; for instance, scanning between ribs.

The pixel size in all cases was 0.012cm, and the average volume of the objects was 5.4cm<sup>3</sup>, leading to inherent volume inaccuracy due to the sampling resolution of approximately  $\pm 0.7\%$ . As discussed in Section 4.2.4, the position of the first and last cross-sections can cause a partial voluming effect — in order to minimise this effect, these cross-sections were fixed at all times to be close to the edge of the scanned object.

Volume measurements were made from the cross-sections by cubic planimetry, and also by linear planimetry, and by calculating the volume from the interpolated surface. The latter two methods are more commonly used than the first, and serve as a comparison with the cubic planimetry technique.

Table 4.1 contains the results for each object and for each sweep pattern. The total number of cross-sections required for the cubic planimetry estimate to be within  $\pm 2\%$  of the real volume is shown in the top four lines of the table. Where *all* the experiments on an object gave volumes within  $\pm 2\%$ , the lowest number of cross-sections investigated is shown. The remaining rows of Table 4.1 show the accuracy of the other two volume measurement methods for the number of cross-sections detailed in the top four rows. 2% was selected as a test point since it is above the limit of resolution (0.7%), but still more accurate than the entire *in vivo* system, assessed for single-sweep data in Section 2.4.

It is clear from this table that volume measurements to an accuracy of within  $\pm 2\%$  are possible for all the objects described in Figure 4.9, and all sweep patterns described in Figure 4.10. Whilst there is *some* variation in the number of cross-sections required to obtain this accuracy, typically only 7 or 8 cross-sections are required *per sweep* (sweep patterns (a) to (c) contain two sweeps, the remainder contain three). The only exception to this, where 12 cross-sections are required, is for sweep pattern (c). This is a good example of the case described in Section 4.2.4,

where the cross-sections are non-parallel, and the dividing plane cuts through them at a highly acute angle, leading to larger errors in the volume estimate.

In all cases, with the single exception of the branching object with sweep (a), the linear planimetry accuracy is worse than that of cubic planimetry — typically greater than  $\pm 5\%$ . This is similar to the previous observations with single-sweep data in Section 2.3, and indicates that those results can be carried through to the multiple-sweep case presented here.

The accuracy of the volume calculated from the surface representation was generally slightly less than that of linear planimetry, with a greater variability across shape and sweep pattern. However, the similarity of the volume measurements to that of cubic planimetry suggests that there were no gross errors in either the surface interpolation or the volume calculation algorithms. This is backed up by the surfaces themselves, a sample of which are shown in Figure 4.9 — in fact, surface visualisation gave sensible results for all the cases in Table 4.1.

## 4.4 *In vivo* ultrasound results

### 4.4.1 Liver

The *in vivo* precision of this volume measurement method was estimated by considering multiple examinations of the same organ. Ten examinations were performed on the livers of each of two healthy subjects. For each subject, five of these examinations involved two ultrasound sweeps, as in Figure 4.2(a), and five involved three sweeps as in Figure 4.3(a). The actual orientation of these sweeps varied somewhat between examinations. Each complete examination was performed in a single breath hold (roughly 20 seconds).

The equipment and method used is described in Appendix A.2. Typically, 20 cross-sections were outlined in each case, with visual feedback from the ‘outline’ window and the interpolated surface, when it was not obvious where the cross-section should be. This whole process (from scanning to volume measurement) took approximately 30 minutes per data set, the vast majority of time being spent on manual segmentation. For the purpose of this experiment, the actual volume measurement was hidden during segmentation, so there could be no chance of increasing or decreasing the size of the cross-sections in an unconscious attempt to make the volumes similar for the same subject.

Figure 4.11 shows the results for both subjects, and the mean and 95% confidence interval in each case (assuming a normal distribution). The volume of the liver of subject 1 was  $1391 \pm 90\text{ml}$  (6.5%) and that of subject 2 was  $1037 \pm 61\text{ml}$  (5.9%). The first five observations were from examinations using two sweeps, and the second using three sweeps. Considered separately, the volume for the two sweep examination of subject 1 was  $1401 \pm 74\text{ml}$  (5.3%) and for subject 2 was  $1027 \pm 52\text{ml}$  (5.1%); and that for the three sweep examination for subject 1 was  $1381 \pm 100\text{ml}$  (7.2%) and for subject 2 was  $1046 \pm 62\text{ml}$  (5.9%). These results indicate that the overall precision of the system is approximately  $\pm 7\%$ , and that using two sweeps was better than three, in this case, giving a precision of approximately  $\pm 5\%$ . It is not clear whether this improvement was due to the use of fewer sweeps (and hence fewer dividing planes) or the improved definition of the liver boundary in the sagittal (as in the right hand sweep of Figure 4.2(a)), rather than horizontal (as in all other sweeps in Figures 4.2 and 4.3), scanning planes.

Surfaces reconstructed from each of the examinations of Figure 4.11 are shown in Figure 4.12, looking along the longitudinal axis to the inferior side of the liver, such that the scan-head was at the top of the surfaces, i.e. the same orientation as in Figures 4.2 and 4.3. Although there is considerable variation due to both scanning pattern and segmentation, all the livers can be

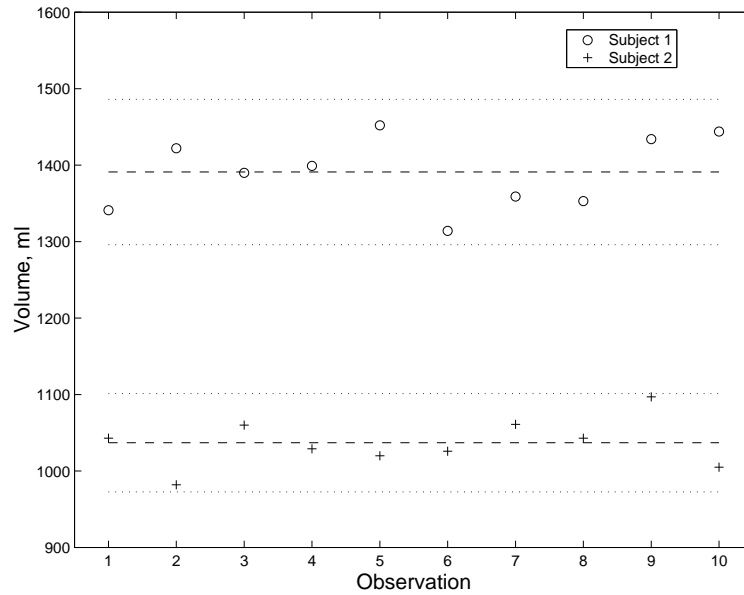


Figure 4.11: **Liver volume for two subjects.** The first five observations used two sweeps, and the second five used three sweeps. Volumes were calculated using cubic planimetry. Dashed lines show the mean, and dotted lines the 95% confidence intervals.

clearly categorised as being from subject 1 or 2. Variation in the surfaces is caused by probe pressure at the top (the shape of the convex curvilinear probe can be clearly seen in all the surfaces), and also difficulty in segmenting the liver from the gall bladder and inferior caval and portal veins, especially given the sparsity of the cross-sections. The fanning action used for the majority of the sweeps also tended to result in oblique incidence at the edges of the liver, particularly in the right lobe. This made it difficult to segment the first and last cross-sections of that sweep, aggravating the partial voluming effect, and in many cases parts of the right lobe were missing from the volume calculation entirely. In addition, some B-scans, for instance the one in Figure 4.2(c), did not cover part of the right lobe, and in this case the segmentation had to be estimated from the remaining data.

#### 4.4.2 Foetus

This technique was also used to examine a foetus at 28 weeks. By this stage in its development, the foetus is usually too large to be scanned with a single sweep. The foetus in Figures 4.13 and 4.14 was over one litre in volume, with a crown-to-rump length of approximately 24cm, and abdominal diameter of 6.5cm.

Acquisition is particularly difficult for foetal scans, since the foetus must keep fairly still during the entire acquisition process, in order to limit the mis-registration in the data. The ability to quickly review the data and assess whether there was much movement is crucial. As with the foetal data in Chapter 2, approximately one in three data sets could be used for volume and surface estimation.

As for the liver, two scanning patterns were employed, involving two and three sweeps. These are shown in Figure 4.13(a) and 4.14(a). The volumes measured from each examination were within 2% of each other, despite the variation in the scanning pattern and the position of the foetus. The dividing planes could also be used to re-slice the data as in Figure 4.13(b)



Figure 4.12: **Reconstructed surfaces of the human liver.** The left hand columns are from subject one, and the right from subject two.



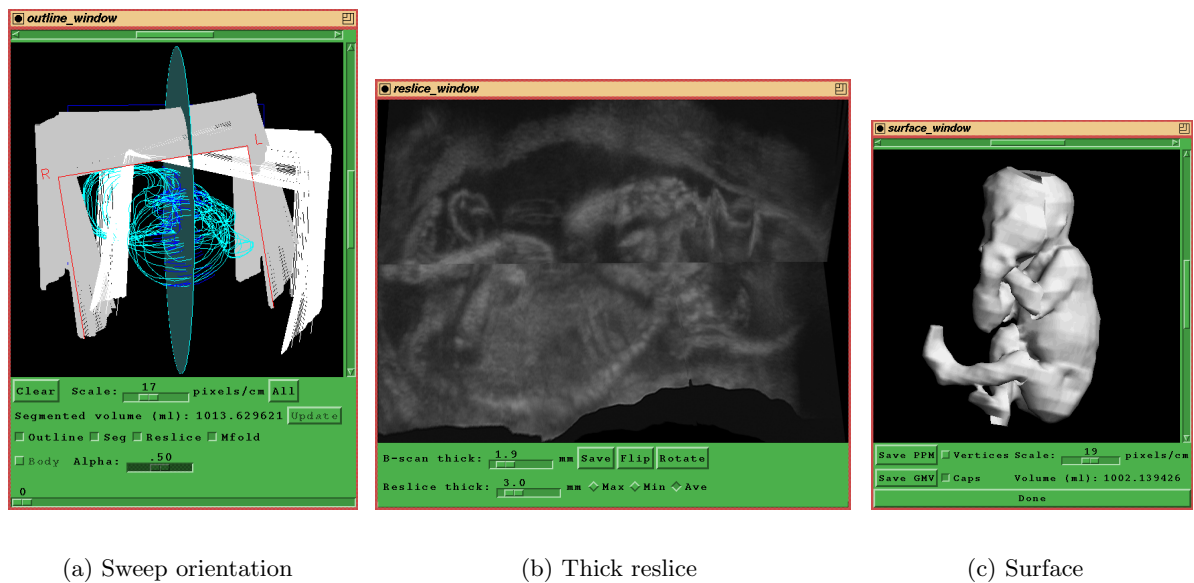


Figure 4.13: **Two-sweep examination of a fetus at 28 weeks.** The scanning pattern and dividing planes are shown in (a). (b) is an averaged, 3mm thick reslice through the data, and (c) shows the surface. The cubic planimetry volume was 1014ml.

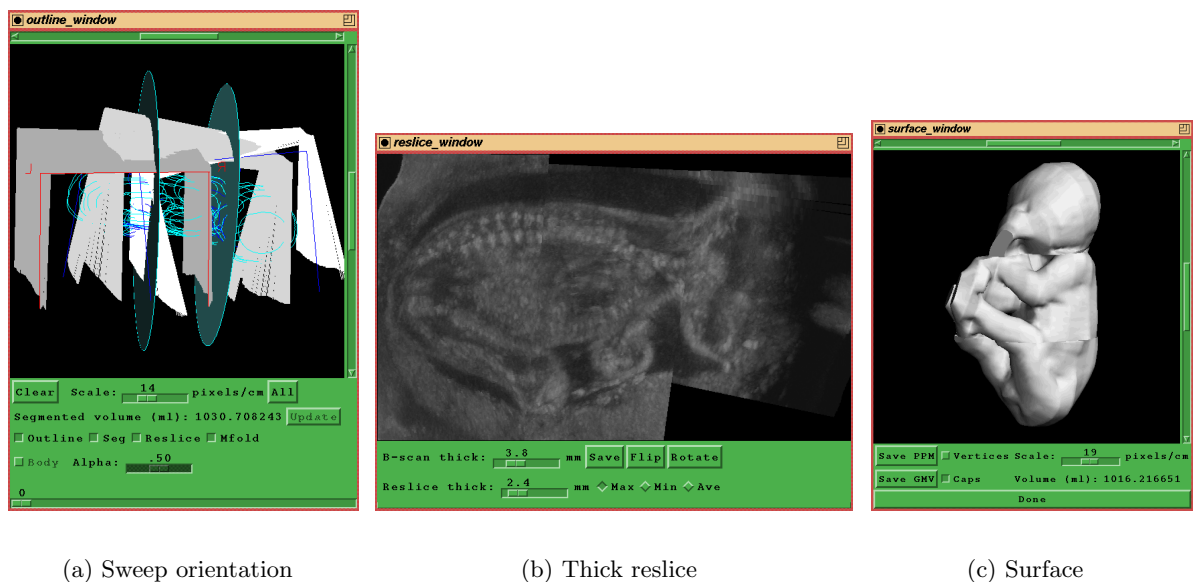


Figure 4.14: **Three-sweep examination of a fetus at 28 weeks.** The scanning pattern and dividing planes are shown in (a). (b) is a maximum intensity projection, 2.4mm thick reslice through the data, and (c) shows the surface. The cubic planimetry volume was 1031ml.

and 4.14(b), generating clear views of the whole foetus which would not otherwise be possible. The reslice of the two-sweep data demonstrates a movement artifact which had little effect on volume calculation, although it is also apparent from the surface reconstruction. Here, the head moved forwards then back to its original position, while the face was being scanned.



## Chapter 5

# 3D volume morphing: an extension of surface interpolation

### 5.1 Current methods of 3D morphing

#### 5.1.1 Comparison to image morphing

Image metamorphosis (or morphing) is the gradual transformation of one image into another, usually by a combination of warping and interpolation, reviewed in [70, 187]. Such techniques are popularised by their use in the film industry, where the morph is used to create the effect that one image ‘becomes’ the other. However, morphing can also be viewed as data interpolation, and similar techniques have been used within the medical imaging community to interpolate between parallel slices of data [154, 164, 171], as mentioned in Section 3.1.5.

3D metamorphosis is the gradual change of one *surface* into another. There has been less work on this subject, and it has been reviewed only recently [106]. 3D morphing can be used for shape transformation (morphing between different objects), animation (interpolating intermediate surfaces between the same object in two different positions) and surface synthesis (editing an existing shape by morphing it with a new one). There is also a close link between the interpolation of 2D cross-sections to form 3D surfaces, and the morphing of 3D surfaces to form 4D time series. It is the ease with which the surface interpolation and visualisation algorithms of Chapter 3 can be applied to 3D morphing which motivates the work in this area.

A major advantage of 3D over 2D morphing is its independence of viewing and lighting parameters — the model itself is morphed, and the rendering is performed on the morphed models. In 2D morphing, the rendering has already been done in creating the initial images, and hence it is impossible to maintain correct lighting, or move the viewpoint, during the morph. There are also situations where the aim is to create intermediate *models* rather than images, for instance improving a model of a beating heart by interpolating sampled data, and in this case there is no option other than to use a 3D morph.

#### 5.1.2 Comparison to functional surface interpolation

Shape-based interpolation, a form of functional interpolation of cross-sections to reconstruct surfaces, has been studied in detail in Section 3.1.2. Such techniques can be adapted to 3D volume morphing simply by extending the interpolation from 2D to 3D, thus creating a 4D (time varying 3D) sequence from two 3D volumes, as opposed to a 3D volume from a set of 2D

images. There are, however, some differences in assumption and philosophy that complicate this otherwise natural extension.

Firstly, when generating a morphing sequence between two objects, it is generally not desired that objects (or parts of objects) should appear out of nothing, or alternatively disappear into nothing. This has no parallel in reality, and hence tends to make the morph unrealistic. However, it is frequently the case that two *cross-sections* of the same object have parts that should not be connected; consider for instance two horizontal cross-sections of the letter ‘J’ — the upper consists of a single circle, and the lower of two, but only the right hand circle should be connected to the upper cross-section. This is because objects, or parts of objects, can be (and generally are) discontinuous in *space*; hence cross-sections of these objects will also be discontinuous. However, objects are never discontinuous in *time*. The assumptions underlying the calculation of correspondence of objects are therefore not the same as those which underly the correspondence of cross-sections.

Secondly, although both problems are highly under-determined and have many possible solutions, it is accepted that in surface interpolation there is some ‘optimum’ surface (even if we do not in fact know what this optimum is), and user interaction in generating this surface is undesirable. This is because the cross-sections are generally of some physical structure, and hence there is a ‘correct’ solution to the surface interpolation problem. Many applications of morphing, however, are not representative of real situations, and the correct solution can be most easily defined as ‘what the user wanted’. The issue here is therefore to provide users with an intuitive way of describing the morph, rather than making the decisions for them. Having said this, there are many morphs which everyone would agree look ‘wrong’ and there is hence scope for some automation to ensure the user is selecting between ‘right’ morphs rather than trying to avoid ‘wrong’ ones.

The third difference is related to the second, in that the desirability of user interaction introduces a new challenge to the problem of 3D morphing, especially since that interaction is with 3D time-varying data, and most display and input technology is designed for 2D. This applies equally to the manual definition of correspondence, and to the assessment of the morphing sequence.

### 5.1.3 Volume-based as opposed to polygon-based or implicit morphing

3D morphing algorithms can be split into three major categories [70, 106], according to the underlying representation of the surfaces being morphed. Polygonal representations can be morphed by interpolating the locations of the surface points [4, 72, 108, 189]. Implicit surfaces (i.e. those defined as an isosurface of a function generated from a set of primitives) can be morphed by interpolating the underlying primitives. Isosurfaces of volume data can be morphed by interpolating the volume data. Each strategy has its own particular strengths and in general it is possible to convert between representations in order to take advantage of these. In particular, most models are initially defined as polygonal meshes, and these can be converted to implicit representations (e.g. the skeleton representation [27] and union of spheres representation [150]), or to volume data [172].

Since volume representations are topology and shape independent (the underlying arrangement of the data stays the same — only the values change) they are inherently suited to handling complex morphing sequences. These representations are very similar to the interpolated distance fields discussed in Section 3.2, save that each voxel contains a signed minimum distance to the *surface*, rather than to a *cross-section* of that surface. Intermediate volumes can be created by

linear interpolation [142, 185]. Alternatively, new surfaces can be created by isosurfacing the initial volumes at a different value, hence generating *offset* surfaces [34].

Unfortunately, this independence from topology and shape makes it difficult to constrain the intermediate surfaces to the same topology and shape as the originals. Indeed, at the mid-point of the sequence, intermediate objects can only exist in regions where the original objects overlap. If there is no overlap, one object disappears and the other appears during the morph, creating a gap in the sequence. The visual effect is similar to cross-dissolving an image — one object ‘dissolves’ into another, but there is no sense of movement of any part of either object. The challenge in volume-based morphing techniques lies in controlling the way in which the volumetric data is interpolated in order to overcome this problem. This is the 3D analogy of the problem of interpolating cross-sections of objects scanned at oblique angles, discussed in Section 3.1.2.

One approach to this problem is to transform the volumes into the frequency domain using either wavelets [81] or Fourier transforms [88]. Interpolation is then performed in the frequency domain before being transformed back to the spatial domain for viewing. The main advantage is that these methods allow information at different frequencies to be transformed at different rates, usually with the emphasis on the low frequencies, which can improve the quality of the morph. General correspondence between the objects can also be established from the low frequency data. However, it is difficult to control the high frequency distortions that can be introduced, and there is little or no scope for user interaction to control the morphing sequence — both the cited methods are entirely automatic. Furthermore, the transformation to and from the frequency domain is both time consuming and memory demanding.

An alternative approach is to warp the source and target volumes before interpolation, or equivalently to vary the interpolation direction across the volume. In this case, the correspondence between the two volumes is defined manually, either with matching feature ‘elements’ [110] (consisting of points, lines, rectangles and boxes), or with matching oriented discs [40] or matching points [44]. Correspondences between various elements are combined by simple inverse distance weighting [40, 110] or used to calculate a smooth warping field [44]. In each case, the quality of the morphing sequence is entirely dependent on the manual definition of correspondence.

A new algorithm for automating correspondence in a framework similar to the latter approach has already been presented in Section 3.2. In the following section, this is expanded and applied to 3D morphing. The aim is to prevent unrealistic morphs by providing automation of correspondence, as in the former approach, whilst maintaining the user interaction inherent in the latter.

## 5.2 A new approach: sphere-guided interpolation

*Sphere-guided interpolation* is the extension of disc-guided interpolation, in Section 3.2, to the interpolation of 3D volumes of data. This approach is relatively fast, and integrates automated and manually defined object correspondence in order to improve the definition of the morphing sequence. Although volume graphics (the storage and display of volumetric data as opposed to polygonal surfaces) has advanced considerably over the last decade [99, 172], most surfaces are still defined as polygonal models, and most graphics hardware is designed to render such models. Regularised marching tetrahedra has already been presented in Section 3.4 for converting volume-based data to a polygonal representation. A technique is also presented in this chapter

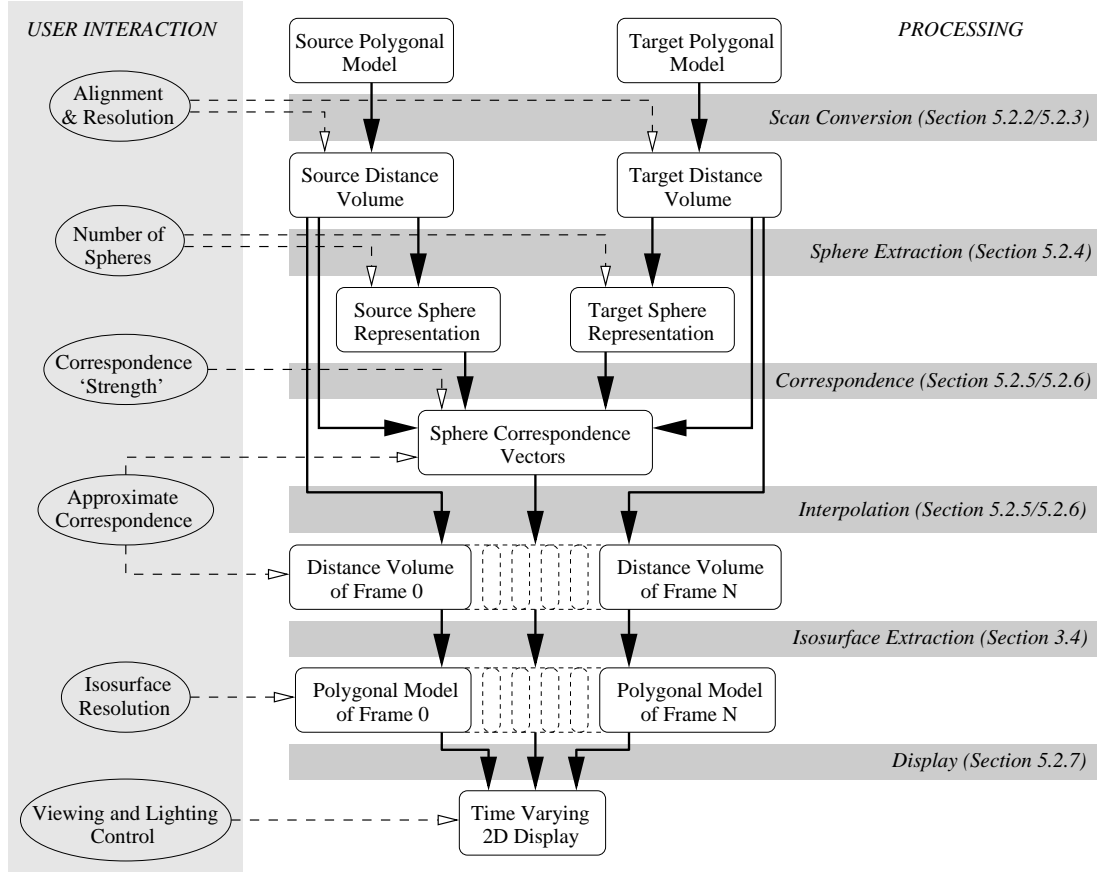


Figure 5.1: **System overview.** The morphing process begins with two polygonal models (top of diagram) and ends with a real-time display of the morphing sequence. User interaction with this process is shown on the left. The processing steps (and sections within this thesis where they are explained) are shown on the right.

for the inverse operation of converting polygonal meshes to volume-based data.

### 5.2.1 Overview

Figure 5.1 shows the main steps in the morphing of two polygonal models, and the points at which the user can interact with this process (though not in the same order as is presented to the user — see Section 5.2.7). The first step is to create the discrete volume representations from the polygonal models. The accuracy of this process determines the quality of the volume data, and hence also the final morphing sequence: or conversely it determines the size of volume required for a given model quality. It is at this point that the alignment and scale of the two models is set<sup>1</sup> — this is controlled by the user. Secondly, sphere representations are calculated from the two volumes. The user can control the coarseness of this representation, as appropriate to the complexity of the surface.

Correspondence between the two objects is then established by using the sphere representations. As with disc-guided interpolation, there are two important features of this correspondence. Firstly, it is not a mapping between spheres as in [150], but rather a correspondence *vector* is

<sup>1</sup>Alignment and scale could equally be determined after conversion to the volume representation, however realigning the volume data introduces sampling errors, whereas realigning the polygonal models does not.

established for each sphere, which is affected by all spheres in the other object. Secondly, since the correspondence is between spheres and not between the original objects, this correspondence is calculated at a *regional* level — i.e. it is not simply a mapping of one object to another. The correspondence calculation can therefore improve the morph even if there is only one object in each model, by determining which parts of which object should be connected. Unlike disc-guided interpolation, the user controls how ‘connected’ the morph is allowed to be, by varying the number of spheres and ‘correspondence strength’: see Sections 5.2.4 and 5.2.5.

Any number of intermediate volumes can be interpolated from the originals, using the sphere correspondence to guide the interpolation. These, in addition to the source and target volumes, are isosurfaced, using regularised marching tetrahedra (Section 3.4) to create polygonal models that can be rendered. In practice, the isosurface triangulation is performed as each intermediate volume is interpolated, and it is this, rather than the volume itself, that is stored. Real-time display of the morphing sequence can be achieved by looping through the sequence of models each time the display is rendered, during which the viewing and lighting can be interactively adjusted by the user.

Each of the steps in this process is discussed in more detail in Sections 5.2.2 to 5.2.5. The incorporation of manual definition of approximate correspondence, and how this is combined with automatic correspondence, is discussed in Section 5.2.6. The display of the morph sequence, and design of the user interface, are explained in Section 5.2.7.

## 5.2.2 Conversion to volume-based representation

The signed minimum distance to surface is not the only possible volume-based representation; indeed others exist which can represent non-closed surfaces within a volume [172]. In this case, however, the morphing algorithm requires the surface to be closed, and it is convenient to define the surface at the zero (rather than some other value) threshold. Intensity-defined volume data can be converted to this representation by thresholding, followed by distance computation, or more accurately converted by using *constrained elastic surface nets* [66]. Implicit surfaces are generally already defined by a distance function, and this function simply has to be evaluated at each voxel location.

In order to be able to represent a polygonal mesh as a distance volume, it must be closed, i.e. it must separate space into *outside* and *inside* regions. In addition, the surface must not be self-intersecting, since this implies a contradictory definition of signed minimum distance at the intersection. The geometric primitives must also be correctly oriented (i.e. the vertices are defined in a consistent order with respect to the direction of the outward surface normal)<sup>2</sup>. These criteria add constraints to the definition of polygonal models, all of which might be considered good practice, but none of which are necessarily required if the model has only to be rendered.

Given that a polygonal model is suitable for representation as a distance volume, the simplest way to convert it is by using a two-step process. In the first step (scan conversion), it is determined which voxels in the volume are inside and which are outside the surface; in the second (distance transformation), the distance of each inside voxel from the nearest outside voxel, and vice versa, is estimated. Both scan conversion [98] and distance transformation [31] can be performed very efficiently, the former by a single pass through the polygons and then the volume, the latter by two passes through the volume. The second step is similar to the distance transformation of cross-sections, and is discussed in the following section.

<sup>2</sup>This criterion does not apply to binary scan conversion.

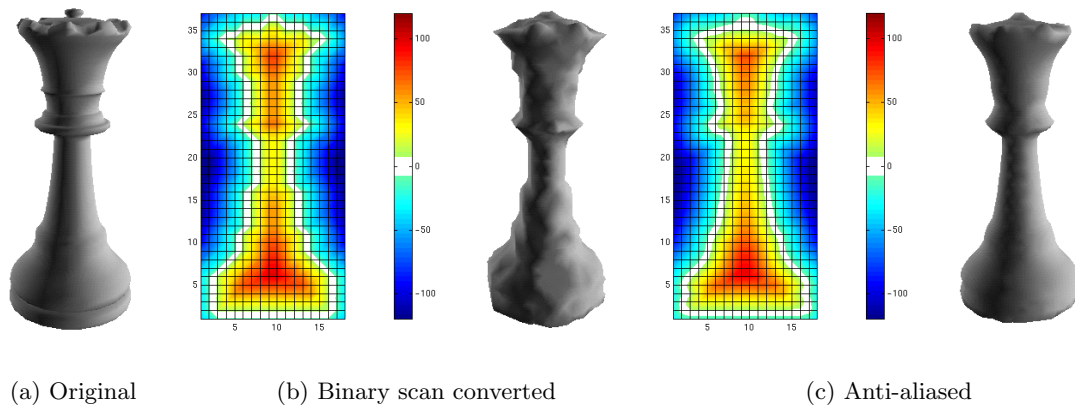


Figure 5.2: **Effect of model conversion.** (a) The original model is a polygonal mesh containing 3600 polygons. (b) and (c) show the results of scan-converting this into a volume of only  $18 \times 18 \times 37$  voxels, using binary and anti-aliasing techniques respectively. This is represented, in each case, by a slice through the volume (with linear interpolation of the values at each voxel), and the re-triangulated zero isosurface (containing approximately 2000 polygons).

Figure 5.2(b) shows results for the generation of a low resolution volume from the source model in Figure 5.2(a). The binary nature of the process is immediately apparent in both the distance volume itself and the triangulated isosurface. The distance values are calculated from voxel borders rather than the original surface, and hence the voxels are themselves apparent on the surface, although this effect is reduced by the use of a high quality isosurfacing algorithm.

The quality of the surface could be improved by calculating the distance volume at a much higher resolution; however, this would also dramatically increase both the storage requirement and processing time for the morph. It is much more efficient to use an anti-aliasing algorithm to initialise the distance volume, rather than binary scan conversion — this results in the much improved distance volume and surface of Figure 5.2(c). Voxels near to the surface are initialised with the exact (Euclidean) signed minimum distance, then this information is propagated to the remainder of the volume, as before, using a chamfer estimate [31]. The two-step nature of this process is similar to a technique employed for *constructive solid geometry* [34]. Like binary scan conversion, only one pass through the polygons making up the model is required (rather than testing each of the polygons at each voxel location, as in [93]).

For each polygon, each voxel inside the 3D bounding box surrounding the polygon is examined, and the distance to the polygon calculated, up to a pre-determined maximum. If it is greater than this maximum, the voxel remains un-initialised. The closest distance to a polygon in 3D space can be either to the plane containing the polygon, to one of the vertices of the polygon, or to one of the edges. It is most efficient to calculate the distance to the plane containing the polygon, then project the point onto that plane and calculate the remaining distances in this plane. An additional advantage of this technique is that if the distance to the plane is already greater than the maximum initialisation distance, there is no need to continue. The distance from a voxel to a polygon is stored if it is less than the initialisation distance, and the voxel is not initialised, or already contains a greater distance value. In practice, an initialisation distance of 1.1 (with respect to the width of a voxel) has been found to be sufficient to correctly define the surface.



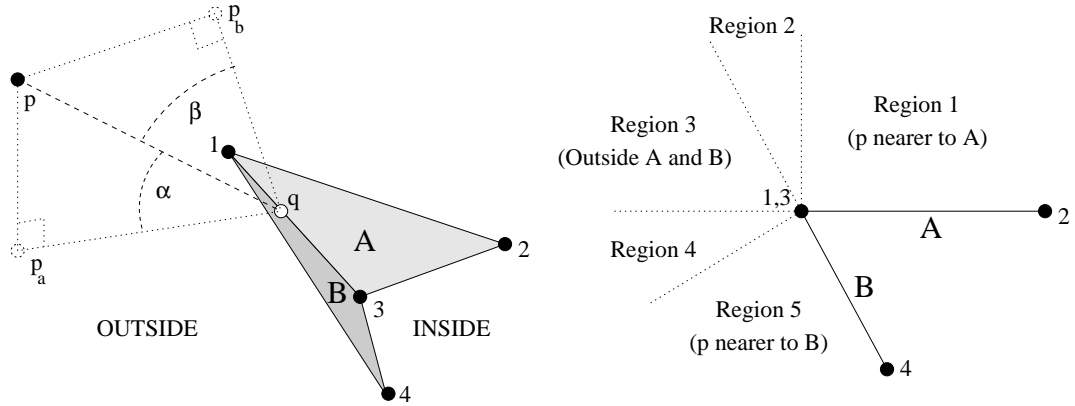


Figure 5.3: **Calculation of signed distance to surface.** If the nearest surface point  $q$  to a point  $p$  lies on the edge of two triangles  $A$  and  $B$ , the sign information from each triangle (i.e. whether the point  $p$  is outside or inside the surface) can be contradictory. The diagram on the right is a view of the triangles looking along the line from point 3 to 1. In regions 2 and 4 the distance to each triangle is the same, but the sign is not. In this case, the sign can be determined by considering the angles  $\alpha$  and  $\beta$  that  $\vec{pq}$  makes with the planes containing  $A$  and  $B$  respectively — the triangle with the largest of these angles has the correct sign.

The sign of the distance is derived from the orientation of the polygon vertices, and it is in the correct determination of this sign that the subtlety in the algorithm lies. For example, Figure 5.3 shows a point that is closest to the edge of two triangles. There are two regions in which the distance to each of these triangles is exactly the same, but the sign is not, and in these regions the angles  $\alpha$  and  $\beta$  must also be considered to determine the correct sign. Simply deciding that the point is outside if either of the triangles indicate such is not sufficient, as although this strategy would work in the case of Figure 5.3, it would not be correct in the complementary case where the inside and outside are swapped. In order to perform the conversion in one pass through the polygons, this angle must be stored at each voxel, along with the distance. This makes the complete test for distance initialisation as follows:

- If the voxel is not initialised, store the new distance and angle.
- Otherwise, if the absolute value of the new distance is less than the absolute value of the stored distance, store the new distance and angle.
- Otherwise, if the absolute value of the new distance is equal to the absolute value of the stored distance, set the sign of the distance to that with the smallest angle, and store this angle.
- Otherwise discard the new stored distance and angle.

This is similar to the technique used in [172], assuming that their ‘line parameter’ is used in much the same way as the angle described above.

### 5.2.3 Distance transformation

The distance transformation used to propagate this information throughout the volume is the 3D equivalent of the method described in Section 3.2.2 [31]. The process described in the previous

section serves as an initialisation to this algorithm. A 22-31-38 chamfer code is used (i.e. the width of one voxel is 22, the diagonal across a side is 31 and the double diagonal across the voxel is 38) to give reasonable accuracy, whilst storing the distance in only two bytes.

As in [34], other surface properties may be stored throughout the volume in addition to the distance, for instance the colour. In the implementation of Appendix D.3, the colour of a voxel close to the surface is initialised from the polygon closest to that voxel, and propagated throughout the volume during the distance transformation process. Voxels farther from the surface inherit the colour of the voxel from which the distance was propagated. This greatly enhances the eventual morphing sequence by allowing the gradual change of colour in addition to shape. Texture which has been mapped to the surface could also be morphed by this technique<sup>3</sup>.

As with disc-guided interpolation, the morphing process involves variation of the interpolation direction across the distance volume, and there is no guarantee that the points to be interpolated will lie within this volume. An estimate of distance from the surface (and colour) at any point in space is required. This estimate does not need to be very accurate, since these points will generally have little effect on the interpolated surface, being themselves far from the original surfaces. A simple method is used to calculate the distance at an arbitrary point:

- If the point is contained within the volume, the distance (and colour) is tri-linearly interpolated from the surrounding points in the volume.
- If the point is outside the volume, the intersection is found of the line joining it to the centre of the volume, with the side of the volume. The returned distance is the sum of the distance *to* this intersection point (using the same distance metric as used in the distance transformation), plus the distance *at* the intersection point. The colour is inherited directly from the intersection point.

### 5.2.4 Sphere representation

Sphere extraction from the 3D distance volume is performed in exactly the same manner as disc extraction from the 2D cross-sections in Section 3.2.3<sup>4</sup>. At each voxel, the sum of the difference of the distance value from each of the 26 neighbouring voxels is calculated. If this is positive, the voxel is at a change of gradient, and is therefore considered to be a candidate for a sphere centre. The sphere radius is simply the value of the distance field at that voxel. This set of spheres is thinned by starting with the largest, and removing all spheres within a certain fraction of that sphere's radius, then iterating.

The number of spheres (and hence the coarseness of the sphere representation) can be controlled by adjusting this fraction. Figures 5.4(b) to (e) show the sets of spheres extracted from the surface in Figure 5.4(a), with a progressively decreasing value for this fraction. Since the spheres are only used to determine correspondence, and not to define the surface itself, it is only necessary to have enough spheres to distinguish the important features of the object — Figure 5.4(d) is the most appropriate choice in this case. User manipulation of this parameter is described in Section 5.2.7. The effect on the morph of increasing the number of spheres is to gradually increase the onset of fine detail from the target surface on the source, and *vice versa*.

<sup>3</sup>To do this, it would be necessary to sample the texture map at the point on each polygon nearest to each voxel, and interpolate this sampled texture map, unless the texture was already supplied as a 3D map.

<sup>4</sup>Only internal spheres are used in this case, since this leads to more correspondence between objects themselves, which is more appropriate for morphing.



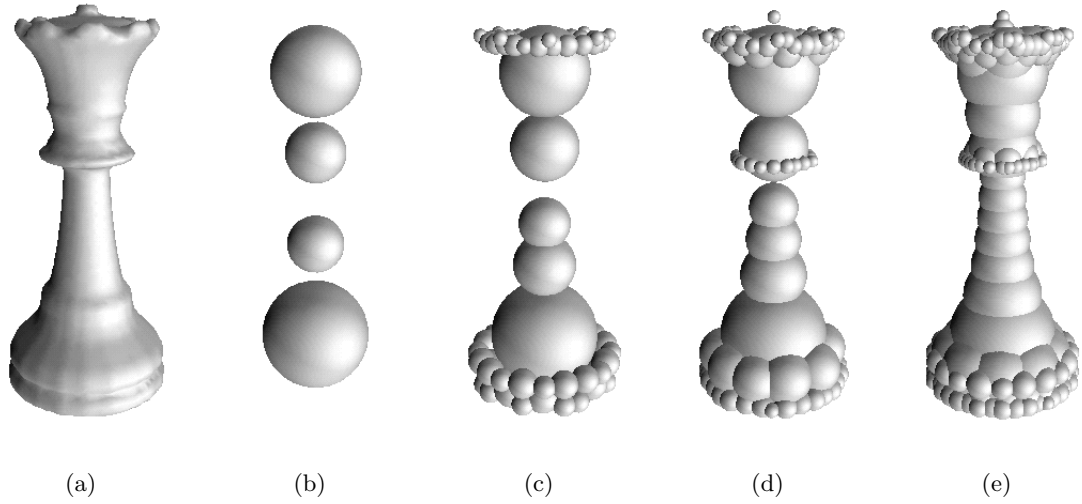


Figure 5.4: **Level of detail in the sphere representation.** (a) is the isosurface of the distance volume for the object in Figure 5.2, containing 8300 polygons. (b) to (e) are the results of progressively increasing the number of spheres in the representation.

### 5.2.5 Interpolation using sphere correspondence

Once again, sphere correspondence is calculated in the same way as disc correspondence for the 2D case (Section 3.2.4), except that the tolerance for connecting spheres from differing objects is extended so that no regions remain unconnected. This ensures that there will be no regions in the morphing sequence which either appear out of nothing, or disappear into nothing. Each sphere from one object is tested against all spheres from the other object. A correspondence vector  $\vec{c}_s$  is calculated for each sphere  $a$  in the source object, which is a weighted sum of the vectors connecting its centre with each other sphere  $b$  in the target object.  $\vec{c}_t$  is similarly calculated for each sphere  $b$  in the target object. The weighting for each pair of spheres  $a$  and  $b$ ,  $\omega_{ab}$ , is:

$$\omega_{ab} = \begin{cases} \frac{1}{(\varepsilon_a^2 + \mu)} + \frac{1}{(\varepsilon_b^2 + \mu)} & \text{if } \varepsilon_a < x \text{ and } \varepsilon_b < x \\ 0 & \text{otherwise} \end{cases} \quad (5.1)$$

$$\varepsilon_a = \left| \vec{l}_{ab} \right| - |r_a - d_s(b)|, \quad \varepsilon_b = \left| \vec{l}_{ab} \right| - |r_b - d_t(a)| \quad (5.2)$$

where  $\left| \vec{l}_{ab} \right|$  is the distance between the spheres,  $r_a$  and  $r_b$  are the radii of the spheres, and  $d_s(b)$  and  $d_t(a)$  are the distance values at the centre of each sphere on the *opposite* volume. The division in equation (5.1) is conditioned by a small constant  $\mu$  (set to the square of half the width of one voxel). This is the same as the disc weighting in equations (3.5) and (3.6), save that the correspondence strength  $x$ , replaces the disc radii — it is this parameter that controls the tolerance for connecting spheres from differing objects.

Figures 5.5(b) to (e) show the correspondence vectors for the two sphere sets in Figure 5.5(a), representing a queen and a pawn. The effect of increasing correspondence strength is twofold: more correspondence vectors are defined (up to the maximum of one per sphere) and these vectors tend to connect across a larger change in shape.

The sphere correspondence vectors are used to guide the direction in which the volume data is interpolated. One vector is calculated for each of the source and target objects,  $\vec{c}_{ps}$  and  $\vec{c}_{pt}$ ,

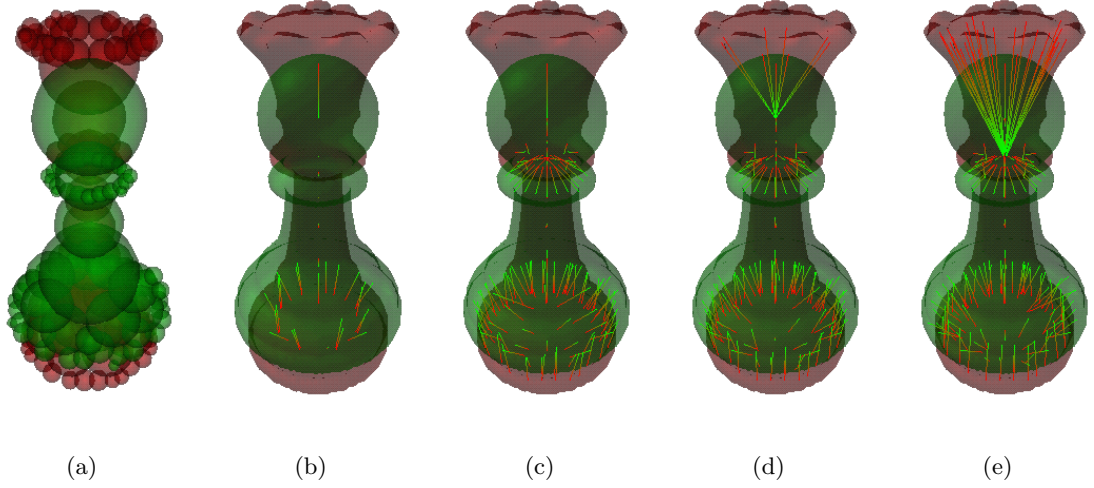


Figure 5.5: **Effect of sphere correspondence strength.** (a) to (e) are examples of the user interface for defining sphere correspondence. In all cases the source and target objects are rendered in translucent red and green, respectively. (a) is the sphere representation. (b) to (e) show the sphere correspondence vectors, with gradually increasing correspondence strength. The colour of each vector varies from red to green to show the direction of correspondence over time.

at each point  $p$ . This is also calculated from the weighted sum of all the sphere correspondence vectors:

$$\vec{c}_{ps} = \frac{1}{\sum_{a \in A} \omega_{pa}} \sum_{a \in A} \omega_{pa} \vec{c}_s, \quad \vec{c}_{pt} = \frac{1}{\sum_{b \in B} \omega_{pb}} \sum_{b \in B} \omega_{pb} \vec{c}_t \quad (5.3)$$

The weights  $\omega_{pa}$  and  $\omega_{pb}$ , for each sphere in the sets of all spheres A and B, in the source and target volumes respectively, is given by:

$$\omega_{pa} = \frac{1}{(\varepsilon_{pa}^2 + \mu)}, \quad \omega_{pb} = \frac{1}{(\varepsilon_{pb}^2 + \mu)} \quad (5.4)$$

$$\varepsilon_{pa} = |\vec{l}_{pa}| - |r_a - d_s(p)|, \quad \varepsilon_{pb} = |\vec{l}_{pb}| - |r_b - d_t(p)| \quad (5.5)$$

where  $\varepsilon_{pa}$ ,  $\varepsilon_{pb}$  and  $\mu$  have similar definitions as in equation (5.1).  $|\vec{l}_{pa}|$  and  $|\vec{l}_{pb}|$  are the distance from  $p$  to the centre of sphere  $a$  or  $b$  respectively.  $d_s(p)$  and  $d_t(p)$  are the distance values at the point  $p$ , in the source and target volumes respectively. Whereas  $\varepsilon_a$  and  $\varepsilon_b$  in equations (5.1) aim to ensure that sphere correspondence vectors connect similar regions,  $\varepsilon_{pa}$  and  $\varepsilon_{pb}$  in equations (5.5) aim to ensure that the vectors used at a specific location are from spheres which dominate the shape at that location. This is exactly the same behaviour as discussed for discs in Section 3.2.5.

Once the point interpolation vector has been determined, the interpolated distance value,  $d(p, t)$ , at point  $p$  and time  $t$  in the morphing sequence, is given by:

$$d(p, t) = td_t(p + (1 - t)\vec{c}_{pt}) + (1 - t)d_s(p - t\vec{c}_{ps}) \quad (5.6)$$

where the time  $t$  varies from 0 to 1,  $\vec{c}_{ps}$  and  $\vec{c}_{pt}$  are given by equation (5.3), and  $d_s(p)$  and  $d_t(p)$  are the distance field at point  $p$  in the source and target volumes respectively, as previously defined.

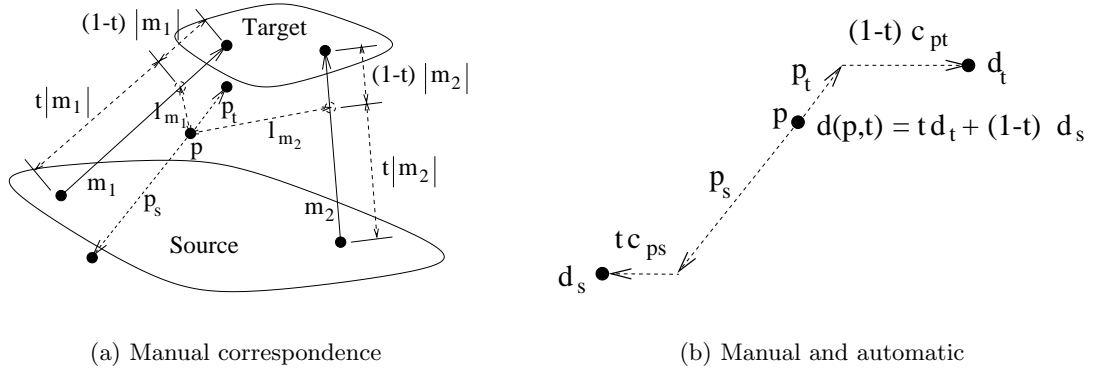


Figure 5.6: **Manual correspondence.** (a) Vectors  $\vec{m}_1$  and  $\vec{m}_2$  connect points in the source and target objects. For any point  $p$  at time  $t$ , two vectors are calculated,  $\vec{p}_s$  and  $\vec{p}_t$ , which give the points corresponding to  $p$  for the source and target objects respectively. These points are used to interpolate the value for point  $p$  at time  $t$  ( $t$  varies from 0 to 1). The calculation of  $\vec{p}_s$  and  $\vec{p}_t$  is by weighted sum, based on the inverse of distances  $l_{m_1}$  and  $l_{m_2}$  to points along the vectors  $\vec{m}_1$  and  $\vec{m}_2$ , which also depend on the time  $t$ . (b) These can be combined with the automatic correspondence vectors  $\vec{c}_{ps}$  and  $\vec{c}_{pt}$  to give  $d_s$  and  $d_t$ , from which the interpolated value  $d(p, t)$  for point  $p$  is calculated.

### 5.2.6 Manual guidance of correspondence

Section 5.2.5 demonstrates how correspondence between the source and target objects can be calculated automatically and used in the morphing sequence. For very complex morphs, for instance as in Figure 5.11, or to add more user control to the morphing sequence, this automatic calculation can be combined with a manual definition of correspondence. Since the sphere correspondence ensures that detailed connectivity is maintained across the objects, it is only necessary to define approximate correspondence manually (e.g. which limb is connected to which limb in Figure 5.11).

#### Defining manual correspondence

Manual correspondence is defined by a set of vectors that connect points in the source object to points in the target object (the user interface for defining such vectors is described in Section 5.2.7). Figure 5.6(a) shows an example where two such vectors,  $\vec{m}_1$  and  $\vec{m}_2$ , have been defined. These vectors are combined with a simple inverse distance scheme to give two correspondence vectors,  $\vec{p}_s$  and  $\vec{p}_t$ , which relate the point  $p$  to corresponding points in the source and target volume respectively. These vectors are dependent on both the position of the point  $p$  being interpolated, and the time  $t$  in the morphing sequence. Rather than calculating the inverse distance to the points at the ends of each manual correspondence vector, it is calculated to a point along the vector. This point moves with time from the source to the target end of the vector. This ensures that locations for which manual correspondence vectors have been defined are guaranteed to be connected at all time points in the morphing sequence.

### Combining manual and automatic correspondence

The manual correspondence method detailed above generates two vectors ( $\vec{p}_s$  and  $\vec{p}_t$  in Figure 5.6) for each point  $p$  in each time frame  $t$ . These vectors give the transformed points in the source and target volumes respectively, which are interpolated to generate the data for this point. If there is no automatic correspondence, the distance is linearly interpolated from these two points, as is the case in Figure 5.12(b).

If only a few manual correspondence vectors have been defined, this scheme alone is not sufficient to define correspondence over the entire object, and needs to be combined with the automatic correspondence vectors. This is done at two stages in the morphing process (Figure 5.1): when calculating sphere correspondence and when interpolating the intermediate volumes.

In calculating sphere correspondence, rather than projecting the centre of a sphere in the source volume to the target volume, the target manual correspondence vector at time 0 is used to transform the sphere centre before sampling the target volume. Conversely, when considering spheres from the target volume, the source manual correspondence vector at time 1 is used to transform the sphere centre before sampling the source volume. Equations (5.2) therefore become:

$$\varepsilon_a = \left| \vec{l}_{a(b+\vec{b}_s)} \right| - \left| r_a - d_s(b + \vec{b}_s) \right|, \quad \varepsilon_b = \left| \vec{l}_{(a+\vec{a}_t)b} \right| - \left| r_b - d_t(a + \vec{a}_t) \right| \quad (5.7)$$

where  $\vec{b}_s$  is the source manual correspondence vector at time 1 for the centre of sphere  $b$ , and similarly  $\vec{a}_t$  is the target manual correspondence vector at time 0 for the centre of sphere  $a$ .  $\left| \vec{l}_{a(b+\vec{b}_s)} \right|$  and  $\left| \vec{l}_{(a+\vec{a}_t)b} \right|$  are the distances between the centres of spheres  $a$  and  $b$ , respectively, and the transformed centres of the alternate sphere.

When interpolating each point at each time frame, the manual correspondence vectors are first used to transform the point to the source and target volumes. These transformed points, rather than the initial point, are used to calculate the two sphere correspondence vectors,  $\vec{c}_{ps}$  and  $\vec{c}_{pt}$ , for the source and target volume respectively. Hence, equations (5.5) are replaced by:

$$\varepsilon_{pa} = \left| \vec{l}_{(p+\vec{p}_s)a} \right| - \left| r_a - d_s(p + \vec{p}_s) \right|, \quad \varepsilon_{pb} = \left| \vec{l}_{(p+\vec{p}_t)b} \right| - \left| r_b - d_t(p + \vec{p}_t) \right| \quad (5.8)$$

where  $\left| \vec{l}_{(p+\vec{p}_s)a} \right|$  denotes the distance from the transformed point  $p + \vec{p}_s$  to the centre of sphere  $a$ , and similarly for  $\left| \vec{l}_{(p+\vec{p}_t)b} \right|$ .

Finally, the manual correspondence vectors  $\vec{p}_s$  and  $\vec{p}_t$  are also used in the interpolation of each point, so equation (5.6) is replaced by:

$$d(p, t) = td_t(p + \vec{p}_t + (1 - t)\vec{c}_{pt}) + (1 - t)d_s(p + \vec{p}_s - t\vec{c}_{ps}) \quad (5.9)$$

as shown in Figure 5.6(b). In effect, both the source and target volumes are warped by the manual correspondence vectors, before either the sphere correspondence, or interpolation, is performed.

#### 5.2.7 User interface design

The user is presented with one window, but three ‘states’, which can be traversed as in Figure 5.7. Manual correspondence is defined interactively in the first state, following which the automatic correspondence parameters can be adjusted in the second. Finally, the morphing sequence itself can be viewed in the third state. Each of these states leads logically to the next; they are all described in the following sections.

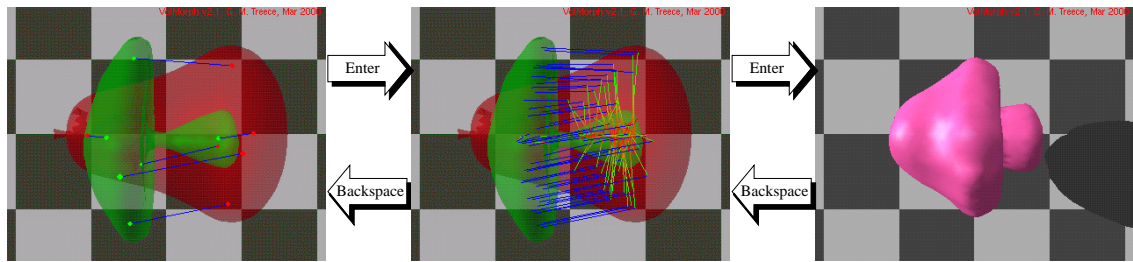


Figure 5.7: **Progression of the user interface.** The user is presented with a sequence of three displays — manual correspondence definition (on the left), sphere correspondence adjustment (centre) and the morph itself (right). This sequence can be traversed by using the ‘Enter’ and ‘Backspace’ keys. The intermediate surfaces for the morph are calculated on starting the final display.

### State one: creation of manual correspondence vectors

Manual correspondence vectors define a 3D correspondence between points in the source and target objects. However, both the display and the input device can only represent 2D information. A technique for defining such vectors in 2D is presented in [70] whereby the user first selects a point on the surface of an object (or on a pre-defined grid), following which the *direction* of a unit vector from that point is chosen, and finally the magnitude of this vector.

There are two difficulties with this approach. Firstly, it would be preferable to define points which are within, rather than on the surface of, the objects. Secondly, the interface is not particularly natural for the user — there are three separate operations in order to define just one correspondence vector. Most 2D drawing programs, on the other hand, use a click-and-drag approach to define vectors. This behaviour can be mimicked by using the objects themselves to provide the extra dimension missing from the input device. The procedure is as follows:

- Both objects are displayed simultaneously in transparent red (for the source object) and green (for the target), as in the left hand diagram of Figure 5.7.
- Clicking and dragging *outside* the objects changes the viewpoint and can be used to examine the objects.
- Once the objects are at an appropriate orientation where corresponding parts can be clearly seen, and are not obscured (either in front or behind), the user clicks on the source (red) object.
- A correspondence point is defined on this object, using the median distance between the surfaces underneath the selected point as the z-dimension.
- As the user drags the mouse, a vector is drawn from this point to a point on the target (green) object, providing the mouse is over part of the target object.
- On releasing the mouse, the z-dimension of the corresponding point is similarly defined as the median of the z-dimensions of the surface under the selected point.
- The spatial location of this vector can now be examined by clicking and dragging outside of both objects, or deleted by right-clicking on the vector itself.

This interface provides a very fast way of defining corresponding points. These points are drawn in the same colour as the object for which they are defined, with the connecting vectors in blue (as in the left-hand diagram of Figure 5.7). It is therefore easy to see which parts of which object are connected. In addition, the way in which the vectors are defined ensures that corresponding points will always be near the centres of the objects that they connect. The one exception to this is if the user clicks over part of an object which has further parts behind it; in this case the median distance over *all* these surfaces is used, which may lie outside any one of the surfaces. This situation can be clearly seen due to the transparency of the objects, and if such vectors are defined, changing the viewpoint makes them apparent, and they can subsequently be deleted.

### State two: selection of automatic correspondence parameters

There are two automatic correspondence parameters which can be adjusted to vary the eventual morphing sequence — the number of spheres and correspondence strength (see Sections 5.2.4 and 5.2.5). Changing the former will cause both the sphere representation and the sphere correspondence to be updated, whilst the latter only affects sphere correspondence. In all but the most complex cases, both of these can be recalculated in real time, providing interactive control of both parameters, as in the central image of Figure 5.7.

Both surfaces are once again rendered transparent red or green, but can be displayed either as the triangulated isosurface of the source and target volumes, or as their sphere representations, as in Figure 5.5(a). The correspondence vectors themselves are each displayed as lines gradually changing from red to green to indicate their sense, as in Figures 5.5(b) to (e). If manual correspondence vectors have also been defined, the part of each sphere correspondence vector due to the manual correspondence is shown in blue, and the remaining part in red to green, as in Figure 5.7(b). Both parameters can be changed using key presses, whereupon both the number and directions of the vectors are updated.

### State three: display of the morphing sequence

The interpolation of the morphing sequence is the most time consuming step in the process; the time taken can vary from a few seconds to a few hours, depending on the complexity of the correspondence, the isosurface resolution and the number of frames (or intermediate surfaces). This interpolation is performed on entering the third state of the user interface. Although the size of the volume is fixed at this point, the resolution of the isosurface compared to the volume (i.e. the approximate size of each of the triangles) is variable. Low resolution morphs can be calculated very quickly and used to assess the approximate behaviour before a high resolution morph is calculated.

The user can vary the viewpoint and model position, in addition to the lighting and speed of the morph, during the sequence. This enables sufficient interaction with the morph to determine if it has the desired effect. If not, then the user can return to the previous states to adjust either the automatic or manual correspondence, and then recalculate the morph once this has been done.



Table 5.1: **Morphing details.** The number of polygons, the size of the discrete volume and the times taken in conversion, user interaction and creating the morph are shown for the sequences in Figures 5.8, 5.9, 5.11 and 5.12. Times are given in hours:minutes:seconds, measured on a Silicon Graphics Indigo 2 R10000 workstation.

	<b>Pawn ⇒ Queen</b>	<b>Tricycle ⇒ Sphere</b>	<b>Dragon ⇒ Creature</b>	<b>Pear ⇒ Mushroom</b>
Source model polygons	2,500	40,000	110,000	900
Target model polygons	4,000	8,000	75,000	240
Dimensions of volume	$30 \times 30 \times 55$	$112 \times 105 \times 108$	$85 \times 136 \times 143$	$35 \times 38 \times 38$
Size of volume	0.19Mb	4.84Mb	6.31Mb	0.19Mb
Scan conversion time	00:00:03	00:00:35	00:01:06	00:00:01
User interaction time	00:00:30	00:01:00	00:30:00	00:00:30
Frames	20	20	20	20
Polygons per frame	$\approx 4,000$	$\approx 38,000$	$\approx 33,000$	$\approx 3,300$
Morph creation time	00:01:31	00:45:20	01:31:00	00:00:33

## 5.3 Results

The algorithms and interface described in Section 5.2 have been implemented using OpenGL<sup>5</sup> in VolMorph<sup>6</sup>, which is freely available for Irix, Linux and Windows platforms, as described in Appendix D.3. The polygonal models used in the following sections can be downloaded from the Geomview<sup>7</sup> distribution (pear and mushroom) or from 3D Cafe<sup>8</sup> (all other models).

Table 5.1 contains details of each of the following examples. Source and target volumes are stored as four bytes per voxel: two for the distance value and two for the colour. All examples are for twenty frame morphs (i.e. eighteen interpolated models), and the morph creation time given in the table is the time to create all of the intermediate models (including isosurface extraction). In each example, the new scheme is compared to a simple morph generated by linear interpolation of the distance values, with no calculation of region correspondence. This is equivalent to shape-based interpolation [142, 154] of the objects.

### 5.3.1 Pawn to queen

Figure 5.8 shows a morph from a white pawn to a black queen chessman. Shape-based interpolation in Figure 5.8(a) results in a variety of unnatural effects. The colour of the green base ‘bleeds’ up into the lower section of the intermediate shape, giving it a green tint. The crown of the queen unfolds from the top of the pawn head rather than growing from the side. Also, the collar from *both* models is apparent in the intermediate morphs, rather than a single collar in the average position.

All these problems can be corrected by using automatic correspondence. The user interaction time for this morph in Table 5.1 is for adjusting the correspondence strength alone, and the whole process (from initial models to display of a twenty frame morph) is completed in only two minutes. Figure 5.8(b) shows the results — both the shape and colour change are now much

<sup>5</sup><http://www.opengl.org>, Silicon Graphics Inc.

<sup>6</sup><http://svr-www.eng.cam.ac.uk/~gmt11/software/volmorph/volmorph.html>

<sup>7</sup><http://www.geom.umn.edu/software/geomview/>.

<sup>8</sup><http://www.3dcafe.com> ©1996–2000 Platinum Pictures. All rights reserved.



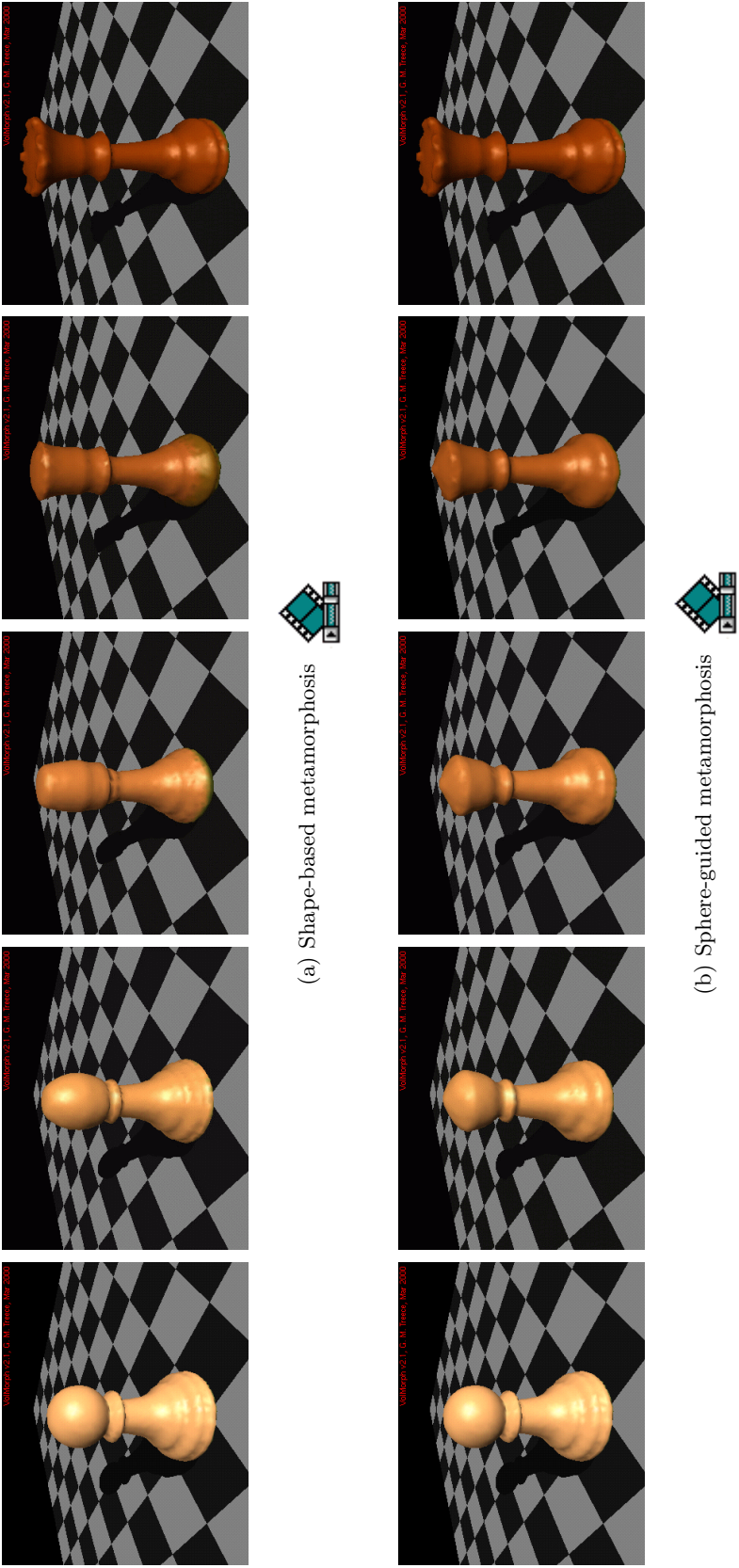


Figure 5.8: Pawn to queen metamorphosis.

more natural.

### 5.3.2 Tricycle to sphere

Figure 5.9 shows a morph from a coloured tricycle to a white sphere. The shape and topology of the objects are very different, and the simple strategy in Figure 5.9(a) is not able to cope with this — for instance the rear wheels and the handlebars disappear completely during the morph. In contrast, the morph of Figure 5.9(b) shows a gradual change of *all* parts of the tricycle to the sphere. The effect is as if the tricycle is gradually inflated; the components of the tricycle are visible at all stages of the morph. Interpolating colour as well as shape adds to this effect by giving visual clues as to which parts of the sphere correspond to which parts of the tricycle.

In this case, the user interaction time in Table 5.1 was for adjusting the number of spheres needed to represent the tricycle. In particular, both box sections (e.g. the back foot rest) and cylindrical sections (e.g. the handlebars) require a lower than usual density of spheres to be selected for good correspondence.

### 5.3.3 Dragon to creature

Figure 5.11 shows a shape-only morph from a dragon to an alien creature (neither of the models are coloured). Although both objects have the same number of limbs and general shape, there is considerable variation in both the detail and the poise. In fact, neither the limbs, tail, nor head are in the same position — as can be seen from the intermediate frame of Figure 5.11(a), where all the limbs have disappeared. In this case, automatic correspondence is not sufficient to define the morph, since both the left arm and the head are farther away from each other than from other body parts (the left arm, for instance, would otherwise be joined to the left thigh).

Approximately three manual correspondence vectors were defined for each limb, placed at the major joints, plus three each for the tail, head and torso, as shown in Figure 5.10(a). This results in the morph of Figure 5.11(b) — the limbs are now connected where the manual correspondence has been defined; however they still disappear in-between these points. There are resulting gaps in the surface at the tail and right claw, as well as a lack of definition at the extremities. The addition of automatic correspondence shown in Figure 5.10(d), results in Figure 5.11(c), in which most of these problems are corrected. However, the large movement of the left arm in contrast with the relatively stationary left leg and torso leads to a lack of definition in the left hand which is difficult to overcome.

Since this morph is more complex, the user interaction time in Table 5.1 is greater; this time also includes the creation of low resolution morphs to check the effect of manual correspondence. The models themselves are also more complex, leading to a larger volume and hence a longer processing time. However, the entire processing was still performed within two and a half hours.

### 5.3.4 Pear to mushroom

Figure 5.12 shows two possible morphs from a pear to a purple mushroom, demonstrating that there are many ‘correct’ solutions to the problem. Figure 5.12(a) is the result of using simple interpolation (the use of sphere-guided interpolation in this case generates the same result). Here the effect is as if the base of the pear has been squashed, resulting in the expansion of the top. On the other hand, Figure 5.12(b) was generated by defining a small set of manual correspondences, as in Figure 5.7<sup>9</sup>. Now the effect is as if the top of the pear has been pushed

<sup>9</sup>The automatic correspondence from Figure 5.7 was not used in this case.

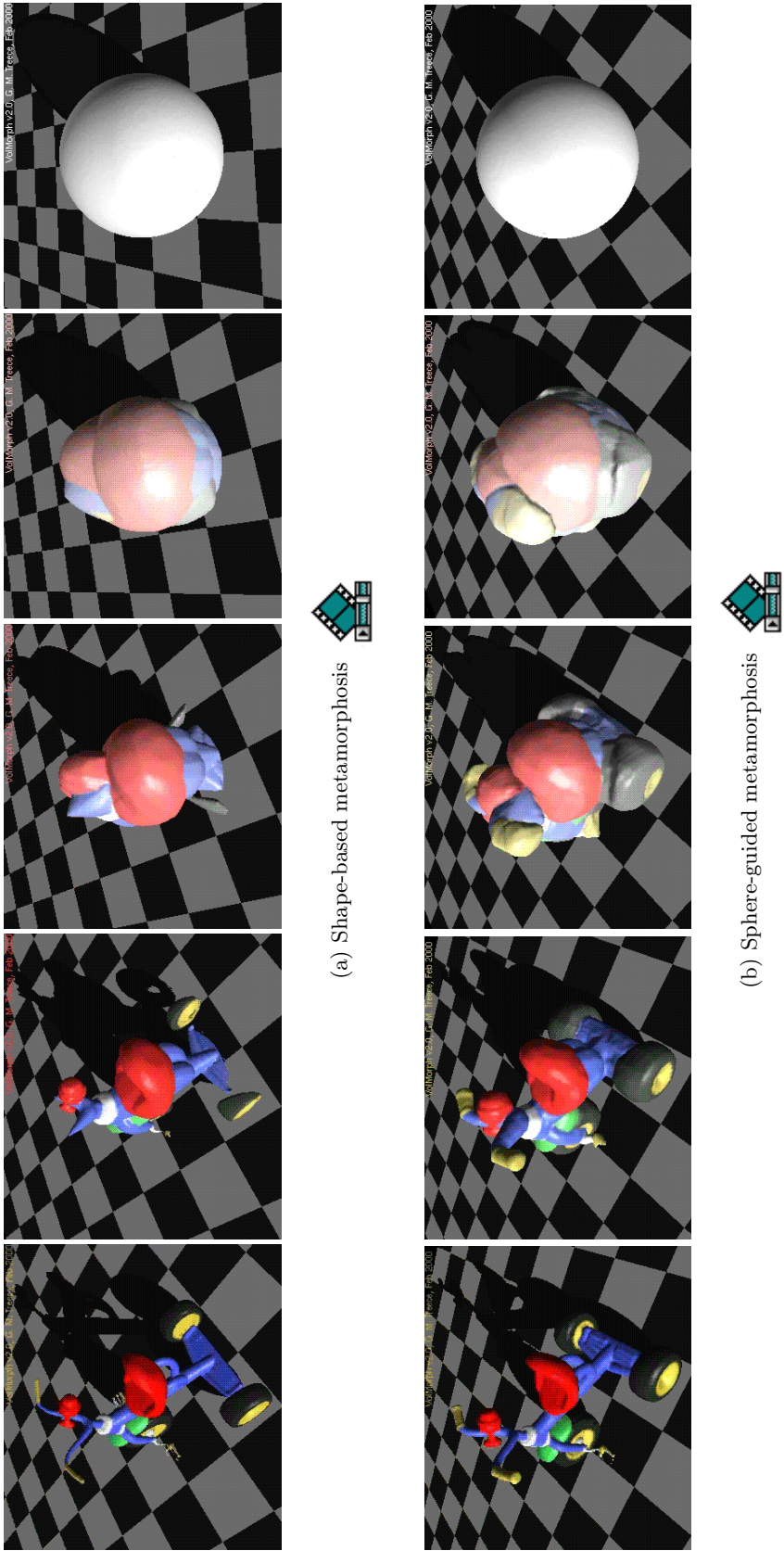


Figure 5.9: **Tricycle to sphere metamorphosis.** The objects and base rotate during the sequence; the light is fixed relative to the viewpoint.

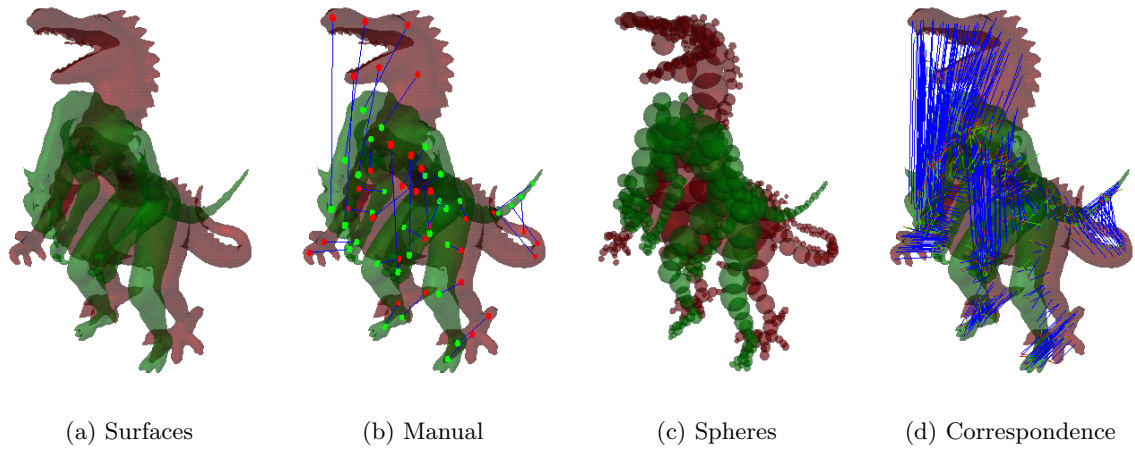


Figure 5.10: **Dragon to creature metamorphosis definition.** (b) shows the manual correspondences defined between the surfaces in (a). (c) shows the sphere representation, and (d) the correspondences calculated from this representation, combined with those in (b).

downwards through the base. Either of these morphs could be the most appropriate in different circumstances.

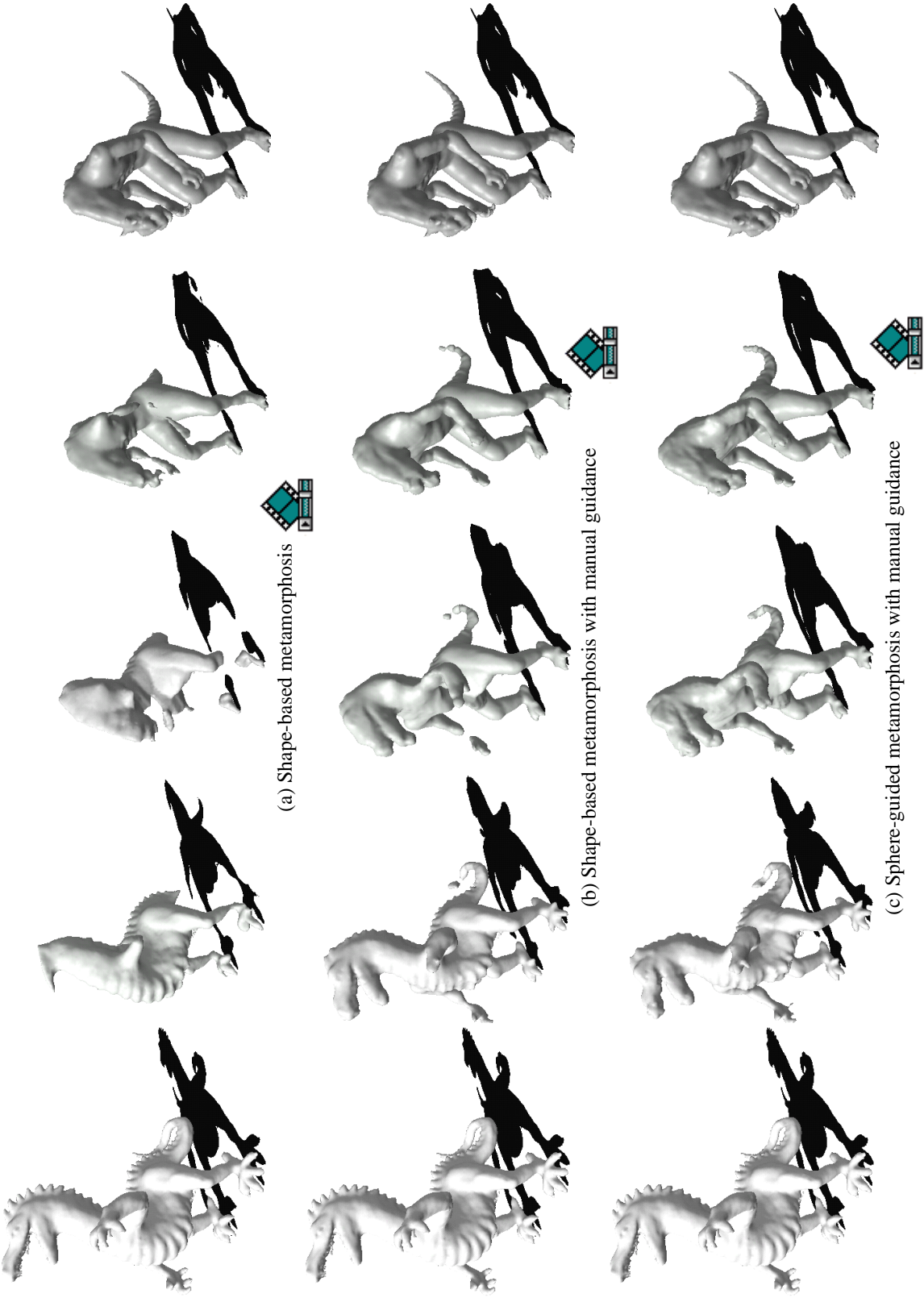


Figure 5.11: Dragon to creature metamorphosis. The manual and automatic guidance is shown in Figure 5.10.



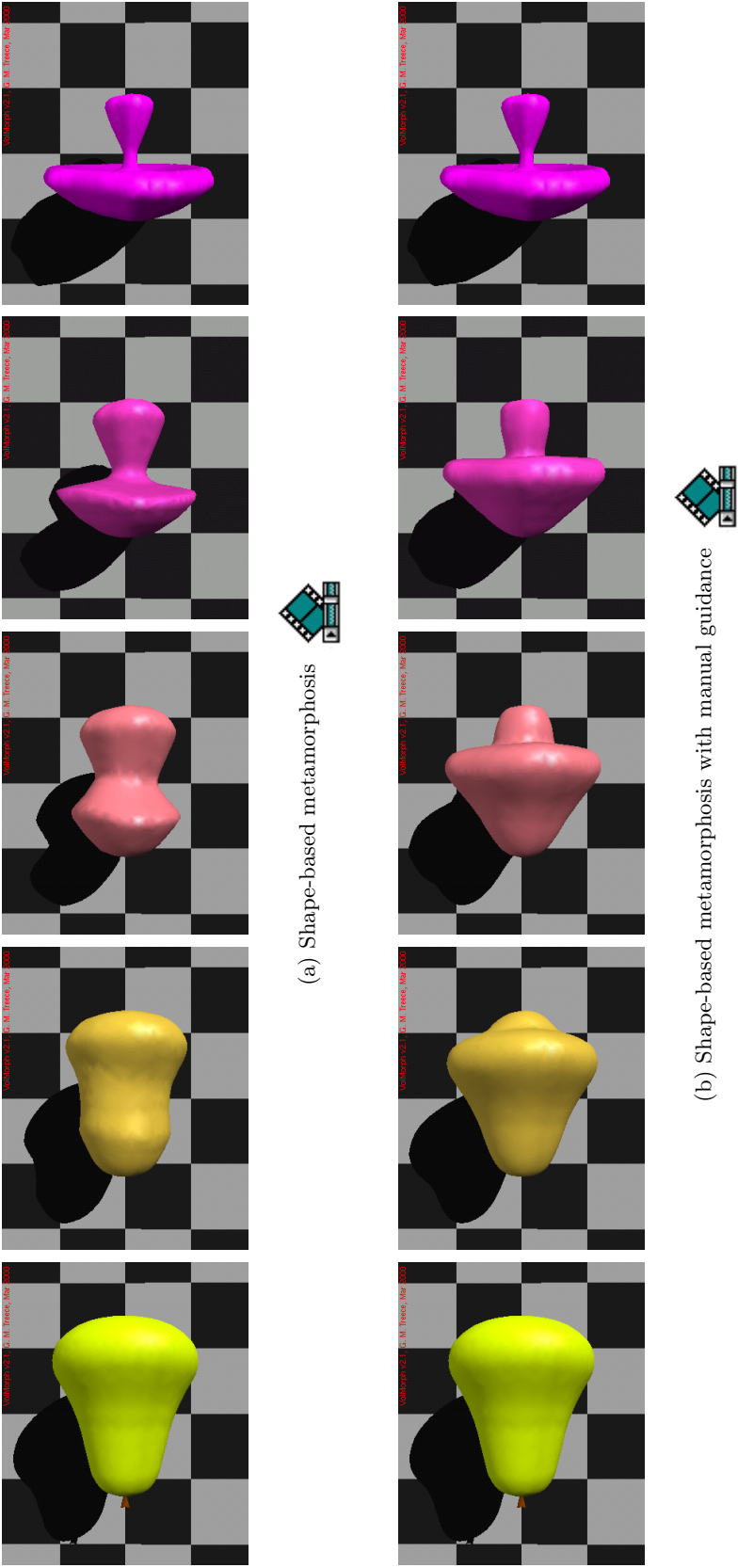


Figure 5.12: Pear to mushroom metamorphosis. The manual guidance for (b) is given in Figure 5.7(a).

## Chapter 6

# Summary and conclusions

### 6.1 3D ultrasound

The motivation for this thesis, as outlined in Chapter 1, is

... not to provide new types of information from 3D ultrasound data, but to improve the *integrity* of such information, the *ease* with which it can be produced, and to expand the *areas* to which it can be applied.

The two types of information addressed in this thesis are volume measurement and surface visualisation. How do the new algorithms developed for each of the above further these aims, when applied to 3D ultrasound data?

#### 6.1.1 Volume measurement

*Cubic planimetry*, presented in Chapter 2, is a technique for measuring volume from a small number of cross-sections, segmented from a set of sequential, freehand, 3D ultrasound B-scans. It can also be applied to multiple sweeps, as shown in Chapter 4.

#### Integrity

Any method for calculating volume based on cross-sections of an object, is clearly more faithful to the measured data than one based on prior assumptions about the shape of the object. In this respect, planimetry is better than both the conventional ellipsoid-based measures in current clinical use, and any method which involves fitting prior models to the data. Simple planimetric volume measures require no such assumptions. Cubic planimetry does make an assumption, that the cross-sectional area varies smoothly throughout the object, in order to calculate volume accurately from fewer cross-sections. Although this assumption is not always correct (the cross-sectional area may not always vary smoothly even for smooth surfaces), it is much less restrictive than the assumption of overall shape involved in non-planimetric methods.

The ability to measure volume from non-parallel cross-sections allows the original measured data to be segmented, rather than an interpolated array. This is of particular importance for multiple-sweep data. In this case, parallel cross-sections could only be generated from data that is both warped (so that it can be registered) and interpolated. Both of these operations reduce the integrity of the data; neither are necessary with cubic planimetry. The multiple-sweep framework is also robust to the registration errors which make interpolation difficult; in



fact organ movement *along* the dividing planes, caused by probe pressure, has no effect on the volume calculation.

The cubic planimetry algorithm itself (i.e. not including registration or segmentation errors) is inherently accurate, demonstrated in Sections 2.3 and 4.3, by simulation across several very different objects and sweep patterns. Accuracies better than  $\pm 1\%$  were achieved, using less than 10 cross-sections in most cases, or for multiple sweeps  $\pm 2\%$ , with 7 or 8 cross-sections per sweep. *In vivo* precision was assessed on examinations of a human kidney (Section 2.4.1), liver (Section 4.4.1) and a foetus at 16 weeks (Section 2.4.3) and 28 weeks (Section 4.4.2). Precision varied from  $\pm 5\%$  to  $\pm 7\%$  in these experiments. A similar *in vivo* accuracy, of  $\pm 7\%$ , was demonstrated by the bladder experiment of Section 2.4.2.

### Ease of use

Cubic planimetry is harder to use than conventional ellipsoid formulae, however it is easier to use than other planimetric 3D ultrasound methods. The reason for this is twofold. Firstly, as already mentioned, the difficulty of segmentation (either by manual or automatic means) is reduced, by the use of the original, rather than interpolated, data. Secondly, the time required to measure the volume is less than with simple planimetry.

This time is reduced in two ways. Since the method is set in the sequential framework, there is no need to generate a voxel array, and thus segmentation can be started as soon as the data has been acquired. In addition, fewer cross-sections are required for an accurate volume measurement, as demonstrated by the experiments on simulated objects in Sections 2.3 and 4.3. The volume calculation itself takes less than one second, and can be updated automatically, as the cross-sections are drawn. If 30 seconds is allowed for segmenting each of 10 cross-sections, then the whole process from scanning to volume measurement can be performed in approximately 5 minutes.

### Areas of application

The areas of anatomy, for which 3D ultrasound can be used to give an accurate measure of volume, have been extended in two ways. Firstly, the use of freehand 3D ultrasound, rather than any of the alternative techniques that acquire fixed volumes of data, allows the 3D system to be used in all of the areas in which a conventional 2D system could be used. Secondly, the use of sequential processing removes the requirement to generate a voxel array. This allows scanning patterns that would result in very poor quality voxel data to be successfully used for volume measurement. It also enables the multiple-sweep framework in Chapter 4 to be used, thus allowing volume measurements from either very large, or very awkward to scan areas. The only area to which the volume measurement algorithm can not be applied is rotational scanning, e.g. of the eye.

In addition, the lack of assumption about the shape of the object under examination makes the method equally viable for objects that have the expected shape, as for those that do not. The latter case is often the more clinically important.

#### 6.1.2 Surface visualisation

*Disc-guided interpolation* and *regularised marching tetrahedra* (RMT), presented in Chapter 3, are techniques for interpolating and visualising a surface from a set of cross-sections. Disc-guided interpolation, as with cubic planimetry, is designed for a small number of cross-sections

segmented from a set of sequential, freehand, 3D ultrasound B-scans. Hence, many of its advantages are similar to those outlined in the previous section. Both of these techniques can also be extended to multiple-sweep data, as shown in Chapter 4.

### Integrity

The surface generated by disc-guided interpolation is derived from the cross-sections, rather than directly from the data. In this respect, its integrity to the data is dependent on the accuracy of segmentation. However, it *is* guaranteed to pass through the cross-sections, and is hence faithful to these. This attribute is shared with all surface estimators which interpolate, rather than fit, a surface. As far as I am aware, this is the first time that a surface interpolator has been developed for data from multiple freehand sweeps.

RMT is also faithful to the interpolated data when generating a triangulated surface for display. The volume and surface error differences, compared to simple marching tetrahedra, are much less than the error inherent in the sampling resolution, as shown in Section 3.6. However, interpolated shading of the surface is greatly improved compared to marching cubes or marching tetrahedra. In particular, features due to the shape of the triangles, rather than the underlying data, are much less apparent with RMT (see Figure 3.19). In addition, unlike other vertex clustering methods, the surface topology in RMT is consistent with the sampled data.

### Ease of use

Disc-guided interpolation shares the same advantages as cubic planimetry of segmentation in the original B-scans, and calculation from a small number of cross-sections. In this case, the use of fewer cross-sections is achieved by introducing the concept of region correspondence into the interpolation. However, this does not introduce unnatural features, which can be generated by other variants of shape-based interpolation. This has been demonstrated on surfaces from simulated objects (Sections 2.3 and 4.3), and on real examples of hepatic ducts, a bladder and a foetus at week 22 (Section 3.5.2).

Both disc-guided interpolation and RMT are relatively fast algorithms, taking typically 10 seconds or less for these ultrasound examples. The reduction in the number of triangles for a given surface resolution with RMT, demonstrated in Section 3.6, also improves the ease with which the surface can be reviewed. RMT generates surfaces with 40% fewer triangles than marching cubes, and 70% fewer than marching tetrahedra, for the same sampling resolution. Hence, once the surface has been created, it can also be rendered faster. This greatly increases the potential for user interaction with the surface, which is one of the main benefits of 3D display.

### Areas of application

Disc-guided interpolation and RMT can be applied to all the situations in which cubic planimetry can be used. Hence, the areas of application are the same as those discussed earlier for volume measurement. This is a result of using a functional surface interpolation technique, which has no topological restrictions. Setting this in a sequential framework, so that registration problems are avoided, also allows both non-parallel scanning and multiple-sweep scanning. However, although a surface can be estimated from any such data, unlike cubic planimetry, the result will be degraded if the scans are overlapping, since this will also cause the surface to overlap itself.

## 6.2 Application to other fields

Although the surface interpolation and visualisation algorithms were designed for freehand 3D ultrasound, they have applications in other areas. In addition, disc-guided interpolation can easily be extended to sphere-guided interpolation for 3D volume morphing, as demonstrated in Chapter 5.

### 6.2.1 Surface interpolation and visualisation

The advantages of disc-guided interpolation over shape-based interpolation carry through to the case of parallel cross-sections, normally present in other medical imaging modalities, e.g. MRI or CT. Where segmentation has to be performed manually, the reduction in the number of cross-sections required for a good surface will be equally welcome for these modalities. Examples of CT and MRI surfaces are shown in Section 3.5.3, for a child's skull, female pelvis and liver. Equally, the quality of RMT as an isosurface visualisation tool is apparent in the display of a child's teeth, and thigh muscles, in Section 3.6.3. Here, RMT is used on thresholded data, and in this case the reduction in the number of triangles is particularly important — indeed it may make the difference between being able to manipulate surfaces in real time or not. For applications where the resolution can be reduced, the tetrahedral grid required by RMT can be formed from a sub-set of the original orthogonal grid. For applications that already require interpolation of the original data, there is no disadvantage in using a tetrahedral lattice for the interpolated data. Isosurface extraction at the highest resolution of the original data is only possible if additional points are interpolated from this data.

In addition, RMT can be used to display surfaces defined by functions, as shown in Section 3.6.3, by sampling these functions to an appropriate grid. This is much more flexible than the common technique of defining a rectangular (x-y) grid whose out-of-plane (z) values are generated from the function, since this is only possible if the function is singularly valued in z, and continuous in both x and y. None of these restrictions apply to the RMT method of surface visualisation.

RMT can also be used in conjunction with the scan-conversion technique outlined in Chapter 5, to re-triangulate closed, polygonal surfaces. In this case, the polygonal surface is first sampled to an appropriate resolution, then RMT is used to create a new triangulated surface. Repeating this at a variety of resolutions generates a set of triangulated surfaces with varying levels of detail, appropriate for viewing at different distances. This is particularly useful for complex virtual reality scenes.

### 6.2.2 3D volume morphing

Disc-guided interpolation allows a surface to be interpolated from a small number of cross-sections, since it can handle cross-sections with very different shapes. In the same manner, sphere-guided interpolation allows intermediate surfaces to be interpolated between two very different source and target surfaces. Sphere-guided interpolation automates the detail of the morph, in order to prevent morphs that are clearly incorrect, but leaves room for a natural level of user control. The speed and versatility of this method in handling complex shape changes enables the user to easily interact with the morphing process. Changing parameters has an immediate effect on the display of correspondence, and low resolution morphs can be calculated in minutes, to assess the eventual effect.

Much of the processing speed of the algorithm is achieved by high quality scan conversion of polygonal models to a volumetric representation, and equally high quality isosurfacing to recreate the polygonal models for display. This can reduce the size of the volume for a given surface quality by more than an order of magnitude. The automation of detailed correspondence allows a simple manual correspondence technique to be adopted, where vectors are defined by clicking and dragging.

## 6.3 Conclusions and further work

Volume measurement and surface visualisation techniques suitable for 3D ultrasound data have been presented.

Using sequential freehand data as the source for these methods widens their applicability, but also increases their complexity. In terms of scanning technique, it is easier to use an integral 3D probe, which automatically acquires an evenly sampled 3D volume, than a conventional probe. The freehand technique requires a steadiness of hand in order to sample the volume appropriately; however it also allows areas to be scanned which would not be possible with any other method. Similarly, using the sequential approach makes the processing of data harder than it would be if a voxel array was used; however it is the use of this approach which enables multiple-sweep scanning for larger, or more awkwardly shaped or located, organs.

Segmentation is the obvious residual problem which must be tackled if the task of volume measurement in 3D ultrasound is to be made any faster or easier. The approach adopted in this thesis is to use manual border tracing, but limit the number of cross-sections required, and ensure that segmentation could be performed in the original B-scans. It is unlikely that it will ever be possible to segment ultrasound data fully automatically — even if object boundaries were clearer, the problem of which object to segment would remain. However, semi-automatic techniques, where the user works together with the computer in the task, may be of more use. Whether this is the case is currently unclear; although semi-automatic techniques will in general increase the *precision* of the segmentation, it is hard to implement them so that they are faster to use than manual techniques.

There is also a trade-off between the number of cross-sections and the ease of segmentation. Many semi-automatic techniques rely on initialising a cross-section in one image, with a predefined cross-section on a neighbouring image. This is more successful if the images are similar, however this may lead to a higher density of cross-sections than is required by the algorithms presented in this thesis, hence also increasing the total segmentation time, even if that for each cross-section is reduced. It should also be noted that the ability to segment the original B-scans may well be crucial to the design of semi-automatic techniques, as well as making manual segmentation easier. Many of the artifacts which make segmentation hard are directional, and if information about the nature of these artifacts can be included in the segmentation approach, this is likely to be much more successful in the original B-scans, where that direction is well defined.

Another conscious decision, taken particularly in the design of the surface interpolation algorithm and multiple-sweep framework, was to be faithful to the original data, rather than trying to accommodate errors through smoothing. Both registration and segmentation errors are presented to the clinician, but in such a way so as not to detract from the remaining data, giving the clinician a clear indication of the quality of the measurements. This has the disadvantage that surfaces will look less realistic than those created by data fitting with a smoothness constraint,

and reslices through multiple-sweep data contain obvious mis-registration artifacts. However, the approach makes it easier to design the algorithms for robustness in all circumstances, without the danger of masking information which may have had clinical importance.

The surface created by disc-guided interpolation is currently constructed piecewise between two cross-sections at a time. Although there are good reasons for avoiding cubic interpolation of the distance field data from more than two cross-sections (outlined in Section 3.2.6), using more than two cross-sections to establish the region correspondence may improve the consistency of the surface. This could be achieved by altering the correspondence weighting for each disc, in some way, such that it included a term based on the difference in correspondence direction to each of the neighbouring planes.

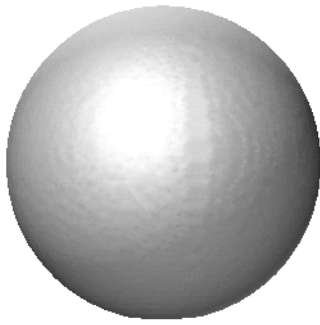
Finally, the choice of experimentation technique has affected the type of conclusions that can be drawn. Simulated experiments were chosen over *in vitro* measurements because although *in vitro* measurements are popular and can be very useful, they neither demonstrate what the *in vivo* system performance will be, nor allow enough control to assess the new algorithms themselves. In this case, the major errors due to segmentation and registration are very similar across all freehand 3D ultrasound systems. The crucial contribution of this thesis is in the design of the *post*-processing algorithms, and these are better compared without swamping the results with errors due to *acquisition*. Repeated *in vivo* observations were performed, and these give an indication of the precision of the whole system, but not accuracy. Had a CT or MRI scanner been available, a better assessment of *in vivo* accuracy would have been possible.

Any algorithms designed for non-parallel planes can also be applied to parallel planes, and this is particularly the case for the surface interpolation and visualisation algorithms presented in this thesis. Disc-guided interpolation is of use in limiting the number of parallel cross-sections required in other medical imaging modalities. The quality and small number of triangles generated by regularised marching tetrahedra is equally useful for any surface which is derived from, or can be represented by, sampled data. In addition, the same framework used in the interpolation of cross-sections can also be used to interpolate 3D volume data, leading to a robust, topologically independent algorithm that is also appropriate for morphing between surfaces.

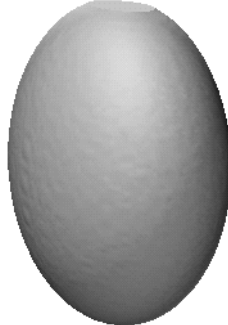
# Appendix A

## Test setup

### A.1 Generating simulated scans



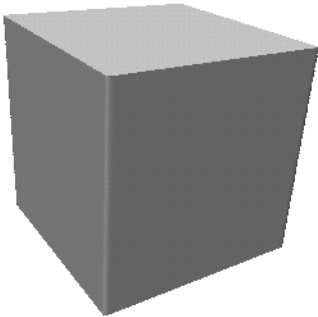
(a) Sphere



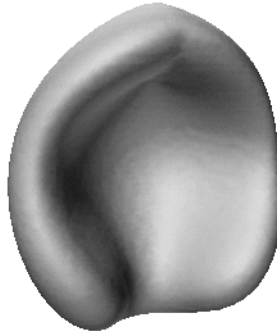
(b) Ellipsoid



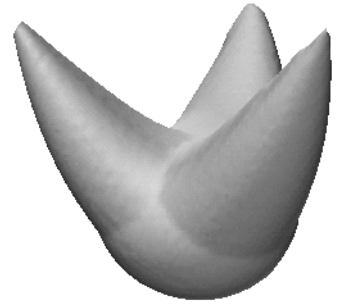
(c) Cone



(d) Cube



(e) 'Baseball glove'



(f) 'Jester's hat'

Figure A.1: **Test objects for simulated scans.**

In order to validate the volume measurement and surface visualisation algorithms, software was written to 'scan' the mathematically defined objects in Figure A.1. This software generated sequences of 'B-scans' of any of these objects, where the image pixels were set to 255 if inside the object, and zero otherwise. The data was saved in the same format as the actual ultrasound B-scans, to allow processing of these simulated scans to be performed with the ultrasound software

described in Appendix A.2. Simulated B-scans were precisely segmented by thresholding at any value between zero and 255.

Objects were chosen to exhibit specific characteristics. The sphere is symmetric and smooth, whereas the ellipsoid is more representative of a real organ. Both the cone and cube have sharp edges, and the variation of the (horizontal) cross-sectional area of the cone is faster than linear. The ‘baseball glove’ object is more complex, and contains both convex and concave regions. The ‘jester’s hat’ object is representative of a branching structure. These two objects were formed by combining warped versions of more simple objects: an ellipsoid for the former, and three cones and a sphere for the latter. The volume of all of the objects was known precisely.

The scanning pattern could be controlled in position, azimuth, elevation and roll. This was achieved by defining the position and orientation of the first and last planes, then interpolating any number of additional planes between them. A random jitter could also be added to the positions and orientations, in order to represent a real sequence acquired with a slightly unsteady hand. In addition, random errors could be introduced to the B-scan locations, in order to represent possible registration errors.

Multiple-sweep simulation was achieved by concatenating the data from several simulated scans of the same object.

## A.2 Acquisition and processing of *in vivo* data

*In vivo* ultrasound data was recorded using a Toshiba Powervision 7000<sup>1</sup> with a 3.75MHz convex curvilinear array probe. A Polhemus FASTRAK<sup>2</sup> magnetic field position sensor was mounted on this probe. The position signal from this, in addition to the video output of the ultrasound machine, were then fed to a Silicon Graphics Indy workstation<sup>3</sup> with an image acquisition card. The position sensor was calibrated, using the technique described in [146], giving positional accuracy over the acquisition volume of typically  $\pm 1\text{mm}$  in all directions.

Acquisition of the ultrasound images and position readings, calibration of the system, and segmentation of the data were all performed using Stradx<sup>4</sup> software [145]. Stradx is a sequential freehand 3D ultrasound acquisition and visualisation tool, which was written in conjunction with Richard Prager and Andrew Gee, at the University of Cambridge Department of Engineering. The volume measurement algorithm of Chapter 2, the surface visualisation algorithms of Chapter 3, and the multiple-sweep framework of Chapter 4, have all been implemented in the latest release, version 6.0, which is freely available on the internet, and included in the CDROM described in Appendix E.

An example of the Stradx interface is shown in Figure A.2. The steps in using this software for volume measurement and surface visualisation are:

**Calibration** Each time the position sensor is mounted on the ultrasound probe, the system is calibrated, by scanning a carefully designed phantom in a water bath [146]. The calibration parameters are calculated automatically from the acquired B-scans, and the whole process takes approximately fifteen minutes.

**Acquisition** The clinician first determines the pattern in which the probe will be moved. Having done so, the sequence is recorded by using the **record** and **pause** buttons in the main

<sup>1</sup>Toshiba America Medical Systems, Tustin, California

<sup>2</sup>Polhemus Incorporated, Colchester, Vermont

<sup>3</sup>Silicon Graphics Incorporated, Mountain View, California

<sup>4</sup><http://svr-www.eng.cam.ac.uk/~rwp/stradx/>



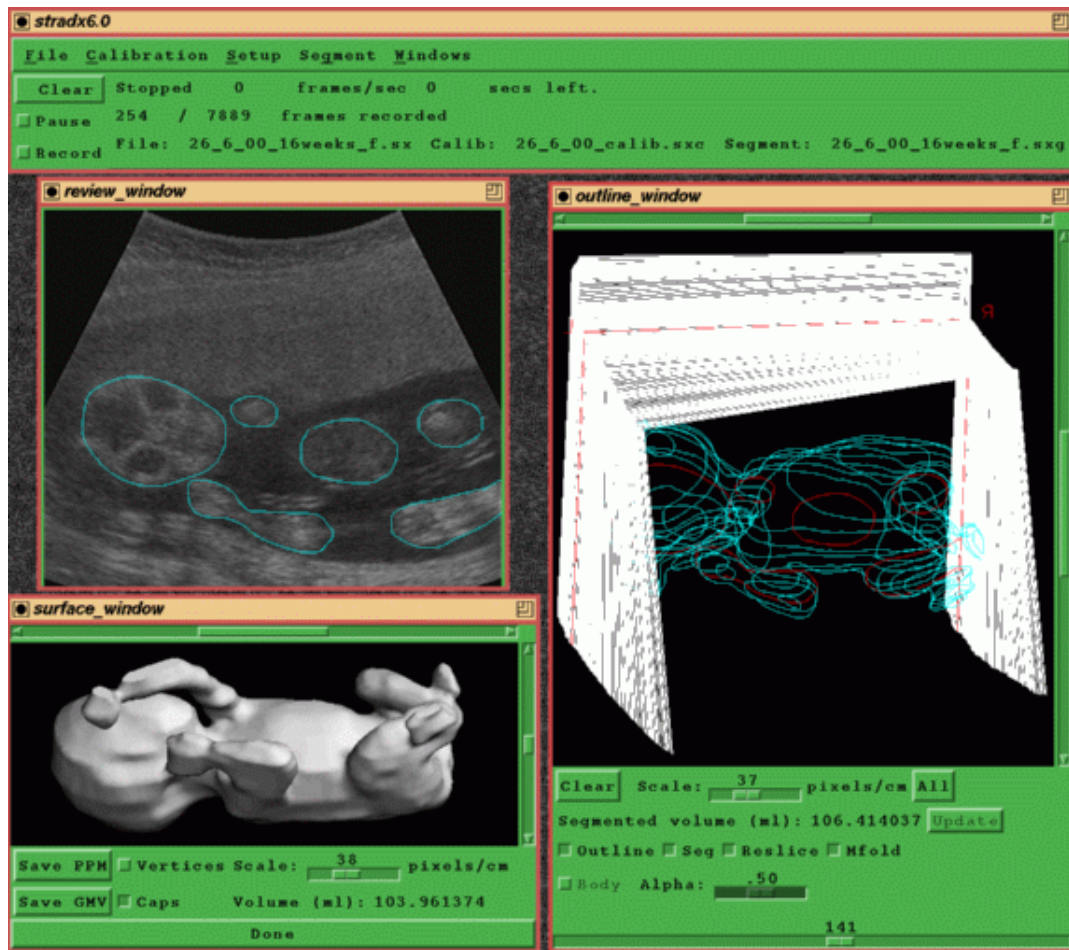


Figure A.2: Stradx v6.0 interface

window at the top of Figure A.2. It is important that the patient is as still as possible, and that the clinician moves the probe with a steady motion. Each of the data sequences in this thesis could be recorded within a single breath hold.

**Review** Once the data has been recorded, B-scans can be reviewed in the **review\_window** of Figure A.2, and their relative positions examined in the **outline\_window**. If much movement is evident from the data, or the sweep did not cover the required volume, the data is re-acquired at this stage.

**Segmentation** This is performed by the clinician on a subset of the recorded B-scans, in the **review\_window** of Figure A.2. The cross-sections are displayed in 3D wire-frame format as they are drawn, in the **outline\_window**. This provides feedback on both the shape and the spacing of the cross-sections, allowing the clinician to concentrate the segmentations on areas with the most complex shape.

**Volume measurement** As the cross-sections are completed, a real-time volume estimate is calculated using *cubic planimetry*. This is shown in the **outline\_window** of Figure A.2.

**Surface visualisation** A surface can be visualised from the current set of cross-sections by opening the **surface\_window** of Figure A.2. The time taken to display the surface is approximately ten seconds, dependent on the complexity of the data and the number of

cross-sections. A secondary estimate of the volume is then calculated from this surface, and displayed in the same window.

**Refinement** The surface visualisation often reveals unexpected features, which, when the data is reviewed, are discovered to be the result of incorrect segmentation. In this case, the segmentation can be edited, and the volume measurement and surface recalculated from the new cross-sections.

These steps are described in more detail, together with the other visualisation algorithms implemented in Stradx, in HTML documentation. This documentation is available with the software on the internet, and on the CDROM described in Appendix [E](#).

## Appendix B

# Algorithms used in volume measurement

### B.1 Area from parametric cubic splines

Given two curves, defined parametrically:

$$\begin{bmatrix} x_i(t) & y_i(t) \end{bmatrix} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} x_{i3} & y_{i3} \\ x_{i2} & y_{i2} \\ x_{i1} & y_{i1} \\ x_{i0} & y_{i0} \end{bmatrix} \quad \text{where } 0 \leq t \leq 1$$

If each curve is connected to the other by two straight lines joining the start points,  $t = 0$ , and the end points,  $t = 1$ , the enclosed area  $A$  (shown in Figure B.1) can be calculated from [182]:

$$A = \left| \int_{t=0}^1 \vec{s} \cdot d\vec{\omega} \right| \quad (\text{B.1})$$

where  $\vec{s}$  and  $\vec{\omega}$  are defined in Figure B.1, and each can be calculated as follows:

$$\begin{aligned} \vec{s}(t) &= \begin{bmatrix} y_1(t) - y_2(t) & x_2(t) - x_1(t) \end{bmatrix} \\ &= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} y_{13} - y_{23} & x_{23} - x_{13} \\ y_{12} - y_{22} & x_{22} - x_{12} \\ y_{11} - y_{21} & x_{21} - x_{11} \\ y_{10} - y_{20} & x_{20} - x_{10} \end{bmatrix} \end{aligned}$$

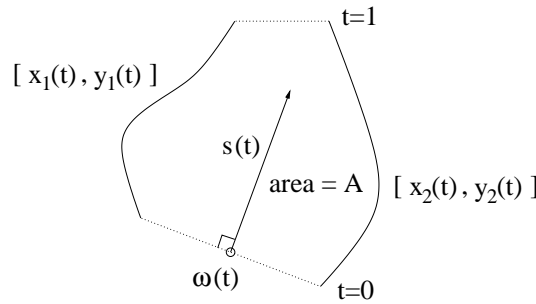


Figure B.1: **Swept area between cubic splines.**  $A$  is the swept area between parametric cubic splines  $[x_1(t), y_1(t)]$  and  $[x_2(t), y_2(t)]$ .  $\vec{s}(t)$  is a vector normal to the line joining the curves at the same value of  $t$ , and  $\vec{\omega}(t)$  is the position vector of the centre of that line.

$$\equiv \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} x_{s3} & y_{s3} \\ x_{s2} & y_{s2} \\ x_{s1} & y_{s1} \\ x_{s0} & y_{s0} \end{bmatrix} \quad (\text{B.2})$$

$$\begin{aligned} d\vec{\omega}(t) &= \frac{1}{2} \begin{bmatrix} dx_1(t) + dx_2(t) & dy_1(t) + dy_2(t) \end{bmatrix} \\ &= \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 3(x_{13} + x_{23}) & 3(y_{13} + y_{23}) \\ 2(x_{12} + x_{22}) & 2(y_{12} + y_{22}) \\ x_{11} + x_{21} & y_{11} + y_{21} \end{bmatrix} dt \\ &\equiv \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \begin{bmatrix} x_{\omega 2} & y_{\omega 2} \\ x_{\omega 1} & y_{\omega 1} \\ x_{\omega 0} & y_{\omega 0} \end{bmatrix} dt \end{aligned} \quad (\text{B.3})$$

Hence the area  $A$  is:

$$\begin{aligned} A &= \left| \int_{t=0}^1 c_5 t^5 + c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t + c_0 dt \right| \\ &= \left| \frac{c_5}{6} + \frac{c_4}{5} + \frac{c_3}{4} + \frac{c_2}{3} + \frac{c_1}{2} + c_0 \right| \end{aligned} \quad (\text{B.4})$$

where the coefficients  $c_5, \dots, c_0$  can be derived from the coefficients of equations (B.3) and (B.2), as follows:

$$\begin{aligned} c_5 &= x_{s3}x_{\omega 2} + y_{s3}y_{\omega 2} \\ c_4 &= x_{s3}x_{\omega 1} + x_{s2}x_{\omega 2} + y_{s3}y_{\omega 1} + y_{s2}y_{\omega 2} \\ c_3 &= x_{s3}x_{\omega 0} + x_{s2}x_{\omega 1} + x_{s1}x_{\omega 2} + y_{s3}y_{\omega 0} + y_{s2}y_{\omega 1} + y_{s1}y_{\omega 2} \\ c_2 &= x_{s2}x_{\omega 0} + x_{s1}x_{\omega 1} + x_{s0}x_{\omega 2} + y_{s2}y_{\omega 0} + y_{s1}y_{\omega 1} + y_{s0}y_{\omega 2} \\ c_1 &= x_{s1}x_{\omega 0} + x_{s0}x_{\omega 1} + y_{s1}y_{\omega 0} + y_{s0}y_{\omega 1} \\ c_0 &= x_{s0}x_{\omega 0} + y_{s0}y_{\omega 0} \end{aligned} \quad (\text{B.5})$$

## B.2 Volume from geometric surface

The volume of a closed, triangulated surface can be calculated by summing the projected volumes of each triangle, shown in Figure B.2, across the whole surface. This is a variant of Gauss' theorem [90]. The area of a triangle, with vertices defined by the position vectors  $\vec{a}$ ,  $\vec{b}$  and  $\vec{c}$ , is given by:

$$\text{area} = \frac{1}{2} (\vec{b} - \vec{a}) \times (\vec{c} - \vec{a}) \quad (\text{B.6})$$

Hence the projected area  $A$  in the direction of projection  $\hat{n}$  is:

$$A = \frac{1}{2} ((\vec{b} - \vec{a}) \times (\vec{c} - \vec{a})) \cdot \hat{n} \quad (\text{B.7})$$

The distance from the centre of the triangle to its projection,  $l$ , is:

$$l = \frac{1}{3} (\vec{a} + \vec{b} + \vec{c}) \cdot \hat{n} \quad (\text{B.8})$$

Hence the enclosed projected volume is:

$$V = Al = \frac{1}{6} [((\vec{b} - \vec{a}) \times (\vec{c} - \vec{a})) \cdot \hat{n}] [(\vec{a} + \vec{b} + \vec{c}) \cdot \hat{n}] \quad (\text{B.9})$$

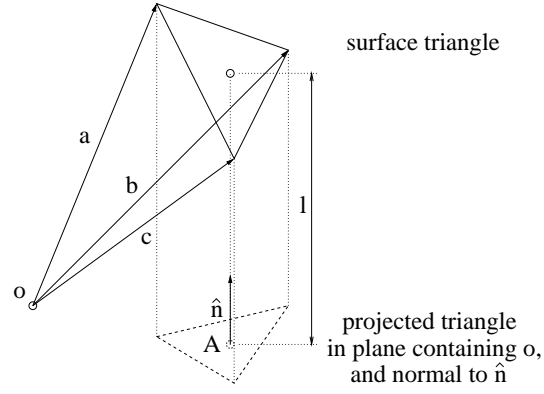


Figure B.2: **Volume of a triangulated surface.** The volume of any closed surface can be calculated from the sum of the enclosed volumes of each projected triangle, as shown above. The choice of projection direction  $\hat{n}$  is arbitrary, as is the location of the origin  $o$ , provided these are the same for all triangles in the surface.

The summation of this volume across all the triangles in the surface gives the total enclosed volume of the surface.

If the surface is clipped by two or fewer planes, the volume enclosed by the surface *within* the planes can also be calculated using equation (B.9). In this case, the projection direction  $\hat{n}$  must be parallel to each of these planes:

$$\hat{n} = \begin{cases} \{0, 0, 1\} & \text{no planes} \\ \hat{n}_1 \times (n_{1z}, n_{1x}, n_{1y}) & \text{one plane, normal to } \hat{n}_1 \\ \hat{n}_1 \times \hat{n}_2 & \text{two planes, normal to } \hat{n}_1 \text{ and } \hat{n}_2 \end{cases} \quad (\text{B.10})$$

# Appendix C

## Geometric algorithms

Some of the methods described in this thesis involve algorithms which are particularly awkward to implement correctly and efficiently in software. Where this is the case, pseudo-C-code, together with a brief outline of the algorithm, have been included in this Appendix.

### C.1 Discrete representation of a cross-section

Manual tracing of the object cross-sections in a B-scan results in a list of connected points defining the object border. In order to use disc-guided interpolation to reconstruct a surface from these cross-sections, a binary image is required, where each pixel is set to ‘1’ only if it is inside the border. The conversion to this representation is similar to the ‘point in polygon’ problem described for instance by O’Rourke [139], except this must be evaluated for *all* the pixels in a binary image containing the cross-section.

This conversion can be efficiently achieved by first discretising the object border, and then ‘filling in’ the inside of the border in a raster scan of the image. During the raster scan, the discretised border image is replaced with an inside-outside ‘state’; this state is initialised as ‘outside’, then toggled each time a border pixel is encountered. Hence the edge of the image is always set to ‘outside’ (i.e. a value of zero), and pixels are set to ‘inside’ (i.e. a value of one) if they are enclosed by an *odd* number of boundaries. Borders within borders, for instance two concentric circles, thus generate a hole within the object.

The initial discretisation of the object border must be calculated with care, to ensure that the final raster scan generates the desired result. As a border crosses each row of the image, only one pixel must be marked in this row, even if the border passes through several pixels; otherwise the inside-outside state will be incorrectly toggled more than once during the scan of this row. However, all the pixels through which the border passes are a part of the object. A similar dilemma arises for borders which are on top of each other — in this case the inside-outside state must not be toggled (the two coincident borders represent a very thin object), but the border pixels *are* a part of the object, as before.

This problem can be overcome by incrementing the pixel at the point where the border crosses an image row, by one; whereas all other pixels through which the border passes are incremented by two. An example of such a calculation is shown in Figure C.1. As the border is discretised, these numbers are accumulated at each pixel. Points at which the border direction changes from up to down, or *vice versa*, also need an additional increment, since they effectively represent two border edges. In the final raster scan of the image, only odd numbers cause the inside-outside state to be toggled, but all non-zero pixels are set to ‘1’ in the final image.

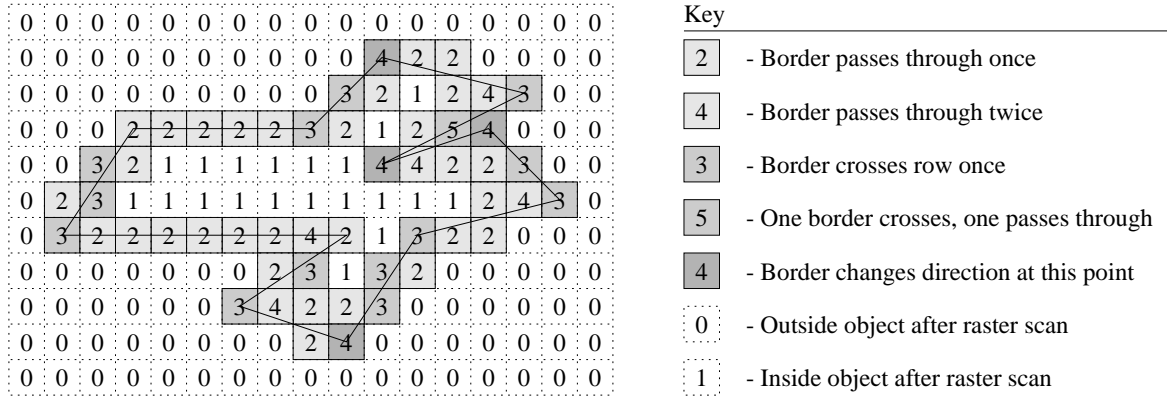


Figure C.1: **Conversion of an object border to a binary image.** Shaded squares are the non-zero pixels after discretisation of the object border. Only odd values cause the inside-outside state to be toggled during the raster scan. After this scan, all the border pixels, and those marked with a '1' above, are 'inside' the object.

This method works reliably, irrespective of the direction (clockwise or anti-clockwise) in which the boundary is drawn, the complexity of the shape, the number of self intersections, or the number of sub-objects (i.e. objects enclosed within holes within other objects). If there is more than one object, all the object borders are discretised, before the final raster scan.

```

/* Convert the closed contour defined by 'vertex[vertices]' to a discrete */
/* inside-outside representation in 'image[height][width]', which is initially blank */
prev_x = vertex[vertices-1].x;          /* Find the previous vertex - in a closed contour, */
prev_y = vertex[vertices-1].y;          /* the last vertex is connected to the first */
v = 2;                                  /* Work out if the y-value was */
while ( prev_y == vertex[vertices-v].y ) v++;                                /* increasing or */
if ( prev_y > vertex[vertices-v].y ) up = TRUE;                               /* decreasing */
else up = FALSE;
for (v=0; v<vertices; v++) {          /* Loop through all vertices */
    new_x = vertex[v].x;                /* Get the new vertex */
    new_y = vertex[v].y;
    if ( new_y == prev_y ) y_inc = 0;   /* Work out which way to increment y */
    else if ( new_y > prev_y ) y_inc = 1;
    else y_inc = -1;
    if ( new_x == prev_x ) x_inc = 0;    /* ... and x */
    else if ( new_x > prev_x ) x_inc = 1;
    else x_inc = -1;
    if ( y_inc == 0 ) {                 /* If this is a horizontal line */
        for (x=prev_x; x≠new_x; x+=x_inc) {
            image[new_y][x] += 2;        /* just mark pixels which it passes through */
        }
    } else {                            /* This is not a horizontal line */
        if ( (up == 1)^(y_inc == 1) ) { /* If this is a turning point */
            image[prev_y][prev_x] += 1; /* additional increment for this point */
            up = !up;                    /* ... and reverse direction */
        }
    }
}

```



```

    }
    f = (new_x - prev_x)/(new_y - prev_y);           /* Calculate gradient of border */
    next_x = prev_x;
    for (y=prev_y; y≠new_y; y+=y_inc) {              /* Loop through all y values, */
        this_x = next_x;                             /* calculating which x values the line passes through */
        mid_x = prev_x + (y-prev_y+y_inc/2.0)*f;
        next_x = prev_x + (y-prev_y+y_inc)*f;
        image[y][this_x] += 1;                       /* Mark the point at which border crosses this y value */
        for (x=this_x; x≠next_x; x+=x_inc) {
            if ( (x*x_inc) ≤ (mid_x*x_inc) ) image[y][x] += 2; /* And also mark where */
            if ( (x*x_inc) ≥ (mid_x*x_inc) ) image[y][x] += 2; /* it passes through */
        }
        if ( mid_x == next_x ) image[y][next_x] += 2;
    }
}
prev_y = new_y;                                     /* Update the previous contour vertex */
prev_x = new_x;                                     /* for the next time */
}
inside = 0;                                         /* Having discretised the contour, now fill in the middle */
for (y=0; y<height; y++) {                         /* Scan image, row by row */
    for (x=0; x<width; x++) {
        inside = inside ∧ (image[y][x]%2);          /* If a crossing point, change 'inside' */
        if ( image[y][x] ) {                         /* make sure that points on the contour */
            image[y][x] = 1;                         /* are always included */
        } else {
            image[y][x] = inside;                    /* Otherwise, use current value of 'inside' */
        }
    }
}
}
}

```

## C.2 Topological checks for clustering

The topological checks described in Section 3.4.3, to determine whether a set of surface intersections can be clustered to form one or more vertices, can all be performed using logical operations on integers. Each of the edges of the tetrahedra are assigned a bit number equivalent to the label, as shown in Figure 3.24. The topological cases A to E are described in Figure 3.25. Gathering surface intersections from a single surface can be done using the hexadecimal masks in Table C.1. Finding holes (i.e. areas with no surface intersections, surrounded by a connected surface) can be done in the same manner. Additional masks in Table C.2 are needed to check for the ‘flat surface’ topological case.

```

/* Topological check to see if intersections around a sample point can be clustered */
/* 'intersections' is a bit field containing the surface intersections */
/* 'distance[p]' gives the sampled distance field at the point 'p' */
if ( intersections == 0x0000 ) return;              /* Quick exit if no intersections */
surfaces = 0;                                       /* Initialise number of surfaces, and */
for (s=0; s<6; s++) surface[s] = 0;               /* intersections for each surface */

```

Table C.1: **Nearest edge masks.** If the edges  $0 \dots 13$  in Figure 3.24 are assigned bit positions  $0 \dots 13$ , then the masks below give the nearest edges. If the current edge is 2 units, there are four neighbours, else if  $\sqrt{3}$  units there are six neighbours.

Edge	0	1	2	3	4	5	6
No. Neighbours	6	4	6	4	6	4	6
Nearest edges	0x321A	0x2015	0x24B2	0x0251	0x006F	0x00D4	0x03B8
Edge	7	8	9	10	11	12	13
No. Neighbours	6	4	6	4	6	4	6
Nearest edges	0x0D64	0x0AC0	0x1949	0x2884	0x3780	0x2A01	0x1C07

Table C.2: **‘Flat surface’ topology masks.** The masks give the vertices of all the *outer* edges, i.e. those marked in grey in Figure 3.24. The first check for a ‘flat surface’ is that there are intersections on both of the adjacent edges, and on neither of the opposite edges.

Edge	0-1	0-3	0-4	0-9	0-12	0-13	1-2	1-4
Adjacent edges	0x0003	0x0009	0x0011	0x0201	0x1001	0x2001	0x0006	0x0012
Opposite edges	0x2010	0x0210	0x000A	0x1008	0x2200	0x1002	0x2010	0x0005
Edge	1-13	2-4	2-5	2-7	2-10	2-13	3-4	3-6
Adjacent edges	0x2002	0x0014	0x0024	0x0084	0x0404	0x2004	0x0018	0x0048
Opposite edges	0x0005	0x0022	0x0090	0x0420	0x2080	0x0402	0x0041	0x0210
Edge	3-9	4-5	4-6	5-6	5-7	6-7	6-8	6-9
Adjacent edges	0x0208	0x0030	0x0050	0x0060	0x00A0	0x00C0	0x0140	0x0240
Opposite edges	0x0041	0x0044	0x0028	0x0090	0x0044	0x0120	0x0280	0x0108
Edge	7-8	7-10	7-11	8-9	8-11	9-11	9-12	10-11
Adjacent edges	0x0180	0x0480	0x0880	0x0300	0x0900	0x0A00	0x1200	0x0C00
Opposite edges	0x0840	0x0804	0x0500	0x0840	0x0280	0x1100	0x0801	0x2080
Edge	10-13	11-12	11-13	12-13				
Adjacent edges	0x2400	0x1800	0x2800	0x3000				
Opposite edges	0x0804	0x2200	0x1400	0x0801				

```

cluster = TRUE;                                     /* and assume we will be able to cluster */
if ( intersections == 0x3FFF ) cluster = FALSE;      /* Topological case A */
else {
    all_edges = intersections & 0x3FFF;              /* Search for surface holes */
    done_edges = 0x3FFF;                             /* by looping through edges */
    todo_edges = 0x0001;
    while ( !(all_edges & todo_edges) ) todo_edges <<= 1;
    while ( todo_edges ) {
        i = 0;
        while ( !((1 << i) & todo_edges) ) i++;      /* Find first edge in list */
        done_edges ∨ = (1 << i);                     /* don't look at this edge again */
        todo_edges |= (all_edges & nearest_edges[i]); /* add near edges to list */
        todo_edges &= done_edges;                     /* but not if already done them */
    }
    /* If there are still edges to check, there is more than one hole */
    if ( all_edges & done_edges ) cluster = FALSE;    /* Topological case B */
    if ( cluster ) {
        all_edges = intersections;                   /* Search for surfaces in a similar */
        while ( all_edges ) {                         /* manner as for surface holes */
            done_edges = 0x3FFF;
            todo_edges = 0x0001;
            while ( !(all_edges & todo_edges) ) todo_edges <<= 1;
            while ( todo_edges ) {
                i = 0;
                while ( !((1 << i) & todo_edges) ) i++; /* Find first edge in list */
                done_edges ∨ = (1 << i);               /* don't look at this edge again */
                todo_edges |= (all_edges & nearest_edges[i]); /* add near edges to list */
                todo_edges &= done_edges;               /* but not if already done them */
            }
            all_edges &= done_edges;
            surface[surfaces++] = done_edges & 0x3FFF; /* Record surface intersections */
        }
        if ( surfaces == 1 ) {                         /* Check for flat surfaces, if only one surface */
            for ( i=0; i<36; i++) {                     /* For each edge */
                if ( !(adjacent_edge[i] & ~done_edges) && /* If all adjacent edges in surface */
                    !(opposite_edge[i] & done_edges) ) { /* but neither opposite edges */
                    /* Test each of the adjacent edges and opposite edges to edge i */
                    a1 = 0;                               /* Find first adjacent edge */
                    while ( (1 << a1) & ~adjacent_edge[i] ) a1++;
                    a2 = a1+1;                             /* and second adjacent edge */
                    while ( (1 << a2) & ~adjacent_edge[i] ) a2++;
                    o1 = 0;                               /* Find first opposite edge */
                    while ( (1 << o1) & ~opposite_edge[i] ) o1++;
                    o2 = o1+1;                             /* and second opposite edge */
                    while ( (1 << o2) & ~opposite_edge[i] ) o2++;
                    if ( ( (distance[a1]≥0) ∧ (distance[o2]≥0) ) || /* Check if both intersections */

```

### C.3 Intersection of a polygon with a set of planes

The set of dividing planes splits space into partitions. Each partition is defined by which side of each of the planes it is on, and can be represented by an integer, with one bit per dividing plane. The (convex) polygon is intersected with each dividing plane in turn, traversing all the vertices each time. The ‘distance\_to\_plane( $x,y,z,i$ )’ function returns the distance of the point  $(x,y,z)$  to dividing plane  $i$ .

```

/* Perform intersection of polygon with one partition, defined by dividing planes */
/* Initial polygon has 'p1' vertices, at locations 'x1[]', 'y1[]' and 'z1[]' */
for (i=0; i<dividing_planes; i++) { /* Loop through all planes */
    d1 = distance_to_plane( x1[p1-1], y1[p1-1], z1[p1-1], i ); /* Distance for last vertex */
    if ( !(partition & (1<<i)) ) d1 = -d1; /* positive if inside partition */
    p2 = 0; /* Initialise new polygon 'p2' */
    for (v=0; v<p1; v++) { /* Loop through all vertices */
        d1 = distance_to_plane( x1[v], y1[v], z1[v], i ); /* Distance for this vertex */
        if ( !(partition & (1<<i)) ) d2 = -d2; /* positive if inside partition */
        if ( d1 > 0 ) { /* Check for an intersection with line from 'p1-1' to 'v' */
            if ( d2 < 0 ) { /* Just gone outside partition - need new vertex */
                x2[p2] = (d1 * x1[v] - d2 * x1[(v-1+p1)%p1]) / (d1 - d2) + 0.5;
                y2[p2] = (d1 * y1[v] - d2 * y1[(v-1+p1)%p1]) / (d1 - d2) + 0.5;
                z2[p2] = (d1 * z1[v] - d2 * z1[(v-1+p1)%p1]) / (d1 - d2) + 0.5;
                p2++;
            } else { /* Still inside partition - keep this vertex */
                x2[p2] = x1[v];
                y2[p2] = y1[v];
                z2[p2] = z1[v];
            }
        }
    }
}

```

```

    p2++;
  }
} else if ( d2 > 0 ) {      /* Just gone inside partition - need this and new vertex */
  x2[p2] = (d1 * x1[v] - d2 * x1[(v-1+p1)%p1]) / (d1 - d2) + 0.5;
  y2[p2] = (d1 * y1[v] - d2 * y1[(v-1+p1)%p1]) / (d1 - d2) + 0.5;
  z2[p2] = (d1 * z1[v] - d2 * z1[(v-1+p1)%p1]) / (d1 - d2) + 0.5;
  p2++;
  x2[p2] = x1[v];
  y2[p2] = y1[v];
  z2[p2] = z1[v];
  p2++;
}
d1 = d2;                      /* Keep record of last distance to plane */
}
copy_polygon( p2, x2, y2, z2, &p1, &x1, &y1, &z1 );      /* Update original polygon */
}

```

## C.4 Clipping a surface to a set of planes

A triangulated surface can be clipped to a given partition (defined by a set of dividing planes, as in the previous section), by clipping each of the triangles in turn to each dividing plane in turn. Each time a triangle is clipped by a plane, one or two new triangles are created — all of these new triangles must be tested against the remaining dividing planes. Up to twice as many triangles can be created, from each original triangle, as there are dividing planes.

```

/* Intersect triangle 't', with vertices 't.a', 't.b' and 't.c', with a partition */
new_tris = 1; new_tri[0] = t;                      /* Initialise list of triangles */
for (i=0; i<dividing_planes; i++) {                /* Loop through all planes defining partition */
  add_tris = 0;
  for (j=0; j<new_tris; j++) {                      /* Loop through all triangles in current list */
    t = new_tri[j];                                /* Get new triangle */
    da = distance_to_plane( t.a.x, t.a.y, t.a.z, i );      /* and distance to plane */
    db = distance_to_plane( t.b.x, t.b.y, t.b.z, i );      /* for each vertex, */
    dc = distance_to_plane( t.c.x, t.c.y, t.c.z, i );      /* checking sign */
    if ( !(partition & (1<<i)) ) { da = -da; db = -db; dc = -dc; }
    abi = 1; aci = 1; bci = 1;                      /* Find intersections with each edge ab, ac and bc */
    if ( da > 0 & db > 0 ) {                          /* Check edge ab */
      ab.x = (da * t.b.x - db * t.a.x) / (da - db);      /* store intersection */
      ab.y = (da * t.b.y - db * t.a.y) / (da - db);      /* if there is one */
      ab.z = (da * t.b.z - db * t.a.z) / (da - db);
    } else abi = 0;
    if ( da > 0 & dc > 0 ) {                          /* Check edge ac */
      ac.x = (da * t.c.x - dc * t.a.x) / (da - dc);      /* store intersection */
      ac.y = (da * t.c.y - dc * t.a.y) / (da - dc);      /* if there is one */
      ac.z = (da * t.c.z - dc * t.a.z) / (da - dc);
    } else aci = 0;
    if ( db > 0 & dc > 0 ) {                          /* Check edge bc */

```

```

    bc.x = (db * t.c.x - dc * t.b.x) / (db - dc);           /* store intersection */
    bc.y = (db * t.c.y - dc * t.b.y) / (db - dc);           /* if there is one */
    bc.z = (db * t.c.z - dc * t.b.z) / (db - dc);
} else bci = 0;
if ( da < 0 ) index = (abi<<2)|(aci<<1)|(bci);               /* Form index from type of */
else index = ~((abi<<2)|(aci<<1)|(bci));                     /* intersection, and sign */
switch ( index ) {
case 0:                                                       /* No intersections - triangle entirely outside partition */
    for (k=j+1; k<(new_tris+add_tris); k++) new_tri[k-1] = new_tri[k];
    new_tris -= 1; j -= 1;
    break;
case 1:                                                       /* ab and ac intersection - one new triangle */
    new_tri[j] = form_triangle( new_tri[j].a, ab, ac );
    break;
case 2:                                                       /* ab and bc intersection - two new triangles */
    new_tri[j] = form_triangle( new_tri[j].a, ab, new_tri[j].c );
    new_tri[new_tris+add_tris] = form_triangle( new_tri[j].c, ab, bc );
    add_tris++;
    break;
case 3:                                                       /* ac and bc intersection - one new triangle */
    new_tri[j] = form_triangle( new_tri[j].c, ac, bc );
    break;
case 4:                                                       /* ac and bc intersection - two new triangles */
    new_tri[j] = form_triangle( new_tri[j].a, new_tri[j].b, ac );
    new_tri[new_tris+add_tris] = form_triangle( new_tri[j].b, bc, ac );
    add_tris++;
    break;
case 5:                                                       /* ab and bc intersection - one new triangle */
    new_tri[j] = form_triangle( new_tri[j].b, bc, ab );
    break;
case 6:                                                       /* ab and ac intersection - two new triangles */
    new_tri[j] = form_triangle( new_tri[j].b, new_tri[j].c, ab );
    new_tri[new_tris+add_tris] = form_triangle( new_tri[j].b, ac, ab );
    add_tris++;
    break;
case 7:                                                       /* No intersections - triangle entirely inside partition */
    break;
}
}
new_tris += add_tris;                                         /* Update number of triangles, then check next plane */
}

```

## Appendix D

# Non-ultrasound software implementations

In addition to implementation of the algorithms described in this thesis in an ultrasound application, several have also been implemented in more general purpose software. As with the ultrasound software, these are freely available from the internet<sup>1</sup>, and are also included on the CDROM described in Appendix E. A brief description of each application is given here. Full instructions in HTML, and example data (where appropriate) are also contained on the CDROM.

### D.1 IsoSurf — surface visualisation for parallel data

IsoSurf (ISOSURFace extraction) implements the surface interpolation and visualisation algorithms in Chapter 3, in a framework suitable for regular voxel data. The binary input data is first read, then thresholded at a user-defined value, resulting in a set of binary cross-sections. These cross-sections can optionally be filtered using a morphological opening and/or closing with a disc size appropriate to the sampling resolution. This operation ensures that the contours do not contain features that are smaller than the sampling resolution.

Surface triangulation, using regularised marching tetrahedra, is performed at a user-defined resolution. If this resolution is greater than the inter-plane spacing (the z resolution), then additional planes are interpolated, using disc-guided interpolation. Shape-based interpolation can optionally be used in place of disc-guided interpolation, in order to compare results. The triangulated surface can be saved in OOGL<sup>2</sup> (‘.off’), or VRML<sup>3</sup> (‘.wrl’) format for rendering with an appropriate 3D graphics viewer.

### D.2 EqnSurf — visualisation of surfaces of equations

EqnSurf (EquationN isoSURFace visualisation) is an implementation of regularised marching tetrahedra for visualising mathematical functions. The functions are supplied as a text string, with the characters ‘x’, ‘y’ and ‘z’ in place of the variables in each of the three dimensions. This function is evaluated over a user specified volume, defined by its centre and range from this centre point. The isosurface at a user specified threshold is extracted from this volume, and displayed, as in Figure D.1.

---

<sup>1</sup><http://svr-www.eng.cam.ac.uk/~gmt11/software/software.html>

<sup>2</sup>Object Oriented Graphics Library

<sup>3</sup>Virtual Reality Modeling Language



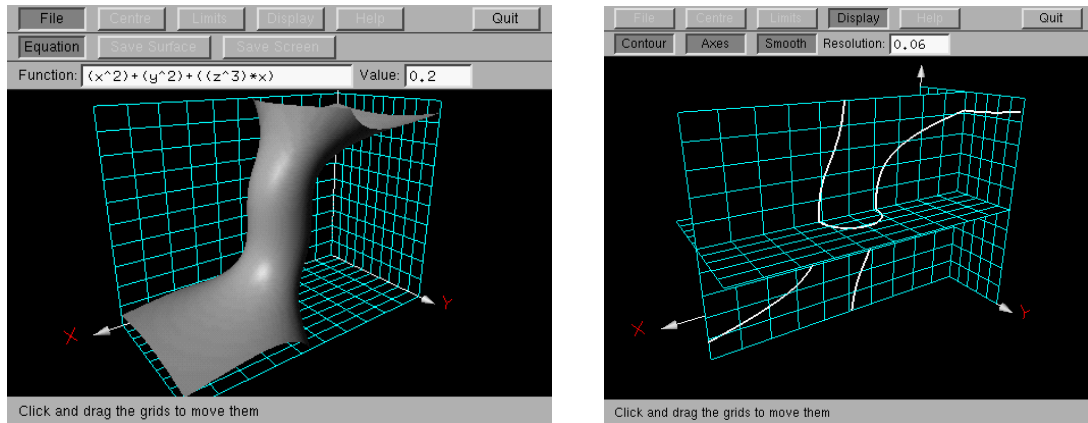


Figure D.1: **Visualisation of equation surfaces.** The triangulated surface can be used both for rendering, and calculating intersections with each axis.

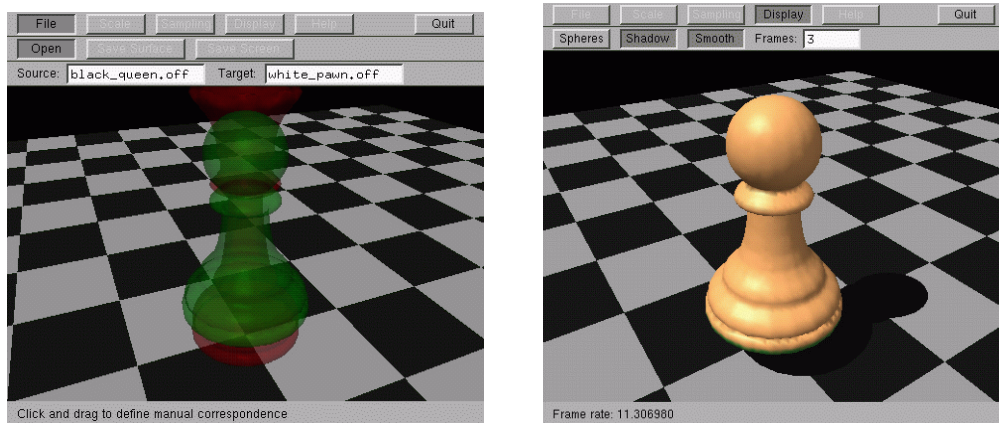


Figure D.2: **Volume morphing of surfaces.** Two views of the user interface are shown.

The viewpoint, and position of the light source, can be rotated by using the mouse buttons. The grids can also be moved by clicking on them, and dragging. In addition to rendering the triangulated surface, it can also be used to calculate the intersection with each of the planes defined by the axes, also shown in Figure D.1. This calculation is updated in real time, such that the intersection changes as the axes are moved.

### D.3 VolMorph — volume morphing of surfaces

VolMorph (VOLUME MORPHing of surfaces) implements the algorithms and user interface described in Chapter 5. Surfaces are input in OOGL ('.off') format, and can be output as either OOGL, VRML ('.wrl'), or as sequences of images.

## Appendix E

# Description of CDROM

A CDROM has been included with this thesis. This contains the thesis itself in PDF<sup>1</sup> format, movies that expand on some of the figures within the thesis, and software implementations as described in Appendices [A.2](#), [D.1](#), [D.2](#) and [D.3](#).

The thesis, movies and software on the CDROM can be accessed with an internet browser, from `index.html`, in the root directory. The movies are in the ‘`movies`’ directory, in both Quicktime<sup>2</sup> and MPEG formats, named with the figure to which they refer. For instance, the movie relating to Figure 3.19d is in the file ‘`movies/figure3_19d.mov`’ (Quicktime version) and ‘`movies/figure3_19d.mpg`’ (MPEG version). The software is available for Windows (98 and NT), Irix (6.3 and above) and Linux, save for Stradx, which is only available for Irix and Linux.

---

<sup>1</sup>Portable Document Format ©2000 Adobe Systems Incorporated

<sup>2</sup>©2000 Apple Computer Inc.

# Bibliography

- [1] R. G. Aarnink, J. J. M. C. H. de la Rosette, F. M. J. Debruyne, and H. Wijkstra. Reproducibility of prostate volume measurements from transrectal ultrasonography by an automated and a manual technique. *British Journal of Urology*, 78:219–223, August 1996.
- [2] R. G. Aarnink, A. L. Huynen, R. J. B. Giesen, J. J. M. C. H. de la Rosette, F. M. J. Debruyne, and H. Wijkstra. Automated prostate volume determination with ultrasonographic imaging. *The Journal of Urology*, 153:1549–1554, May 1995.
- [3] D. Aiger and D. Cohen-Or. Real-time ultrasound imaging simulation. *Real-Time Imaging*, 4:263–274, August 1998.
- [4] M. Alexa. Merging polyhedral shapes with scattered features. *The Visual Computer*, 16(1):26–37, 2000.
- [5] M.-E. Algorri and F. Schmitt. Mesh simplification. *Computer Graphics Forum*, 15(3):C–77–C–86, 1996.
- [6] C. P. Allott, C. D. Barry, R. Pickford, and J. C. Waterton. Volumetric assessment of carotid artery bifurcation using freehand-acquired, compound 3D ultrasound. *The British Journal of Radiology*, 72:289–292, March 1999.
- [7] K. Altmann, Z. Shen, L. M. Boxt, D. L. King, W. M. Gersony, L. D. Allan, and H. D. Apfel. Comparison of three-dimensional echocardiographic assessment of volume, mass, and function in children with functionally single left ventricles with two-dimensional echocardiography and magnetic resonance imaging. *The American Journal of Cardiology*, 80:1060–1065, October 1997.
- [8] P. Arbeille, V. Eder, D. Casset, L. Quillet, C. Hudelo, and S. Herault. Real-time 3-D ultrasound acquisition and display for cardiac volume and ejection fraction evaluation. *Ultrasound in Medicine and Biology*, 26(2):201–208, February 2000.
- [9] C. Arcelli and G. Sanniti di Baja. Finding local maxima in a pseudo-Euclidean distance transform. *Computer Vision, Graphics and Image Processing*, 43:361–367, September 1988.
- [10] K. Baba, T. Okai, S. Kozuma, and Y. Taketani. Fetal abnormalities: evaluation with real-time processible three-dimensional US — preliminary report. *Radiology*, 211(2):441–446, May 1999.
- [11] H. Baddeley, M. Benson, G. Liefman, T. Singcharoen, V. Siskind, K. Soon, and J. Williams. Measurement of liver volume using water delay ultrasonography. *Diagnostic Imaging in Clinical Medicine*, 55:330–336, November 1986.

- [12] C. L. Bajaj, F. Bernardini, and G. Xu. Reconstructing surfaces and functions on surfaces from unorganized three-dimensional data. *Algorithmica*, 19:243–261, September 1997.
- [13] C. L. Bajaj, E. J. Coyle, and K.-N. Lin. Arbitrary topology shape reconstruction from planar cross sections. *Graphical Models and Image Processing*, 58(6):524–543, November 1996.
- [14] F. G. Balen, C. M. Allen, J. E. Gardener, N. C. Siddle, and W. R. Lees. 3-dimensional reconstruction of ultrasound images of the uterine cavity. *The British Journal of Radiology*, 66:588–591, July 1993.
- [15] G. Barequet, D. Shapiro, and A. Tal. Multilevel sensitive reconstruction of polyhedral surfaces from parallel slices. *The Visual Computer*, 16:116–163, 2000.
- [16] G. Barequet and M. Sharir. Piecewise-linear interpolation between polygonal slices. *Computer Vision and Image Understanding*, 63(2):251–272, March 1996.
- [17] W. A. Barrett and E. N. Mortensen. Interactive live-wire boundary extraction. *Medical Image Analysis*, 1(4):331–341, 1997.
- [18] C. D. Barry, C. P. Allott, N. W. John, P. M. Mellor, P. A. Arundel, D. S. Thomson, and J. C. Waterton. Three-dimensional freehand ultrasound: Image reconstruction and volume analysis. *Ultrasound in Medicine and Biology*, 23(8):1209–1224, 1997.
- [19] O. Basset, G. Gimenez, J. L. Mestas, D. Cathignol, and M. Devonec. Volume measurement by ultrasonic transverse or sagittal cross-sectional scanning. *Ultrasound in Medicine and Biology*, 17(3):291–296, 1991.
- [20] M. Belohlavek, D. A. Foley, J. B. Seward, and J. F. Greenleaf. 3D echocardiography: Reconstruction algorithm and diagnostic performance of resulting images. *Proceedings of SPIE*, 2359:680–692, 1994.
- [21] S. Berg, H. Torp, and H.-G. Blaas. Accuracy of *in vitro* volume estimation of small structures using three-dimensional ultrasound. *Ultrasound in Medicine and Biology*, 26(3):425–432, March 2000.
- [22] S. Berg, H. Torp, D. Martens, E. Steen, S. Samstad, I. Høivik, and B. Olstad. Dynamic three-dimensional freehand echocardiography using raw digital ultrasound data. *Ultrasound in Medicine and Biology*, 25(5):745–753, June 1999.
- [23] L.-I. Bih, C.-C. Ho, S.-J. Tsai, Y.-C. Lai, and W. Chow. Bladder shape impact on the accuracy of ultrasonic estimation of bladder volume. *Archives of Physical Medicine and Rehabilitation*, 79(12):1553–1556, December 1998.
- [24] W. Birkfellner, F. Watzinger, F. Wanschitz, G. Enislidis, C. Kollmann, D. Rafolt, R. Nowotny, R. Ewers, and H. Bergmann. Systematic distortions in magnetic position digitizers. *Medical Physics*, 25(11):2242–2248, November 1998.
- [25] E. Bittar, N. Tsingos, and M.-P. Gascuel. Automatic reconstruction of unstructured 3D data: combining a medial axis and implicit surfaces. *Computer Graphics Forum*, 14(3):C–457–C–468, August 1995.

- [26] H.-G. Blaas, S. H. Eik-Nes, S. Berg, and H. Torp. In-vivo three-dimensional ultrasound reconstructions of embryos and early fetuses. *The Lancet*, 352:1182–1186, October 1998.
- [27] J. Bloomenthal and C. Lim. Skeletal methods of shape manipulation. In *Shape Modeling International '99*, International Conference on Shape Modeling and Applications, pages 44–47, Aizu-Wakamatsu, Japan, 1999.
- [28] H. Blum. Biological shape and visual science (Part I). *Journal of Theoretical Biology*, 38:205–287, 1973.
- [29] J.-D. Boissonnat. Shape reconstruction from planar cross sections. *Computer Vision, Graphics and Image Processing*, 44:1–29, 1988.
- [30] F. Bonilla-Musoles, F. Raga, N. G. Osborne, and J. Blanes. Control of intrauterine device insertion with three-dimensional ultrasound: Is it the future? *Journal of Clinical Ultrasound*, 24:263–267, June 1996.
- [31] G. Borgefors. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics and Image Processing*, 27:321–345, 1984.
- [32] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics and Image Processing*, 34:344–371, June 1986.
- [33] G. Borgefors and I. Nyström. Efficient shape representation by minimizing the set of centres of maximal discs/spheres. *Pattern Recognition Letters*, 18:465–471, May 1997.
- [34] D. E. Breen and S. Mauch. Generating shaded offset surfaces with distance, closest-point and color volumes. In *Proceedings of International Workshop on Volume Graphics*, pages 307–320, March 1999.
- [35] R. O. Bude, A. Arbor, and T. A. Tuthill. Ultrasound artifacts peculiar to out-of-plane cross-sectional images reconstructed from 3D volume data sets. *Radiology*, 209P(SS):265, 1998.
- [36] D. Carr and J. G. Duncan. Liver volume determination by ultrasound: a feasibility study. *British Journal of Radiology*, 49:776–778, September 1976.
- [37] E. Catmull and R. Rom. A class of local interpolating splines. In R. Barnhill and R. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 317–326. Academic Press, San Francisco, 1974.
- [38] S. L. Chan and E. O. Purisima. A new tetrahedral tessellation scheme for isosurface generation. *Computers and Graphics*, 22(1):83–90, January 1998.
- [39] H.-H. Chang and H. Yan. Skeletonization of binary digital patterns using a fast Euclidean distance transformation. *Optical Engineering*, 35(4):1003–1008, April 1996.
- [40] M. Chen, M. W. Jones, and P. Townsend. Volume distortion and morphing using disk fields. *Computers and Graphics*, 20(4):567–575, July 1996.
- [41] S.-Y. Chen, W.-C. Lin, C.-C. Liang, and C.-T. Chen. Improvement on dynamic elastic interpolation technique for reconstructing 3-D objects from serial cross sections. *IEEE Transactions on Medical Imaging*, 9(1):71–83, March 1990.

- [42] C.-Y. Chou, K.-F. Hsu, S.-T. Wang, S.-C. Huang, C.-C. Tzeng, and K.-E. Huang. Accuracy of three-dimensional ultrasonography in volume estimation of cervical carcinoma. *Gynecologic Oncology*, 66:89–93, July 1997.
- [43] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers and Graphics*, 22(1):37–54, January 1998.
- [44] D. Cohen-Or, D. Levin, and A. Solomovici. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics*, 17(2):116–141, April 1998.
- [45] L. T. Cook, P. N. Cook, K. R. Lee, S. Batnitzky, B. Y. S. Wong, S. L. Fritz, J. Ophir, S. J. Dwyer III, L. R. Bigongiari, and A. W. Templeton. An algorithm for volume estimation based on polyhedral approximation. *IEEE Transactions on Biomedical Engineering*, 27(9):493–500, September 1980.
- [46] G. Coppini, R. Poli, and G. Valli. Recovery of the 3-D shape of the left ventricle from echocardiographic images. *IEEE Transactions on Medical Imaging*, 14(2):301–317, June 1995.
- [47] D. Crivianu-Gaita, F. Miclea, A. Gaspar, D. Margineatu, and S. Holban. 3D reconstruction of prostate from ultrasound images. *International Journal of Medical Informatics*, 45:43–51, June 1997.
- [48] C. R. Dance. *Computing models from 3D ultrasound*. PhD thesis, Cambridge University, 1997.
- [49] J. Deng, J. E. Gardener, C. H. Rodeck, and W. R. Lees. Fetal echocardiography in three and four dimensions. *Ultrasound in Medicine and Biology*, 22(8):979–986, 1996.
- [50] D. B. Downey, A. Fenster, and J. C. Williams. Clinical utility of three-dimensional US. *Radiographics*, 20(2):559–571, March 2000.
- [51] D. B. Downey, D. A. Nicolle, M. F. Levin, and A. Fenster. Three-dimensional ultrasound imaging of the eye. *Eye*, 10:75–81, 1996.
- [52] W. S. Edwards, C. Deforge, and Y. Kim. Interactive three-dimensional ultrasound using a programmable multimedia processor. *International Journal of Imaging Systems and Technology*, 9(6):442–454, 1998.
- [53] T. L. Elliot, D. B. Downey, S. Tong, C. A. McLean, and A. Fenster. Accuracy of prostate volume measurements in vitro using three-dimensional ultrasound. *Academic Radiology*, 3(5):401–406, May 1996.
- [54] H. Embrechts and D. Roose. MIMD divide-and-conquer algorithms for the distance transformation. Part II: chamfer 3-4 distance. *Parallel Computing*, 21:1077–1096, July 1995.
- [55] R. Entekin, P. Jackson, J. R. Jago, and B. A. Porter. Real time spatial compound imaging in breast ultrasound: technology and early clinical experience. *MedicaMundi*, 43(3):35–43, September 1999.
- [56] A. X. Falcão, J. K. Udupa, S. Samarasekera, S. Sharma, B. E. Hirsch, and R. de A. Lotufo. User-steered image segmentation paradigms: live wire and live lane. *Graphical Models and Image Processing*, 60(4):233–260, July 1998.

- [57] A. Fenster and D. B. Downey. 3-D ultrasound imaging: A review. *IEEE Engineering in Medicine and Biology*, 15(6):41–51, November 1996.
- [58] A. Fenster, S. Tong, S. Sherebrin, D. B. Downey, and R. N. Rankin. Three-dimensional ultrasound imaging. *Proceedings of SPIE*, 2432:176–184, 1995.
- [59] D. Fine, S. Perring, J. Herbetko, C. N. Hacking, J. S. Fleming, and K. C. Dewbury. Three-dimensional (3D) ultrasound imaging of the gallbladder and dilated biliary tree: reconstruction from real-time B-scans. *The British Journal of Radiology*, 64:1056–1057, November 1991.
- [60] J. D. Foley et al. *Computer Graphics: Principles and Practice*. Addison-Wesley Systems Programming Series. Addison-Wesley, second edition, 1996.
- [61] D. S. Fritsch, S. M. Pizer, B. S. Morse, D. H. Eberly, and A. Liu. The multiscale medial axis and its applications in image registration. *Pattern Recognition Letters*, 15(2):445–452, May 1994.
- [62] P. Fritschy, G. Robotti, G. Schneekloth, and P. Vock. Measurement of liver volume by ultrasound and computed tomography. *Journal of Clinical Ultrasound*, 11:299–303, August 1983.
- [63] Q. Gao and F.-F. Yin. Two-dimensional direction-based interpolation with local centred moments. *Graphical Models and Image Processing*, 61:323–339, 1999.
- [64] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97 Proc.*, Computer Graphics, pages 209–216, August 1997.
- [65] A. H. Gee, R. W. Prager, and L. Berman. Non-planar reslicing for freehand 3D ultrasound. In *Proceedings of Medical Image Computing and Computer-Assisted Intervention — MICCAI'99*, LNCS 1679, pages 716–725, Cambridge, UK, September 1999. Springer.
- [66] S. F. F. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *Proceedings of 1998 IEEE Symposium on Volume Visualization*, pages 23–30, October 1998.
- [67] O. H. Gilja, P. R. Detmer, J. M. Jong, D. F. Leotta, X.-N. Li, K. W. Beach, R. Martin, and Jr. D. E. Strandness. Intragastric distribution and gastric emptying assessed by three-dimensional ultrasonography. *Gastroenterology*, 113:38–49, July 1997.
- [68] O. H. Gilja, T. Hausken, A. Berstad, and S. Ødegaard. Measurements of organ volume by ultrasonography. *Proceedings of the Institution of Mechanical Engineers*, 213(H):247–259, 1999.
- [69] O. H. Gilja, A. I. Smievoll, N. Thune, K. Matre, T. Hausken, S. Ødegaard, and A. Berstad. *In vivo* comparison of 3D ultrasonography and magnetic resonance imaging in volume estimation of human kidneys. *Ultrasound in Medicine and Biology*, 21(1):25–32, 1995.
- [70] J. Gomas, L. Darsa, B. Costa, and L. Velho. *Warping and morphing of graphical objects*. Morgan Kaufmann Publishers, Inc., 1999.



- [71] A. S. Gopal, M. J. Schnellbaecher, Z. Shen, O. O. Akinboboye, P. M. Sapin, and D. L. King. Freehand three-dimensional echocardiography for measurement of left ventricular mass: *in vivo* anatomic validation using explanted human hearts. *Journal of the American College of Cardiology*, 30(3):802–810, September 1997.
- [72] A. Gregory, A. State, M. C. Lin, D. Manocha, and M. A. Livingston. Interactive surface decomposition for polyhedral morphing. *The Visual Computer*, 15(9):453–470, 1999.
- [73] A. Guéziec and D. Dean. The Wrapper: A surface optimization algorithm that preserves highly curved areas. In *Visualization in Biomedical Computing*, volume 2359, pages 631–642. SPIE, 1994.
- [74] A. Guéziec and R. Hummel. Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):328–342, December 1995.
- [75] E. Hafner, T. Philipp, K. Schuchter, B. Dillinger-Paller, K. Philipp, and P. Bauer. Second-trimester measurements of placental volume by three-dimensional ultrasound to predict small-for-gestational-age infants. *Ultrasound in Obstetrics and Gynecology*, 12(2):97–102, August 1998.
- [76] T. D. Haig, Y. Attikouzel, and M. Alder. Border marriage: matching of contours of serial sections. *IEE Proceedings*, 138(5):371–376, October 1991.
- [77] M. Hall and J. Warren. Adaptive polygonization of implicitly defined surfaces. *IEEE Computer Graphics and Applications*, 10(6):33–40, November 1990.
- [78] E. Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14(3):95–108, 1998.
- [79] S. Hashimoto, H. Goto, Y. Hirooka, A. Itoh, Y. Ishiguro, S. Kojima, T. Hirai, T. Hayakawa, and Y. Naitoh. An evaluation of three-dimensional ultrasonography for the measurement of gallbladder volume. *The American Journal of Gastroenterology*, 94(12):3492–3496, December 1999.
- [80] T. Hausken, D. F. Leotta, S. Helton, K. V. Kowdley, B. Goldman, S. Vaezy, E. L. Bolson, F. H. Sheehan, and R. W. Martin. Estimation of the human liver volume and configuration using three-dimensional ultrasonography: effect of a high-calorific liquid meal. *Ultrasound in Medicine and Biology*, 24(9):1357–1367, November 1998.
- [81] T. He, S. Wang, and A. Kaufman. Wavelet-based volume morphing. In R. D. Bergeron and A. E. Kaufman, editors, *Proceedings of Visualization '94*, pages 85–92. IEEE Computer Society Press, 1994.
- [82] G. T. Herman, J. S. Zheng, and C. A. Bucholtz. Shape-based interpolation. *IEEE Computer Graphics and Applications*, 12(3):69–79, May 1992.
- [83] W. E. Higgins, C. Morice, and E. L. Ritman. Shape-based interpolation of tree-like structures in three-dimensional images. *IEEE Transactions on Medical Imaging*, 12(3):439–450, September 1993.

- [84] T. C. Hodges, P. R. Detmer, D. H. Burns, K. W. Beach, and D. E. Strandness Jr. Ultrasonic three-dimensional reconstruction: *in vitro* and *in vivo* volume and area measurement. *Ultrasound in Medicine and Biology*, 20(8):719–729, 1994.
- [85] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, July 1992.
- [86] I. M. Hösli, S. Tercanli, A. Herman, M. Kretschmann, and W. Holzgreve. *In vitro* volume measurement by three-dimensional ultrasound: comparison of two different systems. *Ultrasound in Obstetrics and Gynecology*, 11(1):17–22, January 1998.
- [87] D. Howe, T. Wheeler, and S. Perring. Measurement of placental volume with real-time ultrasound in mid-pregnancy. *Journal of Clinical Ultrasound*, 22(2):77–83, February 1994.
- [88] J. F. Hughes. Scheduled Fourier volume morphing. In *SIGGRAPH '92*, volume 26 of *Computer Graphics*, pages 43–46, 1992.
- [89] S. W. Hughes, T. J. D'Arcy, D. J. Maxwell, W. Chiu, A. Milner, J. E. Saunders, and R. J. Sheppard. Volume estimation from multiplanar 2D ultrasound images using a remote electromagnetic position and orientation sensor. *Ultrasound in Medicine and Biology*, 22(5):561–572, 1996.
- [90] S. W. Hughes, T. J. D'Arcy, D. J. Maxwell, J. E. Saunders, C. F. Ruff, W. S. C. Chiu, and R. J. Sheppard. Application of a new discrete form of Gauss' theorem for measuring volume. *Physics in Medicine and Biology*, 41(9):1809–1821, September 1996.
- [91] P. K. Jensen and M. K. Hansen. Ultrasonographic, three-dimensional scanning for determination of intraocular tumor volume. *Acta Ophthalmologica*, 69(2):178–186, April 1991.
- [92] B. Jin, L. Turner, B. Crawford, A. Birrel, and D. J. Handelsman. The development of the baboon prostate: Ultrasound methodology, modelling, and natural history. *Journal of Andrology*, 17(4):342–352, July 1996.
- [93] M. W. Jones. The production of volume data from triangular meshes using voxelisation. *Computer Graphics Forum*, 15(5):311–318, December 1996.
- [94] M. W. Jones and M. Chen. A new approach to the construction of surfaces from contour data. *Computer Graphics Forum*, 13(3):C-75–C-84, September 1994.
- [95] J.-M. Jong, K. W. Beach, J. F. Primozich, R. O. Bergelin, M. Caps, C. H. Chan, and D. E. Strandness Jr. Vein graft surveillance with scanhead tracking duplex ultrasound imaging: a preliminary report. *Ultrasound in Medicine and Biology*, 24(9):1313–1324, November 1998.
- [96] W. Kampmann, M. M. Walka, M. Vogel, and M. Obladen. 3-D sonographic volume measurement of the cerebral ventricular system: *in vitro* validation. *Ultrasound in Medicine and Biology*, 24(8):1169–1174, October 1998.
- [97] M. Kass, A. Witkin, and D. Terzopoulos. Snakes — active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.
- [98] A. Kaufman. Efficient algorithms for scan-converting 3D polygons. *Computers and Graphics*, 12(2):213–219, 1988.

- [99] A. Kaufman, D. Cohen, and R. Yagel. Volume graphics. *Computer*, 26(7):51–64, July 1993.
- [100] T. Kimoto and Y. Yasuda. Shape description and representation by ellipsoids. *Signal Processing: Image Communication*, 9(3):275–290, March 1997.
- [101] D. L. King, A. S. Gopal, A. M. Keller, P. M. Sapin, and K. M. Schröder. Three-dimensional echocardiography: Advances for measurement of ventricular volume and mass. *Hypertension*, 23(1):I–172–I–179, January 1994.
- [102] H.-M. Klein, R. W. Günther, M. Verlande, W. Schneider, D. Vorwerk, J. Kelch, and M. Hamm. 3D-surface reconstruction of intravascular ultrasound images using personal computer hardware and a motorized catheter control. *Cardiovascular and Interventional Radiology*, 15(2):97–101, March 1992.
- [103] F. W. Kremkau. *Diagnostic Ultrasound: Principles and Instruments*. W. B. Saunders Company, fourth edition, 1993.
- [104] R. C. Lalouche, D. Bickmore, F. Tessler, N. J. Mankovich, H. K. Huang, and H. Kangarloo. Three-dimensional reconstruction of ultrasound images. *Proceedings of SPIE*, 1092, 1989. 450–457.
- [105] J. A. M. Laudy, M. M. M. Janssen, P. C. Struyk, T. Stijnen, H. C. S. Wallenburg, and J. W. Wladimiroff. Fetal liver volume measurement by three-dimensional ultrasonography: a preliminary study. *Ultrasound in Obstetrics and Gynecology*, 12(2):93–96, August 1998.
- [106] F. Lazarus and A. Verroust. Three-dimensional metamorphosis: a survey. *The Visual Computer*, 14:373–389, 1998.
- [107] A. Lee, J. Deutinger, and G. Bernaschek. Three dimensional ultrasound: abnormalities of the fetal face in surface and volume rendering mode. *British Journal of Obstetrics and Gynaecology*, 102(4):302–306, April 1995.
- [108] A. W. F. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. In *Proceedings of SIGGRAPH 99*, pages 343–350. ACM SIGGRAPH, 1999.
- [109] M. E. Legget, D. F. Leotta, E. L. Bolson, J. A. McDonald, R. W. Martin, X.-N. Li, C. M. Otto, and F. H. Sheehan. System for quantitative three-dimensional echocardiography of the left ventricle based on a magnetic-field position and orientation sensing system. *IEEE Transactions on Biomedical Engineering*, 45(4):494–504, April 1998.
- [110] A. Leros, C. D. Garfinkle, and M. Levoy. Feature-based volume metamorphosis. In *SIGGRAPH '95*, volume 29 of *Computer Graphics*, pages 449–464, 1995.
- [111] D. Levin. Multidimensional reconstruction by set-valued approximations. *IMA Journal of Numerical Analysis*, 6(2):173–184, April 1986.
- [112] E. D. Light, R. E. Davidsen, J. O. Fiering, T. A. Hruschka, and S. W. Smith. Progress in 2-D arrays for real time volumetric imaging. *Ultrasonic Imaging*, 20:1–15, January 1998.
- [113] X.-Z. Lin, Y.-N. Sun, Y.-H. Liu, B.-S. Sheu, B.-N. Cheng, C.-Y. Chen, H.-M. Tsai, and C.-L. Shen. Liver volume in patients with or without chronic liver disease. *Hepato-Gastroenterology*, 45(22):1069–1074, 1998.

- [114] Y.-H. Liu, Y.-N. Sun, C.-W. Mao, and C.-J. Lin. Edge-shrinking interpolation for medical images. *Computerized Medical Imaging and Graphics*, 21(2):91–101, March 1997.
- [115] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–168, July 1987.
- [116] M. Loughlin, I. Carlbom, C. Busch, T. Douglas, L. Egevad, H. Frimmel, M. Norberg, I. Sesterhenn, and J. M. Frogge. Three-dimensional modelling of biopsy protocols for localized prostate cancer. *Computerized Medical Imaging and Graphics*, 22(3):229–238, May 1998.
- [117] G. Magni, Q.-L. Cao, L. Sugeng, A. Delabays, G. Marx, A. Ludomirski, M. Vogel, and N. G. Pandian. Volume-rendered, three-dimensional echocardiographic determination of the size, shape, and position of atrial septal defects: Validation in an *in vitro* model. *American Heart Journal*, 132(2):376–381, August 1996.
- [118] L. S. Marks, F. J. Dorey, M. L. Macairan, C. Park, and J. B. de Kernion. Three-dimensional ultrasound device for rapid determination of bladder volume. *Urology*, 50(3):341–348, September 1997.
- [119] R. W. Martin, G. Bashein, P. R. Detmer, and W. E. Moritz. Ventricular volume measurement from a multiplanar transesophageal ultrasonic imaging system: an *in vitro* study. *IEEE Transactions on Biomedical Engineering*, 37(5):442–449, May 1990.
- [120] S. Meairs, J. Beyer, and M. Hennerici. Reconstruction and visualization of irregularly sampled three- and four-dimensional ultrasound data for cerebrovascular applications. *Ultrasound in Medicine and Biology*, 26(2):263–272, February 2000.
- [121] E. Merz, F. Bahlmann, G. Weber, and D. Macchiella. Three-dimensional ultrasonography in prenatal diagnosis. *Journal of Perinatal Medicine*, 23:213–222, 1995.
- [122] D. Meyers, S. Skinner, and K. Sloan. Surfaces from contours. *ACM Transactions on Graphics*, 11(3):228–258, July 1992.
- [123] C. Montani, R. Scateni, and R. Scopigno. A modified look-up table for implicit disambiguation of Marching Cubes. *The Visual Computer*, 10(6):353–355, August 1994.
- [124] A. Montanvert and Y. Usson. Discrete distances applied to 2D granulometry and 3D reconstruction. In *8th Scandinavian Conference on Image Analysis*, pages 1153–1160, 1993.
- [125] W. E. Moritz, A. S. Pearlman, D. H. McCabe, D. K. Medema, M. E. Ainsworth, and M. S. Boles. An ultrasonic technique for imaging the ventricle in three dimensions and calculating its volume. *IEEE Transactions on Biomedical Engineering*, 30(8):482–492, August 1983.
- [126] M. Moshfeghi. Directional interpolation for magnetic resonance angiography data. *IEEE Transactions on Medical Imaging*, 12(2):366–379, June 1993.
- [127] N. Nagdyman, M. M. Walka, W. Kampmann, B. Stöver, and M. Obladen. 3-D ultrasound quantification of neonatal cerebral ventricles in different head positions. *Ultrasound in Medicine and Biology*, 25(6):895–900, July 1999.

- [128] B. K. Natarajan. On generating topologically consistent isosurfaces from uniform samples. *The Visual Computer*, 11(1):52–62, 1994.
- [129] M. S. Nathan, K. Seenivasagam, Q. Mei, J. E. A. Wickham, and R. A. Miller. Transrectal ultrasonography: why are estimates of prostate volume and dimension so inaccurate? *British Journal of Urology*, 77(3):401–407, March 1996.
- [130] T. R. Nelson, D. B. Downey, D. H. Pretorius, and A. Fenster. *Three-Dimensional Ultrasound*. Lippincott Williams & Wilkins, 1999.
- [131] T. R. Nelson and T. T. Elvins. Visualization of 3D ultrasound data. *IEEE Computer Graphics and Applications*, 13(6):50–57, November 1993.
- [132] T. R. Nelson and D. H. Pretorius. Visualization of the fetal thoracic skeleton with three-dimensional sonography: A preliminary report. *American Journal of Roentgenology*, 164(6):1485–1488, June 1995.
- [133] K. J. Ng, J. E. Gardener, D. Rickards, W. R. Lees, and E. J. G. Milroy. Three-dimensional imaging of the prostatic urethra — an exciting new tool. *British Journal of Urology*, 74(5):604–608, November 1994.
- [134] F. Nilsson and P.-E. Danielsson. Finding the minimal set of maximum disks for binary objects. *Graphical Models and Image Processing*, 59(1):55–60, January 1997.
- [135] P. Ning and J. Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, November 1993.
- [136] E. O. Ofili and N. C. Nanda. Three-dimensional and four-dimensional echocardiography. *Ultrasound in Medicine and Biology*, 20(8):669–675, 1994.
- [137] Y. Ohashi. Three-dimensional reconstruction of pore geometry from serial sections — image algebraic approach. In R. Pflug and J. W. Harbaugh, editors, *Computer Graphics in Geology*, volume 41 of *Lecture Notes in Geology*, pages 63–76. Springer, 1992.
- [138] R. Ohbuchi, D. Chen, and H. Fuchs. Incremental volume reconstruction and rendering for 3D ultrasound imaging. *Proceedings of SPIE*, 1808:312–323, 1992.
- [139] J. O’Rourke. *Computational Geometry in C*. Cambridge University Press, 1993.
- [140] N. Pagoulatos, R. N. Rohling, W. S. Edwards, and Y. Kim. New spatial localizer based on fiber optics with applications in 3D ultrasound imaging. In *Medical Imaging 2000: Image Display and Visualisation*, volume 3976 of *Proceedings of SPIE*, pages 595–602, San Diego, California, February 2000.
- [141] B. A. Payne and A. W. Toga. Surface mapping brain function on 3D models. *IEEE Computer Graphics and Applications*, 10(5):33–41, September 1990.
- [142] B. A. Payne and A. W. Toga. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications*, 12(1):65–71, January 1992.
- [143] B. A. Payne and A. W. Toga. Surface reconstruction by multiaxial triangulation. *IEEE Computer Graphics and Applications*, 14(6):28–35, November 1994.

- [144] U. G. Pöhls and A. Rempen. Fetal lung volumetry by three-dimensional ultrasound. *Ultrasound in Obstetrics and Gynecology*, 11(1):6–12, January 1998.
- [145] R. W. Prager, A. H. Gee, and L. Berman. Stradx: real-time acquisition and visualisation of freehand 3D ultrasound. *Medical Image Analysis*, 3(2):129–140, 1999.
- [146] R. W. Prager, R. N. Rohling, A. H. Gee, and L. Berman. Rapid calibration for 3-D free-hand ultrasound. *Ultrasound in Medicine and Biology*, 24(6):855–869, 1998.
- [147] D. H. Pretorius, T. R. Nelson, R. N. Baergen, E. Pai, and C. Cantrell. Imaging of placental vasculature using three-dimensional ultrasound and color power doppler: a preliminary study. *Ultrasound in Obstetrics and Gynecology*, 12(1):45–49, July 1998.
- [148] D. T. Puff, D. Eberly, and S. M. Pizer. Object-based interpolation via cores. *Proceedings of SPIE*, 2167:143–150, 1994.
- [149] A. Rahmouni, A. Yang, C. M. C. Tempany, T. Frenkel, J. Epstein, P. Walsh, P. K. Leichner, C. Ricci, and E. Zerhouni. Accuracy of *in vivo* assessment of prostatic volume by MRI and transrectal ultrasonography. *Journal of Computer Assisted Tomography*, 16(6):935–940, November 1992.
- [150] V. Ranjan and A. Fournier. Shape transformations using union of spheres. Technical Report TR-95-30, Imager Computer Graphics Laboratory, Department of Computer Science, University of British Columbia, 1995.
- [151] V. Ranjan and A. Fournier. Matching and interpolation of shapes using unions of circles. *Computer Graphics Forum*, 15(3):C–129–C–142, August 1996.
- [152] R. N. Rankin, A. Fenster, D. B. Downey, P. L. Munk, M. F. Levin, and A. D. Vellet. Three-dimensional sonographic reconstruction: Techniques and diagnostic applications. *American Journal of Roentgenology*, 161(4):695–702, October 1993.
- [153] S. N. Rasmussen. Liver volume determination by ultrasonic scanning. *British Journal of Radiology*, 45:579–585, August 1972.
- [154] S. P. Raya and J. K. Udupa. Shape-based interpolation of multidimensional objects. *IEEE Transactions on Medical Imaging*, 9(1):32–42, March 1990.
- [155] M. Reddy. SCROOGE: Perceptually-driven polygon reduction. *Computer Graphics Forum*, 15(4):191–203, October 1996.
- [156] M. Riccabona, D. H. Pretorius, T. R. Nelson, D. Johnson, and N. E. Budorick. Three-dimensional ultrasound: display modalities in obstetrics. *Journal of Clinical Ultrasound*, 25(4):157–167, May 1997.
- [157] B. Robert, B. Richard, and J.-M. Nicolas. An interactive tool to visualize three-dimensional ultrasound data. *Ultrasound in Medicine and Biology*, 26(1):133–142, January 2000.
- [158] R. N. Rohling. *3D freehand ultrasound: reconstruction and spatial compounding*. PhD thesis, Cambridge University, 1999.
- [159] R. N. Rohling, A. H. Gee, and L. Berman. Three-dimensional spatial compounding of ultrasound images. *Medical Image Analysis*, 1(3):177–193, April 1997.

- [160] R. N. Rohling, A. H. Gee, and L. Berman. Automatic registration of 3-D ultrasound images. *Ultrasound in Medicine and Biology*, 24(6):841–854, July 1998.
- [161] R. N. Rohling, A. H. Gee, and L. Berman. A comparison of freehand three-dimensional ultrasound reconstruction techniques. *Medical Image Analysis*, 3(4):339–359, 1999.
- [162] K. Rosenfield, P. Boffetti, J. Kaufman, R. Weinstein, S. Razvi, and J. M. Isner. Three-dimensional reconstruction of human carotid arteries from images obtained during noninvasive B-mode ultrasound examination. *The American Journal of Cardiology*, 70(3):379–384, August 1992.
- [163] C. F. Ruff, S. W. Hughes, and D. J. Hawkes. Volume estimation from sparse planar images using deformable models. *Image and Vision Computing*, 17(8):559–565, June 1999.
- [164] D. Ruprecht and H. Müller. Deformed cross-dissolves for image interpolation in scientific visualization. *The Journal of Visualization and Computer Animation*, 5(3):167–181, July 1994.
- [165] A. Salustri and J. R. T. C. Roelandt. Ultrasonic three-dimensional reconstruction of the heart. *Ultrasound in Medicine and Biology*, 21(3):281–293, 1995.
- [166] V. V. Savchenko, A. A. Pasko, O. G. Okunev, and T. L. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14(4):181–188, October 1995.
- [167] R. L. Schild, T. Wallny, R. Fimmers, and M. Hansmann. Fetal lumbar spine volumetry by three-dimensional ultrasound. *Ultrasound in Obstetrics and Gynecology*, 13(5):335–339, May 1999.
- [168] P. Schnider, P. Birner, A. Gendo, K. Ratheiser, and E. Auff. Bladder volume determination: portable 3-D versus stationary 2-D ultrasound device. *Archives of Physical Medicine and Rehabilitation*, 81(1):18–21, January 2000.
- [169] C. M. Sehgal, G. A. Broderick, R. Whittington, R. J. T. Gorniak, and P. H. Arger. Three-dimensional US and volumetric assessment of the prostate. *Radiology*, 192(1):274–278, July 1994.
- [170] R. Shahidi, R. Tombropoulos, and R. P. Grzeszczuk. Clinical applications of three-dimensional rendering of medical data sets. *Proceedings of the IEEE*, 86(3):555–568, March 1998.
- [171] W.-S. V. Shih, W.-C. Lin, and C.-T. Chen. Morphologic field morphing: Contour model-guided image interpolation. *International Journal of Imaging Systems and Technology*, 8(5):480–490, 1997.
- [172] M. Sramek and W. E. Kaufman. Alias-free voxelization of geometric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):251–267, July 1999.
- [173] H. Steiner, A. Staudach, D. Spitzer, and H. Schaffer. Three-dimensional ultrasound in obstetrics and gynaecology: technique, possibilities and limitations. *Human Reproduction*, 9(9):1773–1778, September 1994.



- [174] M. K. Terris and T. A. Stamey. Determination of prostate volume by transrectal ultrasound. *The Journal of Urology*, 145(5):984–987, May 1991.
- [175] S. Tong, H. N. Cardinal, R. F. McLoughlin, D. B. Downey, and A. Fenster. Intra- and inter-observer variability and reliability of prostate volume measurement via two-dimensional and three-dimensional ultrasound imaging. *Ultrasound in Medicine and Biology*, 24(5):673–681, 1998.
- [176] J. W. Trobaugh, D. J. Trobaugh, and W. D. Richard. Three-dimensional imaging with stereotactic ultrasonography. *Computerized Medical Imaging and Graphics*, 18(5):315–323, September 1994.
- [177] P. M. Tuomola, A. H. Gee, R. W. Prager, and L. Berman. Body-centred visualisation for freehand 3D ultrasound. *Ultrasound in Medicine and Biology*, 26(4):539–550, June 2000.
- [178] G. Turk and J. F. O’Brien. Shape transformation using variational implicit functions. In *Siggraph 99: Computer Graphics Proceedings*, pages 335–342, Los Angeles, California, August 1999.
- [179] T. A. Tuthill, J. F. Krücker, J. B. Fowlkes, and P. L. Carson. Automated three-dimensional US frame positioning computed from elevational speckle decorrelation. *Radiology*, 209(2):575–582, 1998.
- [180] J. K. Udupa. Three-dimensional visualization and analysis methodologies: A current perspective. *Radiographics*, 19(3):783–806, May 1999.
- [181] J. K. Udupa, D. Odhner, S. Samarasekera, R. Goncalves, K. Iyer, K. Venugopal, and S. Furuie. 3DVIEWNIX: An open, transportable, multidimensional, multi-modality, multiparametric imaging software system. In *SPIE Proceedings*, volume 2164, pages 58–73, 1994.
- [182] Y. Watanabe. A method for volume estimation by using vector areas and centroids of serial cross sections. *IEEE Transactions on Biomedical Engineering*, 29(3):202–205, 1982.
- [183] L. Weng, A. P. Tirumalai, C. M. Lowery, L. F. Nock, D. E. Gustafson, P. L. Von Behren, and J. H. Kim. US extended-field-of-view imaging technology. *Radiology*, 203(3):877–880, 1997.
- [184] P. N. Werahera, G. J. Miller, G. D. Taylor, T. Brubaker, F. Daneshgari, and E. D. Crawford. A 3-D reconstruction algorithm for interpolation and extrapolation of planar cross sectional data. *IEEE Transactions on Medical Imaging*, 14(4):765–771, December 1995.
- [185] R. T. Whitaker and D. E. Breen. Level-set models for the deformation of solid objects. In *Proceedings of 3rd International Workshop on Implicit Surfaces*, pages 19–35. Eurographics, June 1998.
- [186] J. Wilhelms and A. Van Gelder. Topological considerations in isosurface generation — extended abstract. *Computer Graphics*, 24(5):79–86, November 1990.
- [187] G. Wolberg. Image morphing: a survey. *The Visual Computer*, 14:360–372, 1998.
- [188] Y. Zhou, W. Chen, and Z. Tang. An elaborate ambiguity detection method for constructing isosurfaces within tetrahedral meshes. *Computers and Graphics*, 19(3):355–364, May 1995.

- 
- [189] M. Zöckler, D. Stalling, and H.-C. Hege. Fast and intuitive generation of geometric shape transitions. *The Visual Computer*, 16(5):241–253, 2000.