NON-INVASIVE MULTI-VIEW 3D DYNAMIC MODEL EXTRACTION

By
Karl J Sharman
M.Eng

A thesis submitted for the degree of
Doctor of Philosophy

Department of Electronics and Computer Science,
University of Southampton,
United Kingdom.

July 2002

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING

ELECTRONICS AND COMPUTER SCIENCE DEPARTMENT

Doctor of Philosophy

Non-Invasive Multi-View 3D Dynamic Model Extraction

by Karl J Sharman

A non-invasive system is presented which is capable of extracting and describing the three-dimensional nature of human gait thereby extending the potential use of gait as a biometric. Of current three-dimensional systems, those using multiple views appear to be the most suitable. Reformulating the three-dimensional analysis algorithm known as Volume Intersection as an evidence gathering process for abstract scene reconstruction provides a new way to overcome concavities and to handle noise and occlusion.

After analysis of the standard voxel-based three-dimensional representation, a new data representation called 2.75D is suggested which allows the scene to be analysed at the most appropriate resolution, avoiding further discretisation.

With a sequence of three-dimensional frames, another evidence gathering algorithm is applied to extract and describe the motion of moving objects. No current techniques have exploited the sequence as a whole during such an operation and in this thesis, a method to incorporate successive frames, and therefore time, as an additional dimension to the extraction process is described.

Results on synthetic and real images show that the techniques do indeed process a multi-view image sequence to derive the parameters of interest, thereby providing a suitable basis for future development as a marker-less three-dimensional gait analysis system. In particular, the parameters of a ball moving under the influence of gravity are extracted with accuracy from a 3D scene. Also, a walking human is extracted and overlaying the result onto the original images confirms that the correct extraction has been made; the result is also supported by medical studies.

Contents

Acknowledgements xii

Chapter 1 Introduction 1

1.1 Gait as a biometric . . . 2

1.1.1 Current methods of gait analysis . . . 3

1.2 Analysis of possible solutions . . . 3

1.3 Acquiring 3D data . . . 3

1.3.1 Silhouette-based algorithms . . . 5

1.3.2 The visual hull . . . 7

1.3.3 Non-segmented 3D reconstruction . . . 9

1.3.4 Alternative methods . . . 11

1.4 Dynamic extraction . . . 11

1.4.1 Evidence gathering: the Hough Transform and Template Matching . . . 12

1.5 Systems . . . 15

1.6 Contributions . . . 17

1.7 Publications associated with this thesis . . . 18

1.8 Thesis structure . . . 18

Chapter 2 Three-dimensional scene reconstruction 19

2.1 Introduction . . . 19

2.2 Transformations . . . 19

2.2.1 The pin-hole camera model . . . 19

2.2.2 Intrinsic parameters . . . 20

2.2.3 Extrinsic parameters . . . 22

2.2.4 The complete projection . . . 23

2.3 The 3D Hough Transform and VI . . . 24

2.4 The new voxel-based algorithm . . . 25

2.4.1 The hypothesis and overview . . . 25

2.4.2 The confidence measure . . . 27

2.4.3 Voxel selection . . . 27

2.4.4 Shade, occlusions and transparencies 27
2.4.5 Anti-aliasing blocks 28
2.4.6 Voxel sides 29
2.4.7 Constants 30
2.5 Three dimensional reconstruction results 32
2.5.1 The visual hull 32
2.5.2 Conflicting images 34
2.5.3 Phantom shapes 34
2.5.4 Real data 35
2.6 Conclusion 36

Chapter 3 The 2.75D projection 39
3.1 Introduction 39
3.2 The new representation 40
3.3 Formalising VI 42
3.3.1 Definitions 42
3.3.2 The projection into the camera 43
3.3.3 The voxel-based VI 43
3.3.4 The 2.75D projection 44
3.4 Ray casting: the optimum rate 46
3.4.1 The approximate method 46
3.5 Removing the approximation 47
3.6 Implementing VI 48
3.6.1 Complexity analysis 49
3.7 2.75D VI results 50
3.8 Grey scale and colour implementation 51
3.9 Grey scale results 55
3.10 Colour results 57
3.11 Future work 58
3.12 Related work 60
3.13 Conclusion 61

Chapter 4 Three-dimensional dynamic model extraction 62
4.1 Introduction 62
4.2 Background removal 62
4.2.1 2D background removal 62
4.2.2 2.75D background removal 64
4.2.3 3D background removal 64
4.3 Extraction 64
4.3.1 Introduction 64

4.3.2 Evidence gathering 64
4.3.3 Mathematical models 65
4.3.4 Modules, basic shapes and CSG 67
4.3.5 2D basic shape evaluation 68
4.3.6 2.75D basic shape evaluation 71
4.3.7 3D basic shape evaluation 73
4.4 Genetic Algorithms 74
4.4.1 The choice of algorithm 74
4.4.2 Overview 74
4.4.3 Choice of parameters 76
4.4.4 Increasing the size of the peak 77
4.5 Model weighting 78
4.5.1 The problem 78
4.5.2 Suitable values for k 79
4.6 Examples 82
4.6.1 Moving ball 82
4.6.2 Gait analysis 84
4.7 Conclusion 88

Chapter 5 Comparison of reconstruction methods 89
5.1 Introduction 89
5.2 Experimental set-up 89
5.3 Discretisation 90
5.3.1 The perfect reconstruction 90
5.3.2 Reconstructing from perfect images 93
5.4 Camera parameters perturbation 93
5.4.1 Extrinsic parameters 95
5.4.2 Intrinsic parameters 95
5.5 Image noise response 97
5.5.1 Camera focusing 98
5.5.2 Additive Gaussian noise 102
5.6 Discussion and conclusion 103

Chapter 6 Model extraction 107
6.1 Introduction 107
6.2 Synthetic example: analysis of a moving ball 107
6.2.1 Setting the scene 107
6.2.2 2D data preparation 109
6.2.3 2.75D data preparation 109
6.2.4 3D data preparation 110

6.2.5 Parameter extraction 112
6.2.6 Problems with background removal 113
6.2.7 Analysis of an occluded moving ball 113
6.2.8 Conclusion on synthetic data 115
6.3 The real world data capture 116
6.3.1 Introduction 116
6.3.2 Digital Video (DV) 117
6.3.3 Calibration of cameras 118
6.3.4 Interlaced video 119
6.4 Real world example: ball under the influence of gravity 120
6.4.1 The model 121
6.4.2 The source data 121
6.4.3 The reconstructed and filtered data 121
6.4.4 Parameter extraction 127
6.4.5 Conclusion on the ball under the influence of gravity 130
6.5 Real world example: human gait 130
6.5.1 The model 130
6.5.2 The source data 132
6.5.3 Parameter extraction 133
6.5.4 Improving the model 134
6.6 Conclusion 134

Chapter 7 Concluding remarks 141
7.1 Introduction 141
7.2 Scene reconstruction 141
7.2.1 Improvements to the reconstruction 142
7.3 Dynamic model extraction 144
7.3.1 Improvements to the extraction 145
7.4 The complete systems 147

References 149

Appendix A Camera arrangement 157
A.1 Introduction 157
A.2 Camera parameter conversion 157
A.3 Real world camera arrangement 158
A.4 Synthetic camera arrangement I 160
A.5 Synthetic camera arrangement II 161

Appendix B Further model extraction 162
B.1 Introduction 162
B.2 Synthetic example: analysis of a box moving around an arc 162
B.2.1 Setting the scene 162
B.2.2 Parameter extraction 164
B.3 Gait intermediate processing examples 164
B.3.1 Introduction 164
B.3.2 The reconstructed and filtered data 165

Appendix C Pixel ray casting 169
C.1 Overview 169
C.2 Theory 169
C.2.1 The positive direction 171
C.2.2 The negative direction 172
C.2.3 Combining the two equations 172
C.2.4 Confirming the result 173
C.2.5 The y-axis and multiple views. 173

Appendix D Camera synchronisation device 174
D.1 Introduction 174

Appendix E DV Codecs 177
E.1 Introduction 177
E.2 Overview of the standard 177
E.3 The 4:2:0 representation 178

List of Figures

1.1 Volume Intersection. 6
1.2 The visual hull's inactive surface. 8
1.3 Phantom shapes. 9
1.4 Appropriate systems for extracting 3D motion parameters. 16

2.1 The pin-hole camera model. 20
2.2 The rotational angles of a camera. 22
2.3 Flow chart of the new algorithm. 26
2.4 Contributing and non-contributing rays. 28
2.5 Approximate method for the anti-aliasing of rays. 29
2.6 Suitable camera positions. 29
2.7 Visibility of a voxel side. 30
2.8 Open-box source images. 32
2.9 Open-box resulting scene views. 33
2.10 Conflicting open-box source images. 34
2.11 Phantom shape results. 35
2.12 Real scene voxel analysis results. 37
2.13 Filtered and thresholded reconstructed scene from novel positions. . . 38

3.1 The poor sampling aspect of voxel spaces. 39
3.2 Example of a 2.5D image. 41
3.3 The 2.75D representation. 41
3.4 Optimum 2.75D steps. 42
3.5 The non-approximated method. 48
3.6 Comparison between VI and the new representation. 51
3.7 Open-box resulting scene views showing the visual hull. 52
3.8 Occlusion and transparency. 54
3.9 Open-box resulting scene views. 56
3.10 Open-box scene views with conflicting cameras. 58
3.11 Colour 2.75D analysis results. 59

4.1 Evidence gathering implementation. 66

4.2 The four basic shapes. 67
4.3 Projecting a ray and testing its intersection with a basic shape. . . 69
4.4 Reducing the searching area. 72
4.5 2.75D voting as considered from a 2D framework. 72
4.6 GA definitions. 75
4.7 GA's cross-over. 76
4.8 Searching for circles. 78
4.9 Non-weighted voting for circles. 79
4.10 Template matching noisy circles. 80
4.11 Template matching noisy circles, varying k. 81
4.12 The gait model of Cunado et al.. 85
4.13 A gait model description. 85

5.1 Example perfect data. 90
5.2 Discretisation effects using perfect data. 91
5.3 Graph of discretisation effects using perfect data. 92
5.4 Discretisation effects using perfect data. 94
5.5 Investigation of discretisation with VI. 94
5.6 Perturbing the camera angle. 96
5.7 Perturbing the focal length. 97
5.8 Image noise response source data. 98
5.9 Blurred images. 99
5.10 Investigation into blurring. 101
5.11 Investigation into blurring looking at a sphere of radius 240. 101
5.12 Images with introduced noise. 102
5.13 Investigation into additive Gaussian noise. 103
5.14 Investigation into Gaussian noise. 104
5.15 Difference of voting measures for a range of Gaussian noise affected images. 104

6.1 Source moving ball ray-traced images. 108
6.2 2D background removed images. 109
6.3 2.75D reconstructed scene. 110
6.4 2.75D reconstructed scene with background removed. 111
6.5 3D reconstructed scene. 111
6.6 3D reconstructed scene with background removed. 112
6.7 Source moving ball ray-traced images. 114
6.8 2D background removed images. 114
6.9 2.75D reconstructed scene with background removed. 115
6.10 3D reconstructed scene with background removed. 115

6.11 Occluded source moving ball ray-traced images. 115
6.12 The errors in the source data. 117
6.13 The calibration board and the ray-traced estimation of it. 119
6.14 The ball-under-gravity source data. 122
6.15 The ball-under-gravity 2D background removed images. 122
6.16 The 2.75D reconstructed scene from original camera angles. 123
6.17 The 2.75D filtered reconstructed scene. 124
6.18 The 2.75D reconstructed scene from novel camera positions. 124
6.19 The 2.75D filtered reconstructed scene from novel camera positions. 125
6.20 The 2.75D filtered background-removed scene. 125
6.21 The 2.75D filtered background-removed scene from novel camera positions. 125
6.22 The 3D reconstructed scene from original camera angles. 126
6.23 The 3D filtered reconstructed scene. 126
6.24 The 3D reconstructed scene from novel camera positions. 127
6.25 The 3D filtered reconstructed scene from novel camera positions. . . 127
6.26 The 3D filtered background-removed scene. 128
6.27 The 3D filtered background-removed scene from novel camera positions. 128
6.28 Results superimposed onto the original images. 131
6.29 The human gait source data. 133
6.30 2D results superimposed onto the original images. 136
6.31 2.75D results superimposed onto the original images. 137
6.32 3D results superimposed onto the original images. 138
6.33 2.75D and 3D predicted gait cycles. 139

A.1 The camera parameters used in the real-world experiments. 158
A.2 The synthetic camera parameters arrangement I. 160
A.3 The synthetic camera parameters arrangement II. 161

B.1 The box-around-arc model. 162
B.2 Source moving ball ray-traced images. 163
B.3 2D background removed images. 163
B.4 The human gait source data. 165
B.5 The human gait 2D background removed images. 166
B.6 The 2.75D reconstructed scene from original camera angles. 166
B.7 The 2.75D filtered background-removed scene. 166
B.8 The 2.75D filtered background-removed scene from novel camera positions. 167
B.9 The 3D reconstructed scene from original camera angles. 167
B.10 The 3D filtered background-removed scene. 167

B.11 The 3D filtered background-removed scene from novel camera positions. 168

D.1 Camera synchronisation device part I. 175
D.2 Camera synchronisation device part II. 176

E.1 DV sampling with the 4:1:1 representation. 178
E.2 DV sampling with the 4:2:0 representation. 179

List of Tables

4.1 Restriction equations on λ for the basic shapes. 70
4.2 The 7 parameters required for the moving ball model. 82
4.3 The 23 parameters required for simple gait recognition. 86

6.1 Extraction results of a moving sphere of unknown radius. 113
6.2 Occluded and unoccluded extraction results of a moving sphere of unknown radius. 116
6.3 Description of the parameters of the model for the ball under the influence of gravity. 121
6.4 The extracted parameters of the model for the ball under the influence of gravity. 129
6.5 The stages for the extraction of the ball in sequence [07217-07239]. . 130
6.6 The 19 parameters required for simple gait recognition. 132
6.7 The extracted parameters of the gait model. 135

A.1 Summary of the relationship between the parameters used in this report and those used by the 'POV-Ray' ray-tracer. 157

B.1 The moving box model. 164
B.2 Extraction results of a moving sphere of unknown radius. 164

Acknowledgements

I would like to take this opportunity to thank my supervisors, Mark Nixon and John Carter, for their continued support and input throughout the period of this research which has been very much appreciated. I am also indebted to the help of my girlfriend, Johannah Hurley, who scribbled green ink all over my poor use of the English language in the first draft of this thesis. I am also very grateful to Mike Grant and my supervisors who scrutinised the technicalities of this thesis, finding (hopefully) even the smallest mistake in the description of the theory. Acknowledgements also go to Jamie Shutler and Richard French for being good sounding-boards for ideas in the early years of the research.

Apologies go to Sam Chalmers whom I often disturbed by working late at night.

Finally, I would like to thank my parents, my brother, my friends and again my girlfriend for keeping me sane during the occasional troublesome and stressful times that have occurred in the course of this research.

I am also grateful for the tenure of an EPSRC studentship, and the equipment support provided by DARPA's 'Human ID at a Distance' programme, contract no. N68171-01-C-9002.

Chapter 1

Introduction

Emerging work has highlighted the potential of human gait as a biometric [60]. Generalised application of gait recognition mandates research into the development of a three-dimensional (3D) analysis system. This system must be able to handle known traits, especially since gait is inherently self occluding: one leg can obscure the other; arms (and apparel) can hide the legs. Further, for recognition to be of application potential, the system is required to be non-invasive, without subject contact. Finally, it is likely that recognition by gait will encounter images of poor quality (such as surveillance videos) suggesting that the capability to handle noise should be considered at the outset.

Having stated the motivation of this research, it is important to stress that the intention was to produce a generic solution, with gait analysis being used simply as an example. Indeed, in this thesis many other examples are described, including the extraction of the acceleration due to gravity acting on a thrown basketball. Therefore, it is not the aim of this thesis to debate the viability of gait as a biometric, but merely to indicate a method in which gait patterns can be described and extracted.

Recently, a multi-view technique has been proposed for 3D moving object analysis. This uses Volume Intersection (VI) separately on each set of frames that are taken at the same instance of time and then tracks the object through the sequence [10]. We shall show that VI appears to bear close similarity to evidence gathering procedures. This is of special interest since it is well known that evidence gathering has performance advantages in respect of the practical factors discussed above, namely the ability to handle noise and occlusion. By performing VI with evidence gathering we will not only confer noise tolerance but also allow the accommodation of image sequences in their entirety, removing the requirement for tracking.

We show how VI can be formulated in grey-scale, thus removing the need for segmentation (a process called 'voxel coloring' [73]). We also show how image sequences can be processed to extract moving 3D objects. However, as will be discovered, three systems are actually proposed, with this grey-scale VI system being the basis of just one. The other systems include one that is based upon the segmented 2D images without the requirement for 3D reconstruction techniques, and also one that is based upon a novel representation that can analyse 3D scenes with optimum fidelity. The complete systems will be contrasted and compared, and their suitability for different scenarios evaluated.

1.1 Gait as a biometric

Today's society increasingly needs reliable methods of identifying an individual. Older systems rely on ownership, such as of documents but these have proved to be too easily forged; futuristic plans involve implanting micro-chip tags, but this would require a great deal of persuasion for social acceptance. Biometrics enable naturally occurring physiological measurements to be the identifiers of the person.

For nearly a century, the assumption that fingerprints are unique has led to it being used as a biometric in order to capture criminals, and more recently, for access control. The other well documented 'fingerprint' is that of DNA which is also considered to be a precise method for identification, except where identical twins are concerned. There is, however, a requirement for more identifiers to be available, especially for forensic use. For example, a thief would now more than likely wear gloves, and obtaining the DNA of a bank-robber would be practically impossible, given the large numbers of people that would have passed through the scene of the crime.

Other biometrics that are being studied include face recognition, retina identification, hand geometry, voice patterns, and even vein patterns, all of which are physical characteristics, and handwriting which is a behavioural trait. These are all applicable for identification at close proximity, and thus can be used as methods of access control. However it would be impossible to identify an individual from a distance, which is just one of the advantages of gait analysis.

It has long been observed that gait can be used to identify a person, and Shakespeare makes several such claims, for example, in *Julius Caesar, ACT I, Scene iii*

> **Casca** Stand close awhile, for here comes one in haste.
>
> **Cassius** 'Tis Cinna; I do know him by his gait;
> He is a friend.

However, for most cases, there are additional clues to the identity of the person such as the sound of the shoes, the outline of the person, and the clothing. Psychologists have devised experiments, commonly using dynamic point-light arrays, that indicate that individuals can indeed be recognised by their gait. Nixon et al. [60] discuss the validity of using gait as a biometric.

1.1.1 Current methods of gait analysis

There are two lines of research regarding automatic gait recognition: model- and statistical-based algorithms. A number of approaches have been reported [60] but the work presented here focuses more on a model-based system [15]. By utilising the robustness of the Velocity Hough Transform developed by Nash et al. [59], Cunado et al. [15] fit an harmonic model to the upper leg. The variance in the harmonics is used for identification, or more precisely the phase spectrum weighted by the magnitude spectrum. In essence this project is a continuation of this work, adding the third dimension so as to allow for the full range of motion.

1.2 Analysis of possible solutions

From the outset, four requirements were defined, having been specified by the future application of extracting gait patterns suitable for use as a biometric:

1. **Non-invasive**: The system must not place any requirements on the dynamic model being sought, nor must it invade the scene that it is witnessing.
2. **Abstract scene capability**: The system must attempt to handle real-world scenes.
3. **3D temporal model**: The only *a priori* information is a mathematical model for the object being sought.
4. **Noise handling**: The system must be able to handle noise.

Note that, for the latter point, there are actually three sources of noise that must be considered: those due to the initial image capture, those due to other objects in the scene, and those inherent in the subject. Since human gait is used as an example of the technique, it is important to note that gait is self-occluding, and thus the model introduces its own form of noise. Noise can also be introduced by the processing system itself, and as will be demonstrated, this can produce significant effects.

As will be seen, these requirements place considerable restraints on the possible systems that can be employed.

1.3 Acquiring 3D data

Although a 3D temporal model is to be used to describe the dynamics of the object that is to be extracted, this does not rule out the possibility of using standard 2D image sequences as the basis of the extraction of the object. A system that extends the model described by Cunado et al. [15] (see section 1.4.1) to include the third dimension can be envisaged. In fact, as will be seen, this research describes one such system that is based solely upon 2D image sequences, although the use of multiple image sequences really implies that it uses a 3D data source.

A decision was made from the outset of this research: since 3D temporal models were being used, for ease of interpretation it would be logical to analyse data in the 3D domain. This thus implied that a means of acquiring 3D data was required.

There are two distinct methods of capturing 3D data namely active and passive. Active methods include using range scanners which project light onto a surface that reflects it into a sensor, thereby triangulating the surface. Current methods to acquire models by this technology have their emphasis on merging noisy range data [17, 21, 31]. Depending on the nature of the range data, terms such as 'height field' or 2.5D are often used; these latter terms cannot be used, for example, in conjunction with the research by Hilton et al. [32] where a study was made of range data that was not presented on a regular 2D grid. The term 2.5D, however, indicates that a single scan is incapable of describing a full 3D world, but provides depth information to a specific position that standard 2D data does not provide. Medical imaging techniques such as ultrasound provide a (noisy) alternative method of acquiring full 3D data, as explored by Carr et al. [11].

Unfortunately, such active methods are unsuitable for our application, falling foul of the requirement for the system to be non-invasive; laser range scanners are also reportedly poor for certain materials and over long distances. The alternative is thus passive sensing, with CCTV cameras being an obvious and commonplace sensing equipment. Indeed, it would be ideal if the proposed gait recognition system could be used with the security CCTV cameras already in place around our environment.

Since humans can infer 3D geometry with monocular vision, it could be concluded that machines should also be able to do so. Cues such as texture [69], shading [43], and focus [20, 72, 91] (a form of active sensing), are all being investigated [3], however, extrapolation must be made using alternative prior information and expectation. This gives rise to a great number of optical illusions that even the highly developed human vision system suffers from. Focusing also requires a more active system where the individual camera properties can be adjusted in real time. The use of expected information thus deems monocular vision using cues as unsuitable due to the requirement to analyse abstract scenes.

Binocular, stereo, vision is another solution for constructing a 3D world, however, the discretisation of the sampling methods has shown that this is currently infeasible. Many, including Blostein and Huang [8], quantify the errors involved, given the correspondences between the two views. Das and Ahuja [20] formulated the error when combining stereo vision with other depth cue methods, including focus, to reduce the errors involved, however, these alternative methods are not suitable, as discussed above. Since the application of biometrics requires a moderate level of accuracy in the measurements, stereo vision is believed not to be a viable source for 3D data. Stereo vision is also prone to optical illusion, such as the increasingly popular 'magic eye' posters where the eyes observe slightly different views that force the perspective inferred. An alternative to stereo is used by Rander et al. [67] and Kanade et al. [38] where 51 cameras were placed around a scene and multiple pairs of these cameras were used to produce a reconstruction that was better than that achievable from a single stereo pair. However, using so many cameras restricts the use to only a handful of applications, and it certainly does not lend itself to a practical surveillance solution.

An increasingly common choice for the construction of 3D data is to use the more generalised description of multiple views, be it with multiple cameras taking views at the same time, or a single camera that is repositioned around a static scene. Beardsley et al. [7] presented a method of analysing video data of an object, taken by a camcorder; the object was static, and thus the problem is actually not monocular. Their method did not require the cameras to be calibrated and the successful extraction of a building was demonstrated. However, the use of a form of correspondence between images indicates that the algorithm will probably not be successful for images other than those containing planar information. Algorithms that use corresponding points and edges in corresponding images often are unable to handle curved objects—examples include robots that can successfully navigate rooms until they encounter a leg or a waste bin. The alternative to these algorithms are those that work on segmented images; these are also known as silhouette-based algorithms.

1.3.1 Silhouette-based algorithms

The advantage of silhouette-based algorithms over stereo algorithms is that no registration is required between views, thus increasing the ability to handle abstract scenes. Having said this, the assumption being made is that the object to be described can be segmented from the background; as will be found in section 1.3.3 and chapter 2, this assumption can be obviated using colour-based algorithms.

Unfortunately, these silhouette-based algorithms require camera calibration information, for example the cameras' positions and orientations in space, which has often been shown to be unnecessary by the point- and line-based algorithms. This information would not be hard to obtain from static or even dynamic surveillance video cameras, due to the availability of camera calibration algorithms [87, 88], and thus this is the one additional piece of *a priori* information that will be utilised.

The root of all the silhouette-based algorithms is Volume Intersection (VI), as illustrated in figure 1.1. In this figure, two segmented images are seen, forming 2D shapes that are the letters 'V' and 'I'. These segmented images were formed from the capture of some 3D shape with the capture process defined by a mathematical

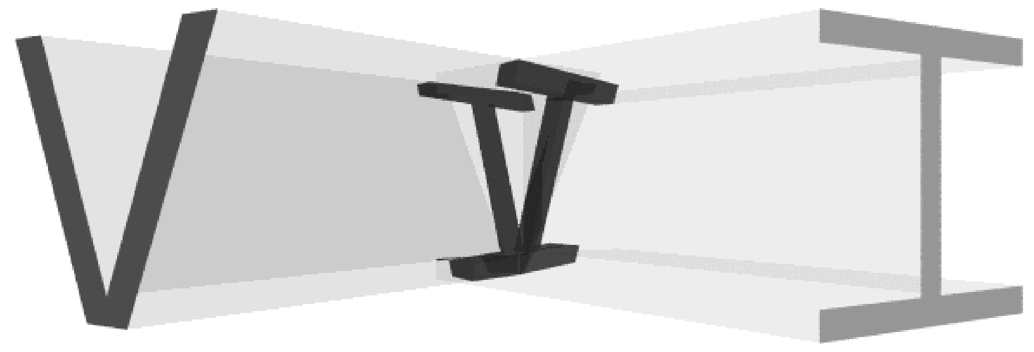


Figure 1.1: Volume Intersection.

mapping. Therefore to calculate the original shape, a reverse mapping is used which projects the segmented images back into the 3D space; it is the intersection of these projecting volumes that contains the original 3D shape.

Much of the early work did not model the camera using a conic or perspective projection (i.e., one in which objects appear smaller as they get further from the camera), but with the cylindrical or orthogonal projection where an object appears the same size no matter how far away it is from the camera. One of the reasons for this was the emphasis on real-time results, which for these early systems resulted in approximations being required. However, the other reason was that many of the early approaches involved projecting the objects as polyhedra, often using rectilinear parallelepipeds as a means of describing the reconstructed objects [54, 41]. Martin and Aggarwal [54] used a three-layered tree structure to represent the object, with the rectilinear parallelepipeds being the 3D equivalent to the pixels in the original image, whereas Kim and Aggarwal [41] used varying sizes of parallelepipeds to represent the object. Kim and Aggarwal [41] also only permitted the use of three views of the object, and these views had to be mutually orthogonal.

These orthogonally projected systems have limited use, since most capture methods use a perspective projection, but using such contour-based methods for perspective projections increases the complexity of the shapes that need to be represented. However, another representation has been suggested that would describe the scene by regular cubes, comparable to the 2D pixel array used for images. Although this was an approximation, the regularity was a great benefit, especially as the results could be described using octrees. Octrees (cf. quadtrees in 2D), represent the object by various sizes of cubes; if a large cube is known to exist solely inside or outside the object, it can remain so, however, a cube that lies on the border is broken down into eight smaller cubes. This is an efficient method for storing the necessary binary data, and the initial approximation is dependent solely on how small the cubes are permitted to become. Hong and Shneier [33] and Noborio [61] demonstrated the use of such a structure, and, similar to the previous work, used edge data to calculate the intersecting cubes, but also used a perspective projection.

Ahuja and Veenstra [4] and Chien and Aggarwal [13] returned to the orthogonal projection and the use of restricted views, but unlike the aforementioned research, no edge data was used, and instead the binary pixel image data was used for the testing, which was in the form of 2D quad-trees or similar. However, it was Potmesil [63] who described the simplest and most versatile method of octree-generation, where perspective cameras from any direction were permitted. This algorithm differed from the others by the fact that the 3D octree was mapped onto the images, instead of the images being projected through the space. To test an octree block for inclusion, the pixels that the block would map to were tested; if all the pixels were foreground or background, the block was allowed or disallowed respectively, whereas if only some of the pixels were foreground, the block was broken down into its eight smaller components which would then be tested individually. Similar but later research by Szeliski [86] differed only in that sub-levels of the octree were only checked after all of the views had confirmed the previous level, therefore offering a possible speed improvement.

The method of perspectively projecting pixels into the 3D space and testing for them individually has many hazards due to the discrete nature of both the images and the 3D grid, as highlighted by the work of Kawato [39, 40]. The research demonstrated the complications caused by the two problems associated with this back-projection method: multiple 'votes' by a single pixel for a close voxel, and no 'votes' by a pixel for a distant voxel. This voting method of 3D structure generation is likened to the Hough Transform, described in sections 1.4.1 & 2.3.

The VI algorithm is a simple yet effective method for the capture of 3D models. With a correct initial segmentation the object is known to exist within the resulting volume. Turntable sequences combined with VI provide a cheap means to produce such models, as demonstrated by Fitzgibbon et al. [24] who assumed no prior knowledge of the system, such as rotation speed, camera position and calibration. However, all of these techniques rely on the object having been segmented from the background, which is not favourable for handling abstract scenes. The binary segmentation also has implications for how well the original object can be described; the object is known to exist within the volume of intersection, but this volume will at best be described by the visual hull of the object.

1.3.2 The visual hull

The concavities that Martin and Aggarwal [54] realised would cause error in the reconstructed object have been investigated and a feature known as the visual hull has been defined by Laurentini [48, 49, 50] and Petitjean [62]. Figure 1.2 shows

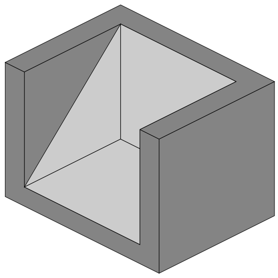


Figure 1.2: The visual hull's inactive surface encloses a region that cannot be determined.

a 3D shape that has a region of uncertainty, represented by the light-grey wedge shape, when it is reconstructed from silhouette images using VI. The content of the grey region is not evident in any view and so will not be reconstructed with fidelity in the 3D model. The visual hull therefore has regions that are part of the original object which are enclosed within active surfaces, and regions that are not, which are enclosed within inactive surfaces.

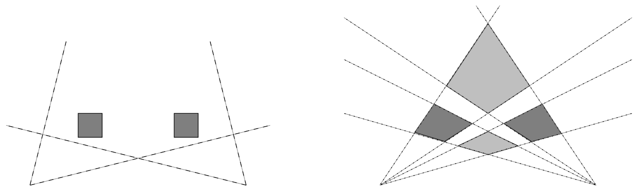
Laurentini [48] actually defined two types of visual hull, depending on the permitted camera locations. For example, in most situations, the camera is at least a small distance away from the object, and therefore it is outside its convex hull. The convex hull of an object is formed by enclosing the object in a surface that is void of concavities; for the example in figure 1.2, the convex hull would actually be a cube. If the viewpoints are restricted to being outside the convex hull, the external visual hull (or just simply the visual hull) is described, producing the response seen in figure 1.2. However, if the viewpoints are permitted to explore positions inside the convex hull, a different hull, named the internal visual hull, is formed. Laurentini [48] defined a set of inequalities relating the described volumes:

$$\mathbf{O} \leq \mathrm{IVH}(\mathbf{O}) \leq \mathrm{VH}(\mathbf{O}) \leq \mathrm{CH}(\mathbf{O}) \tag{1.1}$$

This thus indicates that the convex hull (CH) is the poorest approximation of the object $\mathbf{O}$, followed by the (external) visual hull (VH), and then the internal hull (IVH). Note that even the internal visual hull cannot be guaranteed to describe the original object; for example, a hollow object will always appear solid.

In this thesis, a further hull is described, that being the observed visual hull; since the (external) visual hull is that which is formed, in general, by all possible view points external to the convex hull, a means to describe the hull generated from just the limited number of views was required. In most contexts, it is this hull that is implied, however, in chapter 5 a clearer distinction will be required.

With such restricted viewpoints, a feature of the VI algorithm known as phantom shapes become noticeable. Figure 1.3 illustrates the phantom shapes, showing the



(a) the original scene showing the field of view of two cameras and two subjects.
(b) the reconstructed scene which contains two phantom shapes.

Figure 1.3: Phantom shapes.

original scene containing two squares. As can be seen in figure 1.3b, these squares are resolved to reside within the described intersecting volume, but also two other disconnected blocks are described. Only with more views can these phantom shapes be removed.

1.3.3 Non-segmented 3D reconstruction

There have been few efforts in reconstructing 3D scenes in a similar manner to VI without using segmentation, although in the past five years, this has become a popular research area. As will be seen, this research is mainly being performed concurrently by two research groups, and the overlap between the two approaches is extremely large, and at times the distinction is trivial.

Efforts have been made to produce such grey-scale or colour algorithms based upon VI, stemming from research by Seitz and Dyer [73, 74] entitled 'Voxel Coloring'. Here, construction of the scene was by depth order (starting at the closest) of voxels in the resulting voxel space. Each voxel was determined to be either coloured or transparent, and if coloured it would have occluding properties on later voting hence the need to perform the sweep of the voxel-space in depth order. This research has been continued by Prock and Dyer [65] who explored various approximations that could be made to quicken the voxel coloring algorithm.

Eisert et al. [23] produced a similar result to Seitz and Dyer [73] without the need for depth ordering, by using an iterative approach to the algorithm. The first stage made multiple hypotheses about the shade of each voxel; for each combination of images that could see a voxel, a hypothesis was made, dependent on the difference between the two images' proposed shade for it. Poor hypotheses were removed, for voxels on the surface of the shape relative to a single view, and if all hypotheses were 'removed', the voxel was removed. Also, Culbertson et al. [14] have described the 'generalised voxel coloring' algorithm, which differs from the algorithm of Eisert et al. [23] by the fact that a voxel only has a single hypothesis, determined by the combination of all the views that can see it. Also, a Layered Depth Image (LDI) is used to speed up the process between iterations: LDIs are discussed in section 3.12.

Bonet and Viola [9] have also recently created 'roxels'—responsibility weighted voxels. This reconstruction technique is similar to that of Eisert et al. [23], however, the hypotheses are not binary, thereby permitting semi-transparent and opaque structures to be modelled. Note is also made that occlusion can cause incorrect hypotheses in the first stage of reconstruction where the initial estimate of the scene is made. The new 3D reconstruction algorithm in this thesis has many similarities to the work of Culbertson et al. [14], but introduces the concept of sides of voxels that are visible from different views. This prevents views that oppose each other providing conflicting information since they are sensing rays that could not have originated from the same surface. The commonality between the algorithms outlined above and the one presented in this report was due to the latter being developed independently and at the same time as that of Culbertson et al. [14] and Eisert et al. [23].

Kutulakos and Seitz [45] have directly continued the work of Seitz and Dyer [73, 74] and produced an algorithm termed 'shape by space carving'. The general case is not dissimilar to that of Culbertson et al. [14], but 'hypothesis' erosion (i.e., voxel carving) only occurs at the surface of the predicted object, thus, the algorithm can be more conservative internally, as well as being slow since only a single voxel can be properly carved per iteration. Methods to decrease the processing time were presented, however, these rely on being able to place a depth order on voxels: if this cannot be achieved, six multiple-sweeps should be performed along the different directions along each of the axes, with each only considering the relevant subsets of cameras—in essence it could be concluded that this is providing a similar contribution to the algorithm that voxels with sides could bring.

Non-segmented 3D reconstruction algorithms, as well as having an equivalent visual hull, also suffer from the same limited region of study that VI is burdened with when voxels or octrees are used; all the above non-segmented 3D reconstruction algorithms are voxel-based. If the voxel array is not correctly positioned from the start of processing, the fidelity of the reconstruction will be extremely poor; also, scenes that are effectively limitless, will introduce a great deal of noise due to accidental correspondences between pixels. Slabaugh et al. [81] attempted to solve this 'infinite domain' problem by using an irregular voxel space where the outer voxels were larger than the voxels around the estimated area of interest. In chapter 3, a set of novel algorithms that do not use voxels (thus do not suffer from incorrectly positioned voxel spaces) and do not suffer from the problems of the infinite domain are presented.

1.3.4 Alternative methods

One of the proposals on 3D reconstruction and understanding is the plenoptic function of the scene, introduced by Landy and Movshon [47], which attempts to model the rays of light in a scene. This leads to a 5D space for a static scene, since there are three degrees of freedom associated with the position in space, and two degrees of freedom associated with the angle of the passing ray. Similar terms include the lumigraph [27] and the lightfield [52].

Continuing their research, Seitz and Kutulakos [75] described a plenoptic function of a scene by simply using a variation of their voxel coloring algorithm, thereby reducing the function to just 3D. However, Gortler et al. [27] and Levoy and Hanrahan [52] both used the full plenoptic function definition, but by enclosing the shape within a boundary (as found from the visual hull), they reduced the problem to just four dimensions. This 4D space was subsequently filled by all the information from the viewing cameras, obviating the requirement for correspondence matching. The scene could then be rendered from 'near'-novel views, i.e., from small perturbations of one of the original camera angles. This is currently a computer graphics technique, however, Kutulakos [46] has demonstrated a means of extracting shape information from these structures.

A 'quasi-static' example is the algorithm described by Vedula et al. [90]. This algorithm works in a similar manner to the space-carving and voxel coloring algorithms described above, however, this algorithm uses two adjacent frames from each camera sequence. The motivation behind this is that neighbouring frames will not be that dissimilar and thus ruling out nonsensical results by studying, in essence, flow should improve the reconstruction. The algorithm works using a 6D space, with three dimensions used to indicate the position of the voxel in the first frame and the other three to indicate the relative position of the voxel in the second frame. However, once again, there is the requirement to perform the analysis using depth-ordered pixels, and thus this would not be suitable for the general case.

The last example, which may be worth considering for future work, is that described by Snow et al. [82] where the VI algorithm is reformulated as an energy minimisation problem. The described energy function also deters discontinuities, thus providing noise tolerance to the silhouette images. This added noise tolerance can be of obvious benefit, however, the algorithm can no longer guarantee that a general object lies within the described intersection.

1.4 Dynamic extraction

There is a plethora of algorithms available to extract the dynamic parameters of moving bodies in both the 2D and 3D domains [12], with a large emphasis being placed on describing human motion [25]. There are two distinct categories of algorithms, those that use statistical methods and those that are model-based. Statistical-based methods have currently shown promising results, although by not properly modeling or understanding the scene and the object will undoubtedly lead to problems when a more general solution is required rather than the, normally, restricted scenarios that are used for demonstrations. As already indicated, this work is based upon the use of models which, due to the increase in computing power and descriptive capability, are becoming a more closely studied area of research.

In order to be able to describe the models, it is of course necessary to understand the forms of models that may be encountered. Aggarwal et al. [1, 2] defined a number of categories, including: rigid motion, articulated motion and elastic motion. Many of the motion algorithms [1, 2] require correspondences to be made between successive images. As was emphasised in the previous section, using correspondences places certain assumptions on the objects being viewed; for example, many 2D correspondences will be corrupted by occlusion. The correspondences are generally tracked from frame to frame, and thus they are assumed to move with a fluid motion. The problems of poor correspondences have led to the use of low pass filters and also estimators such as the Kalman filter, so that the object is more successfully tracked [5, 19, 36, 44, 66, 68]. For such tracking algorithms, locating suitable correspondences [78] and selecting the filter are paramount to the success of the system.

The manner in which the object is modelled is very variable, ranging from simple tracking of 2D points [57], to volumes such as that described by McInerney and Terzopoulos [56] who used a representation similar to a 3D snake. One example using VI was reported by Joshi et al. [35] who assumed a rigid 3D model and tracked the intersection through a sequence, refining the fidelity of the intersection on a frame-by-frame basis. Similarly, Bottino et al. [10] described an algorithm where a 3D model was produced using VI and a skeletal model was fitted using a least squares method. There was no explicit filtering performed except that a succeeding frame was seeded from the skeleton described by the previous frame. However, this algorithm was dependent on successful segmentation of the 2D images. The research presented in this thesis is similar in aim, although it is phrased as an evidence gathering technique, describing the object only once the entire sequence (or selected sub-section) is analysed: the emphasis is on noise tolerance since real data is to be analysed.

1.4.1 Evidence gathering: the Hough Transform and Template Matching

Evidence gathering, as the name suggests, is a means by which information is accrued before a decision about any outcome is decided. The most noted algorithm is the Hough Transform (HT) described by Hough [34], who devised it to locate lines in bubble tanks used in nuclear physics experiments. It is a transform as it takes a segmented image and converts it into a parameter space, with this conversion being reversible (although this may be a lengthy process).

The Hough Transform is based upon a voting mechanism, using an accumulator space in which each cell describes the votes accrued to a particular set of parameters; each feature in the segmented image increments all the possible cells whose parameters could have caused the feature to be present. For example, the formulation described by Hough [34] was to locate lines in an image, with a line described by two parameters, namely the gradient, m, and the y-intercept, c. For each feature point in the image, all the possible combinations of m and c were calculated, and the respective cells in the two dimensional (2D) accumulator, or parameter, space were incremented. This resulted in lines that were present in the images voting for a particular combination of parameters more than others, leading to peaks in the accumulator space which could be searched for.

The original formulation, which was brought to the attention of the image processing community by Rosenfeld [70], is hindered by the fact that the parameters are unbounded, and thus the first major improvement to this was made by Duda and Hart [22] who selected a more suitable parametrisation of the problem using a polar coordinate system. Similarly, by changing the types of parameters, other features were found to be extractable; for example circles [42] and even abstract shapes using the Generalised HT (GHT) in which an object is described by a contour model that can be scaled and rotated [6]. Both of these examples require that the edges in the images are extracted by, for example, the Canny edge detector.

Problems, however, arise with the HT regarding the discrete nature of the source data and parameter space. The discrete nature may cause peaks to be spread out over several parameter sets, or a cell with a high vote may actually be caused by many insignificant features; this thus makes the parameter space noisy. Some have attempted to model this into the HT itself, for example the Analytic HT [18], whereas a common method is to perform some form of smoothing filter to the parameter space before the peaks are located. An excellent review paper of the HT, its derivatives, and such post-processing filters was written by Leavers [51].

Bridging the domains, Vaz and Cyganski [89] demonstrated a method in which the GHT is used to locate a 3D shape from a 2D image, by introducing rotation and translation parameters. This is thus one method that could have been selected to analyse 2D images, however, as will be seen, a more generic 3D modelling method was selected. Interestingly, Hamano and Ishii [29] described an algorithm that used the voting structure of the Hough Transform, but where the votes are related to the presence of a 3D point. Multiple images from a moving camera were analysed, and each feature in each image was in essence back projected through the accumulator space; this thus amounts to an alternative description of VI, as will be further examined in chapter 2.

Bridging an alternative domain, there have been many recent analyses of HTs that can be used to extract moving objects in a sequence of frames. It is this research that can be used in place of tracking methods. For example, Nash et al. [59] describe the Velocity HT (VHT), in which a circle moving with constant velocity is sought; this was shown to have increased noise tolerance over that of extracting circles on a frame-by-frame basis and then using regression to calculate the velocity. Similarly, Grant et al. [28] described a Constant Velocity HT (CVHT) which is the temporal extension of a form of the GHT. One of the novel areas of research described in this thesis is a means to analyse abstract 3D objects temporally, but the form of the source data is volumetric rather than surface- or line-based.

Probabilistic HTs

As HTs are used to analyse more complex models, the parameter space increases exponentially in size, thus representing it becomes impractical. Research has been made into probabilistic HTs whose aims are to reduce both the amount of processing required to perform the voting and the size of the resources required to represent the parameter space.

One method is to use a pyramidal or multi-resolutional HT where initially a coarse resolution of parameters is selected, thus enabling an efficient method to produce an estimate. Analysing this estimate, only the ranges of parameters that correspond to the cells with a large number of votes are tested at a higher resolution. This method is used, for example, by Silberberg et al. [79] who extracted a 3D object from multiple orthogonal 2D images using line segments. However, such coarse-to-fine algorithms must be used with caution since, especially for line data, the peaks may be very narrow and thus could easily be missed in the original estimate.

Alternatives to this include the Randomized HT (RHT), developed by Xu et al. [92] and extended by Kälviäinen [37], where it is assumed that if feature pixels in the image are selected at random any significant line will be found since it should have a dominating effect on the accumulator space. Their algorithm also selects two feature pixels at one time and assuming a line exists between the two, hence only one cell in the accumulator space is incremented rather than all the possible lines that could go through one pixel. Also, the accumulator space is represented as a sparse matrix to reduce memory requirements.

Unfortunately the RHT is unlikely to be successful for high resolution and high dimensional parameter spaces, and thus a 'guided' random algorithm has become of great interest to researchers in recent years. Genetic Algorithms (GAs), described in detail in section 4.4, provide a method by which random points in the parameter space can be analysed and if the votes for those points are relatively high, the neighbourhood may also be studied. Yin [94] and Ser et al. [76] used such methods to implement the HT for circles and the GHT respectively; Yamany et al. [93] used GAs with the HT to match a 3D surface model onto 3D object data, although they also limited the range of parameters. Finally, Cunado et al. [15, 16] utilised a GA-based HT to extract 2D gait signatures from people walking orthogonal to a camera for use as a biometric, where a line with oscillatory motion was matched against a thigh in an image sequence. It is upon this that the research presented here is derived from, although the resulting formulation will be seen to be vastly different.

Template Matching

GA-based HTs do not construct the accumulator space, but instead evaluate the fitness of a particular parameter set, i.e., the fitness of an accumulator cell. Stockman and Agrawala [85] and Sklansky [80] have shown that the HT-based algorithms are in fact an efficient description of another algorithm known as Template Matching (TM). This alternative algorithm is actually capable of producing the same accumulator space, although it does this on a cell-by-cell basis, and hence TM is commonly used to evaluate parameters for GA-based HTs; the algorithms, however, should really be termed as GA-TM rather than GA-HT, although this misclassification is due to the direction of the evolution of the research.

Since a cell in the HT accumulator space will sum up to the number of features in the image that indicate the presence of the respective parameters, a single cell can be evaluated with TM by testing all relevant features in the image. For example, when seeking a straight line using the m and c parametrisation, TM will take a cell in the accumulator space and then find all the pixels in the image that lie upon the respective line. However, pixels on the template's line may not actually be classified as being features, and thus many extra comparisons are performed than in the HT algorithm if the entire accumulator space is evaluated.

1.5 Systems

To meet the requirements listed in section 1.2, a number of possible systems was considered as represented in figure 1.4. The systems are all non-invasive, using only video information captured from multiple cameras. The systems also have a good handling ability of abstract scenes since understanding the full scene or even segmenting it is done on a frame-by-frame basis. The systems also share a common method of subject extraction, using evidence gathering which is accredited with good noise handling abilities. The evidence gathering algorithm makes use of temporal 3D templates formed from the subject's mathematical model.

The first system, figure 1.4a, initially extracts any movement by removing the background, the results of which would be passed directly to the evidence gathering

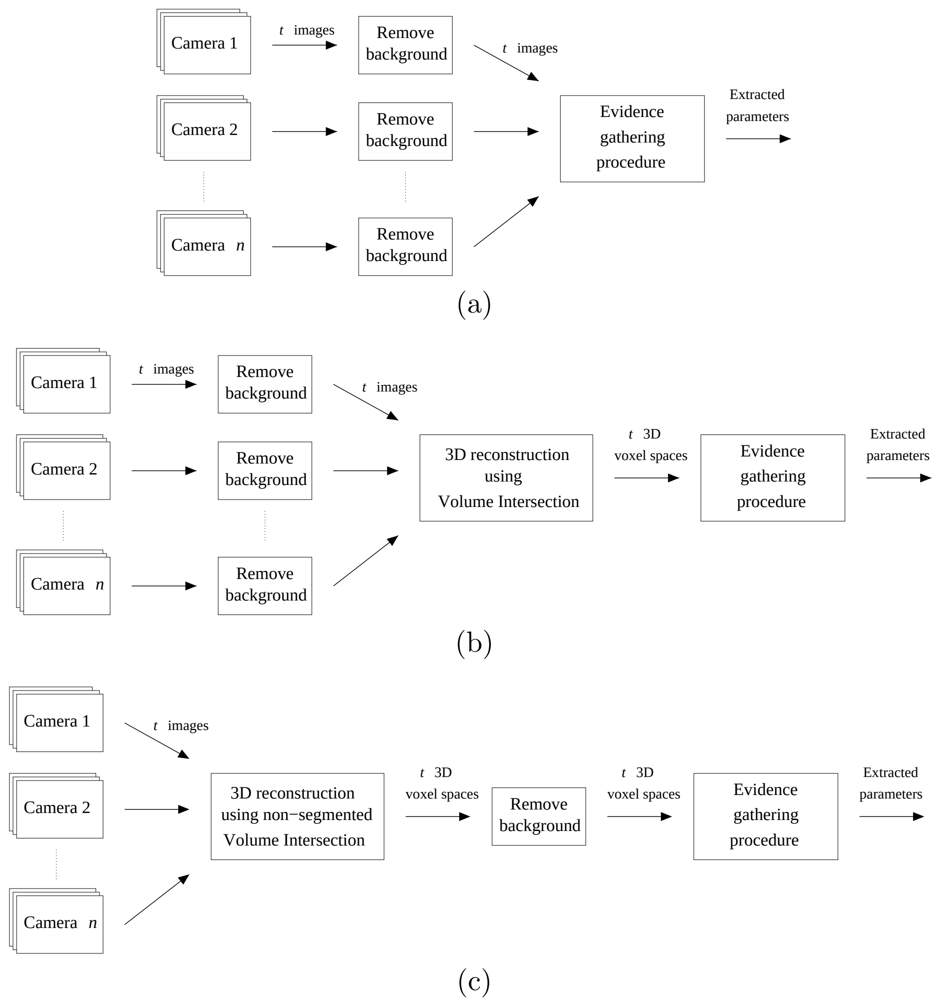


Figure 1.4: Appropriate systems for extracting 3D motion parameters.

algorithm which calculates the best fitness for the dynamic model in the scene. The model is temporal and 3D in nature, but would be mapped onto the 2D static images.

The second, figure 1.4b, would be to extract the moving objects in each of the viewpoints, and then construct a 3D model, using a method such as VI. The evidence gathering procedure would then analyse this 3D data. The results of this method would, however, be similar to those of the first due to the projection of the data being made before the analysis instead of as part of it. There are no advantages over the first implementation, however, it could bring the disadvantage of added noise as a result of the more discrete representation of the data if the voxel representation is used, and if current computing limitations are to be noted.

The third, figure 1.4c, would be to analyse the viewpoints to produce a 3D representation of the world, thus using shade and even colour for correspondence purposes. As with the above methods, the analysis of the scene at this stage is performed on a frame-by-frame basis. The dynamic information in this 3D scene would then be extracted by removal of the background, and the evidence gathering procedure would then be used to parametrise the information. The advantage of this is that understanding the nature of the 3D scene would enable localisation of the subject to be performed more successfully. Since the first two methods yield similar results, both can be claimed to be based upon VI, and as such both would suffer from noise tolerance and also features of the algorithm such as the visual hull and phantom shapes (see figures 1.2 & 1.3). This third method would thus produce cleaner information for the extraction to be performed on.

Two separate approaches have thus been selected for analysis. The first is based upon a 2D representation, where the 3D model is mapped to the 2D space. This is based upon a VI approach (figure 1.4a). The second approach, based upon the figure 1.4c, was split into two distinct representations of the 3D space. These different representations, presented in the following chapters, are a voxel-based grey-scale 3D data representation, and a new 2.75D full colour data representation.

The latter two systems both require the reconstruction of the scene before the dynamic objects can be extracted. The analysis of real world scenes is simplified because no models will be used in the reconstruction; real information will be studied, and producing models for all the information that is likely to be seen in a real world scene is currently impossible. The technique must analyse the information on a frame-by-frame basis; modelling the behaviour of the cloth in trousers is a very difficult process, thus performing the analysis in this way means that all objects can be studied—solid, liquid, elastic etc. Models are applied to extract and describe the various objects in the second stage of analysis of the 3D data.

There will thus be multiple cameras so as to allow the capture of 3D data, and these cameras must also be in synchronisation. This last point is important since conferring information between frames that are not synchronised would reduce the fidelity of the system; the result of the absence of modelling during the 3D generation stage of the algorithm is that there can be no compensation of out-of-phase cameras. Note, however, that the 2D-based algorithm of figure 1.4a would be capable of handling out-of-sync cameras, so long as the relative timings are known. On a practical level, it is believed that, for security applications, obtaining synchronised camera data is not difficult, and may in many cases be the default arrangement for a surveillance system since this allows the data to be more easily switched onto monitors and recording devices.

1.6 Contributions

The grey-scale 3D reconstruction algorithm is novel and is very timely, in view of the current research of contemporaries, with its main differences being not only the use of sides of voxels but also the statistical nature of the scene generation. The basis of the 2.75D representation presented shows a similarity with two other representations that are currently being developed as will be seen in section 3.12. However, unlike these other examples, it is mathematically formalised, and also the non-segmented scene reconstruction algorithm using it is believed to be unique; this gives the ability to examine abstract scenes with the highest fidelity and being able to handle near-infinite fields of view.

All of the research into the extraction of 3D dynamic parameters from 3D dynamic data using evidence gathering in conjunction with constructive solid geometry (CSG) is novel, and thus the three completed systems described are also novel.

1.7 Publications associated with this thesis

During the course of this project, three papers have been written, with the first personally presented at Austin, Texas in April 2000, and the second presented at Amsterdam, Netherlands in July 2000. The third has been accepted for a conference in Padova, Italy in June 2002. The papers are:

K. Sharman, M. Nixon and J. Carter. Non-Invasive 3D Dynamic Object Analysis. *Proceedings of the 4th IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, pages 214–8, 2000.

K. Sharman, M. Nixon and J. Carter. Towards a Marker-Less Human Gait Analysis System. *Proceedings of the XIXth International Society for Photogrammetry and Remote Sensing (ISPRS)*, Vol XXXIII, 2000.

K. Sharman, M. Nixon and J. Carter. Extraction and Description of 3D (Articulated) Moving Objects. *Proceedings of 3D Data Processing Visualization and Transmission (3DPVT)*, pages 664-7, 2002.

1.8 Thesis structure

The new research into the 3D scene capture and 3D dynamic object analysis is described in chapter 2. Chapter 3 introduces the current research into a new representation for the reconstruction of 3D scenes. Chapters 5 & 6 compare and contrast the three systems with analysis of static and dynamic scenes respectively. Finally chapter 7 indicates possible areas of research for future exploration to improve the systems described, and concludes this report.

Chapter 2

Three-dimensional scene reconstruction

2.1 Introduction

All of the systems outlined in chapter 1 require multiple cameras, and thus in order for a 3D model to be produced by correlating their information, it is necessary to explain the basis of the image capture, which performs the reverse process. This chapter presents the matrix transformation of the camera model that was used throughout the research. The new voxel-based reconstruction algorithm is also described, with its roots being shown from the model-less method, Volume Intersection (VI). However, the results presented are for visual analysis only; chapter 5 provides a more thorough analysis of this algorithm having described a suitable manner for doing so in chapter 4.

2.2 Transformations

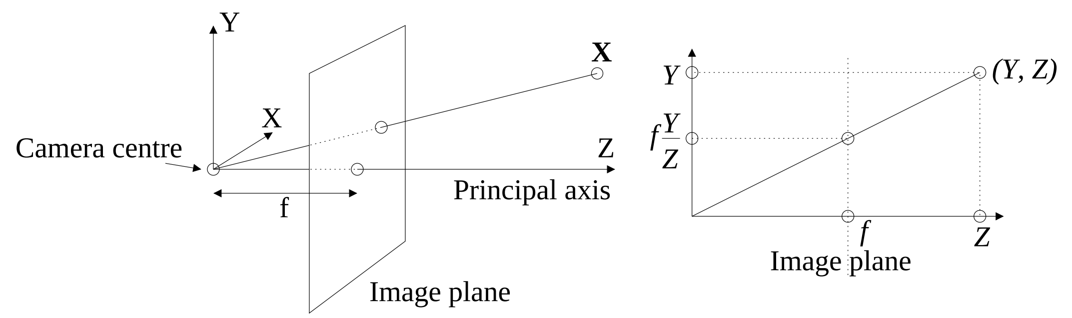
The underlying mathematics used in the capture of images by a camera is the projective transformation; a synonym of this is the principle of collinearity. In this section, this projection is described, starting with a basic model and then developing it into a more descriptive and generic model.

2.2.1 The pin-hole camera model

The basis of the data capture is the pin-hole camera model. Figure 2.1 illustrates how a point in 3D space is mapped onto the 2D image plane. This uses the mathematics of similar triangles, and can be described by:

$$\begin{aligned} x &= f\tfrac{X}{Z} \\ y &= f\tfrac{Y}{Z} \end{aligned} \tag{2.1}$$

where $\mathbf{x} = (x,\ y)$ is the image coordinate, $\mathbf{X} = (X,\ Y,\ Z)$ is the point's 3D coordinate, and f is the effective focal length of the camera. Figure 2.1 also describes



a) 3D representation. b) considering only Y and Z axes.

Figure 2.1: The pin-hole camera model.

the optical centre of the camera, the point through which all rays pass, and the principal axis, which lies perpendicular to the image plane and which passes through the principal point (the image centre).

This equation can be rewritten to show the mapping of the 3D point to the 2D image point, with the use of matrix multiplication only:

$$Z \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2}$$

The increased number of dimensions for the image coordinates enables the factor Z to be removed from the left-hand side of the equation, and instead a proportional factor, λ can be introduced:

$$\lambda \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \tag{2.3}$$

λ can be calculated from the equation yielded by the third row of the image vector, and will later not be equivalent to Z.

2.2.2 Intrinsic parameters

Intrinsic parameters are those that arise as a result of a camera property, for example the focal length f in equation 2.2. They need only be calculated for a particular camera once, assuming that altering the focus does not affect them and that the camera has a fixed focal length. Taking equation 2.3, the camera projection matrix can be formalised, and a 3 $\times$ 3 matrix extracted that will be used to describe all of the intrinsic parameters:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underline{\underline{\mathbf{P}}} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \tag{2.4}$$

where the camera's projection matrix, $\underline{\underline{\mathbf{P}}}$ is:

$$\underline{\underline{\mathbf{P}}} = \underline{\underline{\mathbf{K}}} \left[\underline{\underline{\mathbf{I}}} | \mathbf{0} \right] \tag{2.5}$$

and where $\underline{\underline{\mathbf{I}}}$ is the 3 $\times$ 3 identity matrix, $\mathbf{0}$ is the 3-element null vector, and the matrix $\underline{\underline{\mathbf{K}}}$, commonly known as the camera calibration or intrinsic factor matrix, is given by:

$$\underline{\underline{\mathbf{K}}} = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \tag{2.6}$$

This matrix will now be generalised further by adding another intrinsic parameter, the principal point offset. For all cameras it is very likely that the lens will not be completely parallel and central to the sensing array. Thus an offset would be present for every 2D point, yielding:

$$\underline{\underline{\mathbf{K}}} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \tag{2.7}$$

where $(p_x,\ p_y)$ is the camera's principal point, defined as an offset in the 2D image pixel array.

Up to now, it is assumed that the pixels are square, however, for CCD cameras and for scanning-beam cameras this is not necessarily the case. Adding a scale factor allows rectangular pixels to be described, but for certain types of camera, the pixels may actually be parallelograms, and thus a shearing factor should also be incorporated:

$$\underline{\underline{\mathbf{K}}} = \begin{bmatrix} f & \sigma f & p_x \\ & \kappa f & p_y \\ & & 1 \end{bmatrix} \tag{2.8}$$

where σ is a measure of skewness, and κ is a measure of the aspect ratio of the pixels. This can be simplified by allowing the camera to have two focal lengths, one along each of the axes of the image plane:

$$\underline{\underline{\mathbf{K}}} = \begin{bmatrix} f_x & \sigma f_x & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix} \tag{2.9}$$

For CCD cameras, skewness is rarely an issue, and thus σ can be ignored.

2.2.3 Extrinsic parameters

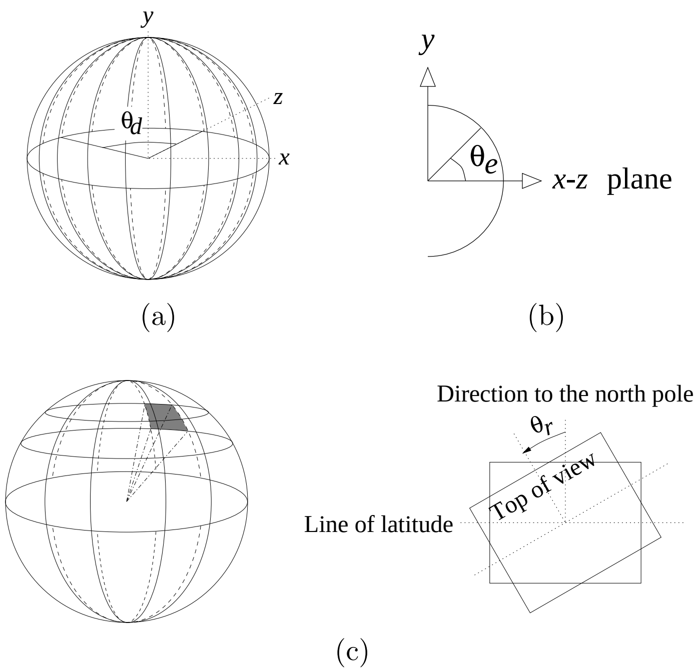
The model so far has not allowed the camera to be placed at arbitrary points in the world—it has been viewing the world from its own camera, or local, geometry. Extrinsic parameters enable a global geometry to be realised by introducing the 3D location for the camera's centre, (x_0, y_0, z_0), and rotation factors for the image plane in the 3D space, $(\theta_d, \theta_e, \theta_r)$, with the latter being described in figure 2.2.

These can be incorporated into equation 2.5, thus:

$$\underline{\underline{\mathbf{P}}} = \underline{\underline{\mathbf{K}}}\,\underline{\underline{\mathbf{R}}}\left[\underline{\underline{\mathbf{I}}}\,|-\mathbf{T}\right] \tag{2.10}$$

where $\mathbf{T} = [x_0, y_0, z_0]^\mathsf{T}$ is the translation vector, and $\underline{\underline{\mathbf{R}}}$ is the 3×3 rotation matrix given by multiplication of the individual rotation matrices:

$$\underline{\underline{\mathbf{R}}} = \underline{\underline{\mathbf{R_d}}}\,\underline{\underline{\mathbf{R_e}}}\,\underline{\underline{\mathbf{R_r}}} \tag{2.11}$$



a) the angle of direction equivalent to longitude, b) the angle of elevation, equivalent to latitude, c) the angle of camera rotation.

Figure 2.2: The rotational angles of a camera.

2.2.4 The complete projection

There are further camera parameters that cannot be represented using this matrix mapping, including radial distortion where an image point is offset by an amount proportional to its distance from the principal point. However, the camera's projection matrix $\underline{\underline{\mathbf{P}}}$, containing 6 extrinsic parameters and 5 intrinsic parameters, can be summarised as:

$$\underline{\underline{\mathbf{P}}} = \underline{\underline{\mathbf{K}}}\,\underline{\underline{\mathbf{R}}}\left[\underline{\underline{\mathbf{I}}}| - \mathbf{T}\right]\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \tag{2.12}$$

where:

$$\underline{\underline{\mathbf{K}}} = \begin{bmatrix} f_x & \sigma f_x & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix} \tag{2.13}$$

$$\underline{\underline{\mathbf{R}}} = \underline{\underline{\mathbf{R_d}}}\,\underline{\underline{\mathbf{R_e}}}\,\underline{\underline{\mathbf{R_r}}} \tag{2.14}$$

and $\mathbf{T} = [x_0 \;\; y_0 \;\; z_0]^{\mathsf{T}}$.

The matrix $\underline{\underline{\mathbf{P}}}$ is a projective transformation and thus has the properties that it is an invertible mapping, and straight lines are mapped to other straight lines.

In this report, the projection matrix $\underline{\underline{\mathbf{P}}}$ is often extended to be a 4×4 square matrix in order that the inverse can be easily evaluated and other manipulations performed. The only effect this has is that image pixel coordinates must be represented by a 4D vector, with the additional element having the value of 1. Note, however, that even after manipulation, the resulting matrix can still be represented by a 4×3 matrix.

Finally, the local 'intrinsic' 3D geometry representation of a camera is defined to be similar to a camera's local geometry, but the 3D space is also affected by the intrinsic parameters, i.e., the entire projection matrix $\underline{\underline{\mathbf{P}}}$ is used for the manipulation, not just the extrinsic part, $\underline{\underline{\mathbf{R}}}\left[\underline{\underline{\mathbf{I}}}| - \mathbf{T}\right]$. Thus with such a representation, the image plane is located at a distance of 1 in the z-axis, and is a regular grid comprised of square pixels. Hence to obtain a pixel's coordinates from a 3D point in such a representation, all that is required is for the x and y coordinates to be divided by the z coordinate.

These and further mathematical relationships regarding the projective, affine and other transformations may be found in Hartley and Zisserman [30].

2.3 The 3D Hough Transform and VI

The 3D Hough Transform (3D HT), like VI, operates on silhouette images, where the segmentation of the object has already been performed. It is thus only feasible to act on images in which the object to be described can be separated from others in the scene, and hence it is common to look only at one object in a scene, with a background that can easily be removed. Both algorithms also require that the cameras are calibrated prior to the combination of the information.

In essence, both methods project the source images through a 3D space. Where the projections intersect with each other, a shape is formed. This can be seen in figure 1.1 where one view sees the letter 'V', and the other the letter 'I'. By projecting these away from the cameras through the 3D space, an intersection is produced. The 3D HT represents this intersection and also the projecting regions in a 3D space represented by volume elements called voxels (cf. picture elements are pixels). VI can use a voxel space, or additionally use an octree, where the space is described by a tree structure with up to 8 possible branch nodes; the latter is not of concern here.

However, projecting the image through the space is problematic for two reasons. The first is that voxels 'close' to a camera will receive many more votes than those far away because many more pixels will be able to vote for them. For example, a voxel directly in front of a camera will receive the number of votes equal to the number of pixels in the foreground of the respective image. This factor, though, does not affect VI, only the 3D HT. The second is that as voxels get further away, the voting will get sparse. More distant voxels appear smaller in the image, and may actually be smaller than a source pixel. When this occurs, pixels need to vote for more than one voxel at a given depth.

To alleviate these problems, the mapping is more simply performed in reverse, i.e., mapping the voxel space onto the images. This introduces its own similar difficulty, when a voxel maps onto several pixels. If this occurs, an average of the area in the image that the voxel maps to must be calculated.

The 3D HT and VI constructs the object within an accumulator of voxels in virtual space. This voxel space must be positioned, rotated and scaled as required so that it covers the entire region of interest. Then, for each voxel in the voxel space, mappings are made to all of the source images, using equation 2.4, taking a corner of the voxel as its 3D point in space. These mappings will thus indicate pixels on the images that correspond to that position in space. In VI, the voxel is set to the value of 1 only if all of these pixels indicate that the point in space is 1. In the 3D HT, a voting system is used, and thus the voxel accrues the value of the number of views that have pixels that indicate that it is valid. This can be seen in the pseudo-code listings 2.1 & 2.2.

```
Initialise accumulator array with the value of 0
For each voxel (accumulator cell),
   For each view
      Map voxel onto an image pixel
      If it maps to a valid pixel
         If image pixel is set
            increment accumulator cell
```

Listing 2.1: 3D HT pseudo-code.

```
Define an accumulator array
For each voxel (accumulator cell),
   transparent = false
   For each view
      Map voxel onto an image pixel
      If it maps to a valid pixel
         If image pixel is not set
            transparent = true
   accumulator cell = NOT transparent
```

Listing 2.2: VI pseudo-code.

If the 3D HT algorithm also recorded the maximum possible number of votes a particular voxel could achieve, i.e., increment another accumulator array without testing to see whether the pixels are set, then a ratio of the number of votes to the maximum number of votes possible could be calculated for each voxel. If the results of this ratio were thresholded at the value of 1, they would yield the same binary results as VI. The accumulator cell nature has been used by many without noting its correspondence with the HT; for example, Snow et al. [82] describe, for use as a comparison, a heuristic algorithm with improved noise tolerance over the standard VI.

Concavities cannot be resolved by either method, as they lead to a visual hull—volumes within the object that cannot be observed. Evidently both are constrained to indicate existence only, although the Hough Transform has marginal noise tolerance since the resulting space need not be thresholded at the ratio value of 1.

2.4 The new voxel-based algorithm

2.4.1 The hypothesis and overview

To increase descriptive capability, grey scale can be incorporated into a new voting process. The motivation behind this is that as humans can see into the visual hull and interpret the shape within by using colour or just grey scale information, then, by introducing grey scale, a machine should be able to do similarly. However, for a model-less algorithm, such as will be described, *a priori* knowledge is not

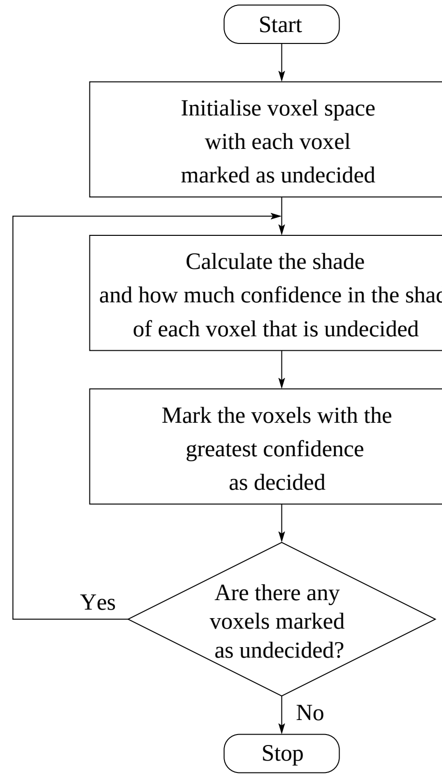


Figure 2.3: Flow chart of the new algorithm.

introduced, thus unexpected results can be produced. For example walls which we assume to be flat, may not actually be predicted to be flat. It is due to the lack of information that the algorithms predict the unexpected—with more cameras such errors could be removed. However, this lack of *a priori* knowledge is of advantage as the algorithms would not suffer from optical illusions which foul our own vision system.

In VI, 2D points are projected as lines through a 3D space, and it is their intersection that describes the visual hull which contains the object. This is thus an attempt to invert the process of the original data capture. Our hypothesis is that rays from a point in 3D space will be of a similar level of intensity. The assumption being made is that the surfaces are Lambertian; these surfaces reflect light with equal intensity in all directions, and thus appear equally bright from all directions. Hence they only exhibit diffuse reflection and do not produce any mirror or specular effects. This also implies that translucent non-diffuse materials are assumed not to be present in the scene.

Figure 2.3 illustrates the iterative nature of this algorithm, with each successive iteration being affected by those preceeding it due to the effects of occlusion, as will be seen in section 2.4.4. The confidence measure is described in the following section, and the manner by which voxels are selected is described in section 2.4.3.

2.4.2 *The confidence measure*

For each voxel, information regarding its shade, and the confidence in the shade is calculated from the rays that pass through it, based upon a statistical measure m, where:

$$m = \frac{\sigma^2 + k_1}{n + k_2} \tag{2.15}$$

where σ^2 is the variance of the grey level of the n contributing rays. k_1 and k_2 adjust the weight of voting for more views (section 2.4.7 describes these and other constants in more detail). This measure, based upon the variance, is suited to the reduction of additive noise, and increases as confidence decreases.

An initial estimate of the scene can be acquired by calculating this measure for every voxel in the space. However, an improved estimate can be made by refining this result by multiple passes, taking into account occlusion by other voxels.

2.4.3 *Voxel selection*

With each iteration, a selection of voxels is deemed to be suitable to fit the observed data. The voxels selected are those chosen from the pool of voxels that have not previously been selected, and are picked due to their high confidence value (a low value of m). Selecting just the voxels with the highest confidence level would be time consuming, since each pass would yield perhaps a single new voxel, thus a band of levels is permitted, given by the equation:

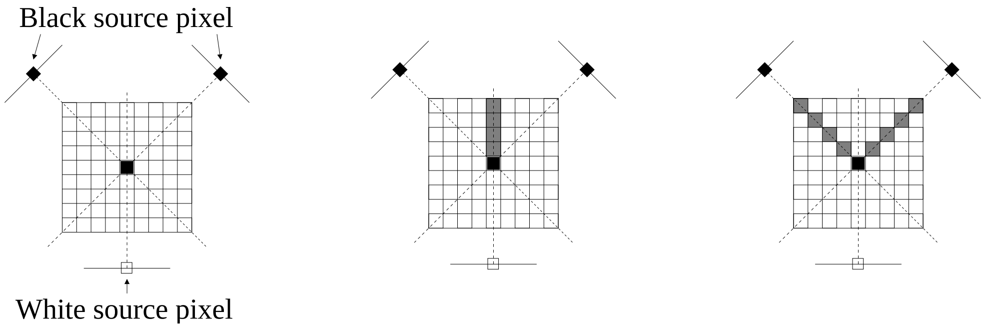
$$m_{min} \leq m < m_{min} k_{mult} + k_{add} \tag{2.16}$$

where k_{mult} and k_{add} are predefined constants. Hence the scene is improved until all of the voxels have either been selected or been concluded as being transparent.

2.4.4 *Shade, occlusions and transparencies*

The actual shade assigned to such a selected voxel is not the mean of the n contributing rays, but is calculated by averaging the values of pixels that lie closest to the mean of all of the rays. This is performed so that rays that clearly do not contribute the same information about the voxel do not influence its shade. Thus if a ray has a shade whose value is greater than a distance of k_{reject} from the mean, it is deemed to be too different and must not contribute to the final value. For example, in figure 2.4a, three camera sources are present, with two indicating a black pixel and the other a white pixel. The camera with the white pixel will not contribute to the resulting colour of the voxel, and thus a black voxel is produced.

For rays that are not believed to contribute to a voxel, it is then predicted that another voxel must lie between the respective source view and the selected voxel, in order for it to have acquired the shade indicated. Therefore the ray should not attempt to project beyond this voxel—it is thus occluding the ray (figure 2.4b).



(a) (b) (c)

(a) 3 sources, producing 2 contributing rays and 1 non-contributing ray to a voxel.
(b) Voxels occluded to the non-contributing source.
(c) Transparent voxels.

Figure 2.4: Contributing and non-contributing rays.

For rays that contributed to a voxel's shade, it is predicted that all voxels between the respective source views and the selected voxel are transparent. The source of such rays is then no longer able to produce any further contribution to the reconstruction. In order to be fair, voxels are selected in a batch, and then processed so that there is no weighting to the first voxel found in the space.

The consequence of these rules is that for each image a depth map is produced, in order that the information regarding how far a pixel can project into the space is retained. Such maps are initialised with a large depth value for each pixel, and are gradually eroded.

2.4.5 Anti-aliasing blocks

As previously described in the 3D HT algorithm, for a voxel that lies close to a view, there may be several pixels that correspond to it. The projected ray must therefore be constructed from the average of the possible contributing pixels in order that the voxel does not under-sample the source image, or conversely, the image oversample or alias the voxel space. For speed, an approximate solution has been implemented where a bounding rectangle of the voxel, taking into account all of the voxel's sides, is found, and the pixels contained within are averaged. This can be seen in figure 2.5 where the dark grey pixels represent the true region that should be considered for the voxel, and the light grey pixels represent the approximated rectangle. Use of bounding rectangles is discussed by Steinbach et al. [83, 84] who use the colour reconstruction algorithm of Eisert et al. [23].

If rays from such blocks are occluded, then all of their respective depth maps must be affected accordingly. Similarly, if a pixel in such a block has a maximum projected depth that does not permit it to detail information regarding a voxel, it must not be included in the production of the anti-aliased ray to that voxel.

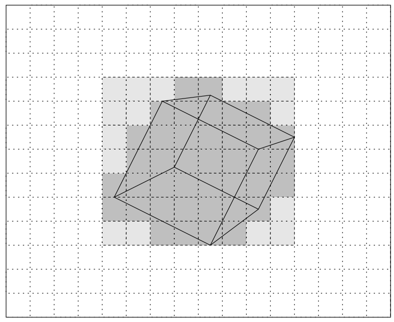
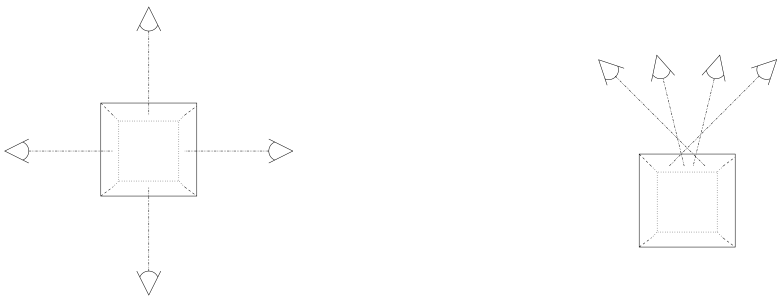


Figure 2.5: Approximate method for the anti-aliasing of rays.



(a) 4 cameras looking in the same plane from these angles will not correlate any information.

(b) 4 cameras looking down at the scene, however, will correlate information.

Figure 2.6: Suitable camera positions.

The maximum projected depth limit for such pixels will not be affected because they will of course be less than the distance to the respective voxel. Note that the approximate bounding rectangle will produce a degraded result as pixels will be incorrectly associated with depths, however, this will only be noticeable at sharp boundary points since otherwise the depths will be similar.

2.4.6 Voxel sides

Returning to the hypothesis, it becomes apparent that rays from opposite directions falling onto a voxel can neither vote against each other nor vote with each other. This is as a result of the voxel being of a finite size—if the voxel was a singular point in space then this would not be an issue. The hypothesis thus also dictates suitable camera positions—placing four cameras equispaced around a plane containing the object would not lead to any correlation between views; however, if all four cameras were to look down onto the object, then a correlation can be made as all views would be able to correlate information regarding the tops of voxels. This can be seen in figure 2.6.

Hence, during the search, voxels are allocated six sides, and thus rays must actually fall onto a side from the correct direction in order for them to contribute

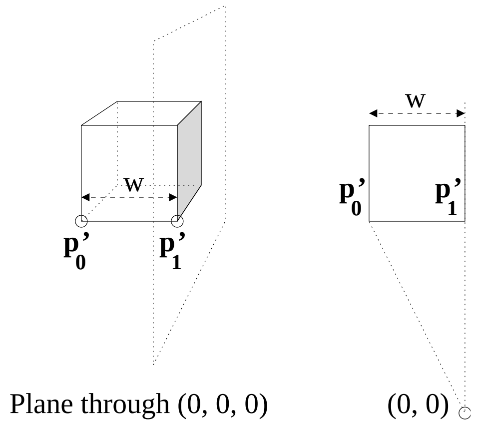


Figure 2.7: Visibility of a voxel side.

to it. The final result, however, for simplicity of later analysis, allocates the voxel with the shade and confidence of the side that has greatest confidence.

To calculate the rays that fall onto, for example, the right-hand side of the voxel (x_0, y_0, z_0), another corner of the voxel $(x_1, y_1, z_1) = (x_0 + w, y_0, z_0)$ is considered, where w is the width of the voxel. Converting both of these to the respective camera's local geometry, using only the extrinsic parameters in the camera model, yields the 3D points $\mathbf{p_0'} = (x_0', y_0', z_0')$ and $\mathbf{p_1'} = (x_1', y_1', z_1')$.

Figure 2.7 illustrates the scenario of a side being at the limit of visibility of a view. In order for the shaded right-hand side of the voxel to be in this state, the face must be directed towards the camera centre, which lies at the origin. Using Pythagoras at this limit yields:

$$|\mathbf{p_1'}|^2 + w^2 = |\mathbf{p_0'}|^2 \tag{2.17}$$

and adding the correct inequality, the condition for the testing of the right-hand side is:

$$|\mathbf{p_1'}|^2 + w^2 < |\mathbf{p_0'}|^2 \tag{2.18}$$

Similarly, the inequality for the testing of the left-hand side is:

$$|\mathbf{p_1'}|^2 > |\mathbf{p_0}'|^2 + w^2 \tag{2.19}$$

2.4.7 Constants

There are five constants that can be adjusted in this algorithm, namely k_1 and k_2, which are used for calculating the confidence (defined in section 2.4.2), k_{add} and k_{mult}, which are used for the selection stepping (defined in section 2.4.3), and k_{reject} (defined in section 2.4.4).

k_1 is required in the confidence measure otherwise voxels that are only in the view of one camera will have a measure of 0 as the variance of one value is 0. The constant thus allows voxels that are in view by more cameras to have a more equal possibility of being selected even though their variance may be higher. Hence it encourages correlation between views instead of favouring the object to be described in regions where there is no correspondence due to only a single camera being able to view that region. It also aids the making of decisions regarding whether voxels must be occluding other views. Voxels in which a number of views produce the same prediction, would have a higher confidence (lower measure) than those in which fewer views can see. Although this effect is required, being able to affect the balance is useful, hence the constant k_2. For example, if all the possible rays passing through a voxel saw similar shades, then σ^2 would be small. In this case, k_2 would be used to reduce the distinction in the denominator of the confidence measure, between the case when, say, there were five rays and when there were four rays passing through the voxel; it would be unfair to greatly favour the case of five rays over that of four rays, given that all of the rays that are being combined indicate a similar shade.

Ideally the constant k_{reject} should be very small, however, this assumes that the cameras are calibrated in terms of colour, and that the scene does not exhibit any specular effects. This constant thus introduces a level of noise tolerance.

The stepping constants k_{add} and k_{mult} must be selected by the balance of three factors. First, larger values encourage a solution to be found more quickly. Second, smaller values encourage a more accurate solution, and third, too small values produce shell like results as only the surfaces of objects produce correspondences. Their values are also dependent on the range of the statistical measure in equation 2.15 and thus also the range of the pixel values.

The values of $k_1 = 800$, $k_2 = 2$, $k_{reject} = 5$, $k_{add} = 1.00$ and $k_{mult} = 1.02$ are used throughout this thesis, having been found by trial and error to be the most suitable for all of the data tested. For more than three cameras these values would need to be amended. k_1 is relatively large as it must be significant compared to the variance of the colours. The worst case value of σ^2 is 16256 (127.5^2); a standard deviation of $\sqrt{800} = 28$ is also approximately 10% of the colour scale. For the situations encountered, k_2 is comparable to the number of views used in these trials, and k_{reject} has been selected for estimating the required noise tolerance in real world data—k_{reject} should ideally be 0 for synthesised data, although a larger value would be preferred due to anti-aliasing effects. Finally, it would be appropriate to select a range of values depending on the magnitude of the best confidence value, i.e., $k_{add} = 0$ and $k_{mult} > 1$. However, a problem exists if the best confidence is 0, as was common with some initial test data, as the range is then all 0. Hence k_{add}

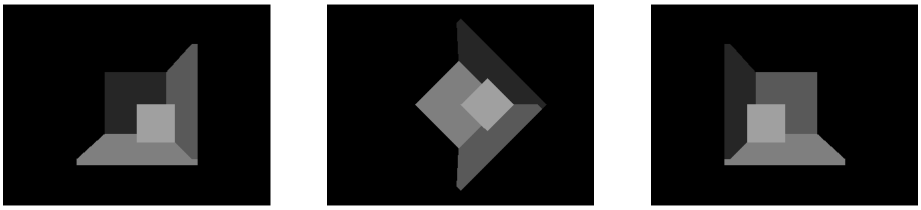


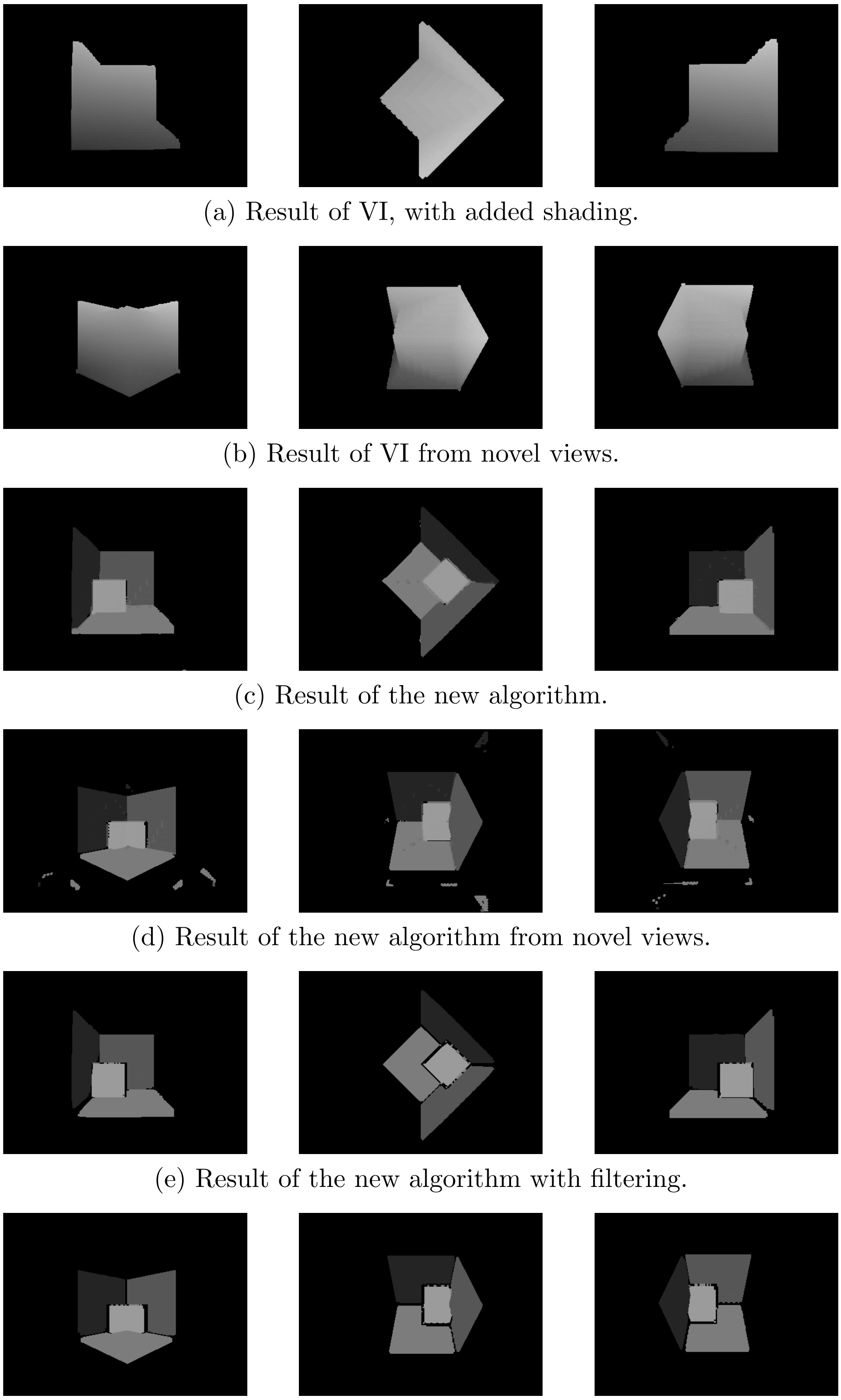
Figure 2.8: Open-box source images.

must also be non-zero. The resulting confidence values for all of the experiments performed frequently were within the range of $(0, 500)$.

2.5 Three dimensional reconstruction results

2.5.1 The visual hull

The 3D reconstruction algorithm, being influenced by the brightness of the pixels from the various views, enables the visual hull problem of the silhouette-based VI algorithm to be overcome. In the example in figure 2.8, the foreground and background can be easily segmented, thus the VI algorithm can be applied. Figures 2.9a & 2.9b show the result of VI, and figure 2.9c demonstrates that this new algorithm, by using shade, can see into the concavity, highlighted by the fact that the small box in the concealed corner is visible. However, from novel views, figure 2.9d, it can be seen that a few other voxels were deemed to be present outside the shape. These are present in the images in figure 2.9c but appear to be correctly positioned. However, they are located in regions where not all of the views can see them, and thus confidence in their presence is lower. Figure 2.9d also indicates protrusions into the inner box; these are as a result of a visual hull that is present for like-coloured objects. From the original images, there was no information that would enable the predication that the walls of the open-box were flat, and this colour visual hull has thus caused these walls to be estimated in incorrect regions. Filtering the confidence levels of the voxels that are to be plotted yields figures 2.9e & 2.9f, with the latter no longer showing the extraneous voxels visible in figure 2.9d. However, it is apparent in these figures that the voxels on the boundaries of the different shaded regions have also been removed. This is because they were formed from 'anti-aliased' rays as described above, and over larger discontinuities, their shade will be affected by even small differences in positioning. For such voxels, the variance in the rays is thus relatively high, and thresholding at a confidence level may also remove these. The confidence of such voxels can be greatly affected by adjusting the constants k_1 and k_2 in the confidence measure equation.



(a) Result of VI, with added shading.

(b) Result of VI from novel views.

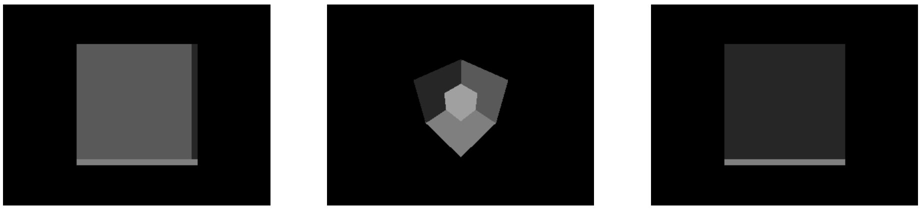
(c) Result of the new algorithm.

(d) Result of the new algorithm from novel views.

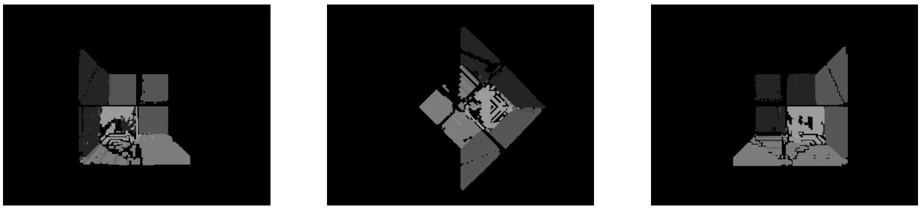
(e) Result of the new algorithm with filtering.

(f) Result of the new algorithm with filtering from novel views.

Figure 2.9: Open-box resulting scene views.



(a) Three more source views of the object.



(b) Result of the new algorithm from the original views.

(c) Result of the new algorithm from the new views

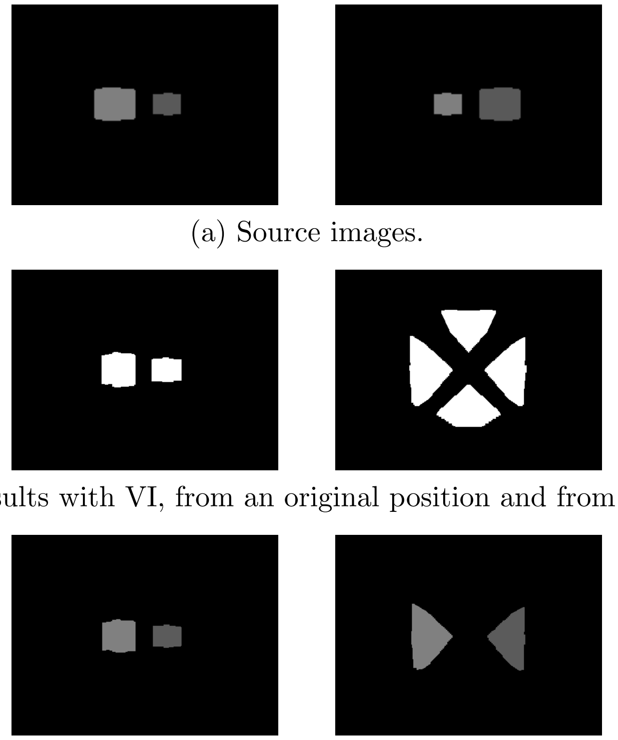
Figure 2.10: Additional open-box source images, which provide conflicting information, and resulting scene views.

2.5.2 Conflicting images

Using VI, as the number of views increases the fidelity of the visual hull also increases. Laurentini [50] discusses the maximum number of images required to reconstruct a visual hull. The same, however, is not true of this grey-scale algorithm, which is also noted for the algorithm by Bonet and Viola [9]. In the above example, only three views were used to reconstruct the volume; in figures 2.10b & 2.10c the effect of adding the conflicting views of figure 2.10a can be seen. The conflicting views are those that cannot see into the shape, and thus do not predict the presence of the box, nor other sides. As still more views are added, the result tends to favour the information from the direction that most cameras face, as would be expected due to the weighting towards more cameras and only selecting a single voxel side.

2.5.3 Phantom shapes

The other well documented feature of VI is the presence of phantom shapes, as illustrated in figure 1.3. Using different shades for the two cubes in that illustration, the results from VI and from the new algorithm can be seen in figure 2.11. As can be seen, the new algorithm has not suffered from the phantom shapes, however, it has not correctly realised that the shapes are box cubes, although this is due to a lack of information in the images. For the case where the two cubes are the same shade, the



(a) Source images.

(b) Results with VI, from an original position and from above.

(c) Result of the new algorithm, from an original position and from above.

Figure 2.11: Phantom shape results.

new algorithm also produces phantom shapes, but there is no information from the input data regarding the presence or lack of presence of such features. There is also an equivalent source of ambiguity in grey-scale scene reconstruction, as described by Seitz and Dyer [73], which can only be removed by increasing the number of views.

2.5.4 Real data

The previous examples are artificial, and as such the algorithms had to be informed that black was the background; without this *a priori* information, interpreting the results is made more difficult. Figure 2.12a demonstrates a real scene where there is no such problem, and thus this shows the strength of the new algorithm as a whole since there is now no segmentation performed. The sequence is taken from an inside data capture session, more fully described in section 6.3, in which a basketball can be seen to have been thrown across a room. Studying the first and last images of figure 2.12b, the voxel space can be seen to be rectangular in shape as its limits are clearly defined within these images. Note that the right-hand door is not properly rendered in the first view of figure 2.12b due to the fact that only this view can see it and thus there is ambiguity over its reconstruction; the radiator, window and left-hand door have been correctly reproduced as they are visible in all three views.

Also apparent in figure 2.12b is the large fluctuation in the shade of the background wall. The cause of this is the selection of the k_{reject} constant in the algorithm. Since the cameras were not calibrated for colour, and fluorescent lighting was used, the shading on the views of the wall differs slightly, and in fact the latter effect actually introduces bands of intensity moving along the wall. Although voxels are on the whole composed from anti-aliased rays, and thus averaging is performed, with these differences the constant k_{reject} must be set to a lenient, i.e., high, value.

The filtering used in figure 2.12c is the background removal filter, which is described in chapter 4. The ball can be seen to have been defined in the voxel space, although there are many anomalous voxels that obscure it from certain angles. In figure 2.13 a further threshold filter is applied to the shade, thus the ball is more clearly visible.

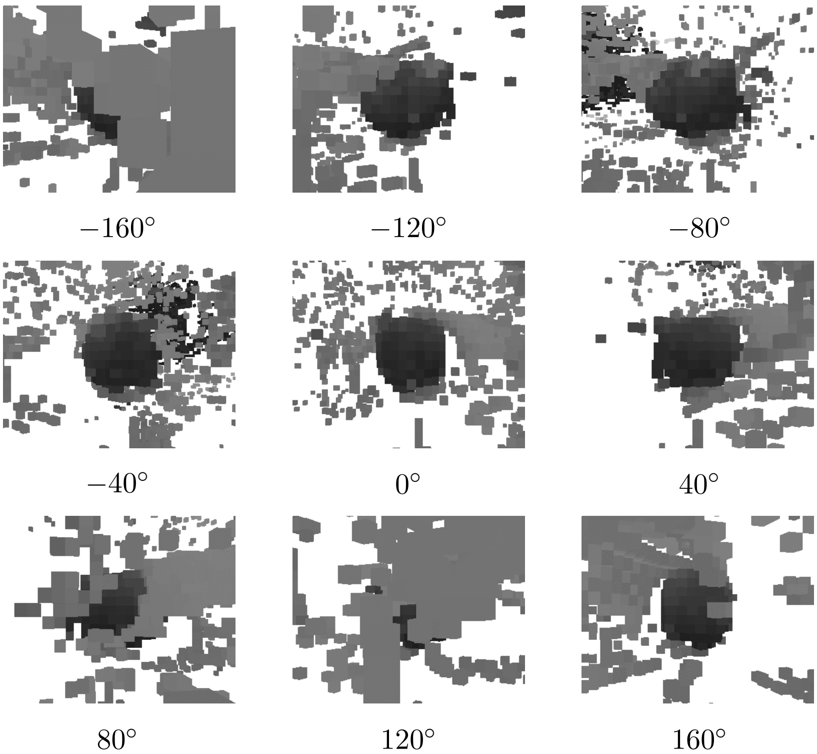
2.6 Conclusion

In this chapter, the algorithm known as Volume Intersection (VI) has been shown to be similar in nature to the 3D Hough Transform (3D HT). The latter is an evidence gathering algorithm, and thus accrues optimal noise performance—a necessity for the study of real world information. The steps to develop the 3D HT into a grey scale 3D reconstruction algorithm have been demonstrated. This new reconstruction algorithm allows arbitrary scenes to be described without the need for segmentation. By removing the segmentation stage, effects such as the visual hull have been removed for images with shading information; segmentation, however, can instead be performed in the 3D domain if required.

The new algorithm is one of the stages in a possible system whose goal is to extract 3D dynamic models from arbitrary scenes. Chapter 4 continues with the second stage and chapter 5 performs analysis of the results of this algorithm, using the method discussed during the development of the second stage. However, the following chapter presents a novel and improved representation for the reconstructed scenes.

(a) The three source images of a basket ball that has been thrown across the scene.

(b) The unfiltered reconstructed scene from the same positions.



(c) Filtered reconstructed scene from novel positions around the ball.

Figure 2.12: Real scene voxel analysis results.

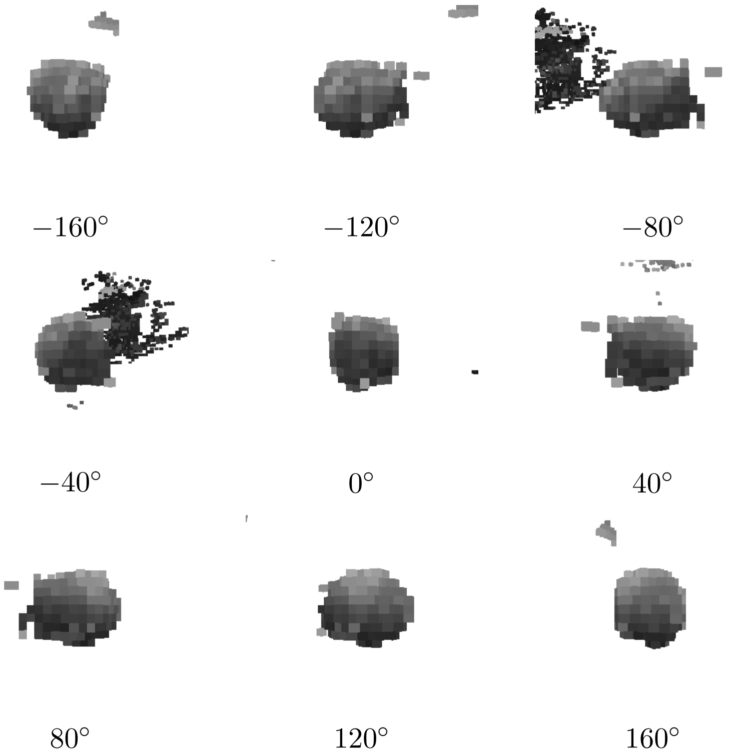


Figure 2.13: Filtered and shade-thresholded reconstructed scene from novel positions. The grey scale has been stretched for clarity.

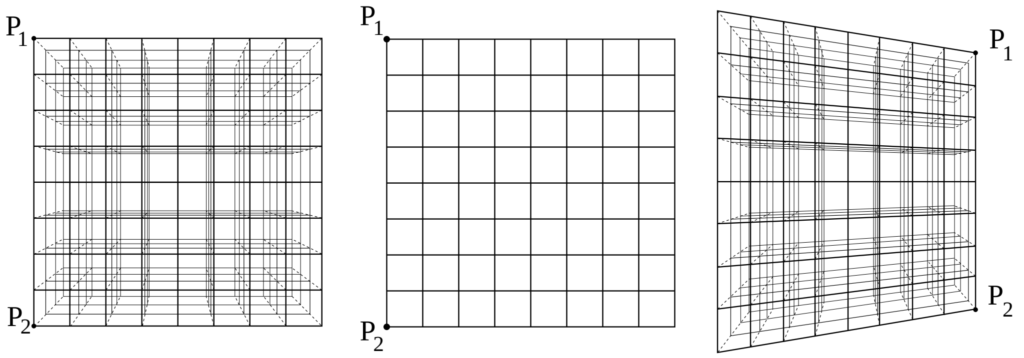
Chapter 3

The 2.75D projection

3.1 Introduction

Voxels are an inherently poor approach to 3D reconstruction. For a regular spaced voxel grid, voxels near to the camera's view may cover a large number of pixels, and hence information regarding an object in the foreground will be discarded as the contributing pixels are merged into a single voxel. For distant objects, there is the possibility of many voxels representing them, however, there is less information regarding such objects, and thus they may be over-sampled.

Although over-sampling is not a problem in 3D reconstruction algorithms, (algorithms including those described earlier), under-sampling certainly is as it may produce correspondence problems and will certainly degrade the fidelity of the system. One simple approach to overcome this is to increase the resolution of the voxel grid, however, this will increase the oversampling of the distant objects and make the voxel grid highly inefficient.



(a) The standard voxel mapping.

(b) The 'ideal' voxel mapping.

(c) 'Ideal' mapping from the side.

Figure 3.1: The poor sampling aspect of voxel spaces. Points P_1 and P_2 are used as an aid to show the orientation of the grid.

Figure 3.1a shows a simple voxel grid mapped onto the 2D page. As is apparent, the closer voxels take a large proportion of the paper; the distant voxels take a smaller proportion of the paper. An attempt to produce an ideal grid could be formed by warping the grid so that near voxels are as well represented as more distant ones, as shown in figure 3.1b. This will only be suitable, however, if the cameras are 'close'; for other orientations this warped grid will degrade the result, as shown in figure 3.1c which is the view from a perpendicular direction.

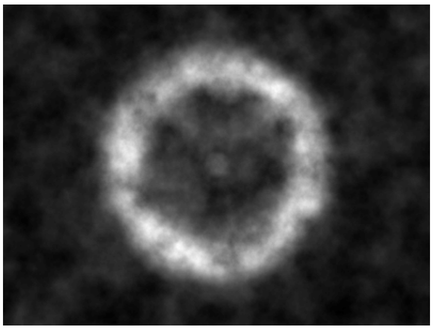
An indication of the failure of the voxel space is that the original source image data is rarely recoverable, thus information is being lost during the reconstruction process. Such losses are certainly not desired for reconstruction algorithms and therefore, this shows the fault in the representation. The ideal space is one that has the sampling structure of the grid indicated in figure 3.1b with respect to all of the views, not just one. This requirement has also been recognised by Slabaugh et al. [81] who indicated that the ideal but unobtainable goal for the voxel space is for each image, voxels should project to the same number of pixels, independent of depth; this was described as the 'constant footprint property'. Confined to using the voxel representation, Slabaugh et al. [81] researched a warped grid in an effort to represent near-objects and far objects in a fixed size voxel grid. However, their representation relies on the grid being carefully placed, otherwise it could result in an even poorer reconstruction if cameras were to be placed at arbitrary positions.

The representation that is presented in this section achieves this constant footprint property, enabling distant and near objects to be correctly interpreted, and can be demonstrated not to lose data for all systems where there is no conflicting information between views.

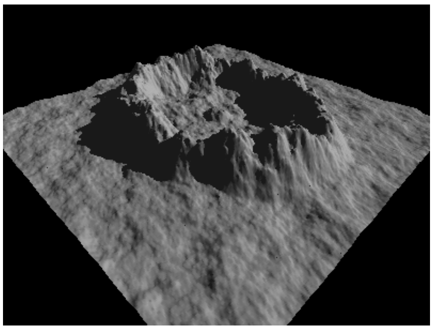
3.2 The new representation

The new representation stems from 2.5D images, or depth or height maps, such as shown in figure 3.2, where each pixel has a single associated depth. However, the restriction to just one depth per pixel shall be removed, and thus each pixel may have many associated depths, as indicated in figure 3.3. This increases the flexibility, allowing the representation of data in a similar manner to that previously described by the grey-scale voxel-based reconstruction algorithm. We call this new representation the 2.75D image. On their own, 2.5D and 2.75D images cannot fully describe the 3D world; only by combining the multiple views with a union operation can this be achieved.

Figure 3.4 shows a pixel from the source image being cast through the unbounded and near-infinite resolution space that is representable in 2.75D. The casting of the pixel forms a ray that covers all of the space that could have caused that pixel to be present in the image. The aim is to project this ray onto the other images and find



(a) The depth image.



(b) The surface plot.

Figure 3.2: Example of a 2.5D image.

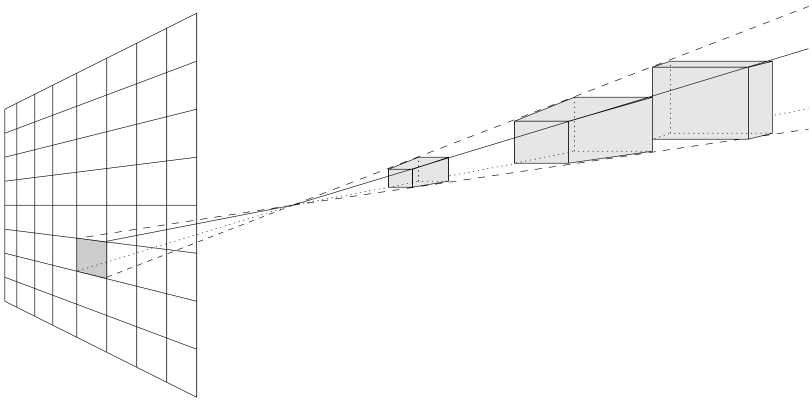


Figure 3.3: The 2.75D representation.

all the possible correspondences it may have. The correspondences are no longer limited to reside within the cubic element structures of voxel spaces.

The ray will be a straight line, due to the normal camera projection models, and will thus also appear as a straight line on the other views (if it appears at all). In figure 3.4, one such source pixel ray can be seen, as can a second view that will provide the correspondences that will indicate the 3D nature of the scene. The ideal 3D model will be one that will make comparisons with the second view at every point along that line. However, that line is discretised due to the nature of images, and thus there is a finite number of points along that line that need to be sampled in order to correctly represent the underlying nature. This is indicated in the diagram by the small perpendicular lines on the ray, which correspond to the limits of the pixels on the second image.

Therefore there is an optimum rate at which the line must be sampled, although note must be made that this rate is not constant but dependent on the current position along the line. This optimum rate will thus ensure that the line is sampled the least number of times but that no pixel along it is missed. This is discussed in section 3.4.

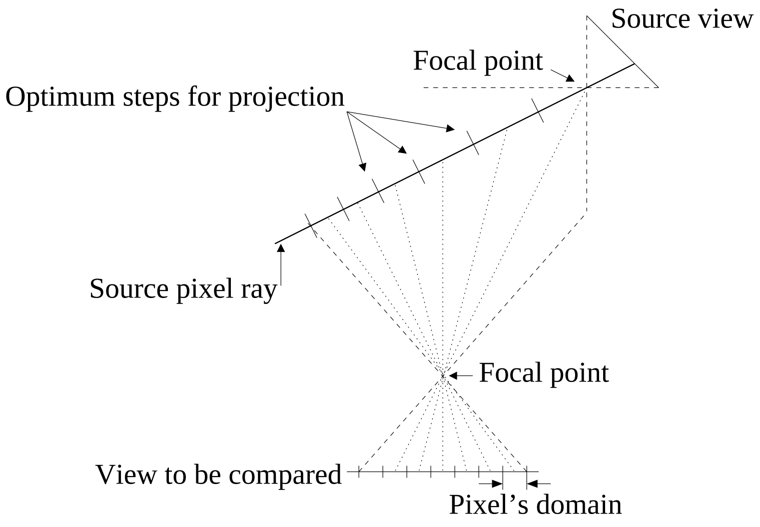


Figure 3.4: Ray casting and the optimum steps for projection of a pixel.

As previously indicated, this new representation allows multiple depths per pixel to be represented. These depths are the orthogonal distances to points on the line, not the actual length along the line, and are efficiently stored in groups, i.e., ranges of depths. The ranges are necessary, otherwise a discrete nature would have to be introduced into the representation. For example, representing all the depths between $1.1 \leq z < 1.9$ would be impossible.

3.3 Formalising VI

The method by which information is gathered and reconstructed using VI in 3D and the new 2.75D representation will now be formalised. The general definitions are first presented, and for completeness, the manner in which the original 2D images are created from the real 3D world is discussed. The voxel-based VI is then formalised and finally the 2.75D VI is described.

3.3.1 Definitions

Let:

$$S = \{s_1, s_2, ...s_{N_S}\} \qquad S \subseteq \mathbb{R}^3 \tag{3.1}$$

be the set of all points in the subject under study.

Given n cameras that witness the subject, each camera will form an image that is segmented into pixels that are and are not part of the subject, labelled '1' and '0' respectively. Thus there are n binary images, $\underline{\underline{\mathbf{I}}}_1, \underline{\underline{\mathbf{I}}}_2, ...\underline{\underline{\mathbf{I}}}_n$, where the j^{th} image has dimensions (w_j, h_j). Each image is described by pixels, with image $\underline{\underline{\mathbf{I}}}_j$ consisting of the array of pixels:

$$\underline{\underline{\mathbf{I}}}_j = \left\{i_{0,0}, i_{1,0}, i_{0,1}, ...i_{w_j-1,h_j-1}\right\} \tag{3.2}$$

Thus a particular pixel in the j^{th} image will be referred to by $i_{\mathbf{p}} \in \underline{\underline{\mathbf{I}}}_j$, or more concisely, $\left(\underline{\underline{\mathbf{I}}}_j\right)_{\mathbf{p}}$ where $\mathbf{p} = [p_0\ p_1]$, $0 \leq p_0 < w_j$, and $0 \leq p_1 < h_j$.

Each camera will effectively project the 3D world into the respective image. Let the transformation performed by camera j be called $P_j(\mathbf{r}) = \mathbf{p}$, where $\mathbf{r}$ is a 3D point in the real world and $\mathbf{p}$ is the 2D integer vector that is used to index the pixels in the image. P_j is equivalent to a projection by the matrix that was defined in equation 2.4 but with the scaling by the respective λ already performed. Note that λ is used in this chapter in a different context, although its actual interpretation is similar.

Finally, let there be a function U such that $U_j(\mathbf{p}, z) = \mathbf{r}$ that, for a given pixel index $\mathbf{p}$ in image j, gives the 3D point $\mathbf{r}$ at an orthogonal distance of z from the camera.

3.3.2 The projection into the camera

The analysis of the image formation shall now be made. To form the binary images in the camera, it can be stated that:

$$\left(\underline{\underline{\mathbf{I}}}_j\right)_{\mathbf{p}} = \left| \begin{array}{ll} 1 & \text{iff } \exists\, \mathbf{e} \in S \text{ st } \mathbf{p} = P_j\left(\mathbf{e}\right) \\ 0 & \text{otherwise} \end{array} \right. \quad \forall\, \mathbf{p} \text{ st } \left\{ \begin{array}{l} 0 \leq p_0 < w_j \\ 0 \leq p_1 < h_j \end{array} \right. \quad \forall\, j \text{ st } 1 \leq j \leq n \tag{3.3}$$

It is important to note that this process is not reversible, i.e., S is not recoverable due to the many-to-one mapping P_j. This indicates that the reconstruction process can only estimate the original subject, as would be expected when reconstructing using a single view.

3.3.3 The voxel-based VI

Volume Intersection (VI) is in essence the extraction of the intersecting volumes that are formed by projecting the binary images. The result is commonly represented in a 3D matrix, $\underline{\underline{\mathbf{M}}}$ whose dimensions are (p, q, r). The elements, m, in this matrix will be referenced using the 3D integer vector $\boldsymbol{\psi} = [\psi_0\ \psi_1\ \psi_2]^{\mathsf{T}}$. Elements, or voxels, in this matrix that are part of the intersection are labelled '1', whilst those that are not are labelled '0'. Hence the matrix elements m are Boolean values.

It has been discussed previously (see section 2.3) that it is simpler to test the individual voxels for their inclusion in the intersection, rather than actually projecting the source images. This is possible since no additional information is required to project a voxel onto a pixel, whereas this is not true for a pixel projected to a voxel. Again, this is due to the many-to-one mapping P_j.

Before formulating the algorithm, two further definitions will be made. Since the matrix is referenced by a 3D integer vector, it would be impossible to change the resolution of the result unless an alternative transformation function was considered that automatically scaled the matrix space. The transformation function $P_j^*(\boldsymbol{\psi})$ will be used to indicate an alternative camera transform function that can be described by:

$$P_j^*(\boldsymbol{\psi}) = P_j(\underline{\underline{\mathbf{W}}}\boldsymbol{\psi}) \tag{3.4}$$

where $\underline{\underline{\mathbf{W}}}$ represents a 4×3 matrix that is capable of altering the orientation, scale factors and origin of the reconstruction matrix space. For example, for adjusting the scale factors and origin only, this would be equivalent to:

$$P_j^*(\boldsymbol{\psi}) = P_j([\psi_0\zeta_0 + \delta_0,\ \psi_1\zeta_1 + \delta_1, \psi_2\zeta_2 + \delta_2]^\mathsf{T}) \tag{3.5}$$

where $\boldsymbol{\zeta}$ is the vector that describes the scale factors, and $\boldsymbol{\delta}$ allows the matrix origin to be arbitrarily positioned.

For each voxel $m_{\boldsymbol{\psi}}$, let there be an associated set $v_{\boldsymbol{\psi}}$ that describes the views that can contribute to it, i.e.:

$$j \in v_{\boldsymbol{\psi}} \text{ iff } \begin{cases} 0 \leq p_0 < w_j \\ 0 \leq p_1 < h_j \end{cases} \quad \text{where } \mathbf{p} = [p_0\ p_1]^\mathsf{T} = P_j^*(\boldsymbol{\psi}) \quad \forall j \text{ st } 1 \leq j \leq n \tag{3.6}$$

This ‘reverse projection’ VI algorithm can now be described by:

$$m_{\boldsymbol{\psi}} = \left| \begin{array}{l} 1 \text{ iff } \left(\underline{\underline{\mathbf{I}}}_j\right)_{\mathbf{p}} = 1 \text{ where } \mathbf{p} = P_j^*(\boldsymbol{\psi}) \ \ \forall\, j \in v_{\boldsymbol{\psi}},\, v_{\boldsymbol{\psi}} \neq \emptyset \\ 0 \text{ otherwise} \end{array} \right. \quad \forall\, \boldsymbol{\psi} \text{ st } \begin{cases} 0 \leq \psi_0 < p \\ 0 \leq \psi_1 < q \\ 0 \leq \psi_2 < r \end{cases} \tag{3.7}$$

This process will rarely be reversible: recovering $\underline{\underline{\mathbf{I}}}_j$ is dependent on the relationship of the discretisation of $\underline{\underline{\underline{\mathbf{M}}}}$, and all of the images $\underline{\underline{\mathbf{I}}}$ and projections P. This would be when the matrix $\underline{\underline{\underline{\mathbf{M}}}}$ has dimensions that do not permit the representation of the data in any one of the images in sufficient resolution or coverage. A trivial example of this would be a matrix consisting of just one voxel attempting to represent a subject that could not be construed as being cubic from any of the views. Another trivial example would be when the matrix is not in view from any image, i.e., the images cannot contribute anything towards the reconstruction. Note, however, must be made that this does not indicate that all views must be used for the intersection.

3.3.4 The 2.75D projection

With this projection the matrix $\underline{\underline{\underline{\mathbf{M}}}}$ will now be redefined as a vector whose elements are matricies:

$$\mathbf{M} = \{\underline{\underline{\mathbf{M}}}_1, \underline{\underline{\mathbf{M}}}_2, ...\underline{\underline{\mathbf{M}}}_n\} \tag{3.8}$$

$$\underline{\underline{\mathbf{M}}}_j = \{m_{0,0}, m_{1,0}, m_{0,1}, ...m_{w_j-1,h_j-1}\} \tag{3.9}$$

Here $\underline{\underline{\mathbf{M}}}_j$ is a two dimensional matrix of dimensions (w_j, h_j) that corresponds to image j. Each element $m \in \underline{\underline{\mathbf{M}}}_j$ is used to represent how each pixel i in $\underline{\underline{\mathbf{I}}}_j$ is reconstructed. This is achieved by allowing each element m to be the set of all orthogonal distances that could be in the original subject. The distances are the actual real values ($\in \mathbb{R}$), thus there is no additional loss of information due to discretisation. The elements m will be infinite in dimension, except for a special case discussed later, when it is possible for an element $m = \emptyset$.

The algorithm projects each image pixel i so that it forms an infinite sheared square-based pyramid. Its cross section at an orthogonal distance of z is then tested for inclusion in subject pixels in all of the other views. If any of the other views indicate that the cross section is not completely within the subject, that orthogonal distance z is not included in the pixel i's respective depth set m. If one of the other views cannot contribute to the projected pixel at the depth of z because it lies outside the bounds of that view, it is not permitted to indicate the suitability of the depth z for that pixel. It is possible for a pixel to be projected and for no depth to be within the bounds of any of the other views, in which case all depths are allowed, i.e., $\mu \in m \;\; \forall \mu > 0 \;\; \mu \in \mathbb{R}$.

For simplicity the equation below does not test for the cross-section being completely within the subject pixels in all of the other views, but just evaluates one point in the cross-section. The full test can be achieved by testing the pixels that contribute to the quadrilateral formed by the projection of the four corners of the source pixel. This is discussed in section 3.5.

The algorithm can thus be demonstrated by evaluating the set of depths of each pixel p in image j:

$$z \in \left(\underline{\underline{\mathbf{M}}}_j\right)_{\mathbf{p}} \quad \text{iff} \quad \left(\underline{\underline{\mathbf{I}}}_k\right)_{\mathbf{q}} = 1 \quad \forall\, k \text{ st} \begin{cases} 0 \leq q_0 < w_k \\ 0 \leq q_1 < h_k \end{cases} \text{where } \mathbf{q} = P_k(U_j(\mathbf{p}, z)) \tag{3.10}$$

This process is reversible except in certain special cases, i.e., the images can usually be recovered. Reconstructing the images from the data is achieved by testing which $\left(\underline{\underline{\mathbf{M}}}_j\right)_{\mathbf{p}} \neq \emptyset$. The special cases are when one pixel is projected completely within the projection cone of another view. An instance can be envisaged where a view's information is tested with and contradicts another view even as $z \to \infty$. For all other instances there is at least the opportunity as $z \to \infty$ that a pixel will not be projected into the viewing area of another camera, and thus the depth be allowed.

3.4 Ray casting: the optimum rate

3.4.1 The approximate method

Having shown the viability of this representation to describe intersecting volumes, it is necessary to analyse it so that an efficient algorithm can be produced. As described, the algorithm projects each image pixel through space. As an approximation, this projection can be thought of as a line growing from the source camera. By projecting this line onto a target view, another is also obtained (see figure 3.4). In this figure the dotted lines denote the limits of each pixel's 'domain' in the second view. Thus it is only necessary to compare the source view with the other view once within each domain.

The projection of this line can be represented by the vector equation:

$$\begin{bmatrix} x_{3d} \\ y_{3d} \\ z_{3d} \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} + \lambda \begin{bmatrix} d \\ e \\ f \end{bmatrix} \tag{3.11}$$

where a, b, c, d, e, f are constants formed from the mapping of the views, and the scaling of λ can be chosen so that it is identical to the projected depth relative to the source image pixel. Hence, $[a \;\; b \;\; c]^{\mathsf{T}}$ is the source camera's origin, and $[(a+d) \;\; (b+e) \;\; (c+f)]^{\mathsf{T}}$ is a point on the ray whose z value is 1 unit from the source camera, both points having been transformed into the other camera's local intrinsic geometry (as defined in section 2.2.4), i.e.:

$$\begin{bmatrix} a & b & c \end{bmatrix}^{\mathsf{T}} = \underline{\underline{\mathbf{P}}}_1 \underline{\underline{\mathbf{P}}}_0^{-1} \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^{\mathsf{T}} \tag{3.12}$$

where $\underline{\underline{\mathbf{P}}}_0$, $\underline{\underline{\mathbf{P}}}_1$ are the projection matrices for the source view and destination view respectively, as defined in equation 2.10, but having been turned into a 4×4 square matrix to enable the matrix manipulation, and:

$$\begin{bmatrix} (a+d) & (b+e) & (c+f) \end{bmatrix}^{\mathsf{T}} = \underline{\underline{\mathbf{P}}}_1 \underline{\underline{\mathbf{P}}}_0^{-1} \begin{bmatrix} \mathbf{p}^{\mathsf{T}} & 1 & 1 \end{bmatrix}^{\mathsf{T}} \tag{3.13}$$

where $\mathbf{p}$ is the source pixel coordinate vector.

A 3D point on the line is mapped onto the destination image by the following equation:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{z_{3d}} \begin{bmatrix} x_{3d} \\ y_{3d} \end{bmatrix} = \begin{bmatrix} \frac{a+\lambda d}{c+\lambda f} \\ \frac{b+\lambda e}{c+\lambda f} \end{bmatrix} \tag{3.14}$$

noting that there is no focal length appearing in the equation as this is accounted for in the camera's local intrinsic geometry.

Thus as λ increases, various pixels on the destination view are visited. However, it is required that all pixels are visited the minimum number of times. By analysing these equations the amount by which λ must increase, $\delta\lambda$, can be formulated. The equation for the required change in λ along the image's x-axis is:

$$\delta\lambda = \left| \begin{array}{ll} \frac{(c+\lambda f)^2}{|(af-cd)|-f(c+\lambda f)} & \text{for} \quad cd \neq af \\ \infty & \qquad\quad cd = af \end{array} \right. \tag{3.15}$$

A similar equation for the image's y-axis can also be formulated, and for a given λ it is the minimum of the two that should be selected. These equations therefore yield the optimum rate by which λ, and thus the projected depth of the source image pixel, must be increased. For multiple views, it is the minimum change in λ over all of the possible destination views that is selected.

3.5 Removing the approximation

Although not yet implemented, a non-approximate method has been envisaged. Instead of the projection of the line, the true square-based pyramid is projected through space. The number of calculations will unfortunately increase four-fold, one for each vertex of the pyramid. It is the cross-section of the pyramid that is projected onto the view to be compared, thus now comparing with an area, not just one pixel.

In figure 3.5, as the source 'pixel' is projected (forming a triangle in 2D) its two sides intersect the destination image at different points for the same orthogonal distance. Over certain depths, both sides of the ray will be projected onto the same destination pixel, and for these regions, the comparison to make is as described for the approximate method. However, for certain regions two destination pixels are required to make contributions, and thus as a form of averaging, a linear combination of the two is required. The contribution at a point along one of the regions is given by:

$$v = \frac{aP_0 + bP_1}{a+b} \tag{3.16}$$

where the a and b are the two lengths as indicated in figure 3.5, which change linearly at these boundary conditions as the orthogonal distance increases, and P_0 is the shade of the respective pixel in the destination view.

For VI, these segments can be split into two regions, one being that which the probability of the pixel being in the image is greater than 50%, and the other being the contrary. Unfortunately when using such a method in the grey and colour algorithms below a non-thresholded value is required. This indicates that there are not a finite number of values, and therefore steps, in these regions. Therefore it is proposed that if there are no other views to segment the space, the region should remain intact and take the value equal to the above equation evaluated at its mid-depth point.

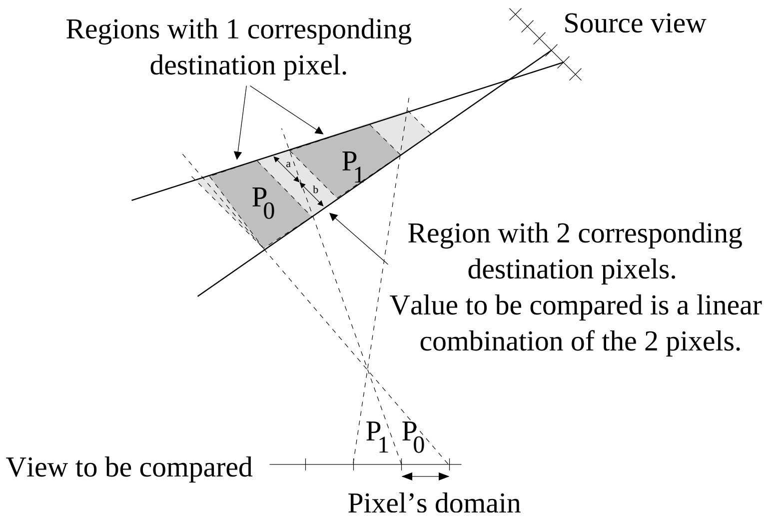


Figure 3.5: The non-approximated method.

3.6 Implementing VI

Combining the formalised definition of 2.75D and the optimum rate equation, the approximate 2.75D VI algorithm can now be described using pseudo-code, as shown in listing 3.1.

It is apparent that two constants are required for this algorithm, namely the ranges that λ is allowed to traverse. Starting λ at the value of 0 is not advised due to the singular nature of the mappings at this depth, thus the minimum value of λ allows this to be overcome. For the examples shown below, an initial value of 1 was selected. The maximum value of λ does not have to be restrictive as it can be defined as the highest value that can be represented. However, restricting the maximum value may be useful especially when there are two views that are nearly co-linear, and thus may continue to produce correspondences beyond that which is required.

It is important to note that although two constants are required for this algorithm, this compares favourably to the many that are required for voxel-based algorithms; the 3D size of the voxel space must be defined, as must its position, orientation and scaling in space, i.e., 12 constants. The selection of the 2.75D constants is trivial.

The implementation of the pixel depth storage uses a singularly linked list for efficiency as it is only required to be traversed in the one direction. Also for efficiency, the linked lists are actually stored in large arrays, otherwise the memory allocation overheads for the many small entries would be very high.

```
For each pixel, in all of the views, that is selected,
   clear respective pixel's depth linked list
   λ = predefined minimum value
   need_new_depth = true
   while λ < λ_max
      λ_next = λ_max
      transparent_at_this_depth = false
      calculate where the pixel at a depth of λ maps to in 3D space
      over all of the other views,
         if this 3D point is visible in this other view,
            calculate the pixel coordinates
            if pixel is not set
               transparent_at_this_depth = true
               need_new_depth = true
         predict λ's change for this view
         λ_next = min(λ_next, prediction + λ)
      if transparent_at_this_depth = false
         if need_new_depth = true
            create a new linked list entry
            entry's min_z = λ
            entry's max_z = λ_next
            need_new_depth = false
         else
            last entry's max_z = λ_next
      λ = λ_next
```

Listing 3.1: The basis of the 2.75D VI algorithm.

3.6.1 Complexity analysis

Consider the case where there are n views, with each being represented by p pixels. Let the number of pixels along the diagonal of the image be d. A ray is projected from a source pixel onto a destination view. The worst case is for the projection to lie along its diagonal since this will yield the most comparisons. As the ray is projected, its next position is calculated from its current one, but this next position does not necessarily lie at the border of the next pixel, just at some point within it. Thus if another destination view was to be considered in parallel with correspondences that interleaved perfectly with the first destination view, there would be no increase in the number of cells tested. The worst case is if the second destination view was described such that all of its correspondences lay within a single cell of the first destination view. In this case there would be $2d$ depth cells for the ray, of which half of them would require to be compared with two views, whilst the others with only one. In general, the worst case is $(n-1)d$ cells for a ray, and a total of $\frac{1}{2}n(n-1)d$ comparisons. A common case would be more similar to the interleaving example, with thus just d cells, and a total of $(n-1)d$ comparisons.

This is because each pixel along the diagonals will only be tested once unless a cell is further segmented by another view's contribution. Each cell will need to be compared with all the views that can see it. Thus the overall order is given by:

$$\begin{aligned} \mathrm{O}(\text{Worst case 2.75D VI}) &= \mathrm{O}\left(np \cdot dn(n-1)\right) \approx \mathrm{O}(p^{1.5}n^3) && (3.17) \\ \mathrm{O}(\text{Common case 2.75D VI}) &= \mathrm{O}\left(np \cdot d(n-1)\right) \approx \mathrm{O}(p^{1.5}n^2) && (3.18) \\ & && (3.19) \end{aligned}$$

where $d \approx \sqrt{p}$.

Analysing listing 2.2, it is clear that the order of processing for the 3D voxel-based VI algorithm is:

$$\mathrm{O}(\text{3D VI}) = \mathrm{O}(nv) \qquad (3.20)$$

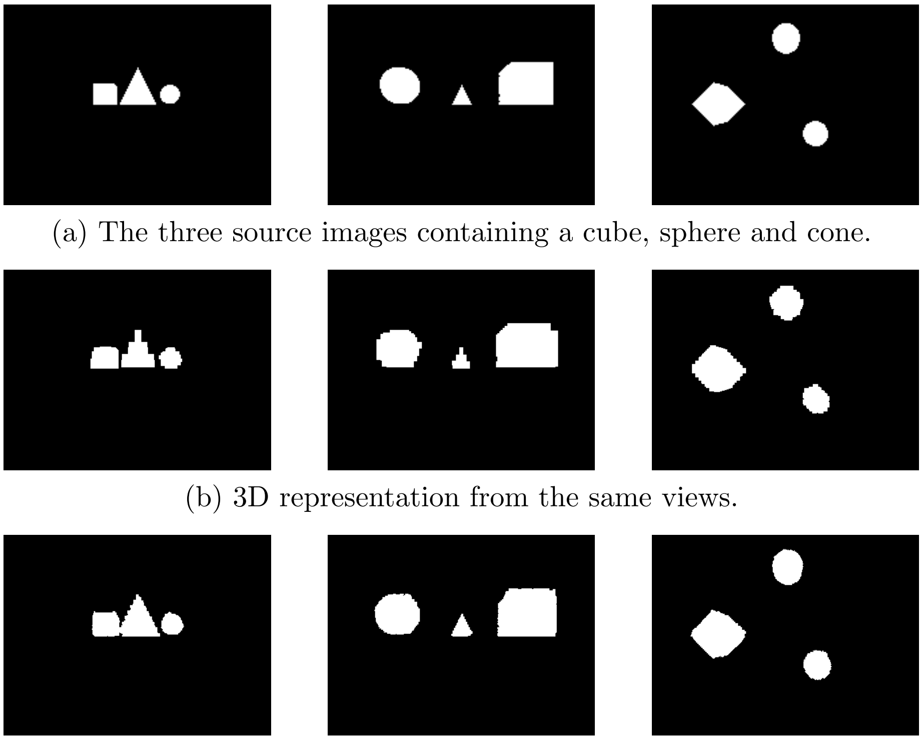
where v is the number of voxels.

It is thus apparent that the 3D voxel-based VI algorithm has a much more favourable order of processing for large n, even though, for a value of $n = 2$, the 2.75D algorithm can yield an order as low as $\mathrm{O}(pd)$ which will commonly be more favourable than $\mathrm{O}(v)$. However, it is believed that if the resolution of the 3D voxel space was increased so that it could properly represent the object, the processing time of the 3D voxel-based VI algorithm would be very much higher than that of the 2.75D algorithm. The choice of the resolution of the 3D voxel space is thus both a compromise in computing memory resources and processing time.

3.7 2.75D VI results

Figure 3.6 demonstrates the results of VI using both the 3D voxel-based and the 2.75D methods. The results from this new representation are similar to the 3D voxel-based solutions, except that they are analysed at the most suitable level of resolution. As can be seen, especially around the edges of the cone, the level of detail in figure 3.6c is much closer to the original data. If the test for depths on each projected pixel is performed as outlined above, it produces exactly the same data as found in the source image. However, figure 3.6c is not identical to figure 3.6a, and although a source of error in the images is the inaccuracy of the rendering of the 2.75D data, the ray-casting approximation of this method is a major factor in the differences. To produce the image, the ray is projected as a square-based pyramid, and though its centroid may be correctly placed within the original source data, the extremities may not be.

The 2.75D VI algorithm suffers from the problems of the visual hull and phantom shapes that the normal voxel-based algorithm is known to be hindered by (see sections 2.5.3 & 2.5.1). This should be expected as it is only the representation



(a) The three source images containing a cube, sphere and cone.

(b) 3D representation from the same views.

(c) Results of the new 2.75D representation.

Figure 3.6: Comparison between VI and the new representation.

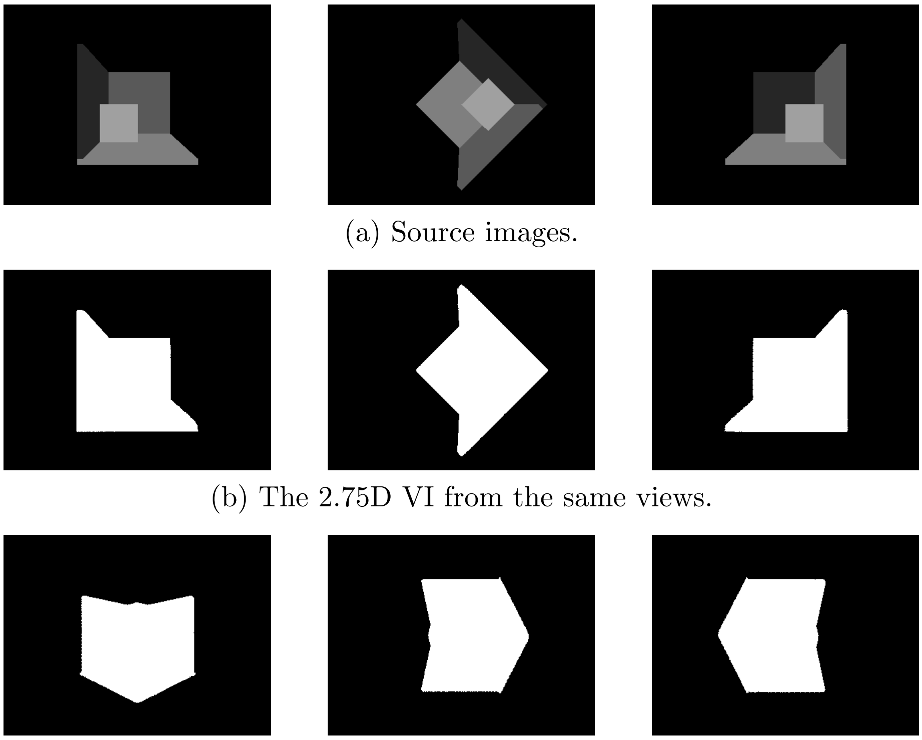
of the data that has changed. The visual hull can be seen in figure 3.7 where the source images are not understood in the manner in which we perceive them.

3.8 Grey scale and colour implementation

There are many similarities between the new algorithm using this new representation and the new voxel-based algorithm discussed in chapter 2. However, the restriction to just six sides can now be removed. The purpose of the sides was to incorporate an understanding of the cooperation between rays, providing a suitable solution to the hypothesis that only rays from the same surface should be combined. In this new representation, the dot-product between projection rays provides a weighting or cooperation factor w:

$$\begin{aligned} 2w - 1 &= \cos\theta = \frac{\mathbf{r_s} \cdot \mathbf{r_d}}{|\mathbf{r_s}||\mathbf{r_d}|} && (3.21) \\ &\approx \operatorname{sign}(\cos\theta)\cos^2\theta = \operatorname{sign}(\mathbf{r_s} \cdot \mathbf{r_d}) \cdot \frac{(\mathbf{r_s} \cdot \mathbf{r_d})^2}{|\mathbf{r_s}|^2|\mathbf{r_d}|^2} && (3.22) \end{aligned}$$

where $\mathbf{r_s}$ is the vector from the source view to the 3D point and $\mathbf{r_d}$ is the vector from a destination view to the 3D point. The approximation is used so that the rather costly square root operation can be avoided.



(a) Source images.

(b) The 2.75D VI from the same views.

(c) The 2.75D VI from novel views.

Figure 3.7: Open-box resulting scene views showing the visual hull.

Thus $w = 0$ for rays in opposition, indicating that no information can be gained from combining their pixels, $w = \frac{1}{2}$ for rays that meet orthogonally, and $w = 1$ for colinear rays, indicating that there is 50% and 100% probability respectively of the rays originating from the same surface (assuming that there is no occlusion, of which the handling is described below).

As with the voxel-based algorithm, there is a confidence measure for each 3D point. This new measure appears to be the same as that in equation 2.15, except that n is no longer an integer representing the number of views, but the sum of cooperations:

$$m = \frac{\sigma^2 + k_1}{n + k_2} \tag{3.23}$$

$$\sigma^2 = \left(\frac{1}{n}\sum_{i=1}^{v}(w_i p_i^2)\right) - \left(\frac{1}{n}\sum_{i=1}^{v}(w_i p_i)\right)^2 \tag{3.24}$$

$$n = \sum_{i=1}^{v} w_i \tag{3.25}$$

where v is the number of views, w_i is the weighting of the pixel from view i, and p_i is that pixel's shade.

It is also feasible to incorporate colour into the measure:

$$m = \frac{\sigma^2_{red} + \sigma^2_{green} + \sigma^2_{blue} + k_1}{n + k_2} \tag{3.26}$$

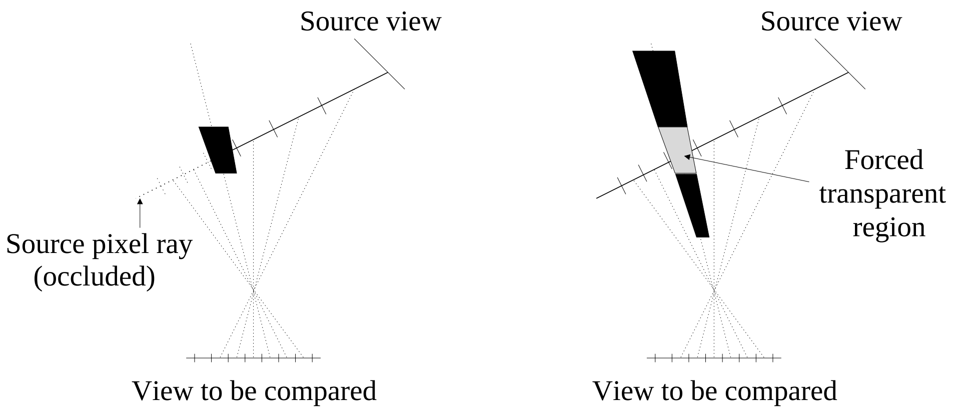
As with the voxel-based system, this is an iterative algorithm with several stages. The first is to analyse the best confidence levels, using equation 3.23, for each pixel, i.e., each pixel is projected through the space and its best confidence level is noted.

However, the projection of pixels is further complicated by occlusions from other views. Once a pixel's set of depths has been decided, it has priority over those that are to be decided, thus when projecting undecided pixels, the decided pixels must be noted. A problem arises when the 'decided' pixel (from a destination view) is not of the same colour or shade as the projected pixel from the source view. If the pixel is similar (i.e., within a range k_{reject}—see below), then the 'decided' pixel makes a standard contribution. However, if the pixel is not similar there are two scenarios depending on the depth set of this 'decided' pixel, as illustrated in figure 3.8. In essence, either the source ray or the destination ray attempt to occlude one another. A previously decided depth along the destination ray is able to occlude the source ray, and thus the source ray is prevented from progressing any further. Such an occlusion only occurs if the source and destination pixels are of a different colour—if they are the same, then the source ray is permitted to continue since the occlusion would have been partly caused by the source ray's previous assessment for the destination ray's depth. However, the second case, is for the source ray to attempt to occlude the destination ray that is of a different colour; the source ray is not permitted to occlude the destination ray since the destination ray has previously been decided. Therefore over such a region, the source ray is not permitted to have any associated depths—it must remain transparent.

The constant k_{reject} rejects pixels because of their shade (in grey-scale analysis) or individual colour component (in full-colour analysis) being different from the source pixel. The destination pixel is not rejected if its value v_d lies within the range:

$$v_s - k_{reject} \leq v_d \leq v_s + k_{reject} \tag{3.27}$$

where v_s is the value of the source pixel. The choice of this constant, k_{reject}, should be to keep its value small. However, note must be made of the noise in the images, both local noise, such as specular effects, and global noise, such as contrast; larger values can reduce the wrong correlations that such errors introduce. Also, note that this rejection principle is different to that of the voxel-based algorithm: in the latter, a ray will not be compared if it is a value of k_{reject} from the mean, but in the former, a ray will be used no matter what its value if it has not already been



(a) Occlusion by a block from another pixel of a different colour.

(b) Force transparency at a depth due to a lack of a depth block from another pixel of a different colour.

Figure 3.8: The occluding and transparency factors when comparing with a decided pixel of a different colour.

decided.

As with the voxel-based system, each iteration permits select depths (cf. voxels), to be predicted, according to their confidence. The range is defined by two constants, namely k_{mult} and k_{add}, which describe the same range relationship as those in the voxel algorithm:

$$m_{min} \leq m < m_{min}k_{mult} + k_{add} \tag{3.28}$$

Thus the best confidence level, i.e., lowest value m_{min}, must first be found. Therefore, for each undecided pixel, the respective ray is projected, and the best confidence level along that ray is recorded. The best confidence level over all the pixels in all of the images is thus also available, and hence the range of confidence levels, similar to that used in the voxel-based algorithm, is calculated. By having noted the best confidence levels of each of the pixels, only those that have a best confidence level within the necessary range need to be re-analysed. This re-analysis re-projects the respective rays, to produce depth lists that describe all the depths, not just the best for the pixel, whose measure of confidence lies within this range. The entire process is then re-iterated until all pixels have been decided.

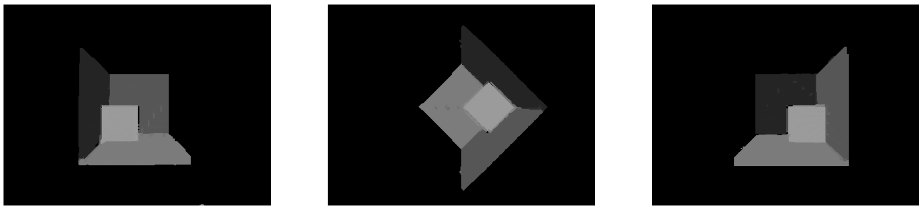
To increase the efficiency of the first stage, on the subsequent iterations of the algorithm, only pixels that could be affected by the selection of pixels in the previous stage are recalculated; this is performed by setting a flag for the destination pixels that are tested by a selected pixel. Also note that pixels that are selected in a batch must not be influenced by the order in which they are processed, otherwise unexpected occlusion effects will occur; the test for occlusion must note whether the destination pixel has only just been decided. Hence each pixel actually has four states, namely *undecided*, *decided*, *selected*, *undecided-but-requires-recalculation*. Only after all of the *selected* pixels have been processed do their states get changed to *decided*, and the *undecided-but-requires-recalculation* get subsequently recalculated and reset to *undecided*.

3.9 Grey scale results

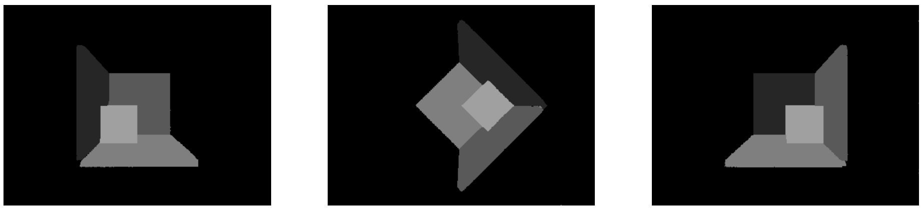
Figure 3.9 demonstrates a selection of results produced with this 2.75D grey scale algorithm. During the construction of these images, the reconstruction algorithms were informed that black pixels were background (transparent); without this, this abstract data becomes difficult to comprehend, especially from alternative viewpoints.

Figure 3.9b compares favourably to the new voxel-based algorithm whose results are shown in figure 3.9a, when viewed from the source angles. Similar to the voxel-based algorithm shown in figure 3.9c, the 2.75D algorithm can be seen in figure 3.9d to introduce artifacts into the reconstruction. Many of these can be removed by using the confidence level as a filter, as shown in figure 3.9f, as they are rays that have failed to confer with any other view and hence are represented in regions that can only be seen by the source view. They are thus rays that stretch to infinity in this case, although for the rendering, this limit is obviously clipped. Figures 3.9e & 3.9f show the filtered voxel and 2.75D results. All of the problems regarding the aliasing around the sharp colour gradients can be seen to have been removed, shown by the lack of a black border around the various coloured regions. Both figures, however, show an identical visual coloured hull, for example in the first image the base can be seen to protrude into the region of the box, and in the second and third, the two sides can be seen to do the same.

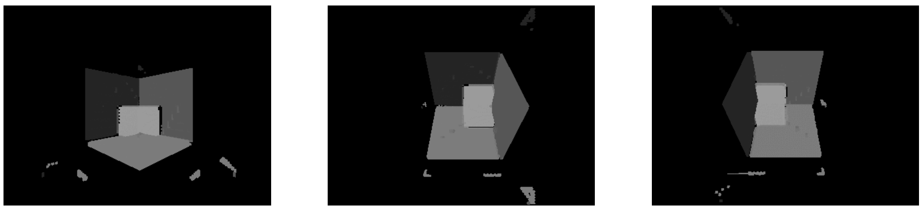
As with the voxel-based algorithm, problems arise when the views provide conflicting images. In fact this can be seen in the 2.75D example where the bottom left of the main structure in figure 3.9b has not been rendered. A further example of this is shown in figure 3.10 where an additional three views are combined. These results have already been filtered at an appropriate level to remove the rays and voxels that are formed from just a single view. It can be seen in these figures that although the conflicting information has produced irregular results, the 2.75D algorithm has produced a more favourable representation. Unfortunately, the results of both algorithms, are heavily dependent on the selection of their constants. Summarising, in the 2.75D algorithm these constants are k_1, k_2, which are used for the fitness rating of a point, the confidence step constants, k_{mult} and k_{add}, and the rejection on



(a) The new voxel algorithm.



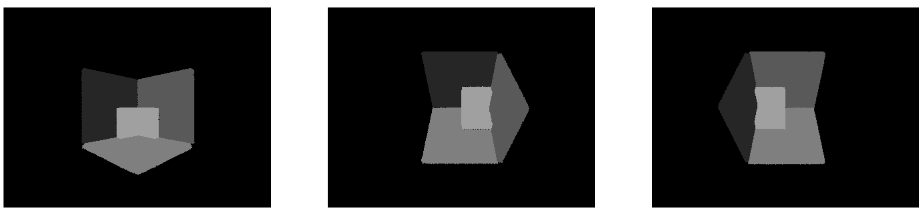
(b) The new grey 2.75D algorithm.



(c) The new voxel algorithm from novel views.

(d) The new grey 2.75D algorithm from novel views.

(e) The new voxel algorithm from novel views with filtering.



(f) The new grey 2.75D algorithm from novel views with filtering.

Figure 3.9: Open-box resulting scene views.

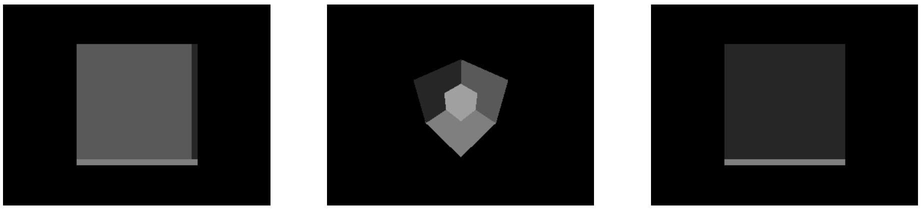
shade constant k_{reject}. For all of the three camera situations the values are 500.00, -0.50, 1.02, 100.00 and 10.00 respectively. For the six camera scenario, k_1 is set to the value of 5000.00 to obtain the best results. These values differ slightly from the voxel-based algorithm. For example k_2 is very much reduced, but this is as a result of the variable n not being the same in both algorithms. In fact k_2 is actually negative to make the contributions of other views more significant—each ray will have a guaranteed condition of $n \geq 1$ since the rays own self-contribution produces a value of $n = 1$. Other contributing views whose additional value to n will be less than 1 will thus not always be significant, and hence the negative value of k_2. k_{add} is also larger in the 2.75D implementation to encourage less shell-like results. The resulting confidence values for all of the experiments performed for the most part were within the range of $(0, 1000)$, which is of the same order to that of the grey-scale 3D algorithm, being $(0, 500)$.

3.10 Colour results

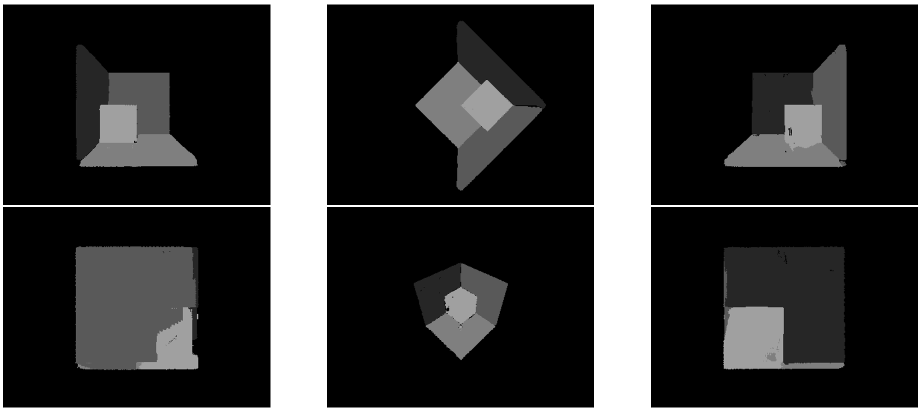
Figure 3.11 demonstrates the results obtained from real data using the colour 2.75D algorithm. The clarity of the reconstructed scene is clear in figure 3.11b, comparable to that of the source images in figure 3.11a. These results demonstrate a considerable improvement over the voxel-based reconstruction shown in figure 2.12, with both smaller features and no limitations on the size of the scene that can be reconstructed; for example, both the radiator pipe-work and right-hand door are properly rendered. However, streaks can be seen across the images which are as a result of the choice of the stepping and rejection constants. Due to the poor source image quality, which is further discussed in section 6.3, the rejection constant had to be kept at the previously indicated value. Having small step constants with these poor images results in only small, or even singular, regions of each ray to be located, yielding almost a wire mesh. Although this does produce results with an improved appearance, it is not suitable for the next stage of the analysis, which is discussed in the following chapter.

Figure 3.11c shows a set of novel views of this reconstructed scene with its background now having been removed (the background removal is also discussed in the following chapter). The lines on the basketball are actually visible from many of these novel views. The ball, however, is not predicted to be spherical—a result of the colour form of the visual hull, and also poor correlations due to the image quality, and specular and lighting effects on the surface. These reconstructions also show degrees of discretisation, but these are solely due to the original discretisation of the source images.

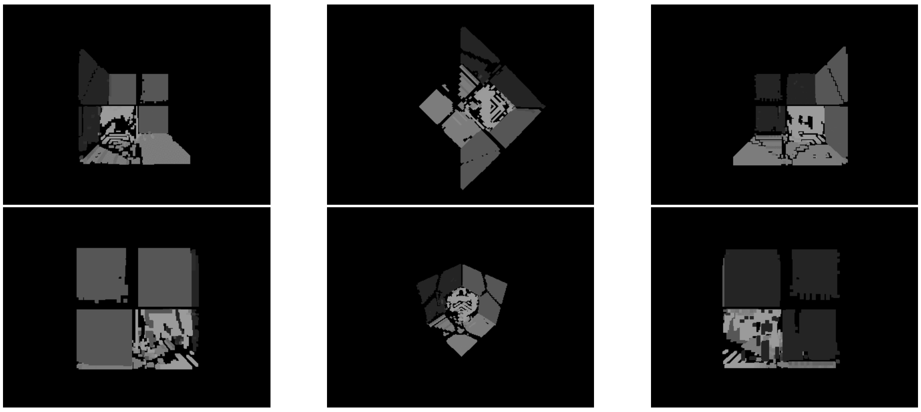
These results indicate that although the VI algorithm is, on the whole, lossless with regards to information, the colour and grey scale algorithms remain lossy.



(a) The three additional source images.



(b) The 2.75D algorithm's results, from the 6 camera viewpoints.



(c) The voxel algorithm's results, from the 6 camera viewpoints.

Figure 3.10: Open-box scene views with conflicting cameras.

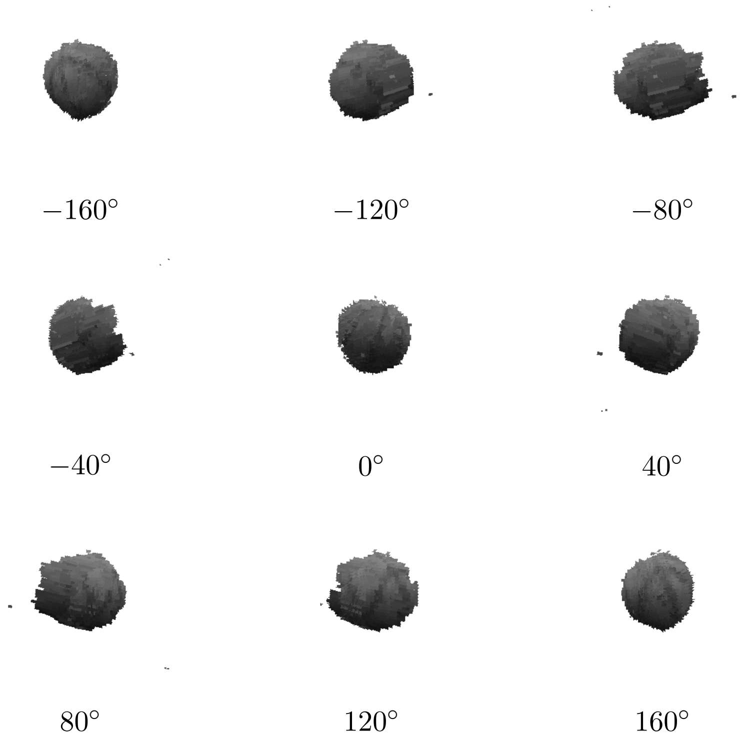
However, this loss is now the fault of only the algorithm; no longer does the underlying 3D representation introduce a major contributory degradation factor.

3.11 Future work

The above examples seem to be very distant from the regular voxel grid structure, however, a predictable structure does lie beneath. Although it was not implemented, it would not be unreasonable to represent each pixel's depth ranges as 'cells' (for they can no longer be called voxels). Cells would become the equivalent to the volume elements that would be described by the small jumps in λ. The cell's depth (λ) range and the pixels that they would have to correspond to in the other images can all be calculated just once. For the images that are used in this work

(a) The three source images.

(b) The unfiltered reconstructed scene from the same positions.



(c) Filtered reconstructed scene from novel positions around the ball. The resulting grey-scale has been stretched for clarity.

Figure 3.11: Colour 2.75D analysis results.

(348 × 280 pixels), this structure would take approximately 0.5 Gb. Once these calculations have been made, the comparisons could be made more quickly, and in fact it would be more comparable in time to the voxel space.

Other future work on this topic should include exploring the relationship between the constants as it is known by experiment that there are dependencies between the step size constants (k_{mult} and k_{add}) and the measure constants (k_1 and k_2). There are also other suggestions that are also common with the voxel algorithm, which are discussed more fully in chapter 7.

3.12 Related work

Although this work is novel, there are two related works, although it could be interpreted as the rectilinear parallelepiped described by Kim and Aggarwal [41] and Martin and Aggarwal [54] under a perspective projection and taken to the extreme. The most similar work is that described by Matusik et al. [55] who described 'Image-Based Visual Hulls'. In essence, this is very similar to the 2.75D VI algorithm described above, although it compares views with a source view one at a time, rather than the above method where a projected depth is compared against all views for inclusion. However, Matusik et al. [55] also described a colour algorithm, but this is actually not a reconstruction algorithm, but merely a means to colour map the original images onto the VI visual hull. Thus, their algorithm is not capable of non-segmented scene reconstruction. However, the fact that such research is being performed is indicative that there is interest in the potential of such techniques.

The other related work is a computer graphics technique called 'Layered Depth Images' (LDIs), as presented by Shade et al. [77]. Its main purpose is to provide an efficient method to describe complex 3D objects by projection of an image. Like the 2.75D algorithm, each pixel can be described by many depths, but unlike the 2.75D algorithm, the pixel can have a different colour at each depth since only one projection is used. However, this representation is unsuitable since each pixel is still treated as a planar object when it is projected—it does not have an associated volume. A reconstruction algorithm is described using just image data but it is lossy since if two depths are similar for a pixel, they are merged together in an effort to reduce the quantity of data. This is not required in the 2.75D algorithm due to the depth ranges that are used. There is also a problem regarding holes for when the distance between neighbouring depths become too large. It is concluded that the algorithm is more suitable for rendering photo-realistic 3D models than for reconstructing them: in fact emphasis in the work of Shade et al. [77] is on using ray-traced images with depth information. Also, the non-solid nature of this representation would be unsuitable for the evidence gathering procedure that is discussed in the following chapter.

3.13 Conclusion

The chapter has introduced the new representation for reconstructing 3D data and has described suitable VI, grey and colour reconstruction algorithms. No information is lost during this transform, unlike the voxel representation, and there is no need to specify what the limits of the reconstruction are as a 'near-infinite' space can be represented; this is required for voxel-based systems where the voxel space must be placed, scaled and rotated into the correct orientation for the region of interest.

The only noted drawback is the order of the processing time as it is considerably more mathematically intensive; the order of the algorithm is unpredictable as it is dependent on how the views are arranged. As an indication, the reconstruction of the real-world sequences in the following section takes approximately two minutes per frame, as opposed to thirty seconds for the voxel-based algorithm, on a 1.4 GHz machine; the images from all three cameras in these sequences are 348×280 pixels. As with the voxel-based algorithm, the 2.75D algorithms are highly suitable for parallel processing, and thus video-rate scene generation is conceivable.

Chapter 4

Three-dimensional dynamic model extraction

4.1 Introduction

This chapter describes the manner in which objects, whose dynamic nature can be mathematically modelled, are extracted and parameters defined. This thus explains the background removal stage and the parameter extraction stage of the processing for all of the three systems discussed in chapter 1. Basic models are used as illustrations of the techniques; further models and also results of synthetic and real data are to be found in chapter 6 where the three systems are compared and contrasted.

Two examples are given to demonstrate the means of 3D dynamic model extraction. One such model should be suitable for the purposes of 3D gait recognition; this model is used to analyse real data in chapter 6.

The implementation of the parameter extraction greatly influenced the solution that is described herein, and thus occasional references are made to this as an illustration of the motivation.

4.2 Background removal

All three systems require the background to be removed from the original data. Although this is actually not essential for the 2.75D and 3D systems since the additional possibility of transparent regions (free-space) has been found to provide a useful segmentation tool, it assists the final stage of object extraction. In the 2D system, removing the background is the core of the segmentation of the image into the dynamic parts which are of interest and static parts which are not.

4.2.1 2D background removal

In order to remove the background from 2D images, the entire sequence must be used, and pixels from different frames but the same position are compared. A simplistic approach to background removal would be to deem pixels as background if all of their three colour components lie within a specific distance, k_1, of the means of the three components of the pixels that reside at the same image coordinate in the sequence.

However, the mean is a poor measure for background removal since foreground pixels will have a large influence on its value. For instance, if, for a particular pixel position, 40% of the pixels were white (value of 1) and 60% of the pixels were black (value of 0), the mean would be 0.4. Therefore, although the values of the black pixels lie closer to the mean than that of the white pixels, the difference is not very significant. The ideal measure is the mode, and in the above example this would produce the required result as it would yield the value of 0, however, the range of pixel values that will actually be encountered compared with the relatively low number of images, makes the mode unsuitable and extremely variable in noisy conditions. The median, which in this example also evaluates as 0, is a more suitable measure in such cases, and provides a more ideal solution than the mean (occasionally the truncated median is used as an approximation of the mode for such examples).

It is common to have regions of images with high turbulence. For example, an outside sequence may contain trees with leaves blowing in the wind. The trees will produce significant fluctuations but are not of interest. Therefore the background removal can be improved further by using the standard deviation, σ, as a measure of turbulence. For regions with high turbulence, the standard deviation is higher, and thus the determination of the background should be not just dependent on the distance measure k_1, but also on σ. Hence the leaves would be ignored, but objects moving occasionally through other parts of the image would be registered as foreground. The use of the variance also enables regions of high specular noise to be ignored.

The resulting binary image can thus be described by:

$$r_{(x,y,t)} = \left| \begin{array}{ll} 0 & \text{if } |p_{(c,x,y,t)} - \text{median}_{(c,x,y)}| < \max(k_1, k_2\sigma_{(c,x,y)}) \ \forall c \in \{1,2,3\} \\ 1 & \text{otherwise} \end{array} \right. \tag{4.1}$$

where k_2 is a scaling constant for the variance, and $p_{(c,x,y,t)}$ describes the source image's colour component c, and pixel position x, y at sequence number t.

There are more advanced methods of background removal, however, this is sufficient for the purposes of this research and performs excellent segmentation. For example, in the later examples, it can be seen to extract both the object and its faint shadow.

4.2.2 2.75D background removal

The masks, i.e., binary images, obtained from the 2D background removal are used to mask the 2.75D data. This thus does not truly mask the result to the visual hull formed from the segmented images, but to an approximation of it. However, the 2.75D algorithm has interpreted the scene using full colour, and has made an improved estimate of the contents of the true visual hull, thus this masking has yielded a possible *better-than-hull* result.

4.2.3 3D background removal

The method selected to remove the background in the voxel algorithm is similar to that used in the 2D system described above, but with only a shade component for each voxel, not a three-colour component. It is complicated, though, by the additional feature of free-space, or transparent, voxels which are in essence, valueless. The algorithm, therefore, must take into consideration the amount of free-space at a particular voxel position throughout a sequence. If there is more free-space than voxels with shade, then those voxels with shade are immediately part of the foreground. However, if there is less free-space than voxels with shade, then the algorithm used for the 2D background removal is applied, but only for the voxels that are not transparent.

This algorithm successfully removes the background from the 3D voxel sequences, although as will be seen, real scenes produce relatively noisy results since they were generated from the noisy source data. An alternative method, as outlined in chapter 7, would be to use a method similar to the 2.75D background removal, i.e., restriction of the data to the visual hull.

4.3 Extraction

4.3.1 Introduction

The manner by which the parameters of an object's mathematically described model are extracted and described is now presented. The basis of this is evidence gathering, or more specifically, the Hough Transform (HT) and Template Matching (TM) as introduced in section 1.4.1. These algorithms are explained in the following section, and subsequently, the method by which the models are described is presented.

4.3.2 Evidence gathering

The basis of both the 3D generation algorithm and the motion analysis algorithm is in evidence gathering, with, as explained in section 1.4.1, this technique's most noted algorithm being the HT [34]. The algorithm uses a resulting accumulator or voting space to gather information from the source image. The algorithm can be described by the pseudo-code listing 4.1.

```
for all foreground pixels in the source image,
   for all possible lines that could have caused that pixel,
      increase vote for the particular line.
```

Listing 4.1: HT for lines pseudo-code.

Peaks in the accumulator space correspond to the lines that are most likely to exist. The initial implementation by Hough [34] was made by describing lines using a Cartesian parametrisation, in terms of the gradient m, and y-intercept, c; the problems of infinities led to an improved parameter space in terms of the polar representation, θ and ρ [22].

The result of the HT is the same as for TM, which is also known as the Hough Transform by back-mapping, but the former can have improved performance since not all of the parameter space will be tested. The TM algorithm can be described by listing 4.2.

```
for a particular line to be tested,
   for all foreground pixels along that line in the image,
      increase vote for the particular line.
```

Listing 4.2: Template matching for lines pseudo-code.

TM does not require an accumulator space to record the parameters of the most suitable line. Thus TM is advantageous when the dimensions or resolution of the result to the problem prevent the use of a HT accumulator space.

The Velocity HT (VHT) was developed by Nash et al. [59] in which moving objects, particularly circles, were sought in a sequence of images. The analysis of all of the information has huge performance advantages in detecting moving objects than using the more common tracking methods where the first frame is the seed for all subsequent frames. The VHT is also much more resilient to occlusion than tracking, which would have to use predictive methods if the loss of tracking was detected. However, the drawback is that the VHT has a greater processing overhead. In fact both the HT and TM are not possible as the number of parameters, and thereby the combinations of the different values of the parameters, increases. This problem was alleviated in the work of Cunado et al. [15] by the use of the approximate parameter searching method that is Genetic Algorithms (GAs—see section 4.4) used in combination with TM; it is this combination that was also used during this research.

4.3.3 Mathematical models

In order to perform TM on data, a fitness function is required. This evaluates a set of parameters for an object, yielding a fitness value. Thus, when searching for a

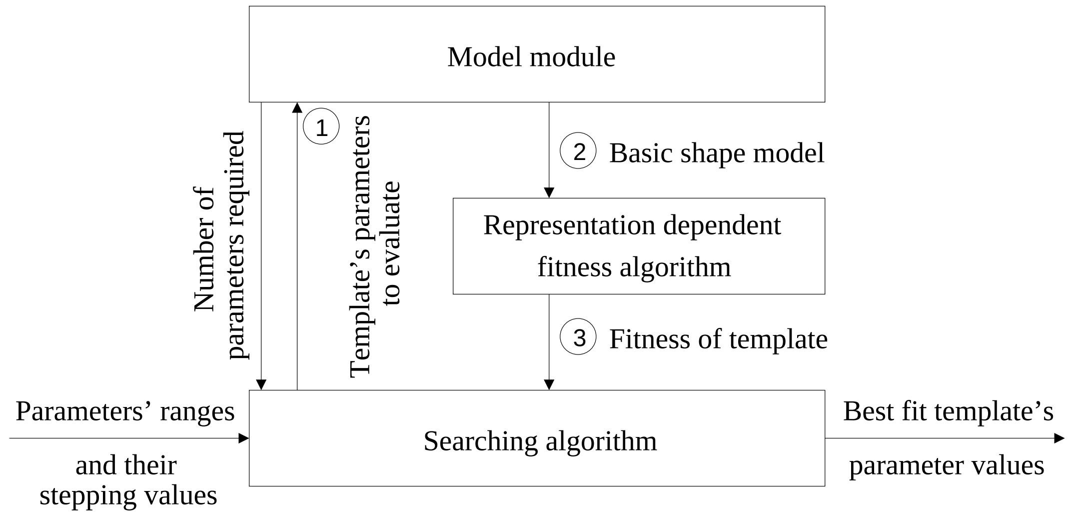


Figure 4.1: The implemented evidence gathering technique, with numbers indicating the order the information is passed through the system. This shows the three separable blocks, where the top two blocks, in essence, perform a template match for a set of parameters. The bottom block is the searching algorithm; for example a grid search algorithm across the whole parameter space, or a Genetic Algorithm.

(limitless) line in a 2D image, two parameters are passed to this fitness function. In the VHT, a circle moving with constant velocity, sought in a sequence of images, is used as an example; in this case, the fitness function would be passed five parameters, these being the circle's image coordinates at time $t = 0$, the circle's velocity components, and the radius.

The models presented here are sought in a similar way; the parametrisation complicated mainly by the addition of the third dimension. For example, a moving sphere is sought using a fitness function with seven parameters, in much the same way as that of the 2D VHT example, although the method of voting will be shown to have added complications.

However, the manner in which they are described in this implementation's framework allows more abstract objects to be sought with relative ease in the design of models. Also, as will be later explained, performing TM on complicated models is not appropriate, and thus other approximate searching methods are used, although these will use the same fitness function as the TM algorithm. This implementation removes the modelling from both the searching method and from the representation of the underlying data, be it 2D, 2.75D or 3D sequences, as shown in figure 4.1. The motivation behind this was to ensure that the models used would be identical, and thereby prevent the possibility of producing errors between the different implementations of the same model. However, it also produces an elegant model description.

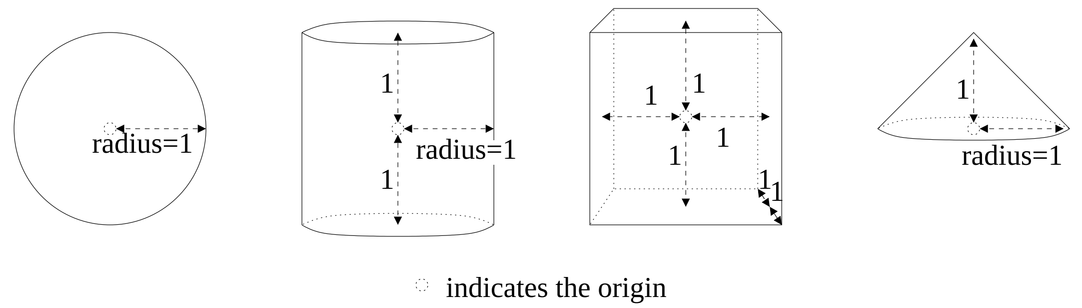


Figure 4.2: The four basic shapes.

4.3.4 Modules, basic shapes and CSG

The use of 'plug-in' modules is a key part of the implementation whereby a model can be used without knowledge of the underlying structures. These modules describe the model by the use of basic shapes; these are the sphere, the cylinder, the cube and the cone (see figure 4.2). The basic shapes can be warped by use of 4 x 3 matrices, thus allowing rotation, translation, scaling, and even shearing. For example, the basic sphere shape contains the volume described by a unit sphere located at the origin; the sphere can be scaled and translated, allowing a sphere of a specific size to be placed at a specified position. Such distorted shapes can also be intersected with each other thus enabling the generation of even more complicated shapes. It is important to note that the objects are solid, and are thus akin to constructive solid geometry (CSG) which is used extensively in computer graphics.

The function provided by each of these modules in essence describes the location, and orientation (etc.) of basic shapes. In addition, the special temporal parameter, i.e., time, is passed too; as well as indicating temporal information, it also governs which frame in the sequence the model is to be compared with. However, time is not deemed part of the parameter description of the model itself; it is simply an additional dimension that enables the model to be described for a particular frame in the sequence.

Thus for a moving sphere, eight parameters are required, which describe the sphere's starting position, velocity, and radius, and also the time for which the model is to be evaluated. The function then calculates where the sphere would be located at a given time, and produces a description of a single basic shape, that being the translated and scaled sphere. This description is then evaluated by the underlying structure, depending upon the representation of the data.

Summarising, each module provides a function, *model*, with the syntax $model(parameter\_list, time)$, which returns a description of a template for the particular set of parameters, in terms of basic shapes, at the specified instance of time.

Since for different models, a different number of parameters is required, another task to be performed by a module is to indicate the number of parameters required and validate the range of values that each of the parameters can take.

Therefore, the underlying representation must interpret the set of basic shapes and produce a measure of fitness; this is now explored.

4.3.5 2D basic shape evaluation

A model has produced a set of basic shapes, with respective warping matrices, for a particular set of parameters, and from these a measure of fitness, or suitability, must be obtained. Thus this stage, in combination with the model module, will evaluate the fitness for a set of parameters, and hence is equivalent to the fitness function that is used during TM.

As a simplified overview, it can be interpreted that shapes are mapped, one at a time, onto each of the binary images (they are binary as they have been segmented to indicate either foreground or background). From this mapping, an intersection is produced, and it is the summation over all of these pixels that produces a manner of voting. It is the addition of the summations over all of the images for a particular frame, and over all frames for a particular parameter description of the model, that produces the complete measure of fitness for a template, i.e.:

$$fitness(parameter\_list) = \sum_{frame=0}^{F} f_{2d}\left(model(parameter\_list, t(frame)), frame\right) \tag{4.2}$$

where $t(frame)$ converts a frame number into a value indicating a useful measure of passing time, and where $f_{2d}(basic\_shape\_list, frame)$ is a function that evaluates a set of basic shapes for a particular frame. Later, it will be shown (see section 4.5) that two measures are actually produced for each model, which are then combined to produce a more suitable fitness function.

The function f, the one related to the formation of the intersections and the voting in general, is now discussed followed by the full analysis of the sphere basic shape.

Overview

Contrary to the description above, the mapping is actually performed in reverse, i.e., the images are mapped onto the object. This is performed for simplicity. Similar to the 2.75D reconstruction algorithm, rays are projected from the cameras. For every pixel in an image from a particular camera, irrespective of whether the pixel is in the foreground or not, such a ray is cast through the space. If this ray intersects at some point with the basic shape, then the pixel has caused a *potential* vote. If the source pixel is actually from the foreground, then the pixel has also caused an *actual* vote.

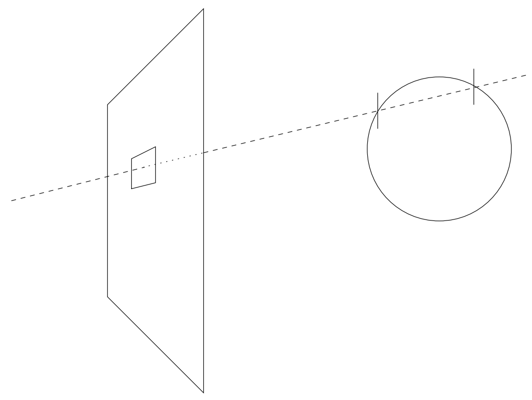


Figure 4.3: Projecting a ray and testing its intersection with a basic shape.

These two voting accumulators will be combined according to section 4.5. Figure 4.3 illustrates projecting a ray and intersecting it with the sphere.

The Sphere

The ray for the pixel p_{p_x,p_y} is projected from the origin of the camera, but using a representation of the space that yields a unit sphere at the origin. This is performed by producing a matrix $\underline{\underline{\mathbf{M}}}$ which is the result of multiplying the matrix that transforms the described sphere to a unit sphere (the inverse object's matrix) by the camera's projection matrix $\underline{\underline{\mathbf{P}}}$. The matrix $\underline{\underline{\mathbf{M}}}$ is a 4x3 matrix.

Thus the line is described by the vector equation:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{X} = \mathbf{X}_0 + \lambda \mathbf{X}_1 \tag{4.3}$$

where:

$$\mathbf{X}_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \underline{\underline{\mathbf{M}}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.4}$$

and:

$$\mathbf{X}_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \underline{\underline{\mathbf{M}}} \begin{bmatrix} p_x \\ p_y \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \tag{4.5}$$

This thus specifies λ as a measure of the unit orthogonal distance from the camera of the projected pixel. For a solid unit sphere, a pixel will be in the object if and only if:

$$x^2 + y^2 + z^2 < 1 \tag{4.6}$$

The values of λ for which the line enters and leaves the sphere are now sought, i.e.:

$$(x_0 + \lambda x_1)^2 + (y_0 + \lambda y_1)^2 + (z_0 + \lambda z_1)^2 = 1 \tag{4.7}$$

Expanding this yields:

$$\begin{aligned} 0 &= \left(x_0^2 + y_0^2 + z_0^2 - 1\right) + 2\lambda\left(x_0 x_1 + y_0 y_1 + z_0 z_1\right) + \lambda^2\left(x_1^2 + y_1^2 + z_1^2\right) && (4.8) \\ &= \left(|\mathbf{X}_0|^2 - 1\right) + 2\lambda\left(\mathbf{X}_0 \cdot \mathbf{X}_1\right) + \lambda^2 |\mathbf{X}_1|^2 && (4.9) \end{aligned}$$

and solving for λ gives:

$$\lambda = \frac{-\mathbf{X}_0 \cdot \mathbf{X}_1 \pm \sqrt{(\mathbf{X}_0 \cdot \mathbf{X}_1)^2 - |\mathbf{X}_1|^2\left(|\mathbf{X}_0|^2 - 1\right)}}{|\mathbf{X}_1|^2} \tag{4.10}$$

It is important to note that if the denominator is 0, the ray can be interpreted as not actually being projected. This should not occur with any of the transformations available, either in the camera matrix definitions, or in the matrix describing the object. For example, scaling an object by the value of 0 is not a reversible operation, and thus is prohibited.

If there are to be solutions, i.e., points at which the projected ray meets the sphere, then the contents of the square root must be greater than, or equal to, 0. Also, at least one of the solutions to λ must be greater than 0, otherwise the object is behind the camera; this is checked by testing whether $\mathbf{X}_0 \cdot \mathbf{X}_1 < 0$ or whether $|\mathbf{X}_0|^2 < 1$, where the latter can be interpreted as the camera being located within the unit sphere and thus guaranteeing a solution. Such checks are performed so that only the full calculation is performed when it is required.

The Cylinder, Cube and Cone

The other three basic shapes derive similar ranges for λ. However, for example, in the cube, a range of λ is calculated for the x-axis, then a second range for the y-axis, and a third for the z-axis; these are then combined producing an intersection of the three ranges. The complete set of shape descriptions are shown in table 4.1.

Shape	Restrictions
Sphere	$x^2 + y^2 + z^2 < 1$
Cylinder	$x^2 + z^2 < 1$, $-1 < y < 1$
Cube	$-1 < x < 1$, $-1 < y < 1$, $-1 < z < 1$
Cone	$x^2 + z^2 < (1 - y)^2$, $0 < y < 1$

Table 4.1: Restriction equations on λ for the basic shapes.

Using the limits of several objects, intersecting objects can easily be accomplished. This involves calculating the intersecting range of λ between all of the different objects. More complicated operations can be envisaged, such as union or difference operators, however, these were not implemented during the course of this research.

Voting

If pixels have passed the tests required, then they will exist within the required intersection. In these cases, as already mentioned, a counter indicating the potential vote is incremented and if the pixel happens to be in the foreground another counter indicating the actual vote is also incremented. These are combined as described in section 4.5.

Streamlining

In the above description, the matrix that warps the space from the global geometry to the basic shape's local geometry has been described. However, the matrix that enables the basic shape to be projected back into the global space, and from whence into a camera's local geometry, is also part of the shape's definition. This enables the limits of the 3D shape to be mapped onto a camera's 2D image plane, thereby restricting the number of pixels that should be tested. For example, the sphere exists within a cube defined by: $-1 < x < 1$, $-1 < y < 1$, $-1 < z < 1$ when described in its local geometry. The eight vertices of the cube are mapped into the global space and then to the local intrinsic geometry of a camera, producing up to eight points on the camera's image (there may be less if points are mapped to behind the camera). Comparing the coordinates of these points, a bounding rectangle is produced, and only pixels in this rectangle are then tested for their intersection with the object, as shown in figure 4.4.

If the camera exists within the shape, then it is debatable whether the object should be ignored since this would be an ill-posed problem—the camera could not be physically placed within the solid object for it to be of use; the implemented algorithm permitted the camera to exist within the shape and hence tested for the volume although this instance did not occur in any of the trials.

4.3.6 2.75D basic shape evaluation

There is great deal of similarity between the 2D and the 2.75D methods, with these methods differing mainly in their voting manner. For each shape, it is the range of λ that is sought, not just the presence of one solution. The 2D method has already indicated how two values of λ, and hence the range, can be calculated.

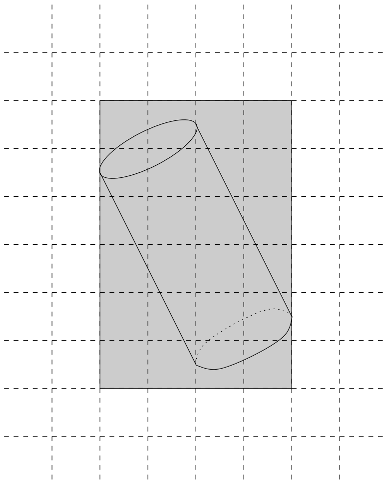


Figure 4.4: Reducing the searching area for a shape using a bounding box formed from the mapping of the object onto the image.

Voting

The voting method is now performed using volumes rather than areas or pixels. The range λ, denoted by $\lambda_0 \leq \lambda < \lambda_1$ corresponds to a volume in the 3D space. It is this volume that is compared with the 2.75D data.

First, considering a two dimensional case, the area of the projection will now be studied. Figure 4.5 shows the area that is being studied; it is a sheared rhombus.

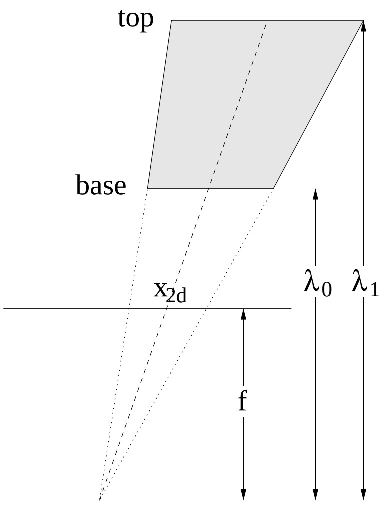


Figure 4.5: 2.75D voting as considered from a 2D framework.

Considering the lengths of the horizontal lines:

$$base = \frac{x_{2d}+0.5}{f}\lambda_0 - \frac{x_{2d}-0.5}{f}\lambda_0 = \frac{\lambda_0}{f} \quad (4.11)$$

$$top = \frac{x_{2d}+0.5}{f}\lambda_1 - \frac{x_{2d}-0.5}{f}\lambda_1 = \frac{\lambda_1}{f} \quad (4.12)$$

$$(4.13)$$

where f is the focal length of the camera.

A sheared shape has the same area as the original, and thus the area of the object can be simply stated as:

$$area = \frac{1}{2}(\lambda_1 - \lambda_0)\frac{\lambda_0 + \lambda_1}{f} \tag{4.14}$$

Increasing the number of dimensions, the volume is thus described by:

$$volume = \frac{1}{2}(\lambda_1 - \lambda_0)\frac{(\lambda_1 + \lambda_0)^2}{f_x f_y} \tag{4.15}$$

Note that there is no need for the other intrinsic parameters, namely skew and the principal point offset, to be included in the calculations, as these will not affect the result. However, if the evaluation is performed upon a λ represented in the local intrinsic space, then $f_x = f_y = 1$, thus the equations are further simplified.

Equation 4.15 demonstrates how a range of λ is converted into a volume, and hence in order to calculate the potential (i.e., the maximum possible) number of votes, no changes are required. However, each projected pixel may not have depths defined for the entire size, and thus the various depth entries must be compared and clipped to the required volume under test. Thus the resulting actual vote is incremented with the sum of all the possible sub-volumes.

Streamlining

The same bounding box method as used in the 2D algorithm to streamline the testing, is employed.

4.3.7 3D basic shape evaluation

The 3D system is the simplest to perform as no mapping is required to particular cameras, because once the scene has been reconstructed, it exists solely within the global voxel space. The voxel space is warped by the inverse shape matrix, and then voxels are tested for inclusion in the various objects.

Voting

Voting is done on a voxel basis—either a voxel is in the foreground or it is not. Again, a maximum possible total is noted, as well as the actual total number of voxels found.

Streamlining

Similar to the 2D and 2.75D methods above, the other matrix supplied with each shape is used to restrict the number of voxels that are to be tested.

4.4 Genetic Algorithms

4.4.1 The choice of algorithm

As indicated in section 4.3.2, it is impossible to construct and search the Hough parameter spaces for models that have a high number of dimensions or even those whose result must be calculated to a high level of accuracy. There are several methods available including the multi-resolutional HT, Simulated Annealing, Gradient Ascent/Descent and Genetic Algorithms. All of these methods are aided by wide peaks in the *voting domain*.

The multi-resolutional, or pyramidal, HT is mainly suitable for few parameters, but its resolution can be gradually increased to the required amount. It can unfortunately select the wrong peak from the outset.

Simulated Annealing attempts to guarantee that the global maximum of the voting domain will be found, although it might take an infinite amount of time to reach it. Multi-start Simulated Annealing would of course be preferable so that many regions can be tested in parallel.

Gradient Ascent attempts to improve upon its estimate of the peak by analysing the peak's immediate neighbourhood and calculating which the preferential direction to move in would be. It, however, can get impeded by local maxima, and once again a multi-start algorithm would be preferable. Unfortunately, the searching of the local space would be costly as the number of dimensions increase.

Genetic Algorithms (GAs) have been used before by, for example, Cunado et al. [15] and Yin [94], to locate peaks in the HT's accumulator space, due to their effectiveness at handling high dimensional space. It is the success these have had that made GAs the method of choice in this research.

4.4.2 Overview

In essence, GAs search parameter spaces by attempting to maximise a fitness function associated with a set of parameters. Searching accumulator spaces of the HT is an ideal application for such a system, as the function is the TM fitness function. Thus this is advantageous over the HT because the accumulator space does not need to be constructed, as the algorithm makes use of the HT-TM dual.

The algorithm implemented is based upon that described by Goldberg [26], however, it uses Gray coding to encode the parameters. This is also discussed by Goldberg [26] stating that it is believed to have advantages due to the unit distance between successive numbers, but no evidence for this has been given.

GAs are obviously based upon biological processes, and are thus described using biological terms. They work initially on a random population, where each member of the population has a chromosome composed of genes that encode the being's

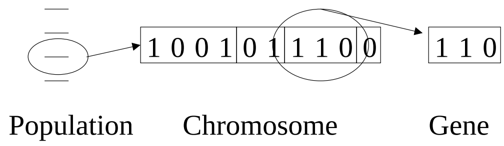


Figure 4.6: GA definitions.

individual parameters (see figure 4.6). The chromosomes are commonly a concatenation of all of the binary digits that are required to represent the different genes. For example, if a parameter can take any even value in the range $(4 < 14)$, then the respective gene is 3 bits in length as there are 8 possible values. It is important to note that with this representation, the number of possible values for a particular parameter must be a power of 2. Other representations exist where the genes are indivisible units which are simply the standard computer representation of their respective value.

For each population member, the parameters are extracted and passed to the function that is to be maximised. For example, in this context, the function is related to the template fitness function described in equation 4.2. The fitness of the population member is based upon this function.

Once the fitness of each member of the population has been calculated, the individual beings are then mated, or allowed to survive. This is performed by picking a pair of beings from the population and from these parents, two children are formed. The selection process is performed using a weighted *roulette wheel*. The wheel has the number of segments equal to the number of members in the population, but the spacing of this wheel is irregular, resulting in making fitter members more favourable for selection that less fit members. The actual spacing need not be directly proportional to the level of fitness; often a small bias is added to ensure the population is kept reasonably rich in values. It is entirely possible, and indeed common, for the same being to be picked twice from the pool, to form the two parents for two new children—the selection process does not remove members from the pool once they are selected.

A random variable is then tested, which thereby decides whether the selected pair are to survive or are to be mated. If a selected pair are to be mated, a random crossover point in their chromosome is chosen. The next generation from such a pair is created by using the first section of one of the beings with the latter part of the other being (see figure 4.7). A similar being is created from the remaining two sections. Since crossovers will commonly occur within a gene, children, although based upon their parents, will also have the potential of representing new values that were not originally within their parents' parameter set.

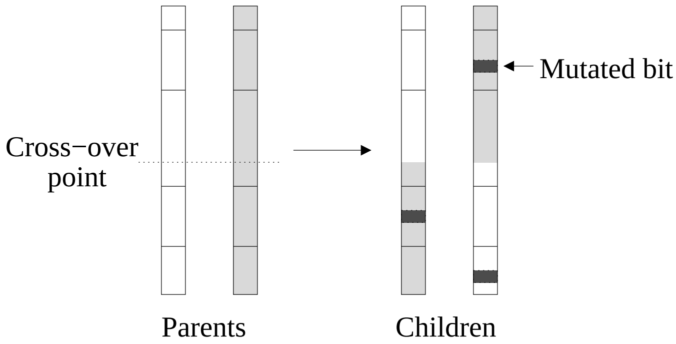


Figure 4.7: GA's cross-over.

To increase the richness of the resulting new population, random errors, or mutations, are allowed to occur during the mating process. If a selected pair are to survive without mating, their genome is also subject to the same random process of mutation.

A further rule, which was implemented, is to keep the member whose fitness is best, thereby ensuring that the system's population will never degrade.

With the new population formed, the process is then iterated until there is sufficient convergence on a particular value. This latter point, however, is subject to the choice of many of the probabilities and scale factors used within the algorithm, and thus it is common to have a stop criterion based upon the number of iterations. Such a criterion was used in this research, and a grid search was then performed on the local neighbourhood afterwards in order to locate the peak in the parameter space more quickly.

4.4.3 Choice of parameters

The Gray coding was used for the parameters as it was believed that by placing subsequent values within a single bit change, the small mutation errors could enable the traversal of the local neighbourhood around the different population members. There are, however, further issues that must be considered when using parameters under a GA.

The first issue is that cyclic parameters should be avoided; cyclic parameters are those that can encode angles. They are hazardous because, for example, 0° is very similar to 359°, and without a special encoding of the parameter they would be placed far apart in the voting domain. Highlighting this:

$$fitness = f(r, \phi, t) = g(r \cos(t + \phi)) \tag{4.16}$$

the above equation would be poor for this reason, however:

$$fitness = f(a, b, t) = g(a \cos t + b \sin t) \tag{4.17}$$

where $a = r\cos\phi$ and $b = r\sin -\phi$, provides exactly the same model, but without the aforementioned problem.

Second, although it is obvious that parameters should be orthogonal, there are sometimes hidden dependencies. For example a simple moving object could be described with a start point and a velocity. However, should the start point be varied during the maximisation of the fitness function (as is likely), the trajectory could be completely altered and the end point of the sequence would no longer be reached without a change in the velocity as well. This has become apparent from watching the results of the GA battle with moving the start point and then adjusting the velocity to compensate so that the appropriate end point can be reached once again. Although a different encoding is often unnecessary, it is worth pointing out that specifying a start and end point may aid convergence to a quicker solution.

4.4.4 Increasing the size of the peak

Increasing the complexity of the mathematical models exponentially increases the size of the accumulator array, thus constructing the accumulator space is infeasible, except for basic models. However, GAs provide a means to search such large dimensional spaces, without the requirement for the space to be represented or constructed, and although their approach is inherently random, with careful use they will converge on or near the peak of the accumulator. This can be aided by ensuring that the peaks of the accumulator array will be wide; peaks can be widened by either blurring the source data, or by searching for volumes rather than edges.

The narrowness of standard HT peaks is due to the data using edges not areas. The first advantage of this is that there are of course reduced computational costs. If, however, areas were used, the peak becomes much wider. In figure 4.8, three lines indicate three different methods of locating a circle using a circle.

The narrowest peak belongs to seeking an edge detected circle with a circle outline. It can clearly be seen to have many local maxima to the sides of the main peak. The curved line indicates searching for an edge detected circle with a filled circle template, or, searching for a filled circle with an outline circle template. The first of these combinations would be prone to noise, as it would quickly seek out regions of high change. The latter of these interpretations would produce a poor response if the template had a larger radius than that of the object being sought; it would suffer the same problems as the last of the graphs. This last graph is seeking filled circles with filled circle templates. The main problem with this is that as there is no edge information, a circle can easily be found lying within a square. If, however, the circle is allowed to expand, then using an appropriate measure, as discussed below, even though a circle may be found within a square, a circle would match another circle much more preferably.

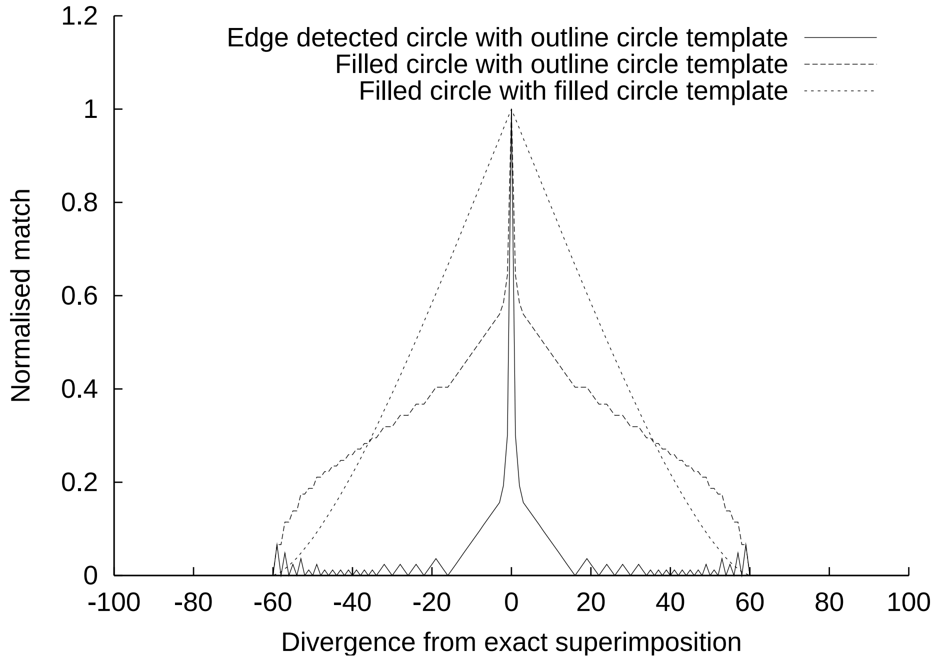


Figure 4.8: Voting spaces resulting from the search for circles of unknown position.

This is thus the reason solid objects are sought, rather than areas, in these 3D analyses. The emptiness of the surrounding space of objects separated them sufficiently that edge information was not required and the templates were able to 'grow' into the required bodies.

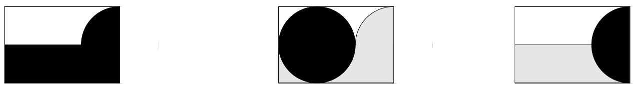
This theory is further illustrated by studying the results of Yamany et al. [93] who researched fitting 3D surfaces to reconstructed objects. One of the examples given demonstrates only a small 15% match, but the description actually appears to yield a good solution. It is thus unsurprising that the parameter ranges for the extraction were restricted. It could be concluded that noise would thus be a significant problem for such systems.

A further method to add edge information is discussed in chapter 7. This was not implemented as the emptiness of the 3D world in the experiments was sufficient for the parameter extraction. However, for further, noisier, analyses this alternative method may prove to be preferential.

4.5 Model weighting

4.5.1 The problem

When dealing with volumes in 3D, or areas in 2D, a template fitness function that is solely dependent on the actual number of votes v_a for a particular template cannot be used if the parametrisation allows the size of the subject to be altered. If, for example, in 2D a circle of unknown radius is sought, the best solution would be the biggest circle that can be represented since it would encompass the greatest area and thus increase the possibility of receiving votes due to noise. This is further illustrated in figure 4.9a where the black circle is sought. The circle in figure 4.9b would receive the same number of votes as the circle in figure 4.9c although it would appear the latter should be the better choice since all of the possible pixels vote for it. To overcome these problems, the number of votes could be divided by the potential number of votes the template could gain, v_p, thus the circle in figure 4.9c would now be preferred since all of the possible pixels in the circle are represented. Also big circles would not be preferable because the ratio of the number of votes to the maximum number of votes would reduce to the level of the noise. However, simple division cannot be performed since the best match would become the smallest circle allowed since this reduces the possibility of erroneous values inside the true data reducing the fitness.



(a) (b) (c)

a) the original image b,c) two possible circles that have equal numbers of votes.

Figure 4.9: Non-weighted voting for circles.

To overcome this, the suitability function is defined as:

$$w = \frac{v_a}{v_p + k} \tag{4.18}$$

where k is a constant related to the level of noise expected.

4.5.2 Suitable values for k

The selection of a suitable value for k is now investigated by expanding upon the example for the search for the circle. We shall say that there is some vote density d_i that exists for the area enclosed inside a circle of radius R, and some vote density d_o for the area beyond. The condition $d_i > d_o$ is assumed, otherwise there would be no reason to be searching for such a circle.

We wish to choose k so that equation 4.18 is a maximum for the circle of radius $r = R$.

The number of votes given to a circle of radius r is given by:

$$v_a(r) = \begin{cases} d_i \pi r^2 & r < R \\ d_i \pi R^2 & r = R \\ d_i \pi R^2 + d_o \pi (r^2 - R^2) & r > R \end{cases} \tag{4.19}$$

and the potential number of votes is given by the area, i.e.:

$$v_p(r) = \pi r^2 \tag{4.20}$$

Let us now say that $k = \pi R^2$. This thus yields:

$$w(r) = \begin{cases} \frac{d_i \pi r^2}{\pi r^2 + \pi R^2} & r < R \\ \frac{d_i}{2} & r = R \\ \frac{d_i \pi R^2 + d_o \pi (r^2 - R^2)}{\pi r^2 + \pi R^2} & r > R \end{cases} \tag{4.21}$$

This produces the results shown in figure 4.10 where the circle radius has been normalised ($R = 1$), as has d_o. The graph shows that for $d_i > 2d_o$, a peak clearly exists at the required radius.

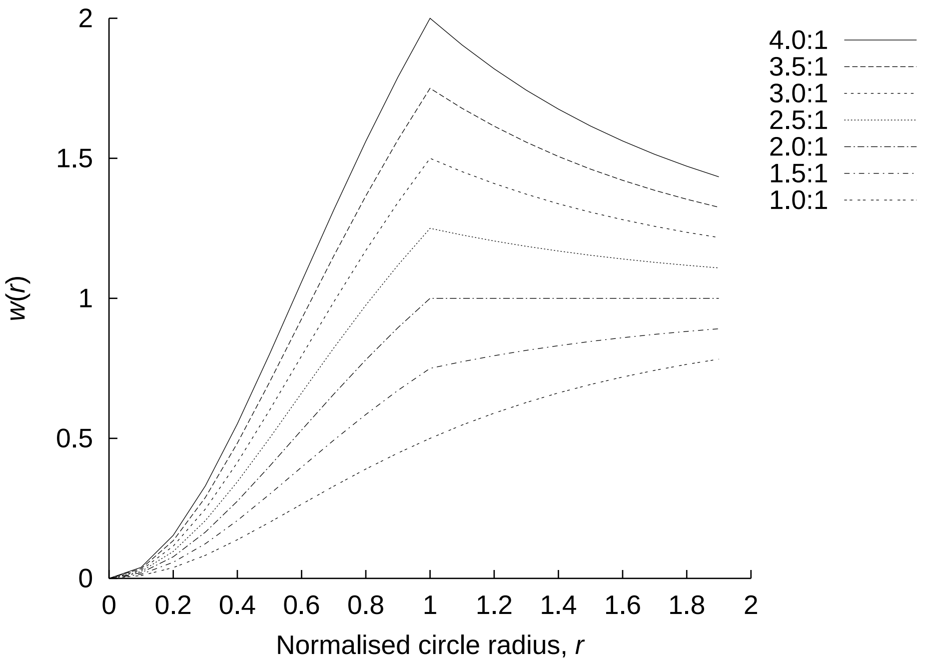


Figure 4.10: The template match of various radius of the template with various relative noise levels of the inside of the circle to outside the circle.

More generally, introducing an area function $a(r)$, the weighting function can be defined as:

$$w(r) = \begin{cases} \frac{d_i a(r)}{a(r) + a(R)} & r < R \\ \frac{d_i}{2} & r = R \\ \frac{d_i a(R) + d_o (a(r) - a(R))}{a(r) + a(R)} & r > R \end{cases} \tag{4.22}$$

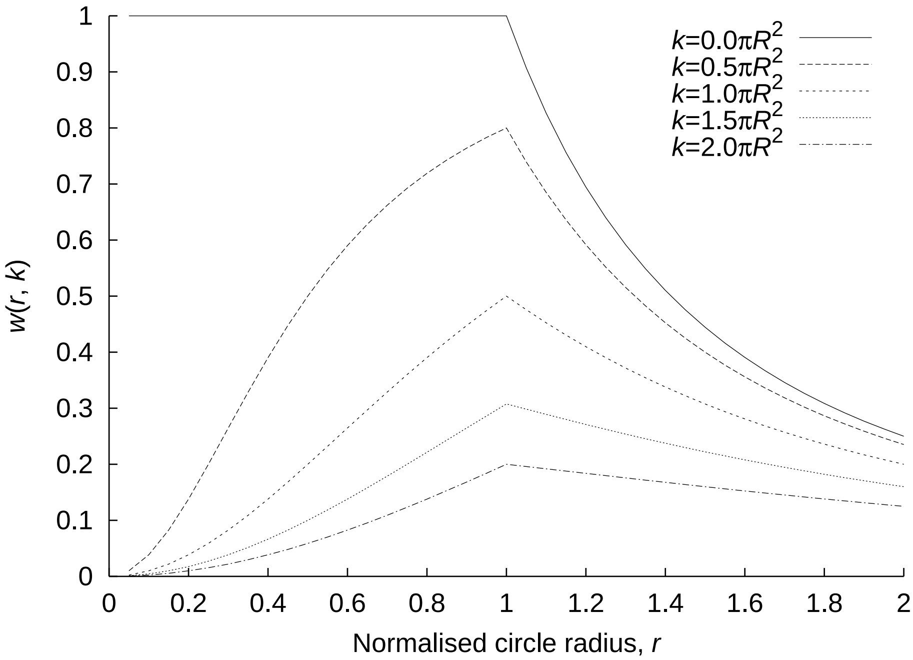


Figure 4.11: The template match of various radius of the template and various constants k, but with a fixed source noiseless circle template.

It can thus be derived that:

$$\frac{dw(r)}{dr} = \begin{cases} \frac{d_i a(R) \frac{da}{dr}}{(a(r)+a(R))^2} & r < R \\ 0 & r = R \\ \frac{(2d_o - d_i) a(R) \frac{da}{dr}}{(a(r)+a(R))^2} & r > R \end{cases} \tag{4.23}$$

Since, for a meaningful template, $\frac{da}{dr} > 0$, as long as $d_i > 2d_o$, a peak is ensured in the weighting function.

In practice, the value of k is estimated to have a comparable value to that of the maximum values expected. An estimate is sufficient since for values close to the peak, the fact that d_i could be less than the local surrounding $2d_o$ would cause a natural expansion into the required area. This can be seen in figure 4.11, where the source radius has been fixed at the radius $R = 1$, but the template size and the constant k are allowed to vary; the vote density within the circle of radius 1 is fixed at 1, and the density outside the circle is 0. This graph clearly indicates that the peak is located at the correct radius of $R = 1$ for such a noiseless example, for any value of k except for $k = 0$. Large values of k can be seen to be a poor choice as the regions neighbouring the peak are less distinguishable; small values are poor as they could become less significant than any noise that may be present.

4.6 Examples

Two examples are outlined below that demonstrate the varied use of models, with the first drawing together all of the techniques discussed.

4.6.1 Moving ball

If it is required that a moving sphere be extracted from a sequence of images from multiple cameras, then a representation must first be specified, i.e., whether 2D, 2.75D or 3D data is required. For the latter two, this would involve analysing the scenes on a frame-by-frame basis, thereby producing a 2.75D or 3D sequence of frames, as explained in chapters 3 & 2 respectively. For all three representations, the data would then be passed through the appropriate background removal filter, as described in section 4.2. The choice of the representation is governed by factors that will become apparent in chapters 5 & 6.

Having the appropriate sequence, a pre-defined model for the moving ball is utilised to give the positions of the basic shapes required.

The model

The moving ball is a sphere that moves at a constant velocity, with its radius also constant but unknown. Therefore there are seven parameters that describe the model. These are described in table 4.2

Parameter	Description
x_0, y_0, z_0	The ball's position at time $t = 0$
v_x, v_y, v_z	The ball's velocity
r	The ball's radius

Table 4.2: The 7 parameters required for the moving ball model.

From these it is apparent that the model will require only the basic shape of the sphere, which would be scaled and then translated, giving the two matrices required for the shape's description:

$$\underline{\underline{\mathbf{T}}}_{shape\_to\_global} = \text{translate}\left(\left[x_0 + tv_x \; y_0 + tv_y \; z_0 + tv_z\right]^{\mathsf{T}}\right) \text{scale}\left(\left[r \; r \; r\right]^{\mathsf{T}}\right) \tag{4.24}$$

$$\underline{\underline{\mathbf{T}}}_{global\_to\_shape} = \text{translate}\left(-\left[x_0 + tv_x \; y_0 + tv_y \; z_0 + tv_z\right]^{\mathsf{T}}\right) \text{scale}\left(\left[\frac{1}{r} \; \frac{1}{r} \; \frac{1}{r}\right]^{\mathsf{T}}\right) \tag{4.25}$$

where the functions 'translate' and 'scale' produce suitable matrices, and thus matrix multiplication is present thereby indicating the 'right-to-left' order of evaluation.

The value of t is passed along with the other parameters, and thus for a particular point in time, and hence frame, for a particular template, the basic shape structure has been defined.

Parameter ranges

Having produced a model, the range of the parameters that describe it must be defined, as must the stepping increment. For example, seeking a ball approximately the same size as a basketball, the range of the radius could be restricted to $100\,\mathrm{mm} < r < 140\,\mathrm{mm}$, and 5 mm steps might be an appropriate discretisation. However, the larger the range, and the smaller the steps, the more possible combinations there are, and thus the GA would take longer to converge.

Evaluation of the model

The GA produces a set of parameters that are required to be tested. The potential vote counter and the actual vote counter are set to 0, and then the model is evaluated over all of the frames, with each frame requiring a new set of matrices for the basic shape.

The shape's description is passed to the function appropriate for the dimensionality of the representation. For the 2D and 2.75D systems, the shape is evaluated for each of the camera views. The matrix $\underline{\underline{\mathbf{P}}}_i \underline{\underline{\mathbf{T}}}_{shape\_to\_global}$, where $\underline{\underline{\mathbf{P}}}_i$ is the projection matrix defined in equation 2.12 for the ith view, is used to map the eight corners of the cube, which encloses the sphere, onto the respective view; each of the pixels that lie within the bounding box formed from these eight mapped points are evaluated. The pixels are first tested to see whether they will intersect with the object, using the matrix $\underline{\underline{\mathbf{T}}}_{global\_to\_shape} \underline{\underline{\mathbf{P}}}_i^{-1}$ to map the local geometry of view i into the local geometry of the shape. If they do intersect with the shape, the values of λ are calculated, checking that at least one of these will lie in front of the camera. In the 2D system, if these tests have passed, then a potential vote is generated, and if the pixel was actually selected, an actual vote is produced. In the 2.75D case, the range of λ is used to calculate the potential vote, and again if the pixel was actually selected, the subsections of the range of λ for which the pixel is defined for are used to produce individual increments to the actual vote.

For the 3D system, the matrix $\underline{\underline{\mathbf{M}}}_{global\_to\_voxel} \underline{\underline{\mathbf{T}}}_{shape\_to\_global}$ is used to map the eight corners of the cube, which encloses the sphere, onto the voxel space, where the matrix $\underline{\underline{\mathbf{M}}}_{global\_to\_voxel}$ performs the necessary scaling and translation of the voxel space (cf. $\underline{\underline{\mathbf{W}}}^{-1}$ defined in section 3.3.3). All of the voxels that lie within the bounding region of the voxel space are then tested for inclusion in the object. The voxels are mapped onto the shape using the matrix $\underline{\underline{\mathbf{T}}}_{global\_to\_shape} \underline{\underline{\mathbf{M}}}^{-1}_{global\_to\_voxel}$, and if they exist within the object, using the test that the resulting x, y, and z fit the equation for a unit sphere located at the origin, they produce a potential vote.

If the voxels are also in the foreground, they produce an actual vote. These votes are accumulated over all of the frames.

The potential vote counter and the actual vote counter are combined using equation 4.18. However, a constant is required which should be estimated prior to the evaluation of the model.

Selecting a constant

As explained in section 4.5, the constant should take the value equal to the potential vote counter of the object being sought. However, the value is not known prior to the analysis, and thus must be estimated. The 3D and 2.75D representations are the simplest for this value to be estimated: for the 3D representation, the volume of the sphere in voxels should be estimated, and multiplied by the number of frames in the sequence. For the 2.75D representation, the volume of the sphere, in the units in which the global representation is defined, must be estimated, and multiplied by both the number of views and the number of frames in the sequence. For the 2D representation, an estimate of the size of the sphere on each of the views should be estimated, and again multiplied by the number of views.

The estimations, however, do not need to be too accurate; if the GA returns an object whose potential number of votes is significantly different to the constant, then the process should be repeated but using the potential number of votes as the new constant. Chapter 7 presents a possible method that should be investigated that could remove the requirement for this constant to be estimated.

The result

The GA will iteratively produce different populations to test, and these are evaluated. Eventually, the system will converge, or the maximum number of iterations allowed will occur. In either case the process stops, and the best population member is found. Once this has occurred, it is useful to check that the best template's parameter set resides at the true peak—it is common for the GA to produce a template that lies near to but not at the peak. Thus performing a complete search of the local neighbourhood will attempt to ensure that the peak found is the true peak of the fitness function.

4.6.2 Gait analysis

Background

Previous work by Cunado et al. [15] investigated a 2D gait extraction and description method where single lines that oscillated in a pendulum manner were sought. From the edge-detected source images, a single pendular line from the thighs was located by an evidence gathering procedure adapted from the VHT formulated by Nash et al. [59]. No tracking was involved, and the evidence gathering nature of the system made it extremely tolerant to noise and occlusion, with the capability to handle time lapse imagery.

By analysing the frequency components of the angles that the pendular lines made with the vertical, as shown in figure 4.12, a biometric was produced from the phase weighted magnitude spectrum. For the small sample of subjects, this method produced a 100% recognition rate.

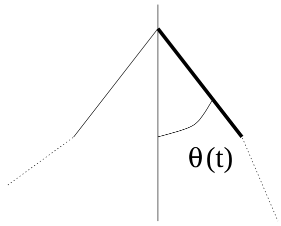
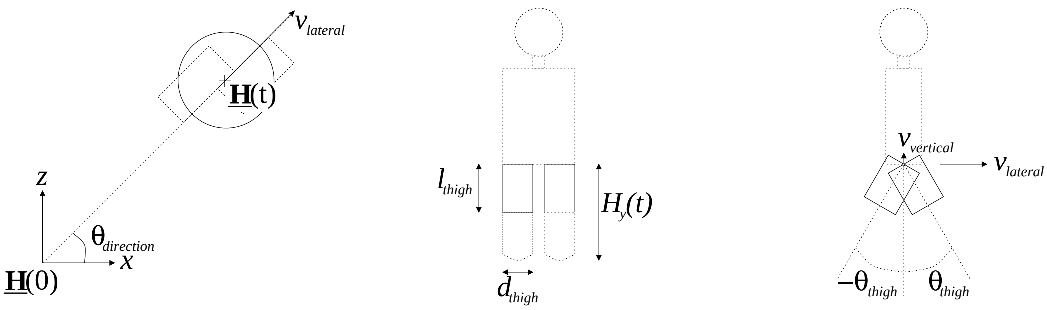


Figure 4.12: The pendular nature of the thighs used by Cunado et al. [15].

Parameter description of the model

The third dimension allows for a much more detailed model to be applied to the data, and subjects would also be permitted to walk at any angle to the cameras instead of the simple planar motion used by Cunado et al. [15], giving true freedom of motion. The swing in the hips and differentiating between the left and right legs are just two examples of the advantage of manipulating the information in 3D. The model, described in figure 4.13 and table 4.3, is used in the later example of human gait description.



(a) plan view (b) from the front (c) from the side

Figure 4.13: A 3D basic human gait model constructed from cylinders.

This model is comprised of two cylinders that represent the thighs; this compares directly with the lines used in the 2D work by Cunado et al. [15]. These oscillate in a pendulum-like motion about a pivot point, that being the hip. This pivot point moves with a constant velocity through the space, although harmonics are applied to this in the direction of the gait. Also, the hip is permitted to have oscillatory motion in the vertical direction.

There is potential for a larger number of harmonics to be present, noting the restrictions placed by Nyquist's sampling theorem, which would be required to describe the oscillations in the thighs, however, previous research into gait has indicated that recognition can be performed with only the first two harmonics. Hence there are 23 parameters which are described in table 4.3.

Parameter	Description
$\mathbf{H_0}$	Hip 3D central position at time $t = 0$.
H_{width}	Hip width.
T_L	Thigh length.
ω_0	Step rate in full gait cycles per second. Period $= \frac{1}{\omega_0}$.
$\mathbf{V}$	Mean velocity of the person in the X and Z directions.
V_2^*, V_4^*	First two harmonics of oscillations of hip position orientated in direction of $\mathbf{V}$.
H_2^*, H_4^*	First two harmonics of oscillations of hip position vertically.
T_0	Mean angle of the thighs.
T_1^*, T_2^*, T_3^*	Fundamental and first two harmonics of oscillations of the thigh angles.

Table 4.3: The 23 parameters required for simple gait recognition. Parameters marked with a '*' are complex.

Understanding the model parameters

Consulting table 4.3, the central position of the hips is given by:

$$\begin{aligned}\mathbf{H}(t) = \quad & \mathbf{H_0} + t\begin{bmatrix} V_x \\ 0 \\ V_z \end{bmatrix} + \begin{bmatrix} \frac{V_x}{|\mathbf{V}|} \\ 0 \\ \frac{V_z}{|\mathbf{V}|} \end{bmatrix} \mathrm{Re}\left\{V_2 e^{-4j\pi\omega_0 t} + V_4 e^{-8j\pi\omega_0 t}\right\} \\ & + \mathrm{Re}\left\{H_2 e^{-4j\pi\omega_0 t} + H_4 e^{-8j\pi\omega_0 t}\right\}\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \end{aligned} \tag{4.26}$$

the angle of the left leg is given by:

$$\theta_l(t) = \mathrm{Re}\left\{T_0 + T_1 e^{-2j\pi\omega_0 t} + T_2 e^{-4j\pi\omega_0 t} + T_3 e^{-6j\pi\omega_0 t}\right\} \tag{4.27}$$

and the angle of the right leg is similar, but lagging by half a period:

$$\begin{aligned}\theta_r(t) &= \mathrm{Re}\left\{T_0 + T_1 e^{-2j\pi\omega_0\left(t-\frac{1}{2\omega_0}\right)} + T_2 e^{-4j\pi\omega_0\left(t-\frac{1}{2\omega_0}\right)} + T_3 e^{-6j\pi\omega_0\left(t-\frac{1}{2\omega_0}\right)}\right\} \\ &= \mathrm{Re}\left\{T_0 - T_1 e^{-2j\pi\omega_0 t} + T_2 e^{-4j\pi\omega_0 t} - T_3 e^{-6j\pi\omega_0 t}\right\} \end{aligned} \tag{4.28}$$

Note that the hip harmonics are at twice the frequency of those in the thighs due to the addition of sinusoidal motion caused by the two legs. Also note that cyclic parameters are not present in the description as these are not trivial to encode successfully into the GA, as indicated in section 4.4.3—the parameter T_0 is in essence cyclic, but in practise it is defined over a limited range and thus can be termed as monotonic.

The legs are assumed to be symmetric in motion, i.e., no limping was modelled. If asymmetric gait was to be considered, it could be modelled as an oscillation of the parameter T_0 as this would yield only relatively small values, but this would also imply that the hips will now have a fundamental frequency equal to that of the thighs. Oscillating T_0 actually produces a multiplicative model which is believed to be advantageous over modelling the thighs separately due to the predicted small amplitudes of the harmonics of T_0. Such alternative descriptions should be investigated in future research.

Production of the basic shape descriptions

Using equations 4.26, 4.27 & 4.28, the two cylinders that represent the template for the thigh motion will thus be explained by the two pairs of matrices. The matrix that converts the left thigh's cylinder from local to global space can be described as:

$$\begin{aligned}\underline{\underline{\mathbf{M}}}_{left} &= \text{translate}\left(\mathbf{H}(t)\right)\text{rotate}_y\left(\tan^{-1}\frac{V_z}{V_x}\right)\text{rotate}_z\left(\theta_l(t)\right)\\ &\quad\ \text{translate}\left(\begin{bmatrix}0\\ -\frac{T_L}{2}\\ \frac{H_{width}}{4}\end{bmatrix}\right)\text{scale}\left(\begin{bmatrix}\frac{H_{width}}{4}\\ \frac{T_L}{2}\\ \frac{H_{width}}{4}\end{bmatrix}\right)\end{aligned} \tag{4.29}$$

Here, the scaling creates the correctly sized cylinder, whose diameter is half of the hip width, and whose overall length is the length of the thigh T_L. Translation is then performed to move the thigh to the correct point relative to the hip position, which involves placing it so that its top is correctly placed at $y = 0$ (relative to the hip) and so it is no longer central to the body, but offset, thereby giving room for the other leg. The rotation about the z-axis places the thigh at the correct angle according to equation 4.27, and the rotation about the y-axis ensures the body is facing the correct direction. The hip position is then moved with the final translation.

Streamlining

Applying this model directly to the data is not suitable due to its complexity. Therefore it is simpler to look for the basic components such as the direction of motion first, and then the various harmonics can gradually be incorporated into the search. However, it is important to note that when fewer harmonics are searched for, values may be produced that are significantly different to when a search is made for more harmonics. For example, this is a noticeable effect between searching for just the fundamental frequency of the thigh angle and searching for the fundamental and first harmonic frequencies.

4.7 Conclusion

The last of the two stages of all of the three systems, these being the background removal and the parameter extraction algorithms, have now been described. An estimate of the order of processing for the general case is unfortunately not possible since technically there is no general model that can be assumed; a qualitative example of the relative processing times for the extraction stages are presented in section 6.6.

A by-product of the extraction algorithm is that two methods to evaluate the quality of the reconstruction algorithms have also been produced. The first method is provided by the TM algorithm itself, which can also evaluate static scenes. Therefore, a reconstructed scene can be tested to see how closely it matches the original model. This is aided by the second by-product, which is that the TM algorithm can be used in reverse so that templates are produced instead of tested, thereby creating perfect data. Both of these methods are used in the following chapter which compares the reconstruction algorithms under various conditions.

Chapter 5

Comparison of reconstruction methods

5.1 Introduction

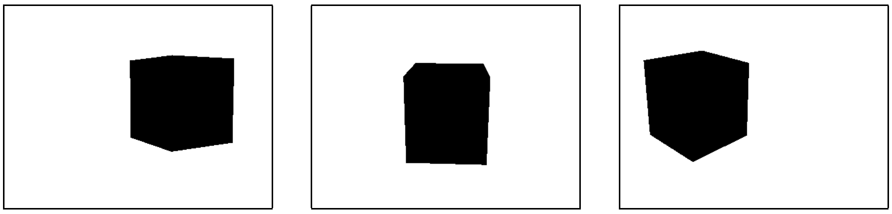
In chapter 4, methods by which objects can be extracted and parametrised by the three systems were described. Since static data is a subset of dynamic data, these different systems are thus capable of demonstrating and evaluating the effectiveness of the reconstruction algorithms themselves. The use of ray-tracing provides an accurate means for a 3D artificial object to be rendered onto 2D images. These images can then be analysed using the various reconstruction algorithms described in chapters 2 & 3 to produce their respective representations of the original scene. Using the parameter extraction algorithms, the parameters of the best matching template can be found, as can a measure of the fitness of the original object in this rendered space. Ray-tracing is not the only source of data; as indicated in chapter 4, the templates of the objects can be used to create perfect data rather than test it.

The aim of this chapter is to compare the different reconstruction algorithms under a number of conditions. These include the investigation into errors caused by the discretisation of the representation itself, and the effects that incorrectly calibrated cameras have on the data.

However, although only two reconstruction algorithms have been described, i.e., the 2.75D and 3D algorithms, the 2D system is also evaluated in many of the circumstances; the 2D reconstruction can be thought of as an implicit part of the extraction phase.

5.2 Experimental set-up

In all of these experiments, three views are used. The intrinsic and extrinsic parameters are the same as those used in the capture of the real data that is analysed



Camera α. Camera γ. Camera β.

Figure 5.1: An example of perfect data with a cube whose scale factor is 240. The irregular ordering of the camera names is for ease of comprehension of the scene.

in chapter 6. The required parameters are described in appendix section A.3.

The resolution of the 3D voxel grid was set so that the descriptions produced were of the same order as the average size of the 2.75D data (in bytes). In fact, the 3D voxel representations were approximately 50% greater in length.

5.3 Discretisation

As described in section 3.3.2, the initial data capture is of course a source of discretisation noise, however, it is the introduction of further discretisation noise that will now be studied. The hypothesis is that the 3D representation introduces further discretisation noise but those of the 2D and 2.75D systems do not.

5.3.1 The perfect reconstruction

This first study shows how the representation affects the resolution of the object's parameters that can be recovered. To perform this, a perfect representation of a cube is created by inverting the function of a cube template so that it produces the 3D data rather than analyses it. The data is perfect since they portray the true template shape, and are thus better than the visual hull (see chapter 1) that the three views would produce from the 2D silhouette images.

The cube is centred at the position $(0, 1000, 500)$ mm; the use of units is arbitrary, but is present to show the correspondence to the real world data that will be seen in chapter 6. Using a varying scale factor, data describing the different 'source' cubes is created. This reconstructed data, be it in 2D, 2.75D or 3D, is then analysed by using template matching (TM), testing for a range of cube sizes. Thus, for each source cube size, s_s, the response of a range of the template cube sizes, s_t, is produced. The peak of this response should be at the point $s_t = s_s$, i.e., the template that best fits the source data is the cube whose parameters are the same as the source. If discretisation is an issue, then for several source cube sizes, the response to a range of template cube sizes will be the same. Figure 5.1 demonstrates the perfect data that is observed by the three cameras for a cube whose scale factor is $s_s = 240$ mm.

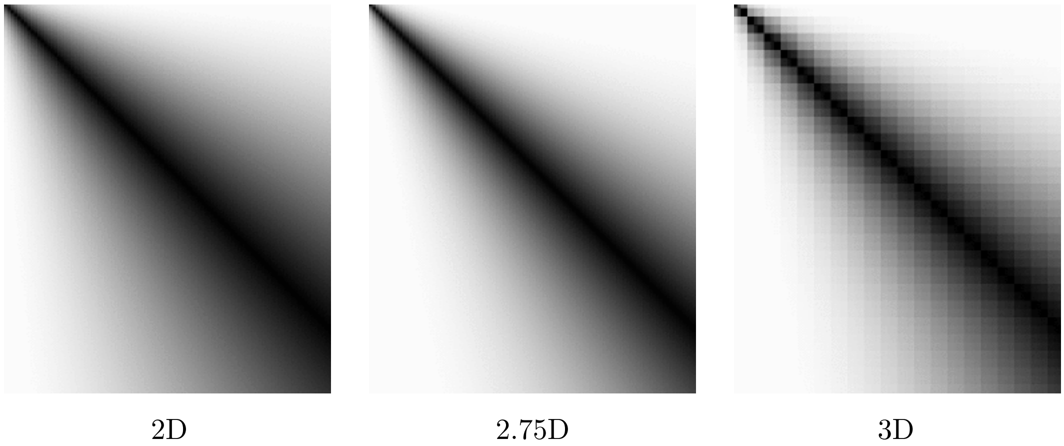


Figure 5.2: Discretisation effects using perfect data of many source cube sizes. Source cube's size increases from left to right; test template's size increases from top to bottom. The darker the point, the higher the fitness.

In this experiment, the suitability of a template is that described in section 4.5. Recapping, the fitness function is:

$$w = \frac{v_a}{v_p + k} \tag{5.1}$$

where v_a is the actual number of votes a template receives, and v_p is the potential, i.e., maximum, number of votes a template could receive. The constant k, is chosen in this experiment, in line with the indication in section 4.5, as being the volume or area of the cube being sought (dependent on the representation being used). Hence the maximum suitability value that is found for any of the algorithms is 0.5.

The sizes of the source cubes and templates start at a value of 10 mm and increase to 500 mm and 600 mm respectively in steps of 5 mm.

The results can be seen in the images of figure 5.2, where grey-scale is used to indicate the suitability of the various combinations. The most noticeable difference in these images is the 'pixelised' nature of the 3D response. This effect is solely due to the voxel representation reducing the fidelity of the result. The 2D response is the best that could ever be hoped to be achieved by a reconstruction algorithm, showing only a slight amount of this phenomenon where it is the resolution of the image that is the limiting factor. Although it is not clear in these diagrams, the 2.75D algorithm has the potential for better localisation than the 2D algorithm; intimate knowledge regarding the 3D nature of the original object is known, represented, in theory, by a set of depth ranges in the $\mathbb{R}$ domain, and are thus providing continuous information.

Figure 5.3 illustrates the response of matching a cube against two different cube sizes (i.e., two vertical lines in the images of figure 5.2). The discrete nature of

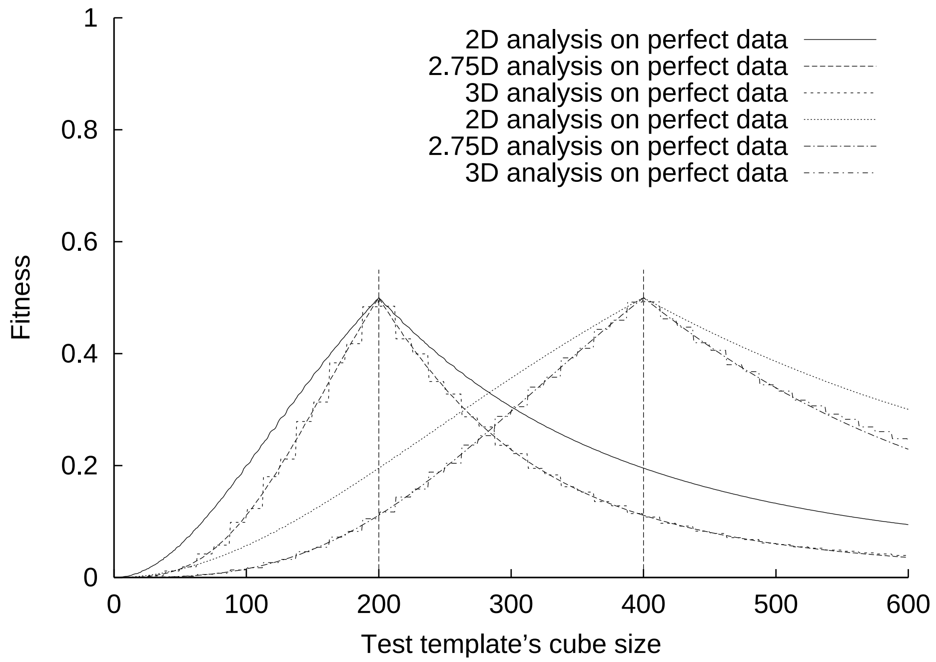


Figure 5.3: Graph of discretisation effects of two sample source cube sizes, using perfect data. The two sizes are indicated by the vertical lines.

the 3D response is very apparent, as is its and the 2.75D algorithm's more peaked response. It is clear that there are 8 steps in the 3D response for each 100 mm that the template size is increased. However, for a scale factor increase of 100 mm, the cube's sides have actually increased by 200 mm, due to the definition of the 'unit cube' being $\{-1 \leq x < 1, -1 \leq y < 1, -1 \leq z < 1\}$. Thus the resolution of the lengths of the sides is 25 mm, which is the same as the resolution of the voxel grid itself. At the depth of the object, the possible resolution capability is approximately 10 mm for each view, although the combination of the various views enables an increased fidelity; only by increasing the resolution of the voxel grid will the maximum fidelity be achieved.

The increased peak effect of the 2.75D and 3D algorithms is due to their use of volume as the means of measurement, instead of areas on the images (2D). This can be described by equations 5.2 & 5.3, for a test cube whose size, s_t, is compared with the source data formed from the source cube, which is of size s_s.

$$m_{area}(s_t) = \frac{1}{s_t^2 + s_s^2} \left| \begin{array}{ll} s_t^2 & \text{for } s_t < s_s \\ s_s^2 & \text{otherwise} \end{array} \right. \tag{5.2}$$

$$m_{volume}(s_t) = \frac{1}{s_t^3 + s_s^3} \left| \begin{array}{ll} s_t^3 & \text{for } s_t < s_s \\ s_s^3 & \text{otherwise} \end{array} \right. \tag{5.3}$$

For this perfect data, the response curves, for all of the algorithms, have a maximum located correctly at the source cube's size, noting that the 3D response has a wide plateau rather than a peak thus making it impossible to be certain of the original source cube's size.

5.3.2 Reconstructing from perfect images

If, instead of using perfect 2.75D and 3D data, the data was reconstructed from the perfect source 2D images using Volume Intersection (VI), different response curves are produced. As in the previous experiment, there is a discrete nature apparent in the 3D response; this can be seen in figure 5.5. However, the difference in the responses is that the peak is no longer located at the correct size, but at a slightly larger value. The cause of this is the observed visual hull; as explained in section 1.3.2, this is not the same as the visual hull of an object, since it is dependent on the positioning of the cameras—the visual hull of an object is as a result of the object itself, and cannot be reduced no matter what external camera positions are used. The visual hull of a cube is thus itself as there are no concavities. The observed visual hull, however, has been formed from the intersection of the three views, and is thus bigger than the source cube. Since the region directly surrounding the cube itself is within this observed visual hull, the density of 'actual' votes directly surrounding the cube is not 0, and may be greater than the crucial factor of 0.5 (see section 4.5). If this is the case, then the weighting factor will cause a bigger cube to be selected. Note, however, if the constant in the model weighting equation is set to 0, the peak is once again correctly located.

Thus it can be concluded that the 2D algorithm does not suffer from the observed visual hull, which impedes the other two algorithms. However, this can only really be stated for perfect data. If, instead, the source data is known to be noisy, then it would not be possible to determine whether the conservative result of the 2D algorithm is preferable over that of the 2.75D and 3D algorithms. Hence, it is the use of the selected voting measure to incorporate noise that has led to this difference. Using the noiseless measure of $\frac{v_a}{v_p}$ would thus reinstate the statement in section 1.5 that there is no advantage or disadvantage in producing 3D data and extracting parameters in the 3D domain, over extracting the parameters in the 2D images, save, of course, the introduction of errors by calculation rounding and discretisation.

5.4 Camera parameters perturbation

A key aspect of the reconstruction algorithms is that the cameras are correctly calibrated. However, as discussed later in section 6.3, the camera set up for the real data was not very accurate; this was not aided by the presence of radial distortion. Thus it is necessary to study the effects of perturbing the various extrinsic and intrinsic camera parameters.

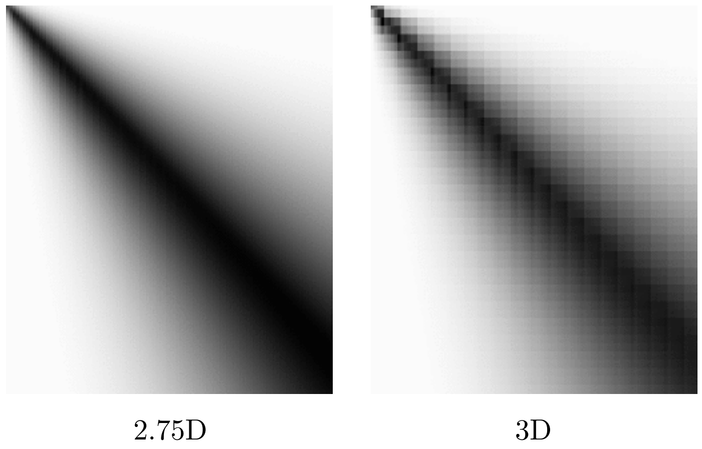


Figure 5.4: Discretisation effects using perfect data of many source cube sizes. Source cube's size increases from left to right; test template's size increases from top to bottom. The darker the point, the higher the fitness.

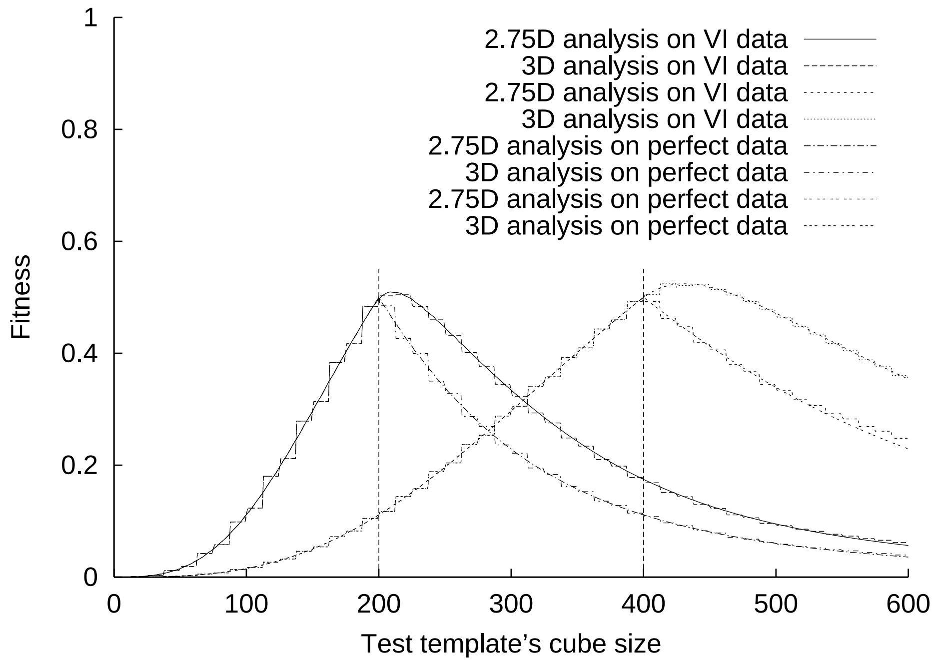


Figure 5.5: Investigation of discretisation. Scenes analysed are reconstructed using VI from perfect 2D images whose source template was of size 200, and also sourced directly from perfect data whose template size was 400.

A cube is placed in a 3D world, centred at the position $(0, 1000, 500)$ mm with the scaling factor of 240, i.e., the cube has lengths of 480 mm. If a camera is incorrectly calibrated, it will view this cube, not with the designated extrinsic and intrinsic parameters, but with perturbed values; these represent the parameters that the camera's calibration should have been described by. Using these perturbed values, 'perfect' 2D images are created of the scene. These are then used to reconstruct the 3D scene, using the parameters that the camera was expected to have, and the extraction process consequently produces a fitness of the original cube.

As there are many combinations, only the parameters of camera γ are perturbed when creating the initial 2D images, and also only one camera parameter is altered at a time.

5.4.1 Extrinsic parameters

Figure 5.6 demonstrates the effect of perturbing the camera's direction angle, that is, the angle about the y-axis. The overall result is not too surprising; as the angle deviates further from the standard value, the fitness reduces, i.e., the cube is gradually eroded. It is also reassuring that the 2.75D and the 3D algorithms produce very similar results, since in essence, the VI reconstruction algorithms are the same.

The linear aspect of the curves is due to the fact that the sine of a small angle is approximately equal to the small angle itself: for a small perturbation angle, the cube is reconstructed at a small distance away from the expected position, where this small distance is approximately proportional to the angle. Hence when searching for the original object, a linearly decreasing volume will be intersecting with the source template's volume.

Although the 2D algorithm produces the same basic curve, the linear section is of much smaller gradient. This is because, although the voting due to the view of γ decreases linearly with angle, the voting due to the other two views is unchanged, and hence the gradient is one third that of the 2.75D and 3D reconstruction results. Hence the fact that the 2D algorithm is less influenced by the perturbation indicates that it is more tolerant to noise; it is due to the fact that the 2.75D and 3D reconstruction algorithms correlate information that the information from all of the views will be affected.

The responses for the other five extrinsic parameters are similar, although the rotation about the z-axis has less of an effect due to the nature of the test images.

5.4.2 Intrinsic parameters

The study of perturbing the two focal lengths and the principal points has been made, with the former producing a more significant effect. It was discovered that a principal point offset of 10 pixels produced only a 4% decrease in fitness for

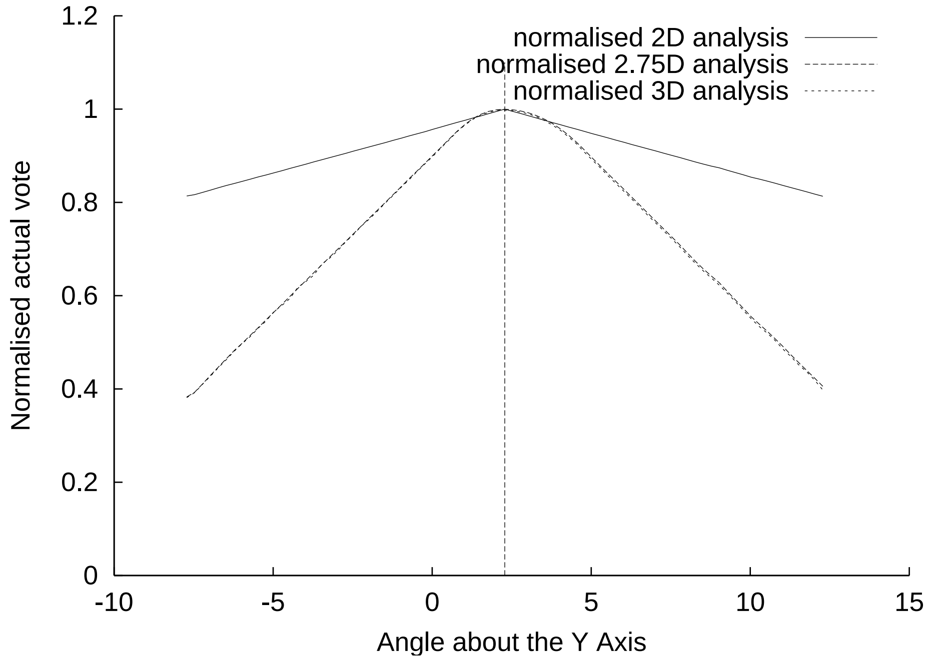


Figure 5.6: Graph of perturbing the angle of the camera about the y-axis.

all three systems. However, for small perturbations, the 2.75D algorithm shows a slightly higher tolerance than the 3D algorithm (a result of discretisation) which shows a higher tolerance than the 2D algorithm; it is the effect of the visual hull that has caused the higher tolerance in the 2.75D and 3D algorithms. For larger perturbations, it is the 2D algorithm that excels due to two views being unaffected and thus both producing a constant level of voting.

The focal length on the other hand produces an interesting effect, as demonstrated in figure 5.7. For large perturbed focal lengths, the original 2D image of camera γ will contain a cube that is actually bigger than it should be. When the intersection of the images is produced, a larger than expected intersection is created. Thus the expected intersection, and hence the cube, lies within this larger region, and thus the tested templates receive 100% of the possible votes. Note that the 3D algorithm never reaches the maximum response value due to a discretisation effect of the projected objects.

For smaller focal lengths, the 2D image of camera γ will contain a smaller cube than normal. Thus the intersection is eroded, and due to the nature of the focal length, this erosion is linearly proportional to it—if the focal length in the direction of the x-axis that is used to produce the initial images is halved, the resulting size of the cube in the image of camera γ is halved. This can be clearly seen in the 2D example. However, in the 2.75D and 3D example, it is the observed visual hull that is actually examined, and as this is bigger than the cube itself, more votes

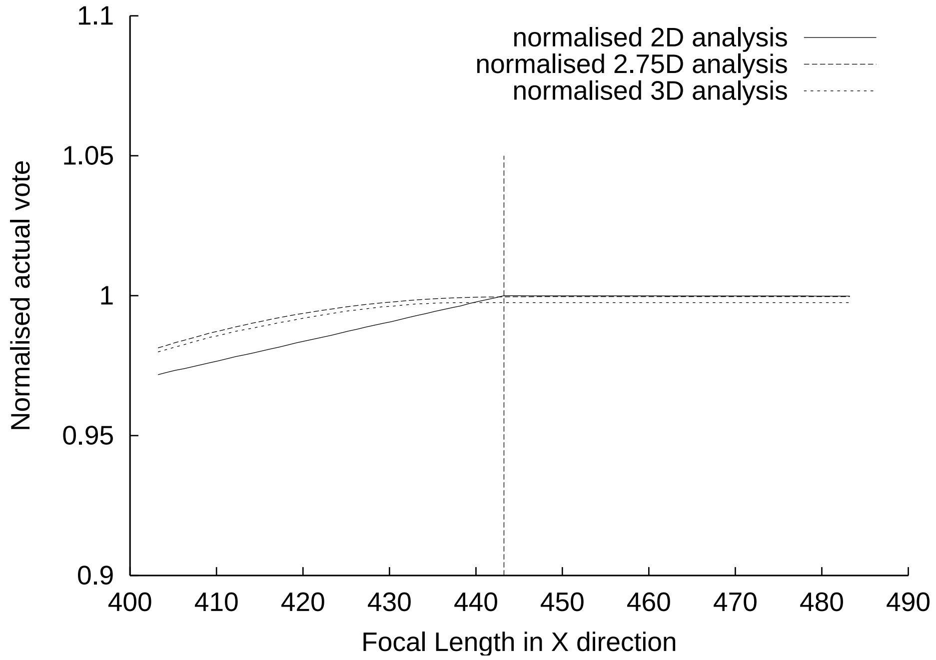


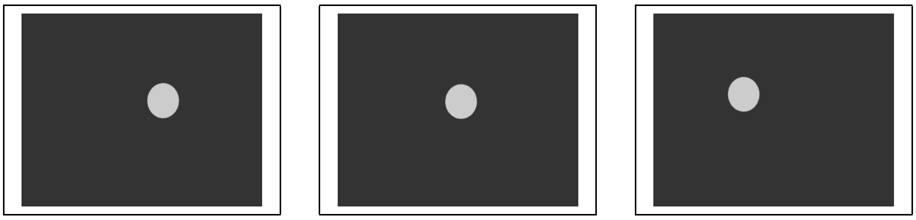
Figure 5.7: Graph of perturbing the camera's focal length in the x-direction.

than would be expected are produced, and hence the apparent increased tolerance of these two algorithms.

5.5 Image noise response

The previous investigations described above used not only perfect data, but perfect binary data, thus analysing the VI algorithms only. The colour and grey scale algorithms of the 2.75D and 3D, being based upon their respective VI algorithm, share the performance described above, however, there are further effects that will affect the output produced. Therefore, the data is now moved from perfect data to a scene that has been ray-traced. However, as there are endless parameters that can now be incorporated into the scene, the scene is kept very basic. The intention is to illustrate the manner in which the algorithms produce their data to enable a better understanding of the processes involved, rather than using more realistic scenes where interactions are harder to describe.

In this section the effects of two different sources of noise will be studied: blurring and additive Gaussian noise. Again, the standard camera parameters are used, the 2D images are affected accordingly, and a scene reconstructed in 3D, but now using the grey-scale voxel algorithm and colour 2.75D algorithm. The scene contains a single sphere, of grey-level intensity 80% on a background of 20%; there is no texture on the objects, but there are anti-aliasing effects around the circumference of the sphere. The sphere is centered at $(0, 1000, 1000)$ mm and has a radius of 240 mm. Figure 5.8 shows the original source images before they are corrupted with noise.



Camera α. Camera γ. Camera β.

Figure 5.8: Image noise response source data, showing a sphere of 80% white on a background of 20% white.

The drawback of such a simple scene is that there is not enough information present to enable the sphere to be localised unaided—the two shades of grey will correspond at many points in the reconstructed scene, and thus a method to distinguish between the foreground sphere and the background is required. To overcome this, the reconstructed scene is thresholded at the intensity level of 50%—anything darker is assumed to be background and is not included in the count, and *vice versa*. Thus, this simple object can now be extracted using a sphere template, and during the extraction stage, the fitness of spheres with a range of radii are produced. With a better noise tolerance, the template of the sphere that has the best fitness should have a radius that is close to the source data radius. The range of radii is $40 \leq r \leq 440\,\text{mm}$ with steps of $1\,\text{mm}$.

The 2D algorithm is included in the graphs for completeness, however, this is only an indication of how suitable such a simple threshold method is at overcoming the introduced noise. Note that for the additive Gaussian noise, the same randomly corrupted source images are used for the three different representations.

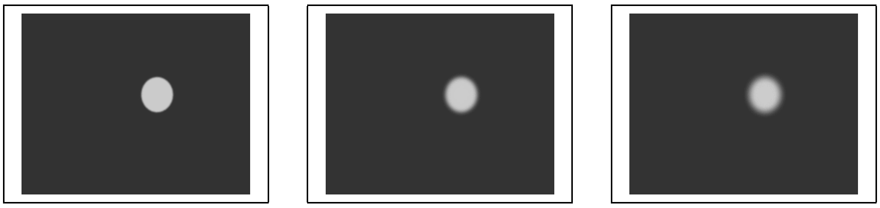
5.5.1 Camera focusing

The experiment

To simulate the effect of camera focus, Gaussian smoothing filters of various sizes are applied to the image viewed from camera γ before they are passed to the reconstruction algorithms. The 2D Gaussian distribution, based upon the 1D equivalent, can be described as:

$$G(x, y) = \frac{1}{2\pi\sigma^2}\mathrm{e}^{-\frac{x^2+y^2}{2\sigma^2}} \tag{5.4}$$

This function is isotropic, i.e., circularly symmetric, and produces a bell-like response. This response, however, is infinitely wide, and thus a cut-off point must be used after which all values are assumed to be 0. The common choice for the x values to be clipped at is $-3\sigma < x < 3\sigma$, and similarly for the y values. Due to the common use of the 1D version of the Gaussian distribution in statistics, the variable σ is commonly referred to as the standard deviation of the filter.



$\sigma = 1$ $\sigma = 3$ $\sigma = 5$

Figure 5.9: Example blurred images of camera γ with their respective standard deviations.

Thus, for $\sigma = 1$, a 5×5 convolution matrix is described by:

$$\begin{Bmatrix} 0.00 & 0.01 & 0.02 & 0.01 & 0.00 \\ 0.01 & 0.06 & 0.10 & 0.06 & 0.01 \\ 0.02 & 0.10 & 0.16 & 0.10 & 0.02 \\ 0.01 & 0.06 & 0.10 & 0.06 & 0.01 \\ 0.00 & 0.01 & 0.02 & 0.01 & 0.00 \end{Bmatrix}$$

where the Gaussian distribution has been evaluated for each integer-indexed element point, with the central element having the index $(0, 0)$.

In this study, a range of σ, $0.1 \leq \sigma \leq 5.0$ with steps of 0.1, is used, with examples shown in figure 5.9. However, for small values of σ, the convolution matrix will rarely be suitable as the contents will not sum to the value of 1, and hence for all of the convolution matrices used, a normalisation process is performed. Also, a fixed size of convolution matrix was utilised due to the fact that the convolution will reduce the size of the image by one less than the size of the matrix itself. Hence, in order to be consistent, convolution matrices were thus all 17×17, yielding a reduction in image sizes of 16 pixels in both the width and height.

Results

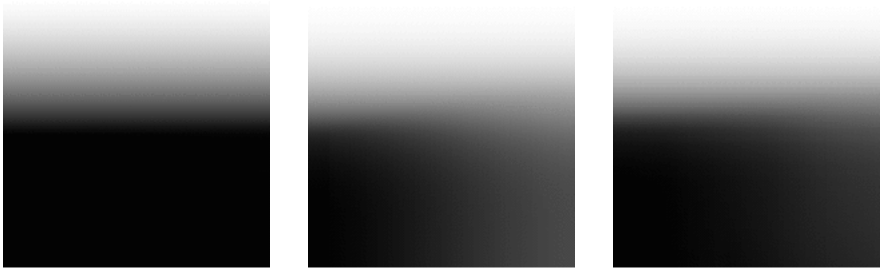
Figure 5.10 shows both how the normalised actual vote accumulator and the full measure are affected by increased blurring. Normalisation of the actual vote is performed by dividing it by the potential vote for the sphere whose radius is 240 mm (which is also the same as the constant k in the full fitness measure of equation 5.1). It is clear in figure 5.10a that for larger radii, the actual vote is gradually reduced as the blurring is increased, but only for the 2.75D and 3D algorithms; the fact that the 2D algorithm is not affected is not surprising since the threshold level at 50% white will yield very similar templates for all blurring levels due to the symmetric nature of the shading of the foreground and background. The reduction in the 2.75D and 3D algorithms is expected as the blurring will cause different shades to be present in the image of camera γ, and these will not correlate with the other images successfully. For increased blurring, the left and right-most extremes of the circle in image γ will not correspond with the other views as well as the centre, and thus the sphere will gradually become ellipsoid in nature. The top and bottom-most extremes of the circle will be less effected due to the camera arrangement.

Figure 5.11 corresponds to the central horizontal line of figure 5.10a, that being the source radius of the sphere. Note that the 2D algorithm does not reach its maximum value. This is due to the ray-tracer producing anti-aliased pixels in the source images that, if they are only to be marginally included into the sphere, have a value of less that 50% white, and hence are excluded from the thresholded version. When the template is tested against the thresholded images, inclusion is all that is required, not the proportion of the pixel to be included; an alternative would be to use sub-pixel sampling in the TM process, as discussed in chapter 7.

Both the 2.75D and 3D plots in figure 5.11 can be seen to tend towards a constant for larger values of the standard deviation. This is directly as a result of the size of the convolution matrix being too small for the standard deviation; if a larger matrix is used, further degradation is seen.

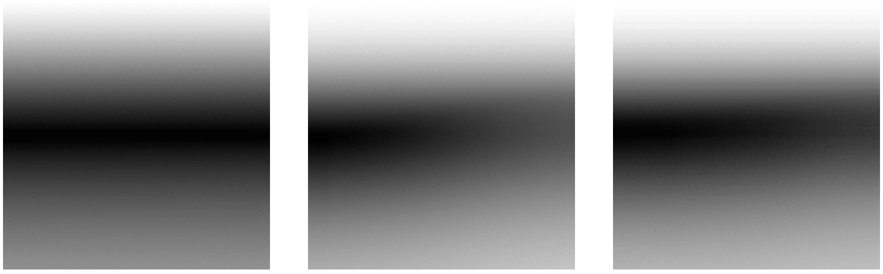
It can be seen in figure 5.11 that the 2.75D algorithm degrades more quickly at first. This is due to the inherent filtering in the 3D algorithm; a small level of blurring will not greatly affect the anti-aliased rays used in the voxel-based algorithm if the blurring is limited to the source region of the ray. Taking this to the limit, if the voxel space was a single, giant voxel that encompassed all of the pixels of all of the views, then no amount of blurring would affect the rays projected onto the voxel.

Turning to figure 5.10b, the full measure can be seen to emphasize the region of the correct radius of the source sphere. However, the peak does not always correspond directly with the expected sphere; the difference between this peak measure and that of the expected radius is actually very small. As an indication of the magnitude of the original measures involved, the full measure of the values of the template radius of 240 mm are half of the values indicated in figure 5.11 (as for this radius $v_{potential} = k$, and the normalised actual vote is given by $\frac{v_{actual}}{k}$). Since the difference in these values are negligible, it is concluded that the use of this measure has successfully selected the correct radius given the various levels of blurring inflicted onto the source image. The 3D algorithm, however, is more affected than the 2.75D and 2D algorithms, showing an increased tendency to underestimate the size of the sphere as the blurring is increased. Interestingly, the 2.75D algorithm actually shows an improvement in the parameter extraction for small standard deviations, thus the blurring is actually an aid: due to the sphere turning slightly ellipsoid, the actual vote will decrease, however, a template whose radius is slightly smaller will also have a similar decrease in actual vote. It is the noise handling ability of the fitness measure that results in the selection of the bigger circle.



2D 2.75D 3D

(a) Plots showing the values of just the actual vote parameter.



2D 2.75D 3D

(b) Plots showing the values of the full fitness measure.

Figure 5.10: Investigation into blurring using Gaussian filters. Standard deviation, σ, increases from left (0.1) to right (5.0) , and the fitness of templates of spheres whose radii increases from top (40 mm) to bottom (440 mm) is shown by the grey-level where the darker the shade indicates the higher the fitness.

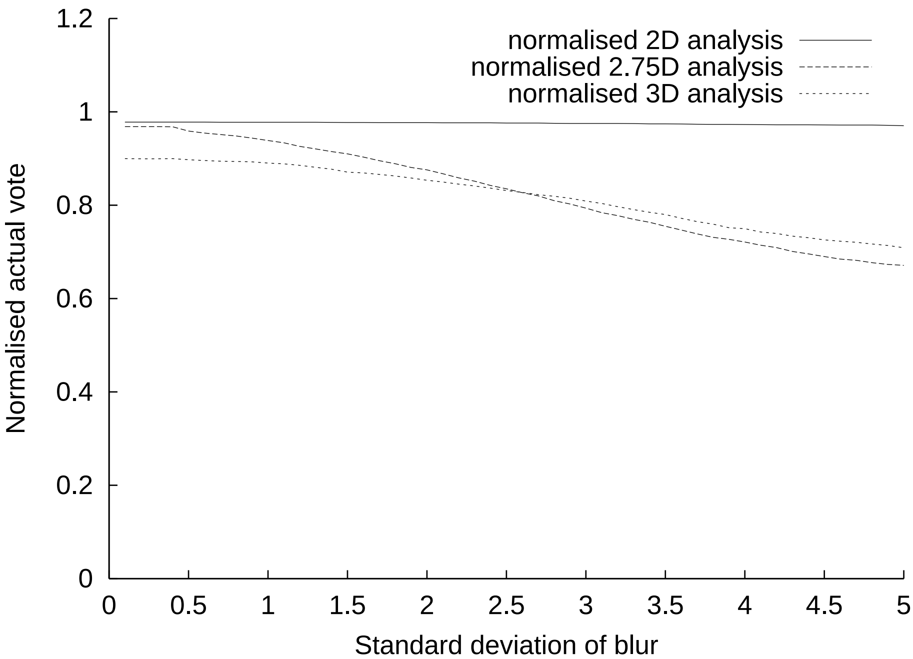
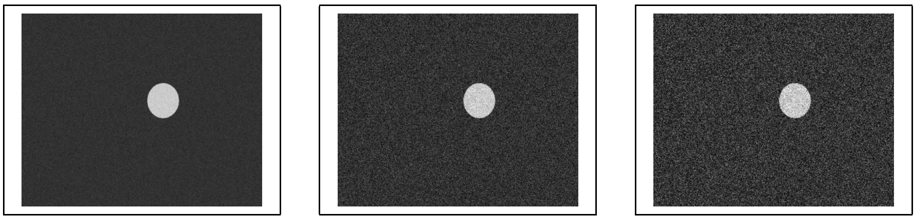


Figure 5.11: Investigation into the effects of Gaussian smoothing. This graph shows the standard deviation of the filter against the normalised actual vote for the template whose sphere has the expected size of 240 mm. Normalising is performed by removing the potential vote contribution in the measure weighting equation.



$\sigma = 10$ $\sigma = 30$ $\sigma = 50$

Figure 5.12: Example images of camera γ with added noise. The standard deviations of the noise is shown.

5.5.2 Additive Gaussian noise

A common method of evaluating the noise handling ability of a system is to subject the source data to additive Gaussian noise. Similar to the previous experiment, the image of camera γ is affected by different levels of such noise. For each pixel in the image, a random variable with a Gaussian distribution is evaluated and added to the pixel's value, the result of which is then clipped to be within a valid range [64].

In this experiment, the standard deviation of this Gaussian distribution was altered from the value of 0 (i.e., no introduced error), to the value of 50; all of the images in this thesis are represented by the range of (0, 255) in the grey-scale and colour channels. An example selection of standard deviations of this noise applied to the image can be seen in figure 5.12.

Results

Introducing the noise causes each non-correlating ray to burrow holes through the reconstructed shape. Figure 5.13 shows how the actual vote and the full measure are affected by the increasing noise levels, though it must be stressed that the shading in these results has been normalised for each level of noise: without this, the results would show very little for high levels of noise as there is a rapid decrease in both the actual vote and the full measure, i.e., it would fade rapidly to white. This can be seen in figure 5.14 where the template whose sphere's size matches that of the source sphere is examined.

In figure 5.14 the 2.75D and 3D algorithms can clearly be seen to be greatly affected even by small levels of noise. The latter, however, is affected less due to the inherent smoothing nature that thus has the result of averaging out small amounts of local-regional errors. The non-correlating rays that have produced holes that tunnel through the reconstruction is the cause of this huge degradation, which in effect reduces both the sphere and surrounding background into discrete rays where the pixels happen to correlate. The 2D algorithm is less affected due to the fact that a pixel must be deviated by a value of about 60 before it is counted as the background, and that two of the views remain unaffected.

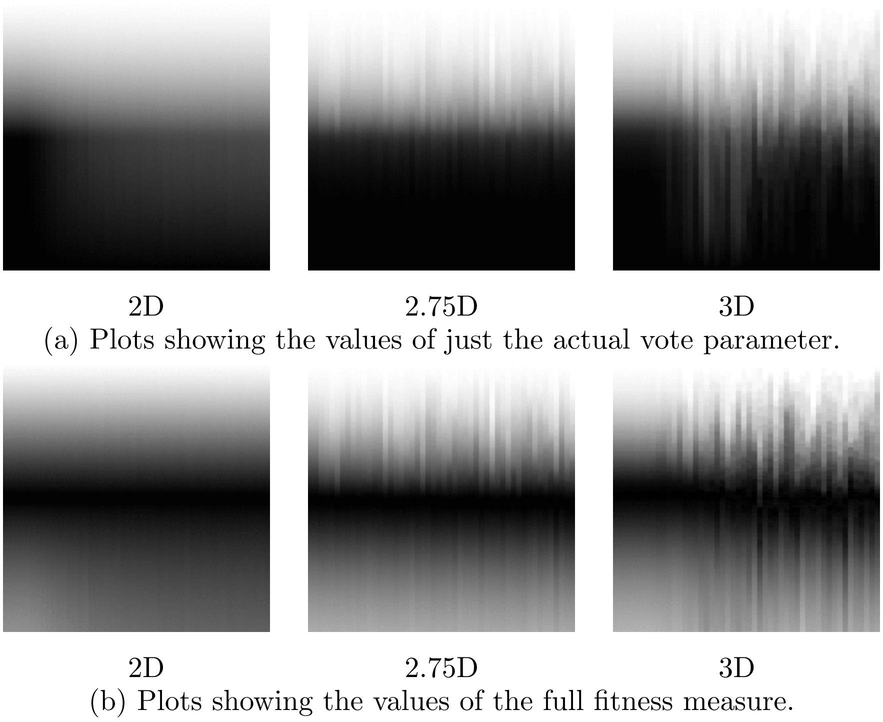


Figure 5.13: Investigation into introducing additive Gaussian noise. Standard deviation, σ, increases from left to right, and the fitness of templates of spheres whose radii increases from top to bottom is shown by the grey-level where the darker the shade indicates the higher the fitness. The fitness has been normalised for each level of noise, i.e., over each vertical line, and thus direct comparison cannot be made with neighbouring levels of noise.

However, as indicated in figure 5.15, the full measure has provided a worthy tolerance to these levels of noise. The 2D trace appears to be most greatly affected, although it is important to note that the magnitude of the measures in question are no longer similar (again the measure of the template whose radius is 240 mm is half that of the values in figure 5.14). In fact, the relative values at $\sigma = 20$ between the best template (with possibly a different radius) and that of the template whose sphere has a radius of 240 mm (shown in figure 5.14), is 0.1% for the 2D algorithm, and 1% for the 2.75D and 3D algorithms. The measure is able to produce a suitable estimate of the radius as the sphere is eroded approximately consistently over its entire volume, and thus the actual votes of the range of spheres are affected similarly.

5.6 Discussion and conclusion

In this chapter it has become apparent that the 2D system, given a perfect background removal method, is superior to both the 2.75D and the 3D voxel-based systems. It has better tolerance to noise in one image as this noise does not affect the interpretation of the other views. However, it is unlikely that a perfect background removal method is available for real data; the 2D system can also only

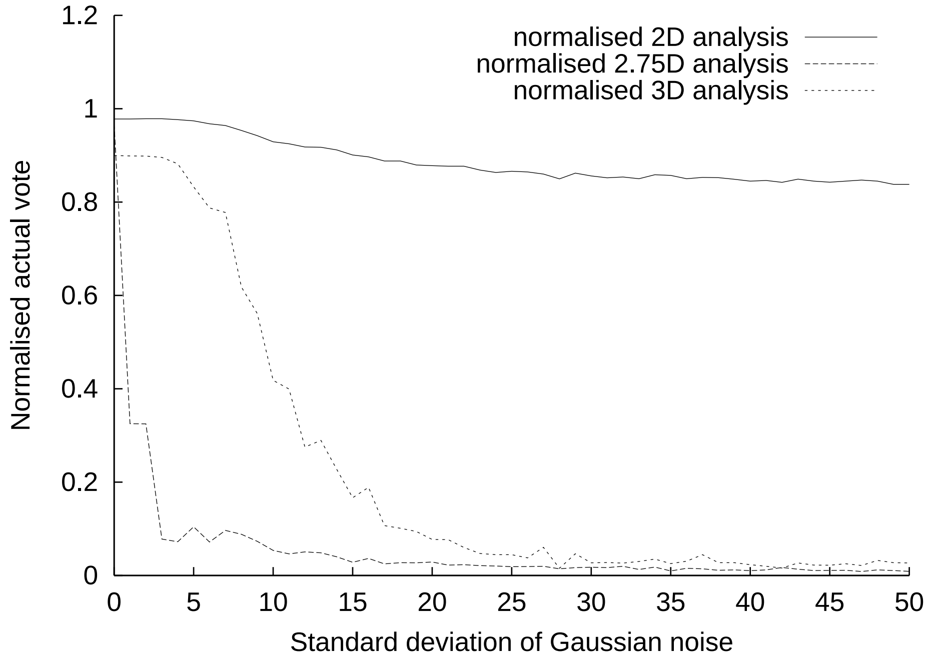


Figure 5.14: Investigation into the effects of additive Gaussian noise. This graph shows the standard deviation of the noise against the normalised actual vote for the template whose sphere has the expected size of 240 mm. Normalising is performed by removing the potential vote contribution in the measure weighting equation.

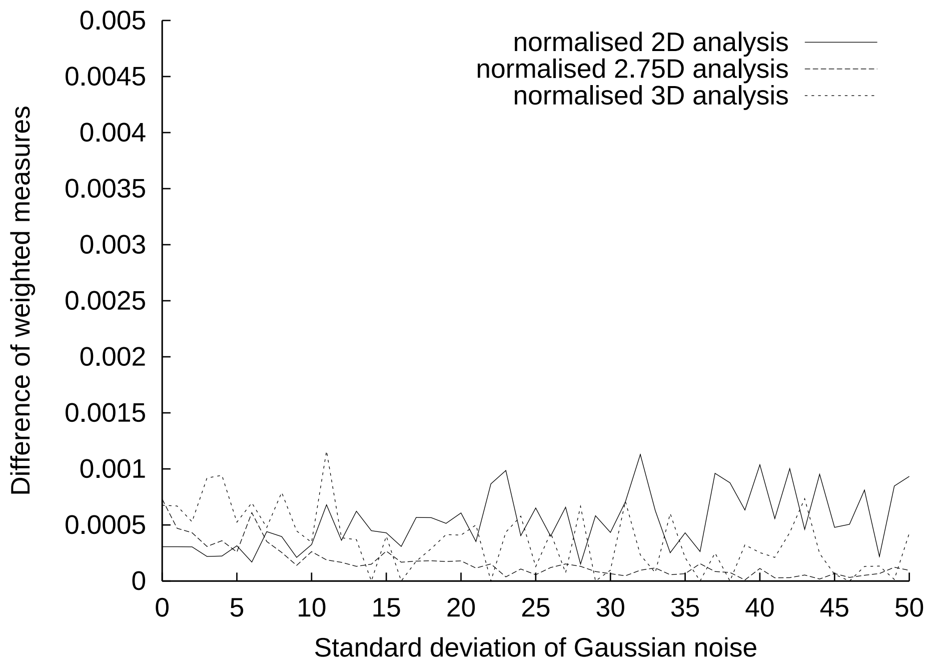


Figure 5.15: For the quantity of noise in the images, this graph shows the difference of the voting measures between the template whose radius is 240 mm and the template that yields the highest fitness measure.

perform as well as the underlying VI, and thus it is subject to problems with phantom shapes and the visual hull. The 2.75D and 3D systems were developed in an effort to eradicate these two artifacts, and perfect background removal is no longer a necessity since empty space becomes a useful segmentation aid of the reconstructed scenes.

It has been shown that the 3D system is hindered due to the underlying voxel structure that places limits on the resolution of the extracted parameters of an object. The 2.75D system on the other hand has been shown to use a representation that essentially provides the same resolution as the idealised 2D system.

The 2.75D and 3D systems have been shown to suffer from the effects of poor camera calibration in a similar manner, but the Gaussian smoothing operator has been seen to have a slightly greater influence on the reconstructed scene in the 2.75D, mainly due to the inherent non-ideal smoothing performed in the 3D system. This inherent smoothing operator also aids the 3D system when Gaussian noise is applied to the source images. However, the full fitness measure provides a useful means to overcome even large levels of introduced noise, and using this, there is no discernible difference between the 2.75D and 3D algorithms when one of the images has been corrupted by Gaussian noise or Gaussian smoothing.

In conclusion, the 2D algorithm should be used in circumstances when a perfect extraction method is available. If this is not possible, then if the highest fidelity extraction is required, the 2.75D system is the logical choice. However, there is an increased processing overhead, as already stated in chapter 4 for the 2D and 2.75D system, although it should be noted that scene reconstruction has to be performed for the latter.

In section 5.3.2 it was noted that the 2.75D and 3D algorithms will commonly overestimate object sizes due to the observed visual hull. However, the converse effect is also possible, and can be seen in the ray-traced results shown above, for example in figure 5.11, the template whose radius is 240 mm does not attain 100% of the vote when no noise is applied. This failure is due to the anti-aliased pixels, which for the 2.75D and 3D algorithms do not correlate well with other pixels in the other views; the latter, however, is less influenced due to the inherent limitation of the resolution. Similarly, there is also a small, but noticeable, effect for the 2D algorithm, but it is entirely dependent on the background filter: a converse 2D example is that background removal commonly results in a halo surrounding the extracted object to be classed as the foreground, due to anti-aliasing. This halo would thus become part of the segmented area and thus augment the projected volume, causing an overestimation of the original object volume to be made.

In the following chapter, the full dynamic systems are evaluated, using both artificial and real data. There is no longer an ideal background removal method, thus enabling a more practical comparison and demonstration to be made of the systems.

Chapter 6

Model extraction

6.1 Introduction

In the previous chapter, static scenes were analysed and parameters were extracted and compared against those that were used to create the original scene. In this chapter, synthetic dynamic scenes are analysed in a similar manner, where it shall be seen that multiple frames provide a means of improving the performance of the extraction. As an illustration of the applicability of the techniques, real world data is also analysed; the first example of this is of a ball thrown through the air, with the ball's radius and the acceleration due to gravity providing a means of verifying the parameters extracted by the complete systems. The second example is of a walking human, and although there are no parameters that can be used for verification, visual inspection and the likelihood of the gait cycle itself are useful qualitative measures.

6.2 Synthetic example: analysis of a moving ball

6.2.1 Setting the scene

In this experiment, a synthetic moving ball model is used, as detailed in section 4.6.1, where the motion of the centre of a translating sphere of radius r is described by:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} + t \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{6.1}$$

where $[s_x\, s_y\, s_z]^\mathsf{T}$ is the sphere's initial coordinates, and $[v_x\, v_y\, v_z]^\mathsf{T}$ describes the velocity along the three axes.

The synthetic data was creating using a ray-tracer ('POVRay') and consisted of the moving sphere that had a contrasting colour and luminance to the other parts of the scene, those being the sky and the tile-patterned floor. No shadows or

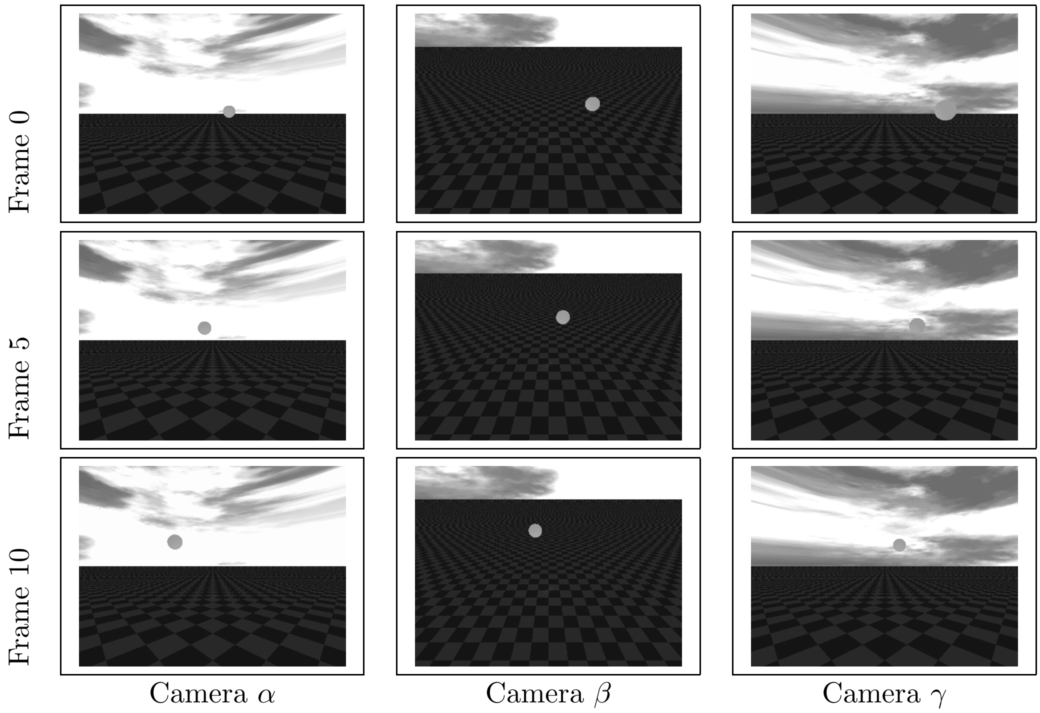


Figure 6.1: Source moving ball ray-traced images.

specular effects are present in this example. Two separate studies were performed, with the difference being the positioning of the three cameras that were to view the scene. The arrangements of these cameras are described in sections A.4 & A.5 in the appendices; in the first of these, the three cameras are on one side of the object, and in the second they are surrounding it. The cameras were positioned above the floor and at a distance greater than 5000 mm from the centre of the region of interest. The images were 400 $\times$ 300 pixels, and thus with the camera's angle of divergence being 90° (focal length $f_x = 200$), implies that at the centre of the region of interest, the resolution possible is approximately 25 ± 5 mm, with the range allowing for the resolution to be variable depending on position in the scene.

Eleven frames were rendered for each sequence, and one hundred sequences in total were analysed; each sequence had a random set of parameters, limited only by the fact that the sphere must appear within the region of interest. The region of interest was delimited by the voxel space which represented the region of (-1512.5 mm$\leq x <$1512.5 mm, 0 mm$\leq y <$2025 mm, 0 mm$\leq z <$2025 mm) with each voxel being (25 mm)^3. A scaling of 0.1 s/frame was applied to convert the frame number (0…10) to an instance in time.

Figure 6.1 shows three frames from the three views used for the first camera arrangement, i.e., that which is described in appendix section A.4.

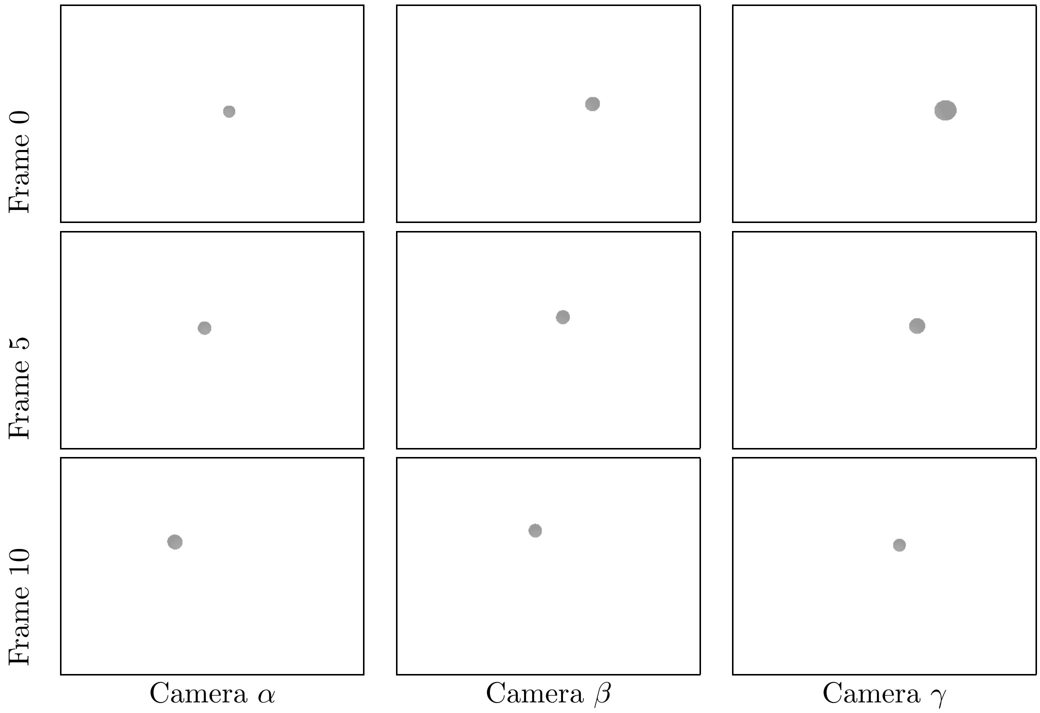


Figure 6.2: 2D background removed images.

6.2.2 2D data preparation

Before parameter extraction can be performed for the 2D algorithm, the background must be removed from the original data. This was performed in accordance with the method described in section 4.2.1; the resulting images that correspond to those displayed in figure 6.1 can be seen in figure 6.2.

6.2.3 2.75D data preparation

The ray-traced images were combined on a frame-by-frame basis, using the full colour 2.75D reconstruction algorithm described in chapter 3. Figure 6.3 illustrates those reconstructed scenes rendered from the source camera positions, and thus can be compared directly to the source ray-traced images shown in figure 6.1. As can be seen, the sky is improperly rendered, which is not surprising since it is actually an object located at an infinite distance away and thus any coincidental correspondences in the near-ground will disrupt its reconstructed accuracy due to occlusion. The patterned ground is also improperly reproduced from the two side views; the repeating pattern causes many correspondences to be made, in essence a form of aliasing at different depths, and is caused by the phantom shapes that are present in the algorithm for like-coloured objects. There are two causes of the improper appearance of the floor: first, the obliqueness from the opposite camera of the flooring causes the wrong height of this pattern to be selected due to minor errors in the rendering, and second, not all of the views can see these parts and thus the confidence in them are lower and hence filtered out of the rendering. The

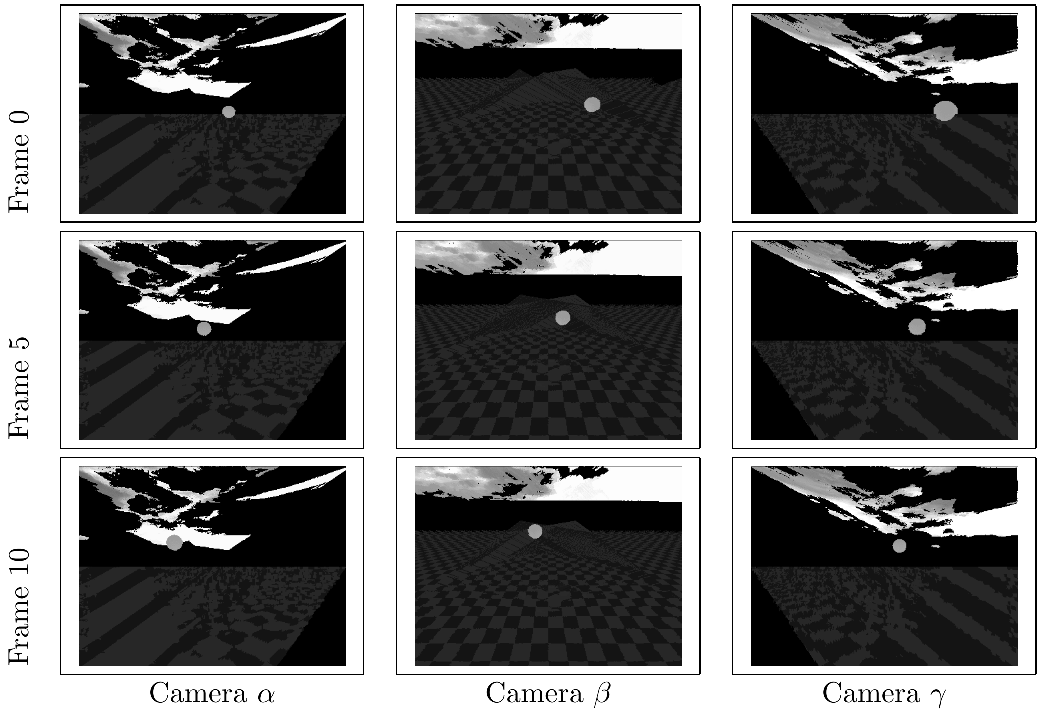


Figure 6.3: 2.75D reconstructed scene.

central view is successful as the correct and 'lower' floor is selected since most parts can be confirmed by all three cameras and since the region it views is less oblique to all three cameras.

Having reconstructed the sequence, the background is then removed; this is performed in accordance with the algorithm described in section 4.2.2, with the results being illustrated in figure 6.4.

6.2.4 3D data preparation

As with the 2.75D system, the 3D system also requires the data to be reconstructed. This is performed using the grey-scale voxel-based algorithm described in chapter 2, with the results corresponding to the images in figure 6.1 being in figure 6.5. As can be seen, the viewable region is very small in relation to the size of the images, with the limits of the voxel space clearly visible. However, over this region the flooring has been successfully located, although the same aliasing problem as was seen in the 2.75D system is present. An estimate of the sky has also been made, however, this is also unsuccessfully reconstructed since it lies outside the voxel space. This sky introduces a great deal of noise as different parts become obscured by the moving sphere, and this can clearly be seen in figure 6.6 where the background has been removed by the algorithm described in section 4.2.2. However, the ball is present, and is solid in structure.

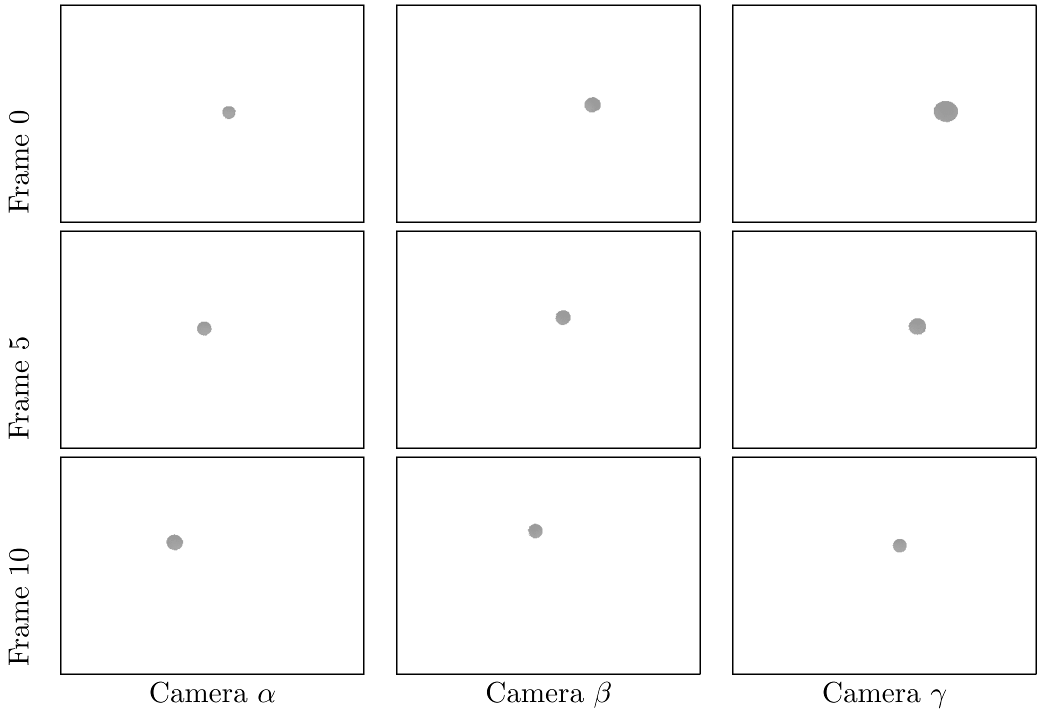


Figure 6.4: 2.75D reconstructed scene with background removed.

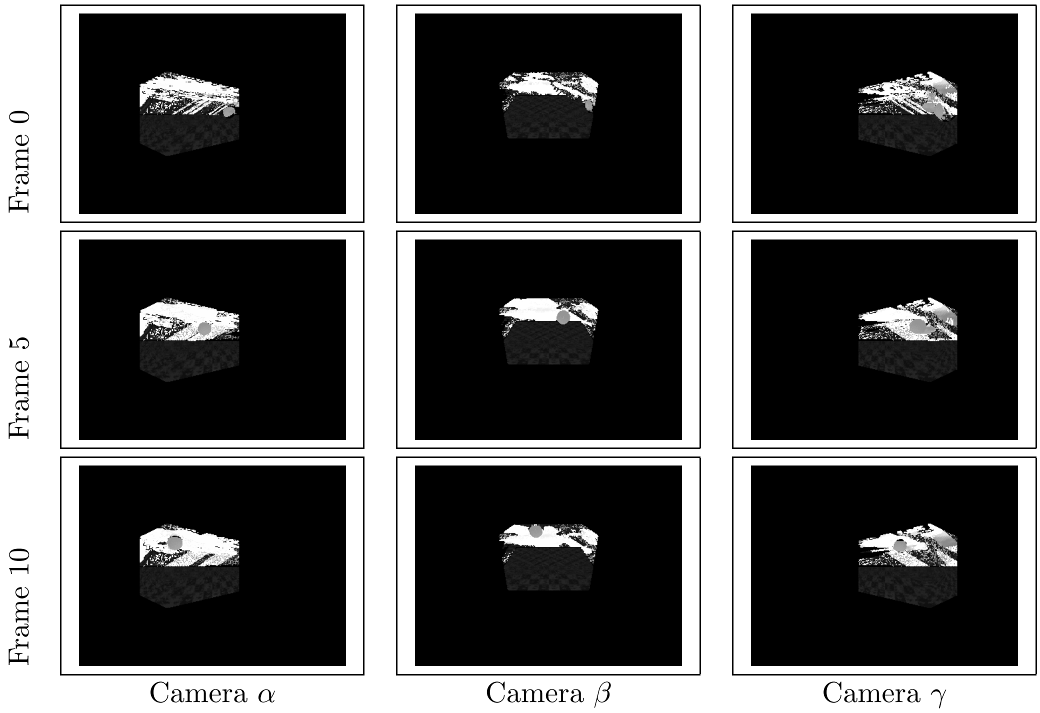


Figure 6.5: 3D reconstructed scene.

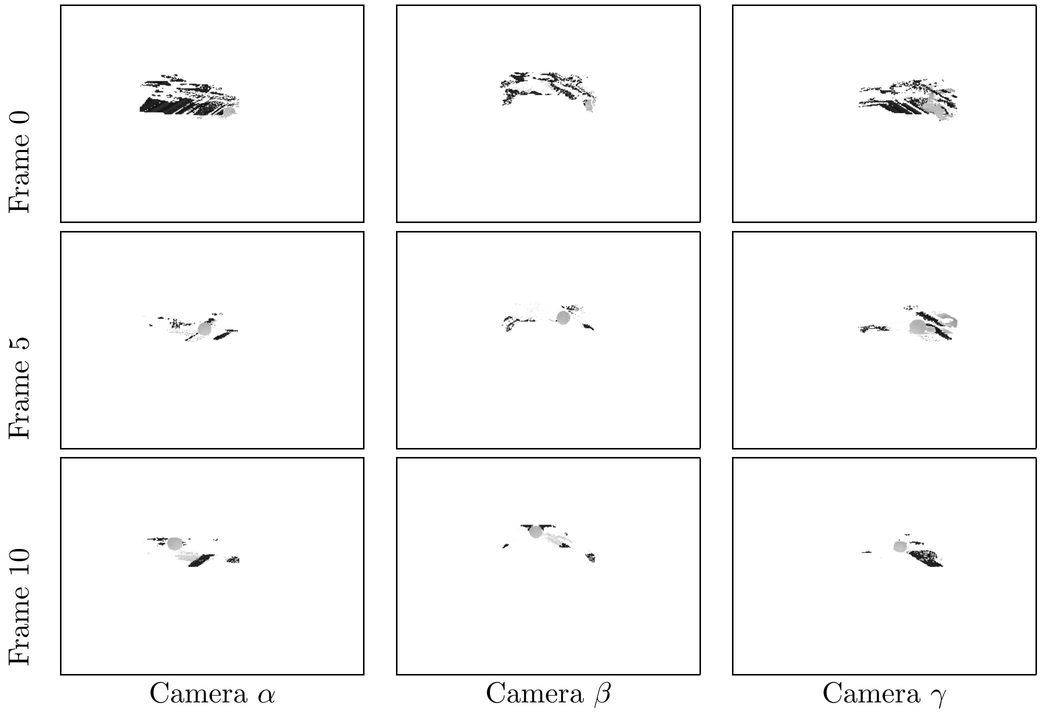


Figure 6.6: 3D reconstructed scene with background removed.

6.2.5 Parameter extraction

Using the parameter extraction algorithm described in chapter 4, the different sequences of eleven frames were then subjected to the moving sphere model template. For the first ten of the 100 sequences, the Genetic Algorithm (GA) was used to scan a much larger parameter space. Noting that this successfully located the regions of interest in this large parameter space, a smaller one was selected that could thus be grid searched. This was performed for speed and so that the peak could be found more assuredly; as already stated in chapter 4, after a GA search of the parameter space, a more thorough search of the indicated neighbourhood should be performed. The range was ± 80 units of the parameter with steps of 10 for all of the parameters except the radius which was ± 20 with steps of 4 units. The constant required for extraction was calculated from the potential vote of the template that was described by the original set of parameters, i.e., the 'perfect' constant was calculated.

Table 6.1 shows the error in the extracted parameters, i.e., the distances between the extracted parameters and the respective source parameter used to create the sequence. The results do not show any statistically significant trends since most of the parameters were calculated to within the possible resolution caused by the discretisation in the image sequences. However, of notable interest is the fact that the 3D algorithm also performs well even though it uses a reduced resolution intermediate data structure. It is the combining of the frames using evidence gathering on the sequence as a whole that leads to this improved parameter extraction performance.

Analysis method	Camera arrangement	Stat	Fitness	Error of the extracted parameters: Initial position s_x	s_y	s_z	Velocity v_x	v_y	v_z	Radius
2D	Arrangement 1	μ	0.418	3.4	23.4	3.7	6.8	17.1	6.4	3.2
		σ	0.081	10.9	13.9	11.0	21.8	22.3	19.7	10.0
	Arrangement 2	μ	0.388	5.3	20.4	6.1	11.0	18.9	13.5	6.4
		σ	0.092	13.0	9.8	13.8	25.4	26.2	28.3	13.5
2.75D	Arrangement 1	μ	0.373	9.2	17.0	14.6	5.3	11.9	12.0	10.9
		σ	0.086	4.8	8.5	6.5	11.4	13.8	13.1	2.6
	Arrangement 2	μ	0.227	7.4	20.7	7.2	13.6	11.0	11.8	10.6
		σ	0.068	9.6	7.0	9.2	17.0	13.8	15.2	4.4
3D	Arrangement 1	μ	0.354	17.3	13.5	15.0	13.0	12.5	20.5	5.8
		σ	0.094	12.6	15.6	18.0	15.8	16.3	19.6	9.2
	Arrangement 2	μ	0.359	16.1	7.1	7.3	13.3	11.8	14.1	5.7
		σ	0.088	9.8	9.1	7.7	15.9	10.4	12.6	5.5

Table 6.1: Extraction results of a moving sphere of unknown radius. In camera arrangement 1 the cameras are located to one side of the region of interest whereas in arrangement 2, the cameras surround this region.

6.2.6 Problems with background removal

The example sequence illustrated in the above figures has demonstrated that the background has been successfully removed. However, for a slower moving ball, or a ball that appears to be moving at a slower rate with respect to one of the views, the non-textured ball may actually be deemed to be background. An example of this is illustrated in figure 6.7. When the background is removed using the 2D background removal algorithm, with corresponding images shown in figure 6.8, parts of the ball are deemed to be the background, and parts of the floor and sky are incorrectly deemed to be foreground. For more complicated models this would be a problem, however, since non-corresponding parts of this moving ball are incorrectly labelled, there is no noticeable degradation in the 2.75D and 3D background removed data as shown in figures 6.9 & 6.10. In the former, however, the restriction of the views to project within the mask of the 2D background removed images causes a reduction in the votes since not all of the views can produce votes throughout its solid structure.

6.2.7 Analysis of an occluded moving ball

In section 6.2.6, a source of noise was illustrated that was due to the motion itself. This noise, however, as explained, did not have a detrimental effect on the extracted parameters. Another source of noise, which evidence gathering is commonly shown to be highly tolerant to, is occlusion. To demonstrate this, the first camera arrangement (see appendix section A.4) was used, but the moving ball was occluded by three cylinders, as shown in figure 6.11; only one trial of this has been performed. Table 6.2 indicates the source parameters of the moving ball and those extracted

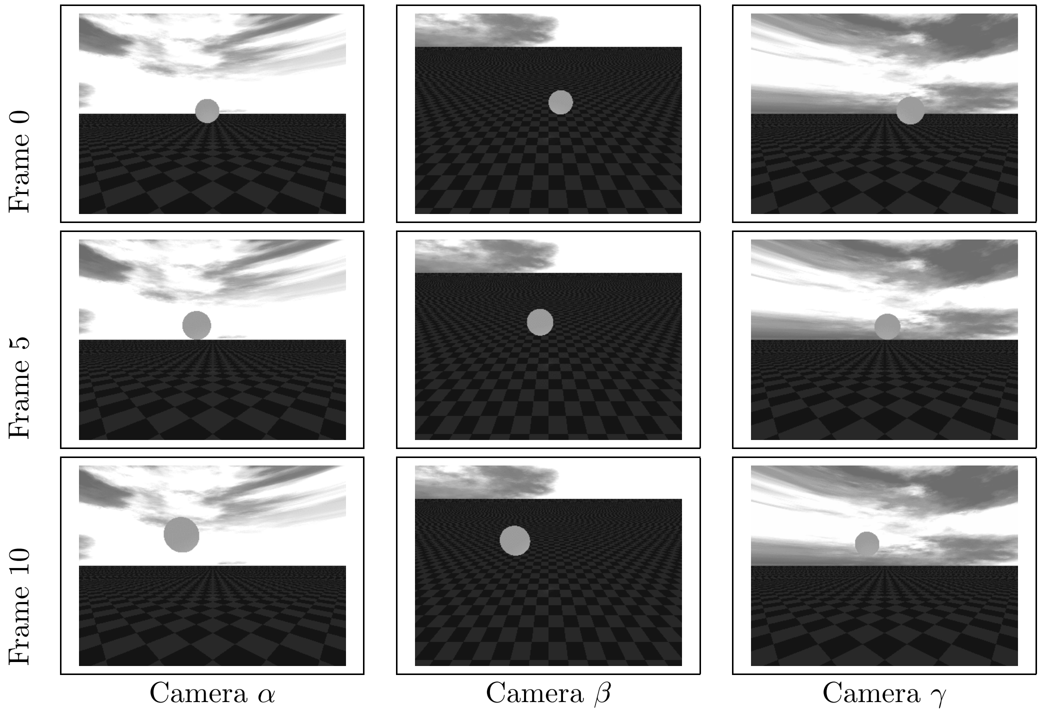


Figure 6.7: Source moving ball ray-traced images.

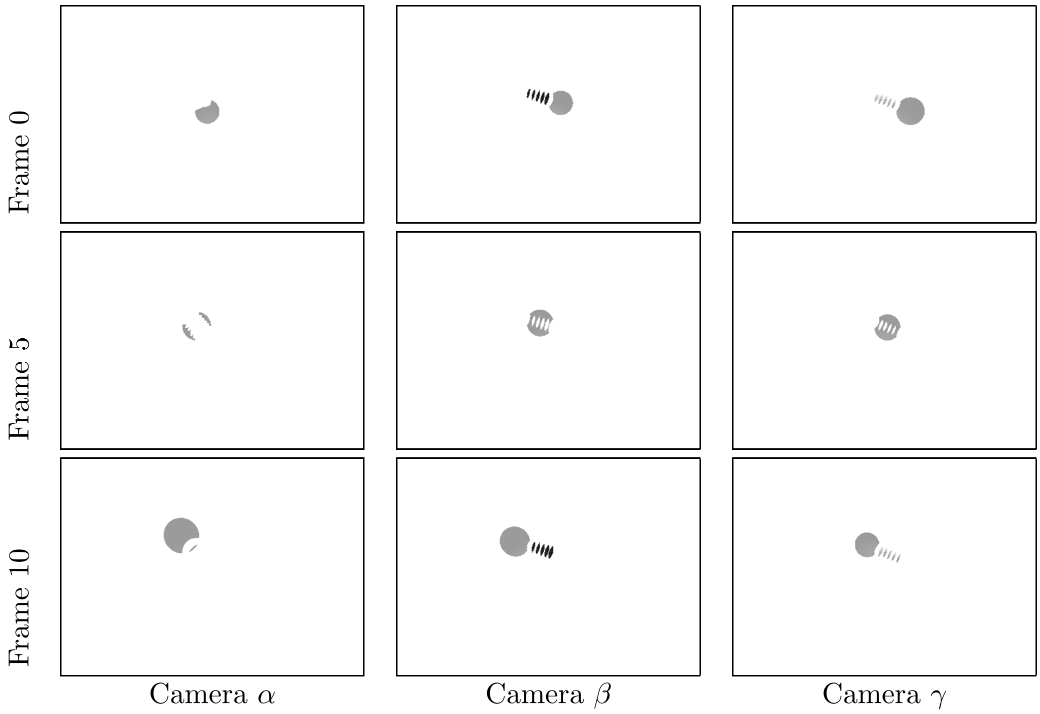


Figure 6.8: 2D background removed images.

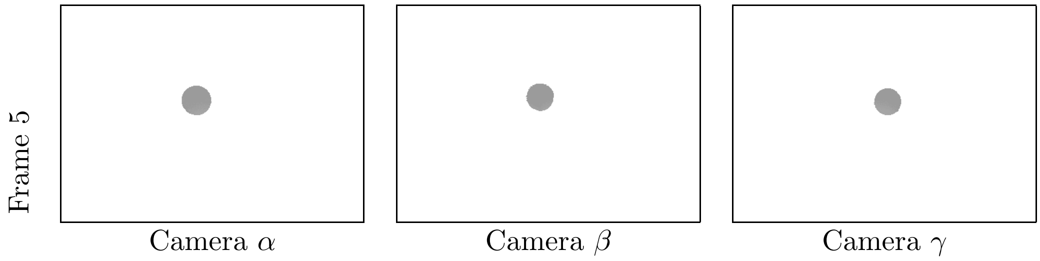


Figure 6.9: 2.75D reconstructed scene with background removed.

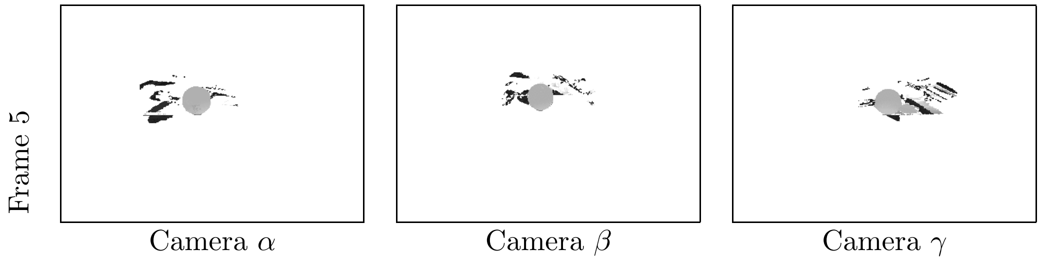


Figure 6.10: 3D reconstructed scene with background removed.

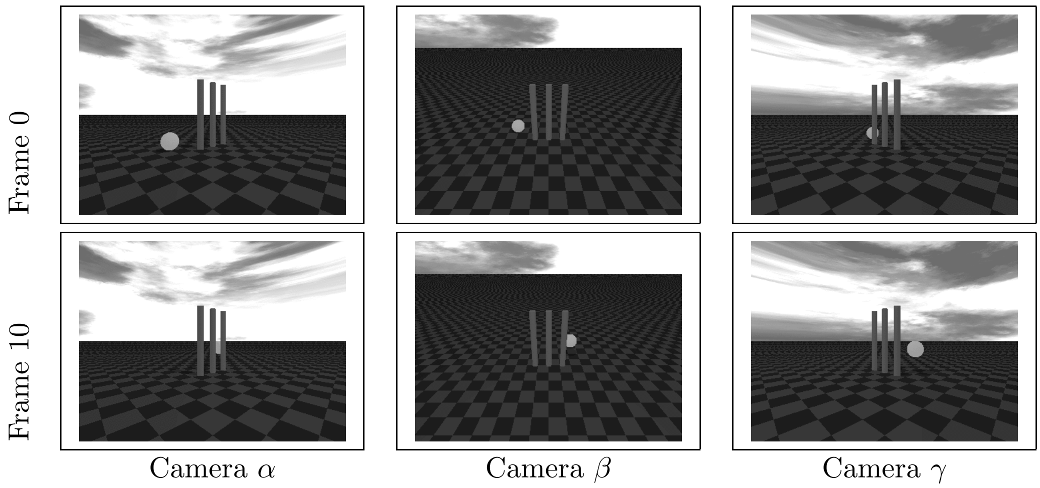


Figure 6.11: Occluded source moving ball ray-traced images.

from the scene with and without these occluding cylinders. As can be seen, there is very little difference in the extracted parameters, although there is a noticeable reduction in the fitness measure. This latter point should be expected since the ball is not properly reconstructed due to the occlusions. The 2.75D and 3D algorithms' fitness measures are more greatly influenced as they are a volume measurement and thus are affected more than the 2D area measurement.

6.2.8 Conclusion on synthetic data

The above experiments have shown the high tolerance that the systems have to the different sources of noise, and the success that they have at extracting the simple artificial model. There are no discernible differences between these systems

	Fitness	Initial position s_x, s_y, s_z $\pm 10\,\mathrm{mm}$			Velocity v_x, v_y, v_z $\pm 10\,\mathrm{mm\,s^{-1}}$			Radius $\pm 4\,\mathrm{mm}$
Source data	-	-1000	300	500	2000	500	0	250
2D unoccluded	0.495	-1000	320	500	2000	510	0	250
2.75D unoccluded	0.422	-990	310	510	2000	510	-10	258
3D unoccluded	0.447	-990	290	500	2010	500	-20	242
2D occluded	0.345	-1000	320	500	2000	500	10	250
2.75D occluded	0.100	-990	310	520	2000	490	20	258
3D occluded	0.121	-990	280	520	2020	520	20	246

Table 6.2: Occluded and unoccluded extraction results of a moving sphere of unknown radius. Values indicated are the actual extracted values not the error; the source parameters are also listed for comparison. The column headings indicate the step size of the respective parameter.

demonstrated by this simple example. A further model, which is a box moving around an arc, can be found in appendix section B.2; this shows a similar pattern of results.

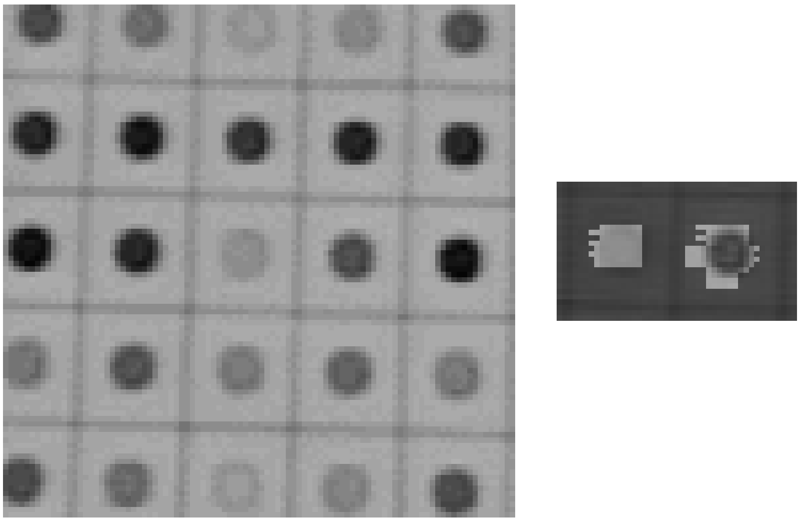
6.3 The real world data capture

6.3.1 *Introduction*

Due to the success with the artificial examples, three cameras were also used for the real world data; this also has the advantage of making the source data distinct from stereo vision. Thus in essence, three CCTV cameras were placed around the scene, recording at a frequency of 25 Hz, (with a shutter speed of 250 Hz). The apparatus used was unfortunately more complicated than just this set of CCTV cameras. Digital video (DV) was used as the recording medium as it enables convenient and consistent data play-back, and thus digital cameras were used. However, the requirement of synchronised cameras made the use of analogue cameras necessary—currently only NTSC, not PAL, digital cameras are capable of this. Thus video output from the analogue cameras were fed into digital cameras and recorded, with the analogue cameras synchronised using the circuit described in appendix D. This introduces several disadvantages, including interlaced images instead of progressive scan, and a slow feedback loop when attempting to focus the cameras—the data had to be played back on a computer to examine the quality of the reproduction. Also, problems with the DV standard 4:2:0 and unsuitable codecs became a common source of frustration.

Originally, an outside session was attempted, however, the uncommonly bright sunshine introduced too great a contrast between the parts of the scene that were in the direct sunlight and those that were in the shade, with the latter disappearing into the darkness. Thus an inside recording session was used, where the conditions were more suitable for the cameras. However, no special lighting was used—standard fluorescent strip lighting was present, which unfortunately introduced lighting flicker; this can be seen in the resulting images as bands of different shades travelling through the images.

The data capture system was also not ideal for the tuning of the cameras since the digital cameras were unable to input from an external source and output to a computer at the same time, and thus the feedback loop in the focusing of the cameras was inadequate, yielding source data that was blurred (see figure 6.12a).



(a) (b)

Figure 6.12: The errors in the source data. a) Magnified example of the test board showing the presence of blurring. b) Magnified and enhanced centre and right-of-centre circles of (a) showing significant colour bleeding that is due to the DV bit-rate reduction algorithm.

6.3.2 Digital Video (DV)

Digital video was chosen because of the convenient nature of the data storage and the reproducibility of the data, however, there are many associated artifacts with DV which are centred around its compression. One such artifact is that the blue, and to a lesser extent the red chrominance information are poorly represented, producing significant colour leaking, as can be seen in figure 6.12b. It is noted that there is poor representation of the chrominance values when PAL analogue sampling is used, however, these effects are different.

The other common problem with DV is poor codecs that do not decode the image data stream correctly, usually making approximations so that the information is presentable in real time. As the systems described will be off-line, the highest quality information is preferred. The other common error in codecs that was discovered is the misinterpretation of the interlaced structure of the 4:2:0 data stream; this is evident by the misalignment of colour on moving objects. Appendix E gives an expanded description of the DV standard, and details how this misinterpretation of the data stream is commonly made.

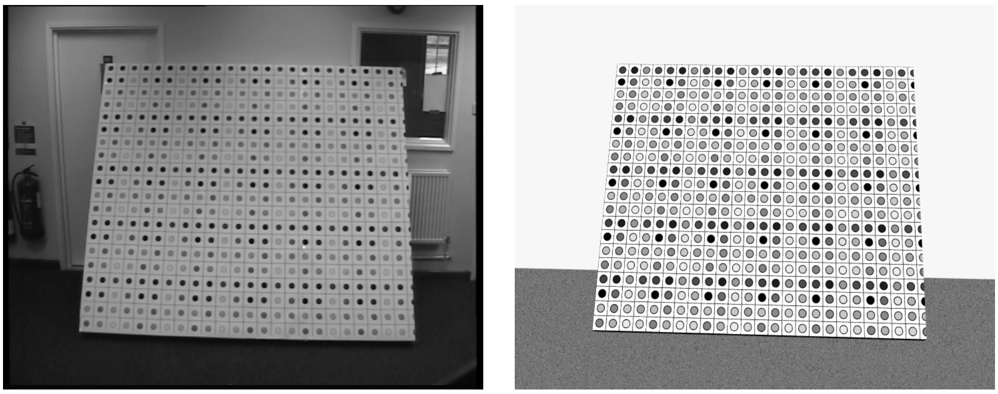
6.3.3 Calibration of cameras

One key aspect of the reconstruction process is that the cameras have been calibrated, with regards to their positions and orientations in space, and also to their interpretation of colour. A test screen was constructed that was intended to perform just this purpose, and a full interlaced image can be seen in figure 6.13a. The circles were filled with the different primary colours, (red/green/blue) at two intensities, the printing primary colours (cyan/yellow/magenta) also at two levels of intensity, and also four levels of gray scale. Note that for the data capture, the cameras' white balance functions were turned off. This ensured that each camera recorded a consistent representation of the scene, rather than varying it according to the colour that appeared brightest.

As well as for colour calibration, the circles were intended to be used for the spatial calibration: using a simple segmentation algorithm and then calculating local moments, the centre of each could be specified to a sub-pixel accuracy. For the calibration to be performed, the points in the images are related to those on the physical board. The board and the points on it are assumed to be the $z = 0$ plane—any offset to the board's position and orientation are performed after the calibration process. Knowing the physical points that each of the centres of the circles in the images correspond to, Tsai's camera calibration algorithm [87, 88], which is freely available on the Internet, can be applied to produce the required intrinsic and extrinsic parameters for the camera, as well as a constant that describes the amount of radial distortion in the image. However, there is an assumption that the camera does not lie perpendicular to the board, since in this case the distance to the board (the $z = 0$ plane) and the focal length are only known up to a scale factor; a value of at least 30° to the normal of the board is recommended. Note must be made that Tsai's camera calibration algorithm uses right-handed geometry rather than the left-handed geometry that is used here.

However, this board was not actually fully used for three reasons. First, there was poor colour in the resulting images, due to both the original lighting conditions and the DV colour representation. Second, the details were too blurred for localisation to be accurately enabled, and third, camera γ was pointing in a direction that was too close to the normal of the board, and thus introduced large errors in the spatial calibration if small errors were present in the localised points on the board. However, the internal and external parameters of cameras α and β were confirmed using this method.

Instead, at the time of filming, physical measurements of distances between the cameras and the board were made, enabling their extrinsic parameters to be calculated by trigonometry. By taking excess measurements at the time of filming, the extrinsic parameters were calculated to be within an accuracy of 5 mm. To calculate the intrinsic parameters, the four corners of the board were labelled in the image of each of the cameras, and a simple search was performed to find the three angles and the two focal lengths that would map the physical positions of the board's corners onto the respective points in the images; this used a method of least squares for judging the fitness by the errors incurred. For a visual inspection, a ray-traced scene was generated, as is depicted in figure 6.13b, which was then compared with the original image. By producing this ray-traced scene, the presence of radial distortion became more noticeable, however, the level was deemed to be only about four pixels at the worst case, and thus was believed to be relatively insignificant for the purposes of the experiments reported here.



(a) (b)

Figure 6.13: (a) The calibration board and (b) the ray-traced estimation of it.

No formalised method was made to colour-calibrate the cameras as the objects under study had good contrast, and the presence of the strobing lights was more significant. However, for one of the experiments described below, a manual estimate of the calibration was made, and the images were slightly altered.

6.3.4 Interlaced video

It was originally believed that interlaced video was a problem rather than an asset of the system. However, it was soon realised that by reducing the size of the images and increasing the temporal rate from 25Hz to 50Hz, this was not the case. Scaling the images so that they were half size considerably reduced the processing time for reconstruction. Having more fields than required provided an alternative way of confirming the results obtained—the results of the even fields could be compared with those produced from the odd fields. Alternatively, for short sequences the fields could be combined to form a 50Hz sequence.

To remove the interlacing from a video stream, a common method is to fabricate the lines in-between those of the field under study by directly averaging them with the rows above and below, or duplicating the row above or below if the row is at the limits of the image:

Let $\mathbf{R}_i$ be a vector of pixel values in row i of an image, $0 \leq i \leq n$, where $n \geq 2$.

Let the resulting image be described by $[\mathbf{R}_0'^{\mathsf{T}}\ \mathbf{R}_1'^{\mathsf{T}}\ ...\ \mathbf{R}_n'^{\mathsf{T}}]^{\mathsf{T}}$

Let $\phi = 1$ if the image resulting from the odd fields is sought, or $\phi = 0$ if the image resulting from the even fields is sought.

$$\mathbf{R}_i' = \begin{cases} \mathbf{R}_i & for\ (i\&1) = \phi \\ \mathbf{R}_{i+1} & for\ ((i\&1) \neq \phi), i = 0 \\ \mathbf{R}_{i-1} & for\ ((i\&1) \neq \phi), i = n \\ \frac{1}{2}\left(\mathbf{R}_{i-1} + \mathbf{R}_{i+1}\right) & otherwise \end{cases} \tag{6.2}$$

where '&' denotes logical AND.

Since this fabrication does not add any information, the images can be scaled by half, vertically, without any loss; the images were also scaled by half horizontally for ease of comprehension, although this does discard information. It is worth noting that scaling an image alters the focal lengths of the camera that viewed it. This initial fabrication is necessary (in form, since these two manipulations can be performed in one step), since the row n of the odd field is not from the same spatial location as row n of the even field. By performing this manipulation, this is taken into account, enabling the resulting lines to be compared. However, for regions in the image of high discontinuity, comparing similar lines is not ideal. For example, behind the board in figure 6.13a is a window whose sill has many fine sharp horizontal details. These details will not be visible by both fields, and thus the two frames produced from the two fields must still be handled separately. This is necessary for the background removal stage, otherwise these regions will produce a high standard deviation, and thus always be labelled as foreground. Although unnecessary, throughout the experiments, the two sets of frames were processed separately, to ensure a consistent approach to the data handling. This thus produced two sets of results for each original sequence, which can then be compared; in order to compare the results, note must be made of the significant time delay between the two sub-sequences.

6.4 Real world example: ball under the influence of gravity

To test the performance of the systems, a method was required that could extract parameters from the real world scene of which some could be verified. The model selected was that of a ball under the influence of gravity. Two parameters were known in this experiment, that of gravity ($9800\,\mathrm{mm\,s^{-2}}$), and that of the ball's radius ($120\,\mathrm{mm}$).

6.4.1 The model

The eight motion parameters, as shown in table 6.3 dictate the ball's position at a given time as:

$$\mathbf{p} = \mathbf{p}_0 + t\mathbf{v}_0 + \begin{bmatrix} 0 \\ 0 \\ \frac{gt^2}{2} \end{bmatrix} \tag{6.3}$$

Parameter	Description
$\mathbf{p}_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$	The position of the ball at time $t = 0$
$\mathbf{v}_0 = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$	The components of the velocity of the ball at time $t = 0$
g	The acceleration due to gravity ($9800\mathrm{mms}^{-2}$)
r	The radius of the ball ($120\mathrm{mm} \pm 5$)

Table 6.3: Description of the parameters of the model for the ball under the influence of gravity.

The ball is modelled using the basic shape of the sphere, centred on $\mathbf{p}$, with radius r.

6.4.2 The source data

For this experiment the cameras were not calibrated for colour, which thus also introduces noise into the reconstruction. This was found to be not necessary since the ball was a significantly different colour from the surroundings. Figure 6.14 shows a selection of frames from the three cameras from a single image sequence.

6.4.3 The reconstructed and filtered data

2D

Having removed the background, segmented images were produced. Those corresponding to the images in figure 6.14 can be seen in figure 6.15.

A large threshold was required for this data, with the significant extraneous movements being sourced from the person who had thrown the ball and also the shadow on the wall.

2.75D

The results of the reconstruction of the source views from the 2.75D scene generation can be seen in figure 6.16. It is clear that although noise is present, the original scene

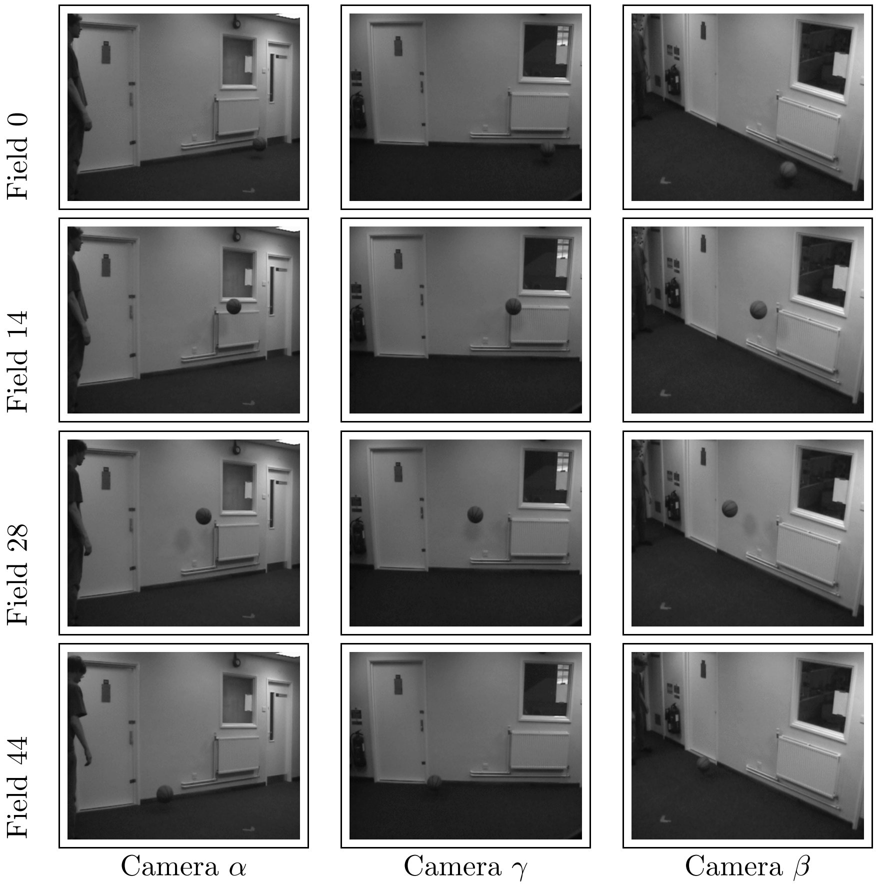


Figure 6.14: The ball-under-gravity source data [sequence 07217-07239].

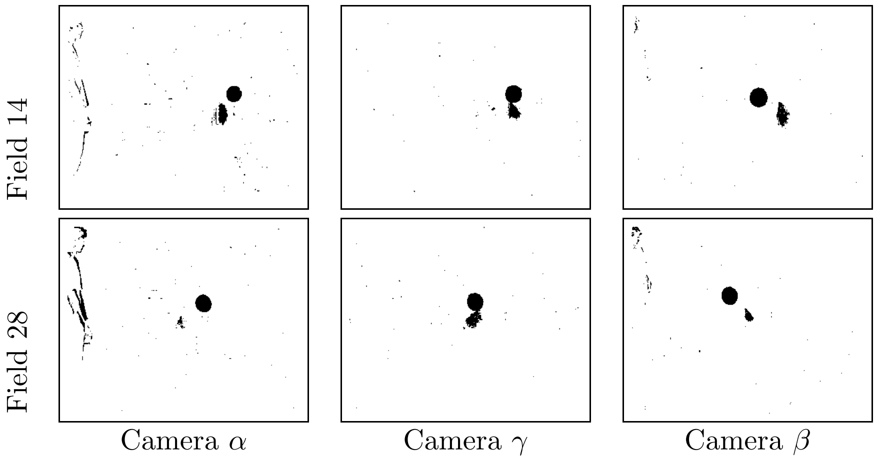


Figure 6.15: The ball-under-gravity 2D background removed images [sequence 07217-07239].

has been reproduced with high fidelity with this representation of data. The noise that is obviously present is caused by the high threshold placed on the rejection of differing pixels. This high threshold was required due to the noise in the source images caused by the imperfect lighting. The streaks visible are thus projected from other views that confer a similar colour.

When producing the images of figure 6.16, all of the information was used, however, some may be inappropriate since the projected ray's limits may not be visible by all of the source cameras. By reducing the threshold of uncertainty in the rays, it can be seen in figure 6.17 that certain areas of the image are now missing in the reconstruction. This includes the area above the door which only camera α can see, and also the area of the window which is due to none of the views looking at the same part of the adjoining room. It is worth noting that the sheet of paper that was adhered to the window, to hide a monitor's strobe effect with the camera, has been resolved.

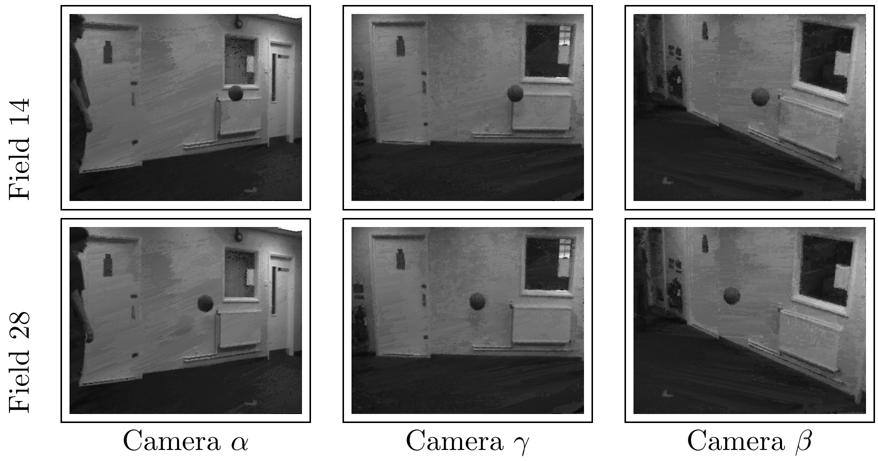


Figure 6.16: The ball-under-gravity 2.75D reconstructed scene from original camera angles.

Figure 6.18 shows the generated scene from three novel angles. It would appear that from such angles, all the 3D nature is lost. An obvious feature is the white streak in view δ, caused by the ceiling light in camera α. Since that light could not be correlated with any view, the representation placed it where the other two views could not see it, and hence from a near novel position such uncertainties become apparent.

By utilising the same restriction as before on the level of uncertainty allowed in the reconstructed scene, a more meaningful result is obtained as in figure 6.19. Cameras δ and ζ, which are located between the original camera positions, bear a close resemblance to the scene, however, camera ϵ indicates that the scene has not been successfully reproduced as expected. In this view, which has an acute vertical angle to the scene, the wall is clearly seen to protrude into the room. This, however,

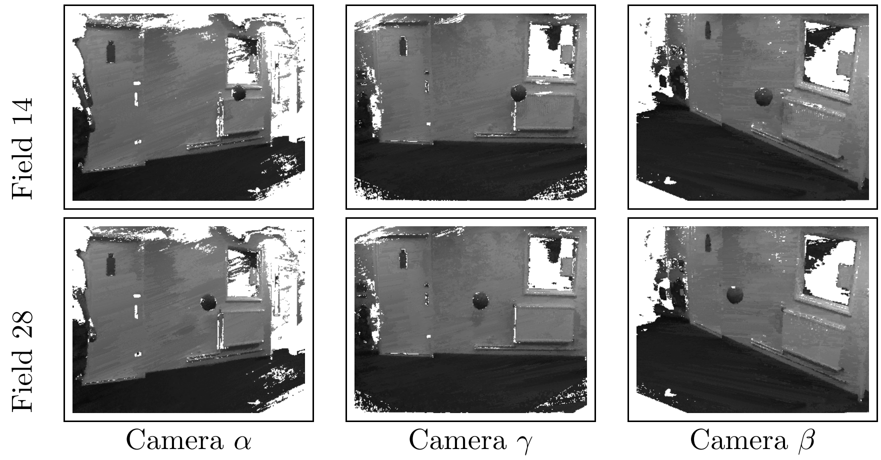


Figure 6.17: The ball-under-gravity 2.75D reconstructed scene as seen from original camera angles but filtered.

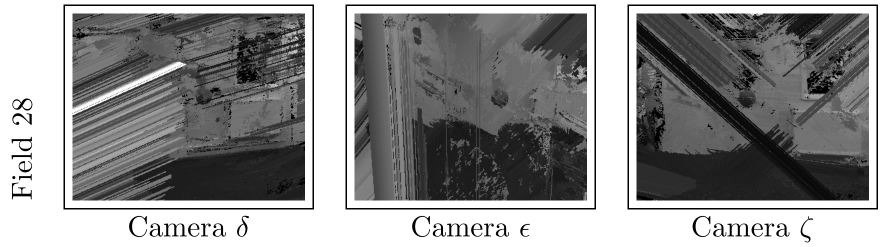


Figure 6.18: The ball-under-gravity 2.75D reconstructed scene as seen from novel camera.

is to be expected, as was found in the previous synthesised examples, because the original views do not have enough information on the wall to be able to indicate its flatness. The limits of the main protrusion are clipped by the original camera α resolving the radiator with the other views, and camera β resolving the door. This thus shows that texture is fundamental in abstract scene reconstruction.

Finally, the results of removing the background, or more precisely, restriction to areas formed from masking the individual view projections with the respective segmented 2D images, can be seen in figures 6.20 & 6.21. No filtering is required to clearly see the reconstructed ball. The small areas of pepper noise in the segmented 2D images can be seen to produce long rays from the source camera positions in the 2.75D data.

Although this reconstructed ball appears spherical from the source camera views, this is not so from the novel views, especially obvious in the acute view ϵ. As before, the true spherical nature cannot be resolved with just three views, and thus the straight lines are the edges of the regions of correlation between the views. The novel views also show how the blurred outline of the ball was not resolved but placed at close or distant positions relative to the cameras, as can be seen, for example, in

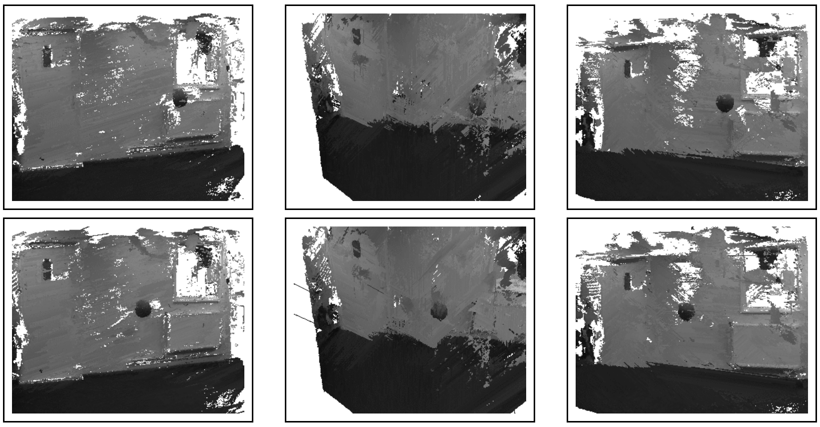


Figure 6.19: The ball-under-gravity 2.75D filtered reconstructed scene as seen from novel camera.

camera δ of figure 6.21 where lines can be seen to be present in the top right and bottom left of the image.

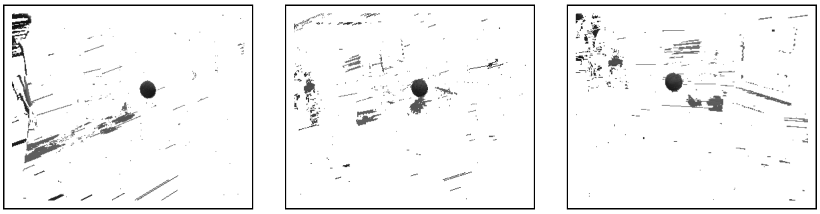


Figure 6.20: The 2.75D filtered background removed scene from original camera angles.

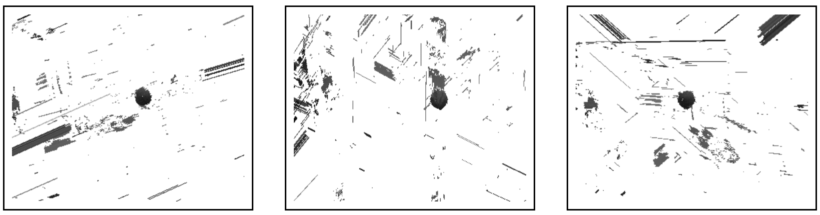


Figure 6.21: The 2.75D filtered background removed scene from novel camera angles.

3D

The results of the 3D reconstruction can be seen in figure 6.22. As with the 2.75D analysis, the scene appears to be constructed well, although there is decreased fidelity due to the lower resolution data structure. In a similar manner to the 2.75D process, voxels of reduced certainty can be removed. This is shown in figure 6.23. It can be seen that, along with the regions in the 2.75D method (figure 6.17), there are regions that were not recovered. The main additional areas for zero contribution are from sharp edges, especially noticeable around the bottom of the wall where there are two examples of such sharp edges. The contribution to voxels at these points in space will be the anti-aliased region of the original images, and thus unless a very similar contribution was made, rejection is inevitable, with another preferred point being chosen. The ball does not suffer from this edge effect, but it is the only dark object in the vicinity, and it is also relatively large.

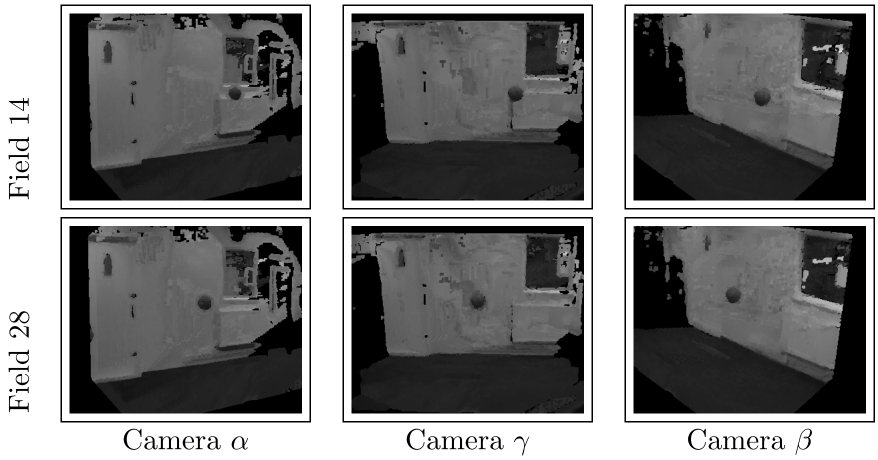


Figure 6.22: The ball-under-gravity 3D reconstructed scene from original camera angles.

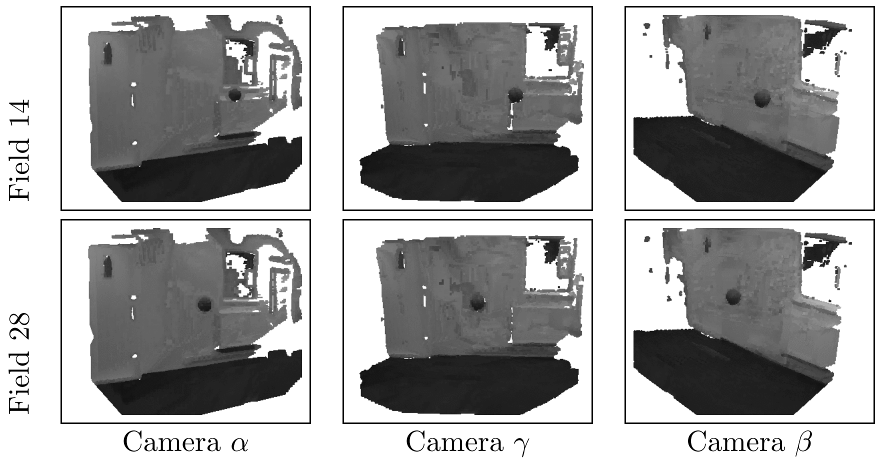


Figure 6.23: The ball-under-gravity 3D reconstructed scene as seen from original camera angles but filtered.

The 3D data can also be rendered from the novel positions used in the 2.75D results above (see figure 6.24). Again, like the 2.75D results, without limiting the uncertainty for voxels, there are large quantities of information that restrict the comprehension of this reconstructed data. Figure 6.25 shows this reconstructed data from new views. Comparing these results with those for the 2.75D case (figure 6.19) indicates that there are similar, but now more intensified protrusions present that obscure the ball. They are intensified by the inability to handle the edge information appropriately.

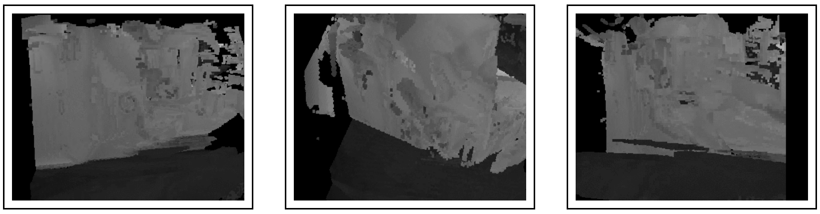


Figure 6.24: The ball-under-gravity 3D reconstructed scene as seen from novel camera.

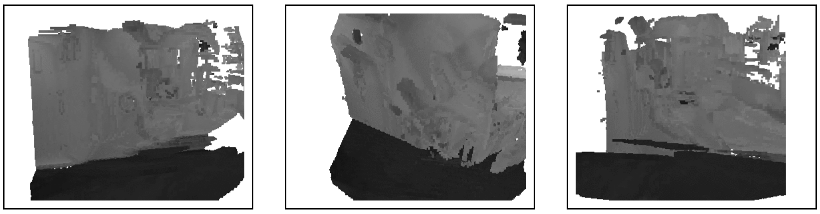


Figure 6.25: The ball-under-gravity 3D filtered reconstructed scene as seen from novel camera angles.

The removal of the background, as shown in figure 6.26 produces a scene which clearly has located the moving ball. There is, however, increased noise in these images. The reconstructed data was more susceptible to noise from the capture process than the 2D segmentation (and hence also the 2.75D background removal); the use of a VI technique instead for the background removal, in a similar manner to the 2.75D algorithm, is left for future work.

Figure 6.27 shows the ball, but it is now entwined within the noise of the voxel space. As with the 2.75D method, the ball is not spherical, but pointed, with the restrictions placed being along the projection rays from the cameras.

6.4.4 Parameter extraction

There were three stages in the extraction process of the ball. For the first two a GA was used, with an increasing level of resolution but decreasing ranges of parameters. During the first pass it was informed that the ball was moving quickly from left to right, or right to left, or slowly with no discernible direction, depending on the sequence under study. The initial values and steps for the various parameters for the first sequence (that used in the above diagrams) can be seen in table 6.5. Although the GA successfully located the region of the peak in the voting space, only about 50% of the peaks were found (there were ten separate trials for each sequence),

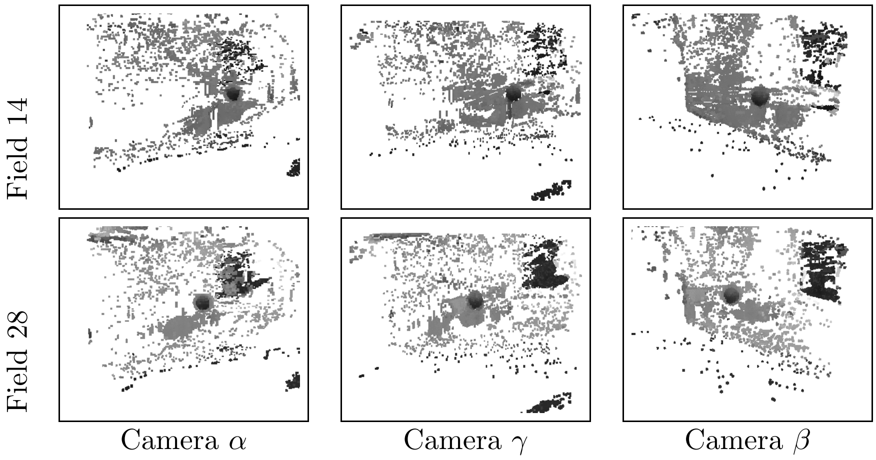


Figure 6.26: The 3D filtered background removed scene from original camera angles.

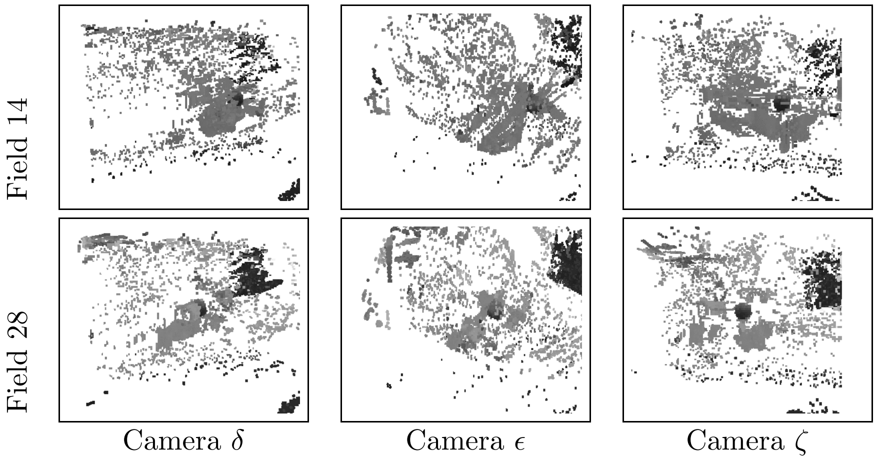


Figure 6.27: The 3D filtered background removed scene from novel camera angles.

and thus the third stage was a very localised grid search of the parameter space. Eventually the GA would have found the peaks as there was often only a single bit change required in the genome, however, the GAs were not run until complete convergence as this is intensive work that does not have any advantages over a grid search of the region; the GAs were terminated after 500 iterations (there were 501 population members).

To provide a means of confirmation, the odd and even sequence members, i.e., those formed from the odd and even fields, were tested separately. The results of the four sequences using the three systems can be seen in table 6.4, where the results from the odd and even fields are separately listed.

The results in table 6.4 clearly show a correlation between the different algorithms and also a correlation between the odd and even frame analyses of the same algorithm. The value for gravity has been successfully predicted, well within the

Seq #	System	x_0 mm ±10	y_0 mm ±10	z_0 mm ±10	v_x mms^{-1} ±10	v_y mms^{-1} ±10	v_z mms^{-1} ±10	g mms^{-2} ±10	r mms ±4
1	2D	1280	220	1130	-2130	4360	70	-9880	128
		1280	230	1120	-2130	4340	90	-9870	128
1	2.75D	1280	210	1120	-2120	4350	100	-9860	120
		1280	220	1130	-2120	4310	90	-9780	120
1	3D	1280	210	1110	-2130	4340	100	-9850	120
		1280	210	1110	-2130	4350	100	-9890	124
2	2D	-630	370	720	1530	5490	-270	-9760	124
		-630	380	720	1530	5460	-280	-9710	128
2	2.75D	-640	360	720	1560	5480	-270	-9750	112
		-640	360	720	1560	5480	-270	-9750	112
2	3D	-640	350	710	1550	5490	-280	-9760	116
		-640	350	710	1540	5490	-280	-9750	116
3	2D	760	190	460	-1010	5080	340	-9800	124
		760	200	460	-1010	5050	340	-9750	128
3	2.75D	770	180	460	-1010	5080	350	-9800	116
		770	190	450	-1010	5080	370	-9760	116
3	3D	750	180	450	-1000	5090	350	-9850	120
		760	170	450	-1010	5110	350	-9850	120
4	2D	180	300	760	260	4960	210	-9850	124
		180	300	760	260	4960	210	-9840	124
4	2.75D	190	290	770	250	4940	210	-9800	116
		190	290	770	250	4950	210	-9820	116
4	3D	180	280	760	260	5000	200	-9930	120
		180	280	760	250	5000	200	-9910	120

Table 6.4: The extracted parameters of the model for the ball under the influence of gravity.

limitations caused by the radial distortion effects: the error caused by a radial distortion that gives rise to two pixel deviation, as was noted during the calibration (it was four pixels, but the images have been halved in size), would yield a distance error of about 20 mm; the sequences were in the region of one second long, and thus the ball would reach the apex of its flight at about half a second, and thus $80\,\mathrm{mm\,s^{-2}}$ would be the estimate of the error present. Only the 3D algorithm for the last example exceeds this error margin, but note must be made that there are also errors in the estimates of the extrinsic and intrinsic parameters of the cameras.

More significantly, the radius is always underestimated by 2.75D algorithm and overestimated by the 2D algorithm. This is because the aliasing around the edge of the ball is not included within the representation of the ball of the 2.75D, but is for the 2D. The 3D algorithm also will not correlate the aliasing, but the error of the voxels is 25 mm, not the 10 mm resolution used for the 2D and 2.75D at the depths of interest and thus the effect is very much less noticeable. As with the synthetic

		x_0 mm	y_0 mm	z_0 mm	v_x mms^{-1}	v_y mms^{-1}	v_z mms^{-1}	g mms^{-2}	r mm
Stage 1	Minimum	0	0	-500	-6300	0	-2500	-14000	100
	Maximum	3100	1500	1000	-1000	5000	2500	-7000	220
	Step	100	100	100	100	100	100	1000	8
Stage 2	Minimum	1000	50	600	-2400	4000	-100	-11000	100
	Maximum	1500	525	1375	-1625	5775	475	-8025	148
	Step	25	25	25	25	25	25	25	4
Stage 3	Minimum	1260	200	1110	-2130	4270	70	-9960	116
	Maximum	1290	230	1140	-2100	4380	100	-9700	128
	Step	10	10	10	10	10	10	10	4

Table 6.5: The stages for the extraction of the ball in sequence [07217-07239].

example of the moving ball, the 3D algorithm is able to predict results that exceed its underlying representation's resolution.

Figure 6.28 demonstrates the even frames of the first sequence with the results superimposed. The white circles indicate the limit of the tested basic shape; they were created from the perfect templates described by the respective parameters, and the perimeters of these templates were then extracted and overlaid onto the original sequence. These results indicate that the 2D algorithm has found the result, although it must be noted that the white line covers the blurred region surrounding the ball that should not actually be included. The 2.75D and 3D algorithms, which opted for a smaller, and for these examples, the correct physical radius, both have results that can be seen to lie within the visible limits of the ball, but are inconsistent in their position within these blurred limits.

6.4.5 Conclusion on the ball under the influence of gravity

This first example has demonstrated that the three techniques are capable of extracting basic dynamic models from sequences of images, with the known parameters being within the tollerances allowed for in the system. The next example is of a more complex model, that of human gait.

6.5 Real world example: human gait

6.5.1 The model

In section 4.6.2, a more complicated model was discussed that is thought capable of extracting and describing human gait for biometric purposes. In this example, the same model is used but the higher, fourth harmonic, components are not used. Thus instead of the 23 parameters, there are only 19, as listed in table 6.6.

This model has many more parameters than the previous examples, and assuming that few are known or can be estimated, this makes the search somewhat awkward—there is little point in performing a GA search of such a large parameter space as there would be little probability of the peaks actually being found. Thus

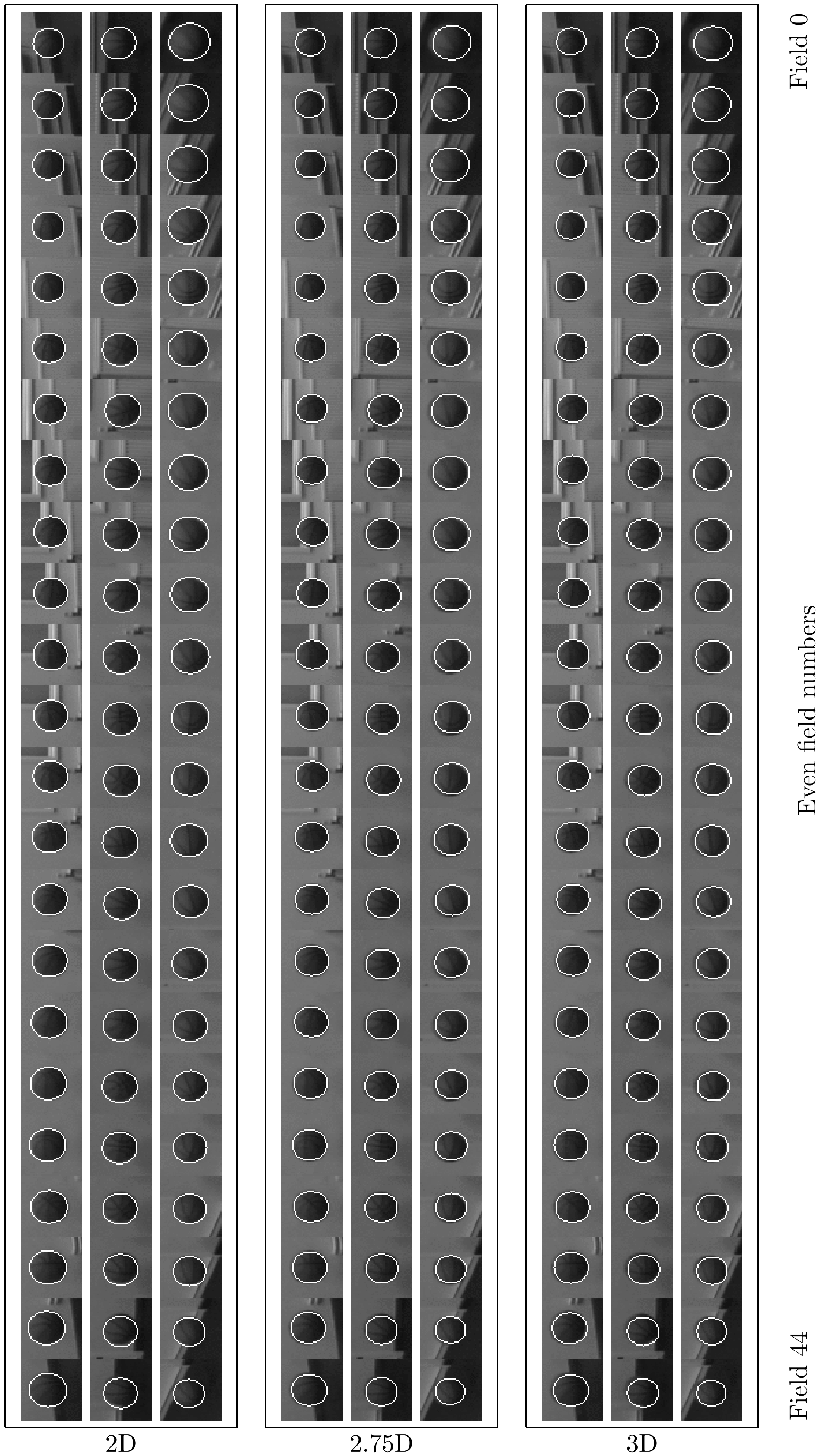


Figure 6.28: The results of the three systems superimposed onto the original images from cameras α, γ and β, which are then clipped. The white pixels indicate the outermost pixels of the sphere.

Parameter	Description	Model 1	Model 2	Model 3
$\mathbf{H_0}$	Hip 3D central position at time $t = 0$.	Yes	Yes	Yes
H_{width}	Hip width.	Constant	Constant	Constant
T_L	Thigh length.	No	No	Constant
ω_0	Step rate in full gait cycles per second. Period $= \frac{1}{\omega_0}$.	Yes	Yes	Yes
$\mathbf{V}$	Mean velocity of the person in the X and Z directions.	Yes	Yes	Yes
V_2^*	First harmonics of oscillations of hip position orientated in direction of $\mathbf{V}$.	Yes	Yes	Yes
H_2^*	First harmonics of oscillations of hip position vertically.	Yes	Yes	Yes
T_0	Mean angle of the thighs.	Yes	Yes	Yes
T_1^*	Fundamental oscillations of the thigh angle.	Yes	Yes	Yes
T_2^*	First harmonic oscillation of the thigh angle.	No	Yes	Yes
T_3^*	Second harmonic oscillation of the thigh angle.	No	No	Yes

Table 6.6: The 19 parameters required for simple gait recognition. Parameters marked with a '*' are complex.

the parameter extraction is performed using three different models with each being described by successively more parameters, as detailed in table 6.6.

For the first two models, the thigh length is not specified, instead it is assumed to be half the mean height of the hip above the ground, only in the third stage is a constant assigned to this value which was slightly smaller than this half mean height. All three models were fed a constant for the hip width; this was only an approximation and could easily be replaced with an unknown parameter if so desired.

Having said that there were three models, there were in fact many iterations of the third model, each with successively smaller steps in order to improve the accuracy of the extraction. Also, the discovered constants found in a lesser model could not be used directly in the more complicated model since, for example, incorporating another frequency component can drastically alter the fundamental oscillation, thus a margin must be allowed.

6.5.2 The source data

The source data suffers from the same flaws as the ball-under-gravity example described above in section 6.4, with blurring being quite noticeable. In this example a manual attempt at colour calibration was made by sampling various colours on the test board and adjusting the contrast and brightness of the three different channels, although this was not an ideal method due to the lack of pure colours on the test-board—much had been blurred with the circle's black circumference. The location also was not ideal as the room was small and thus subjects showed an inconsistent gait pattern as they walked since they had to first find their natural pattern and then prepare to stop before hitting another wall. Only two gait cycles have been analysed, however, the second was corrupted so much by this that the results did not visibly match the data very well; in hindsight it would have been better to analyse this gait pattern over a single gait cycle rather than over the entire sequence of about three gait cycles.

In a similar manner to the ball analysis, not all of the sequence was used for analysis, but instead one frame in four, noting that each original frame actually produced two frames due to their interlaced nature. Thus the frames $0, 4, 8 \ldots 116$ were analysed throughout, but also the set of frames $2, 6, 10 \ldots 118$ were analysed separately from when the more complicated model was utilised; in the final stage, all four sets were analysed.

Figure 6.29 shows a selection of frames from the three cameras from a single image sequence; examples of the intermediate stages of the processing can be found in appendix section B.3.

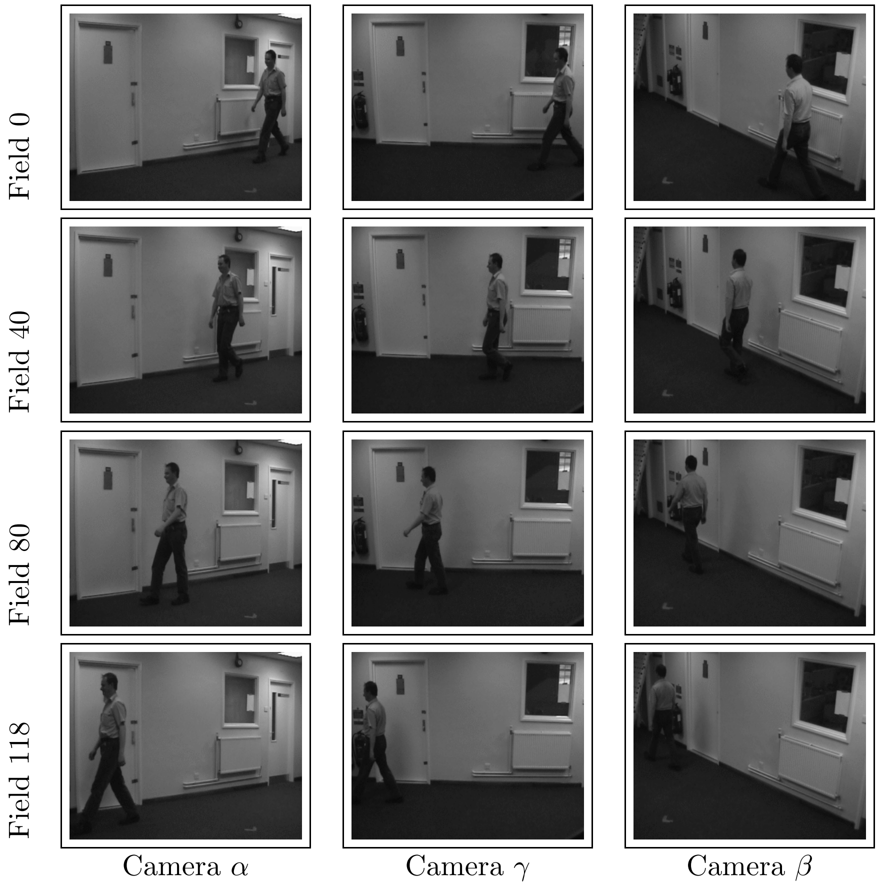


Figure 6.29: The human gait source data [sequence 11487-11546].

6.5.3 Parameter extraction

The results of the four sets of the frames from the sequence, using the three systems, can be seen in table 6.7. It can be seen that the 2D algorithm has produced significantly different values, and has instead chosen to place the hip higher and oscillate the legs less; this is more clear from figure 6.30. The 2.75D and 3D algorithms, however, have produced very similar results, as can be seen in figures 6.31 & 6.32.

These results are also visually accurate, except in the last few frames where the subject can be seen to slow in the region of the wall; the results of 2.75D algorithm show a marginal improvement over that from the 3D algorithm. The extracted parameters from the 2.75D and 3D algorithm give rise to the gait cycles as described in figure 6.33; the described cycles fit within the limits found by medical studies [58].

6.5.4 Improving the model

The 2D algorithm failed to locate the gait since there was nothing to prevent the legs being predicted higher in the body. Without using edge information, there is no advantage in placing the legs at the correct position or higher in the abdomen. This problem, however, can be circumvented in two ways: first, anchoring could have been made on the model to, for example, the knee or more simply the shoulders—the model would be encouraged to correctly place the moving legs. Second, the region of space surrounding the legs could be analysed for free space, as described in chapter 7. These methods would thus ensure the advantage of using solid objects which yields increased peak widths in the parameter space.

For future extractions of gait patterns, the initial searches can be refined by allowing only sensible amplitudes and phases, using the ranges suggested by medical studies. In these cases it would be simpler to make the higher harmonics a multiple of the fundamental gait frequency, thus allowing the extracted parameters to be compared more easily, rather than calculate their phases relative to the fundamental gait frequency.

Section 4.6.2 has already briefly described a method in which asymmetric, i.e., limping, gait can be modelled.

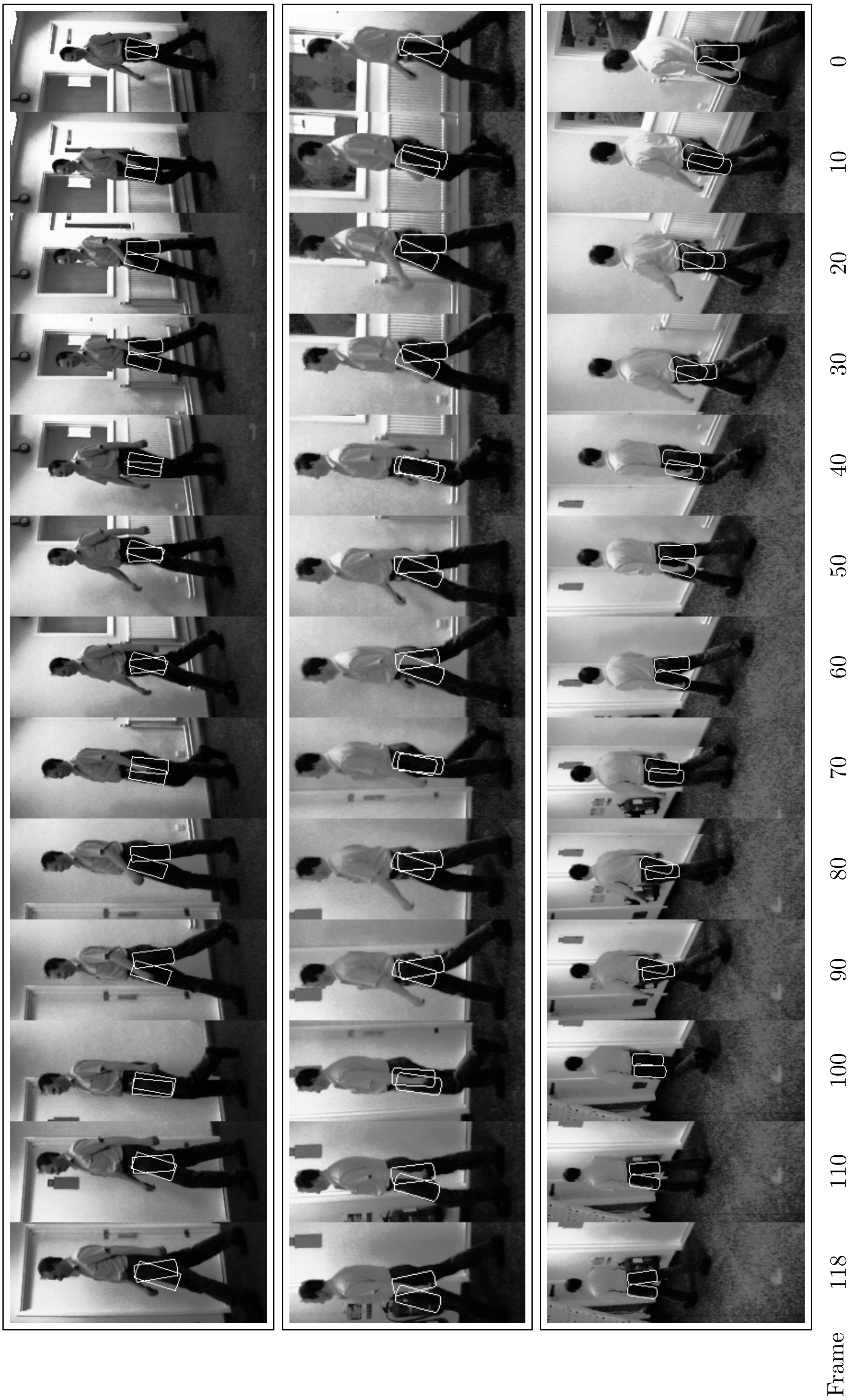
6.6 Conclusion

In this chapter various models have been applied to the respective objects in synthesised and real sequences. The examples have all shown similar and accurate results, except for the 2D algorithm which was unable to successfully extract the human gait.

The 2D algorithm is also expected to fail for a more dynamic scene where the subject cannot be successfully removed from the scene. In this situation no part of any image may be static, in which case the whole image would be included. The 3D algorithm has in these examples overcome the limitations of the lower resolution of its underlying structure, although there are small indications in the real data examples that discretisation effects may be restricting the accuracy of the analysis. For examples with relatively more chaotic motion, these restrictions should become more noticeable. The 2.75D algorithm has been seen to be consistent in all of the

	Step	2D		*Step*	2.75D				3D			
		0	2		0	1	2	3	0	1	2	3
Fitness	-	0.519	0.517	-	0.403	0.403	0.402	0.401	0.428	0.426	0.424	0.424
$\mathbf{H_0}(x)$	10	1400	1400	10	1370	1370	1370	1370	1380	1380	1380	1380
$\mathbf{H_0}(z)$	10	970	970	10	970	970	970	970	970	970	970	970
H_{width}	constant 320											
T_L	constant 350											
ω_0	0.025	0.825	0.825	0.025	0.825	0.825	0.825	0.825	0.825	0.825	0.825	0.825
$\mathbf{V}(x)$	10	-1250	-1240	10	-1240	-1240	-1240	-1240	-1250	-1250	-1250	-1250
$\mathbf{V}(z)$	10	0	0	10	0	0	0	0	0	0	0	0
$Re(V_2)$	10	0	0	10	10	10	10	10	20	20	10	20
$Im(V_2)$	10	30	20	10	20	20	20	20	30	20	30	20
$\mathbf{H_0}(y)$	10	970	970	10	840	840	840	840	820	820	820	820
$Re(H_2)$	10	-10	-10	10	-20	-20	-20	-20	-20	-20	-20	-20
$Im(H_2)$	10	0	0	10	0	0	0	0	0	0	0	0
T_0	1	8	10	0.25	7.75	7.75	7.50	7.25	7.50	7.50	7.50	7.50
$Re(T_1)$	1	12	10	0.25	20.75	21.00	20.75	20.75	20.50	20.50	20.00	20.50
$Im(T_1)$	1	-10	-11	0.25	-17.00	-17.25	-17.25	-17.25	-16.50	-16.50	-16.50	-17.00
$Re(T_2)$	1	-2	-4	0.25	-5.75	-5.50	-5.50	-5.75	-6.00	-7.00	-5.50	-7.00
$Im(T_2)$	1	1	2	0.25	3.75	3.75	3.50	3.50	2.00	4.00	2.00	3.50
$Re(T_3)$	1	2	-1	0.25	2.00	2.25	2.50	2.00	2.00	2.50	2.00	2.00
$Im(T_3)$	1	1	3	0.25	2.00	2.00	2.25	2.50	2.00	2.50	2.00	2.50

Table 6.7: The extracted parameters of the gait model. The numbers beneath the system's dimension indicate which set of frames are used (modulo 4 arithmetic on frame number).



Camera α Camera γ Camera β

Figure 6.30: The results of the 2D systems superimposed onto the original images.

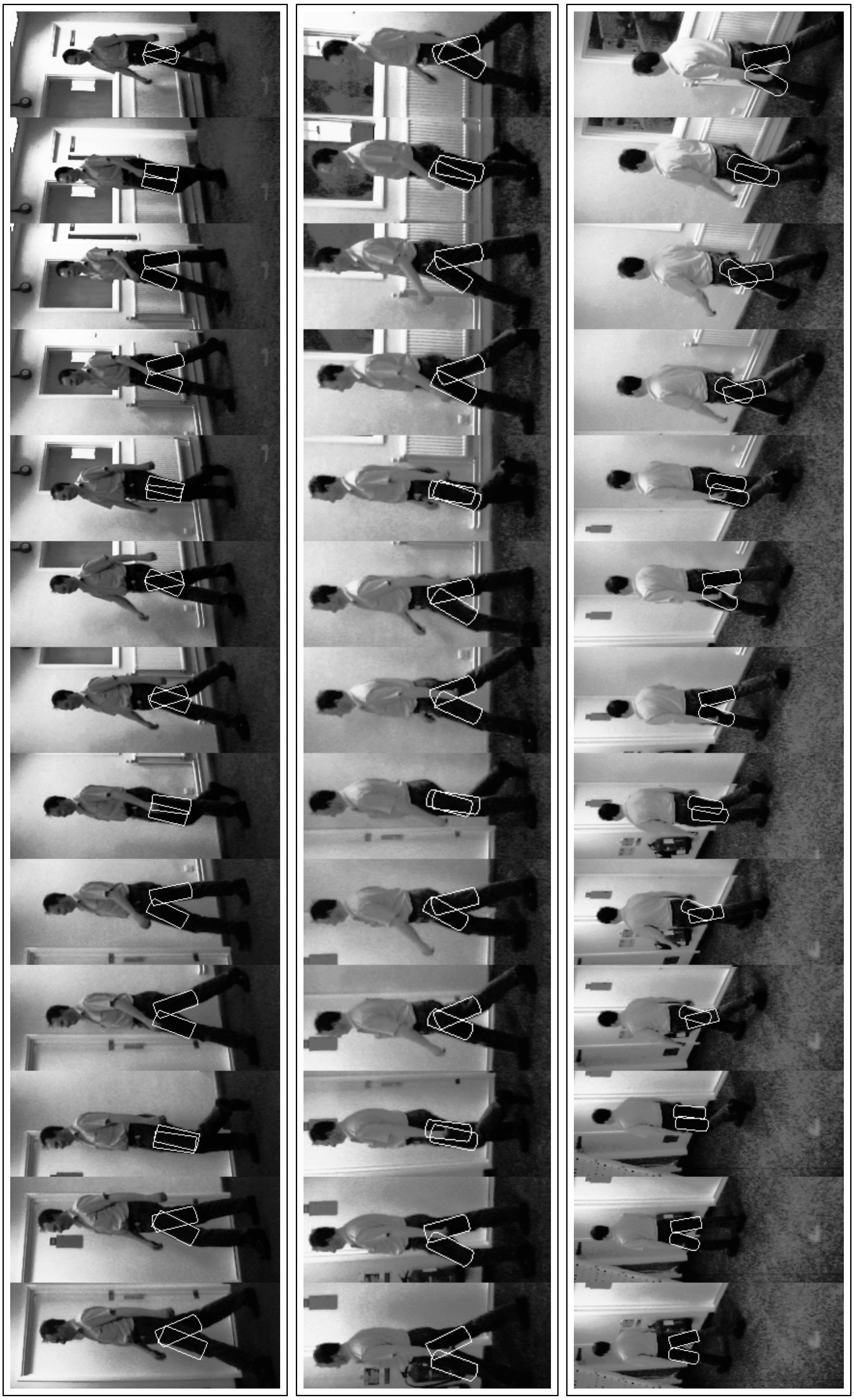


Figure 6.31: The results of the 2.75D systems superimposed onto the original images.

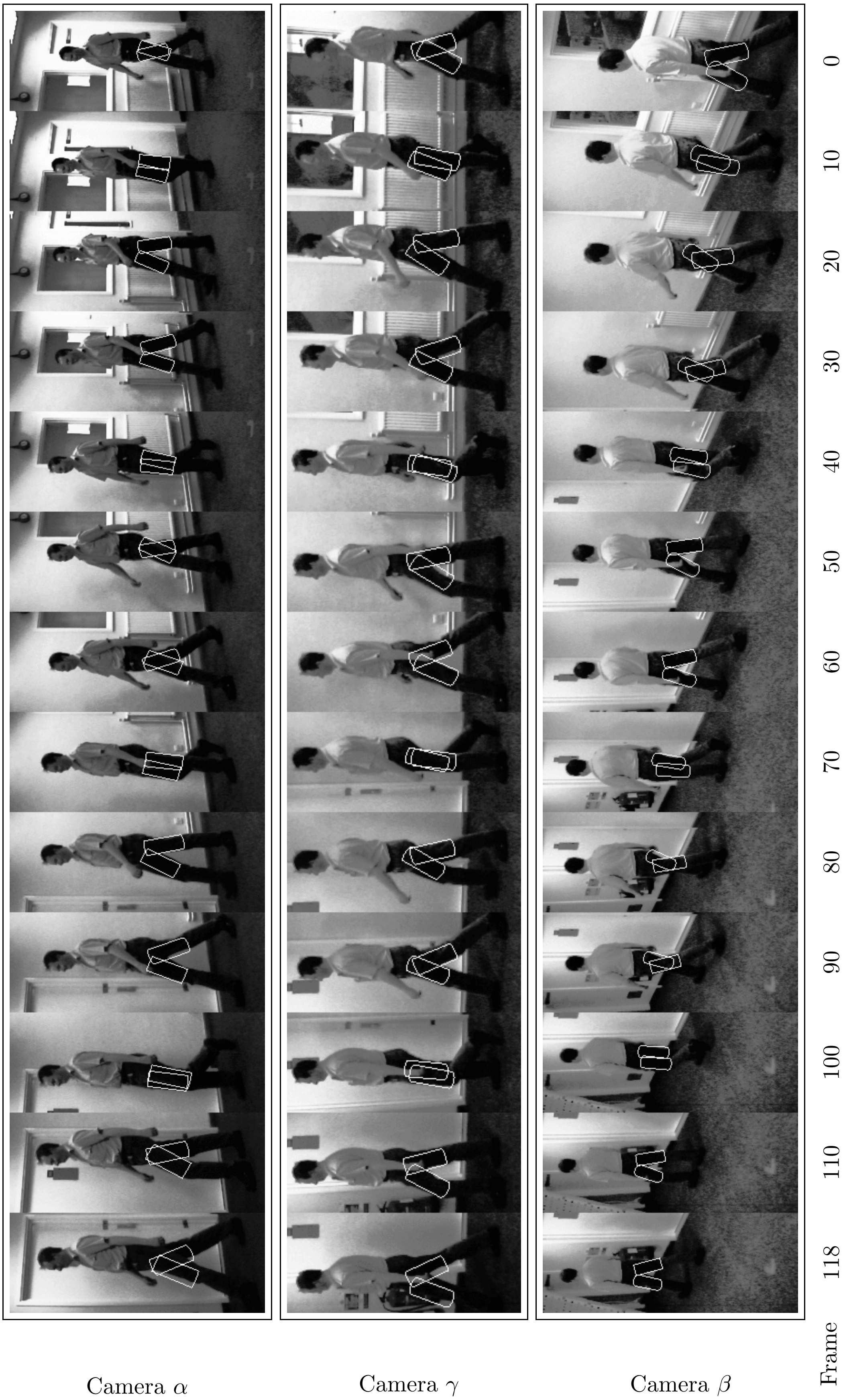


Figure 6.32: The results of the 3D systems superimposed onto the original images.

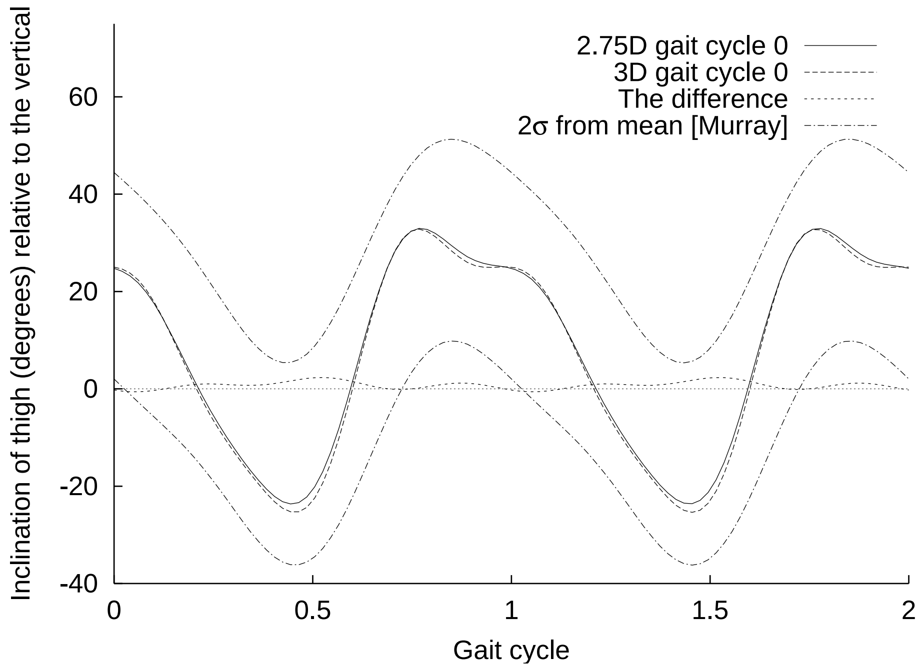


Figure 6.33: The gait cycles predicted from the 2.75D and 3D GA model, showing the 2σ deviation described by Murray [58].

examples and no situation where this algorithm would fare significantly worse than the other two can be invisaged.

However, the 3D algorithm does have the advantage that it is much faster to perform. For example on a 1.4 GHz machine, the reconstruction stage of the moving ball model is approximately 25 seconds for the 3D algorithm as compared to 5 minutes for the 2.75D algorithm, and 15 minutes for the 3D extraction stage, as opposed to the 20 minutes and 45 minutes for the 2D and 2.75D algorithms respectively (approximately 6,000,000 templates tested). These values indicate, however, that it would not be favourable, given the same number of views, to increase the resolution of the 3D voxel grid so that it becomes comparable to the underlying representation of the 2D and 2.75D algorithms. However, for larger views, the number of possible correspondences that could be made by the 2D and 2.75D systems would increase, unlike the 3D system where the voxel-grid resolution is independent.

Summarising, the 2D algorithm is ideal if the object can be segmented from the images and if the dynamic motion is not restricted within its own visual hull; the 3D algorithm should be used if the motion is simple and a less accurate method is required; the 2.75D algorithm should be used for more complicated scenes where an accurate extraction of the parameters is required, possibly having used the 3D algorithm to locate the region of interest in the parameter space.

However, the 2D algorithm has other advantages: first, the views do not need to be synchronised temporally, although the relative timing is required—each view is not compared with the others and it is only the evidence gathering procedure that finds the best fit to the observed data. The 2D algorithm can operate on single view sequences, and thus provide a means to extract 3D models from a sequence of 2D images. This latter feature can also be performed by the 2.75D algorithm, however, this would be of no additional value.

Concluding, in this chapter, the three complete systems have been demonstrated and the results compared, showing in these examples, little difference. Scenarios where significant differences occur have been discussed, as has the relative merits of the different algorithms.

Chapter 7

Concluding remarks

7.1 Introduction

In this thesis, three systems have been presented that are capable of extracting dynamic objects from a sequence of images. A gait recognition system was not the aim of this research, however, it is a possible application that could be investigated, especially as it has been shown that complex gait signatures can be produced. This chapter concludes the work presented and suggests other future directions and improvements that could be made to the algorithms.

7.2 Scene reconstruction

In chapter 1, it was concluded that the most suitable method to extract 3D dynamic objects was to do so either directly from the 2D images or from reconstructed 3D data. The latter resulted in a survey of different methods that have been developed elsewhere, which culminated in the decision to use an algorithm based upon Volume Intersection (VI). This has been shown in chapter 2 to bear close resemblance to an evidence gathering procedure and thus noise tolerance can be introduced if required. Other algorithms were also presented that enabled non-segmented scenes to be reconstructed, including 'voxel coloring' [73]. The advantage of these is that more abstract scenes can be analysed, and the problems regarding the visual hull and phantom shapes are reduced.

Initial research investigating the 3D reconstruction problem led to an algorithm similar to that described by Culbertson et al. [14] and Eisert et al. [23], although it introduced the concept of sided-voxels and a statistical presence of voxels. This has been shown to produce good reproductions of real scenes, as well as confer information in synthetic scenes. The sides on the voxels were introduced so that facing views could not contradict each other; for example, looking at a sheet of paper that is coloured differently on either side.

After further analysis of the 3D results it was discovered that the voxel space was a poor representation for scene reconstruction from 2D images. The failure was due to both the inefficiency of the representation if the scene was to be reconstructed with the highest fidelity, and the limited range of the voxel grid: outside scenes would be noisy if distant objects were in view as these would not be correlated correctly. This led to a more ideal representation which was described as being 2.75D (see chapter 3). This representation is an extension of 2.5D images, where each pixel has a single depth, except in 2.75D, each pixel has many associated depths. This has been shown to lead to a more natural description of the scene, and also to permit near-infinite scenes to be reproduced. However, the order of processing can be up to the order of the cube of the number of views. This representation is thus only practical for a limited number of cameras.

As well as demonstrating how VI can be described mathematically in this representation, a colour 2.75D reconstruction algorithm has been formulated. This uses the angle between two cameras at a point to describe how likely it is that they would see the same colour; this is thus an improvement on the sided-voxels used in the 3D algorithm. The 2.75D algorithm has been shown in chapter 5 to naturally represent the object at the highest resolution.

7.2.1 Improvements to the reconstruction

Unfortunately, comparisons have only been made between the systems that have been created specifically for this research and not with others elsewhere. As described in section 1.3, there have been many recent efforts in the area of non-segmented 3D reconstruction, in particular algorithms derived from, and including, 'voxel coloring' by Seitz and Dyer [73]. It would be advantageous to compare the 3D reconstruction algorithm described in this thesis with such algorithms. Also, the recent use of warped voxel spaces by Slabaugh et al. [81] so that larger regions of interest can be studied could also be incorporated.

Research should also be made into a voxel coloring or space carving algorithm that would be applicable to the 2.75D algorithm. It would most likely be implemented as depth carving, with each pixel initially being defined over all possible depths which would then be gradually eroded.

However, other systems aside, there are many further improvements that could be made to the algorithms, including the implementation of the less-approximate 2.75D algorithm which is described in section 3.5.

Surface influence

Non-segmented 3D reconstruction is naturally noisy due to the large number of similar shaded pixels in an image. The most common noise is a form of salt-and-pepper noise, where spurious voxels are deemed present. Snow et al. [82] noticed that such noise was a problem with VI, and thus described an energy-based algorithm to calculate the object, which could introduce noise tolerance. Implementing this for non-segmented scenes is not envisaged, but instead the reconstructed space should be influenced by the local neighbourhood.

The 3D algorithm should first produce a fitness measure for each voxel in the standard manner, but then each voxel should be re-examined, noting the fitness measure of the 26 neighbours; even a simple averaging function might suffice to produce the desired effect. The 2.75D algorithm is complicated by the fact that when a pixel is projected, the neighbours along the same ray may correspond to inconsistently sized volumes. Therefore it would be sensible to integrate the neighbouring fitness over a fixed range of depth relative to the point location.

Understanding the constants

There are five constants for both the 2.75D and 3D reconstruction (excluding the constants relating to the 3D voxel grid resolution, placement and scaling), and currently these have only been assigned values experimentally. It would thus be advantageous to perform a study to investigate their effects on reconstruction. This should include experiments to analyse the effects of perturbing the colour calibration on a camera. Such studies on the constants were not performed systematically during the development in the present study due to the drive to produce a fully working system; the computing equipment available at that time is also significantly inferior to that which is available now and would have taken many months to complete.

Using the visual hull

As already mentioned in section 4.2.3, it would be useful to use the VI visual hull as a filter on the 3D voxel data. This should reduce the noise in the images viewed. However, the 2.75D algorithm is also not properly restricted to the visual hull, but instead each view is masked by the segmented image. Again, restricting the 2.75D algorithm to the visual hull should enable better-than-hull results to be obtainable. Calculating the VI visual hull takes a fraction of the time of the colour algorithms, and thus there would not be a significant increase in workload. In fact, the extraction stage would take less time as there would be less data to process, and thus this should compensate the time to processes the additional VI visual hull.

Marching cubes

Although the 2.75D algorithm has not been presented for use as a static model capturing method, it could in theory be used in this manner. It would thus be appropriate to investigate methods to extract surface detail for use in computer graphics. Lorenson and Cline [53] described an algorithm that converted a voxel space into a triangular mesh that represented the surface of the object in the space.

This was performed by considering which of the eight corners of the voxel could be deemed as inside or outside the object, thereby producing a few triangles for each voxel which represented the segmenting surface; this algorithm was termed 'marching cubes'. It would not be unreasonable to assume that an equivalent algorithm exists for the 2.75D representation, which would thus introduce the advantages of near-infinite sized scenes and the more efficient high fidelity representation.

Self calibration

An unsuccessful attempt has been made to automatically calibrate the cameras using a test board, as explained in section 6.3, with the problem arising from the poor angle to the board from one of the cameras. However, even for the other cameras, there was a considerable error between the predicted and calculated calibration values. To simplify the calibration, the radial distortion, being a 2D process, could actually be measured prior to the camera's use in the data capture by studying a regular pattern from a position that is normal to it. If the camera's field-of-view is not altered, this radial distortion will be constant. Modelling the radial distortion prior to the 3D spatial calibration will undoubtedly improve the estimate for the camera's parameters, and will thus also improve the reconstruction and extraction results.

7.3 Dynamic model extraction

In chapter 4 a generalised method to represent arbitrary 3D dynamic objects was presented using a method akin to constructive solid geometry (CSG), a tool commonly used in computer graphics. A selection of basic shapes are transformed to describe the object at a particular moment in time. The transformations are described by the model's governing parameters, and examples have included a simple translating sphere to a human thigh gait model.

The models describe templates that are used in a template matching (TM) algorithm to find the best match in a sequence. TM is known to produce the equivalent result to the evidence gathering procedure of the Hough Transform (HT) which is well regarded for its high noise tolerance. The generic nature of the description allows the same model to be used for extracting the best match from segmented 2D images and 2.75D and 3D reconstructed scenes.

In order to extract the objects using TM, it was known that even for moderately complicated models representing and processing the various data structures required would be infeasible, and thus an alternative method was sought. Genetic Algorithms (GAs) were introduced as the method to search these high dimensional spaces. GAs search the spaces in a evolutionary style, but as with all approximate searching methods, it is necessary to increase the probability of a part of the peak in the space to be visited. It was thus shown that searching for volumes in 3D, or areas in

2D, maximised this probability by widening the peak. However this also introduced localisation issues, and thus a weighting factor had to be used in conjunction with the ability for shapes to dynamically alter in size.

Using the segmented 2D image data and the 2.75D and 3D reconstructed scenes, this dynamic modelling has been shown to be capable of extracting relatively simple synthetic static and dynamic objects, with the former examined closely in chapter 5 where, a study was made for the effects of noise on the three systems. Both synthetic and real objects have been extracted in chapter 6, ranging from describing a simple translating sphere, to producing a feasible human gait pattern.

7.3.1 Improvements to the extraction

As with the reconstruction stage, a useful comparison to make would be between the systems described in this thesis and others that are being used elsewhere. The work of Bottino et al. [10] is of interest as it uses VI for reconstruction and then matches a body to the shape before tracking it through a scene. This would thus give an opportunity to demonstrate the expected effectiveness of the evidence gathering approach presented here. Comparisons with other human model tracking algorithms and statistical-based human gait analysis systems would also be beneficial, however, it would require a huge effort to implement them.

In chapter 6, many suggestions for improvements to the extraction stage were made, which are now summarised below.

Volumetric Edge Detection

In the extraction stage, constructive solid geometry (CSG) was used to describe the templates which were thus volumetric in nature, but there was a notable problem with this representation. All of the previous work on the Hough Transform (HT) that were described in section 1.4.1 used edge-detected 2D images, and thus the equivalent in 3D would be to use surface detection. It was explained in section 4.5 that using edge-detected 2D images produces very narrow peaks and thus would be less suitable for searching with a GA. However using edge-detected 2D images helps to ensure that a circle will not be recognised so easily as a square, and likewise in 3D, the surface of a sphere would not be recognised as that of a cube.

To overcome this problem, the CSG objects described in chapter 4 were allowed to grow, and with the fitness weighting, resulted in a solid sphere matching another sphere better than a cube. This method is not suitable if the objects are assumed to be constant in size, and thus an alternative method must be used.

An alternative method would be to use surface-detected 3D data and extend the work of Samal and Edwards [71] who researched the extraction of natural shapes using the HT with 2D edge-detected images. It was realised that, for example, recognising a picture of a leaf is awkward due to the many perturbations that could cause problems which thus cause a poor peak to be produced. Therefore Samal and Edwards [71] suggested the use of a 'template' that was a wide line, thus in essence producing an algorithm that was an amalgamation of the area and edge-detected methods; this would certainly produce a more suitable peak for a GA. Surface-detected 3D data will also significantly reduce the amount of data to be tested and thus would improve the processing time.

However, the 2.75D and 3D data produced are relatively noisy in character, and thus surface-detecting is probably not suitable. An alternative method would be to use the volumetric method presented in this report in conjunction with halos-of-void. For example, for the human gait model, the thigh can be defined as a solid cylinder, but surrounding its curved edge, a region of space can be tested for the absence of data. The halo would act in a contrary manner to the main solid cylinder, thus any data present in it would be a penalty and a void would be a benefit. This method would thus still be volume-based, and incorporate a form of edge-detection. A comparison of this volume-based method and the surface-detected method should be made to establish which is most suitable. Additionally, a geometric-based algorithm could be formulated that uses the 'marching cubes' algorithm highlighted above.

Removing a constant

During the extraction, a fitness was assigned to a template using the equation:

$$w = \frac{v_a}{v_p + k} \tag{7.1}$$

where v_a and v_p are the actual and the potential vote of the template respectively, and k is a constant. It was found in section 4.5 that this constant should ideally be the potential vote of the required template, but this is obviously an ill-posed problem since the ideal template is not known. However, a method has been conceived to remove the requirement to calculate the constant in advance. On the first iteration of the GA, a moderate value of k could be selected, even a value of 1 would probably be acceptable; note that a value of 0 should never be chosen as this will lead to only small objects being located, as explained in section 4.5. On subsequent iterations, this constant is revised so that it equals the potential vote of the best fitting object. Although at first, the constant will change rapidly, since the GA will not have properly started to converge, the effect of changing the weighting factor will not be too significant on the overall process. The choice of the initial value may actually be unimportant if the templates are restricted to a minimum size. Thus it can be concluded that the constant is effectively being removed from the set-up of the extraction stage although it is still present in principle.

Alternative searching methods

A GA was chosen as the tool for searching for features in the large parameter space due to the success others have had with them to perform a similar task. However, there are many other techniques, Simulated Annealing and Gradient Ascent which could also be investigated. Multi-resolutional methods were actually employed in part in the extraction stage, as can be seen in chapter 6.

Sub-pixel sampling

If optimum fidelity is required without regard of the processing cost, the final suggestion for improvement in the extraction stage is to analyse the object using sub-pixel and sub-voxel accuracy; this would thus allow, for example, a sphere template to be more sphere-like in nature. Thus when comparing voxels in the template with those in the data, a weighting factor could be used to describe how much of the voxel in the template actually lies within the template's sphere. This weighting factor would thus influence the contribution that a voxel in the data provides.

Implementing this, however, is only suggested when the voxel space becomes relatively coarse for the size of object to be extracted. In many cases it may not actually produce a significant improvement, though this is dependent on the type of motion. For example, a sphere traveling at exactly one voxel per frame in the direction of one of the voxel space's axes cannot be resolved any better than to the accuracy of one voxel.

7.4 The complete systems

In chapter 1 it was concluded that there were two suitable methods to extract 3D dynamic objects from arbitrary scenes. One would be to use a sequence of 2D segmented images and search for a 3D mathematically described model mapped onto the 2D images using evidence gathering as the extraction tool. The second would be to reconstruct the scene prior to segmentation and then search for the 3D model in this 3D space. Three systems have been developed: the proposed segmented 2D method and the 2.75D and 3D reconstructed scene methods.

These three systems have been analysed in chapters 5 & 6 and the suitability to various scenarios discussed. It has been shown that whilst the 2D system is more tolerant to noise caused by poor calibration of cameras, it is highly dependent on successful segmentation. It is surmised that, for example, gait extraction using the segmented 2D system would not be possible in crowded scenes. However, the 2D system can be used in conjunction with unsynchronised cameras, which the other two systems cannot.

The 3D system has been shown to introduce noise due to its inappropriate underlying representation, which have been seen to slightly influence the extracted results. Nevertheless it has shown favourable results. The 2.75D system has been demonstrated to extract the parameters of objects with the highest fidelity, however, the order of processing makes it unsuitable for large numbers of cameras.

This research has led to a novel approach to 3D voxel-based reconstruction as well as a novel representation, named 2.75D. Also, a novel method has been presented to extract 3D models from 2D segmented images and 2.75D and 3D reconstructed sequences using a description akin to constructive solid geometry; these systems have been contrasted and compared. In conclusion, three systems have thus been described which are capable of estimating 3D motion parameters with success from multi-view images by non-invasive means.

References

[1] J.K. Aggarwal, Q. Cai, W. Liao, and B. Sabata. Articulated and elastic non-rigid motion: A review. In *Proc. of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 2–14, 1994.

[2] J.K. Aggarwal, Q. Cai, W. Liao, and B. Sabata. Nonrigid motion analysis: Articulated and elastic motion. *Computer Vision and Image Understanding (CVIU)*, **70**(2):142–56, 1998.

[3] N. Ahuja and A.L. Abbott. Active stereo: Integrating disparity, vergence, focus, aperture, and calibration for surface estimation. *IEEE Transactions on PAMI*, **15**(10):1007–29, 1993.

[4] N. Ahuja and J. Veenstra. Generating octrees from object silhouettes in orthographic views. *IEEE Transactions on PAMI*, **11**(2):137–49, 1989.

[5] M. Armstrong and A. Zisserman. Robust object tracking. In *Proc. of the Asian Conf. on Computer Vision (ACCV '95)*, volume 1, pages 58–62, 1995.

[6] D.H. Ballard. Generalising the Hough transform to arbitrary shapes. *Pattern Recognition*, **13**(2):111–22, 1981.

[7] P. Beardsley, P. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In *Proc. of the European Conf. on Computer Vision (ECCV '96)*, volume 2, pages 683–95, 1996.

[8] S.D. Blostein and T.S. Huang. Error analysis in stereo determination of 3-D point positions. *IEEE Transactions on PAMI*, **9**(6):752–65, 1987.

[9] J.S. De Bonet and P. Viola. Roxels: Reponsibility weighted 3D volume reconstruction. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV '99)*, volume 2, pages 418–25, 1999.

[10] A. Bottino, A. Laurentini, and P. Zuccone. Toward non-intrusive motion capture. In *Proc. of the Asian Conf. on Computer Vision (ACCV '98)*, volume 2, pages 416–23, 1998.

[11] J.C. Carr, W.R. Fright, A.H. Gee, R.W. Prager, and K.J. Dalton. 3D shape reconstruction using volume intersection techniques. In *Proc. of the IEEE Sixth Int. Conf. on Computer Vision (ICCV '98)*, volume 1164, pages 1095–100, 1998.

[12] C. Cedras and M. Shah. Motion–based recognition: a survey. *Image and Vision Computing*, **13**(2):129–55, 1995.

[13] C-H. Chien and J.K. Aggarwal. Model construction and shape recognition from occluding contours. *IEEE Transactions on PAMI*, **11**(4):372–89, 1989.

[14] W.B. Culbertson, T. Malzbender, and G.G. Slabaugh. Generalized voxel coloring. In *Proc. of the Int. Workshop on Vision Algorithms. Vision Algorithms: Theory and Practice. (Lecture Notes in Computer Science Vol. 1883)*, pages 100–15, 2000.

[15] D. Cunado, J.M. Nash, M.S. Nixon, and J.N. Carter. Gait extraction and description by evidence–gathering. In *Proc. of the Int. Conf. on Audio- and Video-Based Biometric Person Authentication (AVBPA '99)*, pages 43–8, 1999.

[16] D. Cunado, M.S. Nixon, and J.N. Carter. Extracting a human gait model for use as a biometric. In *Proc. of the Colloquium on Computer Vision for Virtual Human Modelling*, 1998.

[17] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. of Computer Graphics (SIGGRAPH '96)*, volume 30, pages 303–12, 1996.

[18] D. Cyganski, W.F. Noel, and J.A. Orr. The analytic Hough transform. In *Proc. of the SPIE—The Int. Soc. for Optical Engineering*, volume 1260, pages 148–59, 1990.

[19] N. D'Apuzzo, A. Gruen, R. Plankers, and P. Fua. Least squares matching tracking algorithm for human body modeling (TC-V06-03). In *Proc. of the XIXth ISPRS Congress*, 2000.

[20] S. Das and N. Ahuja. Performance analysis of stereo, vergence, and focus as depth cues for active vision. *IEEE Transactions on PAMI*, **17**(12):1213–9, 1995.

[21] C. Dorai, G. Wang, A. Jain, and C. Mercer. From images to models: Automatic 3D object model construction from multiple views. In *Proc. of the IEEE Int. Conf. on Pattern Recognition (ICPR '96)*, volume 1, pages 770–4, 1996.

[22] R.O. Duda and P.E. Hart. Use of the Hough Transformation to detect lines and curves in pictures. *Communications of the ACM*, **15**(1):11–5, 1972.

[23] P. Eisert, E. Steinbach, and B. Girod. Multi–hypothesis, volumetric reconstruction of 3–D objects from multiple calibrated camera views. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, volume 6, pages 3509–12, 1999.

[24] A.W. Fitzgibbon, G. Cross, and A. Zisserman. Automatic 3D model construction for turn–table sequences. In *Proc. of the European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE '98)*, pages 155–70, 1998.

[25] D.M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding (CVIU)*, **73**(1):82–98, 1999.

[26] D.E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison Wesley, 1989.

[27] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The lumigraph. In *Proc. of Computer Graphics (SIGGRAPH '96)*, pages 43–54, 1996.

[28] M.G. Grant, M.S. Nixon, and P.H. Lewis. Extracting moving shapes by evidence gathering. *Pattern Recognition*, **35**(5):1099–114, 2002.

[29] T. Hamano and K. Ishii. Structure from motion by voting algorithm. *Systems and Computers in Japan*, **24**(4):23–33, 1993.

[30] R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision.* Cambridge University Press, 2001.

[31] A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *Proc. of the European Conf. on Computer Vision (ECCV '96)*, volume 1, pages 117–26, 1996.

[32] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Implicit surface–based geometric fusion. *Computer Vision and Image Understanding (CVIU)*, **69**(3):273–91, 1998.

[33] T-H. Hong and M.O. Shneier. Describing a robot's workspace using a sequence of views from a moving camera. *IEEE Transactions on PAMI*, **7**(6):721–6, 1985.

[34] P.V.C Hough. Method and means for recognising complex patterns. *US Patent, 3,069,654*, 1962.

[35] T. Joshi, N. Ahuja, and J. Ponce. Structure and motion estimation from dynamic silhouettes under perspective projection. *Int. Journal of Computer Vision*, **31**(1):31–50, 1999.

[36] S.K. Jung and K.Y. Wohn. Tracking and motion estimation of the articulated object: a hierarchical kalman filter approach. *Real Time Imaging*, **3**(6):415–32, 1997.

[37] H. Kälviäinen. *Randomized Hough Transform: new extensions.* PhD thesis, Lappeenranta University of Technology, 1994.

[38] T. Kanade, P. Rander, and P.J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Multimedia*, **4**(1):34–47, 1997.

[39] S. Kawato. 3D shape recovery by octree voting technique. In *Proc. of the SPIE—The Int. Soc. for Optical Engineering*, volume 1820, pages 40–9, 1993.

[40] S. Kawato. Extraction of three–dimensional structure from images of multiple viewpoints by two-step voting. *Systems and Computers in Japan*, **26**(10):59–67, 1995.

[41] Y.C. Kim and J.K. Aggarwal. Rectangular parallelepiped coding: A volumetric representation of three–dimensional objects. *IEEE Journal of Robotics and Automation*, **RA-2**(3):127–34, 1986.

[42] C. Kimme, D.H. Ballard, and J. Sklansky. Finding circles by an array of accumulators. *Communications of the ACM*, **18**(2):120–2, 1975.

[43] R. Kimmel, K. Siddiqi, B.B. Kimia, and A.M. Bruckstein. Shape from shading: Level set propagation and viscosity solutions. *Int. Journal of Computer Vision*, **16**(2):107–33, 1995.

[44] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In *Proc. of the European Conf. on Computer Vision (ECCV '94)*, volume 1, pages 189–96, 1994.

[45] K. Kutulakos and S.M. Seitz. A theory of shape by space carving. *Int. Journal of Computer Vision*, **38**(3):199–218, 2000.

[46] K.N. Kutulakos. Shape from the light field boundary. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '97)*, pages 53–9, 1997.

[47] M. Landy and J.A. Movshon, editors. *Computational Models of Visual Processing*, chapter 1: The Plenoptic Function and the Elements of Early Vision. E.H. Adelson and J.R. Bergen. The MIT Press, Cambridge, Mass., 1991.

[48] A. Laurentini. The visual hull concept for silhouette–based image understanding. *IEEE Transactions on PAMI*, **16**(2):150–62, 1994.

[49] A. Laurentini. How far 3D shapes can be understood from 2D silhouettes. *IEEE Transactions on PAMI*, **17**(2):188–95, 1995.

[50] A. Laurentini. How many 2D silhouettes does it take to reconstruct a 3D object. *Computer Vision and Image Understanding (CVIU)*, **67**(1):81–7, 1997.

[51] V.F. Leavers. Which Hough transform? *CVGIP: Image Understanding*, **58** (2):250–64, 1993.

[52] M. Levoy and P. Hanrahan. Light field rendering. In *Proc. of Computer Graphics (SIGGRAPH '96)*, pages 31–42, 1996.

[53] W.E. Lorenson and H.E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. In *Proc. of Computer Graphics (SIGGRAPH '87)*, pages 163–9, 1987.

[54] W.N. Martin and J.K. Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Transactions on PAMI*, **5**(2):150–8, 1983.

[55] W. Matusik, C. Buehler, R. Raskar, S.J. Gortler, and L. McMillan. Image-based visual hulls. In *Proc. of Computer Graphics (SIGGRAPH 2000)*, pages 369–74, 2000.

[56] T. McInerney and D. Terzopoulos. A finite element model for 3D shape reconstruction and nonrigid motion tracking. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV '93)*, pages 518–23, 1993.

[57] D. Meyer. Human gait classification based on hidden markov models. In *Proc. of 3D Image Analysis and Synthesis*, pages 139–46, 1997.

[58] M.P. Murray. Gait as a total pattern of movement. *American Journal of Physical Medicine*, **40**(1):290–329, 1967.

[59] J.M. Nash, J.N. Carter, and M.S. Nixon. Dynamic feature extraction via the velocity Hough transform. *Pattern Recognition Letters*, **18**(10):1035–47, 1997.

[60] M.S. Nixon, J.N. Carter, D. Cunado, P.S. Huang, and S.V. Stevenage. *BIOMETRICS: Personal Identification in Networked Society*, chapter Automatic Gait Recognition. Kluwer Academic Publishing. Ed. A.K. Jain et al., 1999. pp231–50.

[61] H. Noborio. Construction of the octree approximating three-dimensional objects by using multiple views. *IEEE Transactions on PAMI*, **10**(6):769–82, 1988.

[62] S. Petitjean. A computational geometric approach to visual hulls. *Int. Journal of Computational Geometry and Applications*, **8**(4):407–36, 1998.

[63] M. Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics and Image Processing*, **40** (1):1–29, 1987.

[64] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press, second edition, 1996.

[65] A.C. Prock and C.R. Dyer. Towards real-time voxel coloring. In *Proc. of the Image Understanding Workshop*, pages 315–21, 1998.

[66] A. Rahimi, L-P. Morency, and T. Darrell. Reducing drift in parametric motion tracking. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV 2001)*, volume 1, pages 315–22, 2001.

[67] P.W. Rander, P.J. Narayanan, and T. Kanade. Virtualized reality: Constructing time-varying virtual worlds from real world events. In *Proc. of the IEEE Conf. on Visualization*, pages 277–83, 1997.

[68] J. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV '95)*, pages 612–7, 1995.

[69] E. Ribeiro and E.R. Hancock. Estimating the 3d orientation of texture planes using local spectral analysis. *Image and Vision Computing*, **18**(8):619–31, 2000.

[70] D. Rosenfeld. *Picture Processing by Computer.* Academic Press, London UK, 1969.

[71] A. Samal and J. Edwards. Generalized Hough transform for natural shapes. *Pattern Recognition Letters*, **18**(5):473–80, 1997.

[72] Y.Y. Schechner and N. Kiryati. The optimal axial interval in estimating depth from defocus. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV '99)*, volume 2, pages 843–8, 1999.

[73] S.M. Seitz and C.R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '97)*, pages 1067–73, 1997.

[74] S.M. Seitz and C.R. Dyer. Photorealistic scene reconstruction by voxel coloring. *Int. Journal of Computer Vision*, **35**(2):151–73, 1999.

[75] S.M. Seitz and K.N. Kutulakos. Plenoptic image editing. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV '98)*, pages 17–24, 1998.

[76] P.K. Ser, C.S.T. Choy, and W.C. Siu. Genetic algorithm for the extraction of nonanalytic objects from multiple dimensional parameter space. *Computer Vision and Image Understanding (CVIU)*, **73**(1):1–13, 1999.

[77] J.W. Shade, S.J. Gortler, L-W. He, and R. Szeliski. Layered depth images. In *Proc. of Computer Graphics (SIGGRAPH '98)*, volume 32, pages 231–42, 1998.

[78] J. Shi and C. Tomasi. Good features to track. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '94)*, pages 593–600, 1994.

[79] T.M. Silberberg, L. Davis, and D. Harwood. An iterative Hough procedure for three–dimensional object recognition. *Pattern Recognition*, **17**(6):621–9, 1984.

[80] J. Sklansky. On the Hough technique for curve detection. *IEEE Transactions on Computers*, **27**(10):923–6, 1978.

[81] G.G. Slabaugh, T. Malzbender, and W.B. Culbertson. Volumetric warping for voxel coloring on an infinite domain. In *Proc. of the European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE 2000)*, pages 109–23, 2001.

[82] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2000)*, volume 1, pages 345–52, 2000.

[83] E. Steinbach, B. Girod, P. Eisert, and A. Betz. 3-D object reconstruction using spatially extended voxels and multi-hypothesis voxel coloring. In *Proc. of the IEEE Int. Conf. on Pattern Recognition (ICPR 2000)*, volume 1, pages 774–7, 2000.

[84] E. Steinbach, B. Girod, P. Eisert, and A. Betz. 3-D reconstruction of real-world objects using extended voxels. In *Proc. of the IEEE Int. Conf. on Image Processing (ICIP 2000)*, volume 1, pages 569–72, 2000.

[85] G.C. Stockman and A.K. Agrawala. Equivalence of Hough curve detection to template matching. *Communications of the ACM*, **20**(11):820–2, 1977.

[86] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, **58**(1):23–32, 1993.

[87] R.Y. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '86)*, pages 364–74, 1986.

[88] R.Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, **RA-3**(4):323–44, 1987.

[89] R.F. Vaz and D. Cyganski. 3D object orientation from partial contour feature data. In *Proc. of the SPIE—The Int. Soc. for Optical Engineering*, volume 1349, pages 452–9, 1990.

[90] S. Vedula, S. Baker, S. Seitz, and T. Kanade. Shape and motion carving in 6D. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2000)*, volume 2, pages 592–8, 2000.

[91] Y. Xiong and S.A. Shafer. Depth from focusing and defocusing. In *Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR '93)*, pages 68–73, 1993.

[92] L. Xu, E. Oja, and P. Kultanen. A new curve detection method: Randomized Hough Transform (RHT). *Pattern Recognition Letters*, **11**(5):331–8, 1990.

[93] S.M. Yamany, M.N. Ahmed, and A.A. Farag. A new genetic-based technique for matching 3-D curves and surfaces. *Pattern Recognition*, **32**(10):1817–20, 1999.

[94] P-Y. Yin. A new circle/ellipse detector using genetic algorithms. *Pattern Recognition Letters*, **20**(10):731–40, 1999.

Appendix A

Camera arrangement

A.1 Introduction

This chapter details the various camera arrangements used in experiments in this project.

A.2 Camera parameter conversion

The ray-tracer uses a different camera description method to that used by the implemented code. The relationship between those used in this report (see section 2.2.4) and those used by the ray-tracer can be seen in table A.1. As can be seen many more parameters are required by the ray-tracer to achieve the camera description in an effort to be flexible for the user. This is, however, only a summary of the basic manipulation, since further commands can be used by the ray-tracer to change the view.

'POV-Ray' camera parameter	Type	Relationship
location	3-Vector	Directly yields the position of the camera, $\mathbf{T}$
lookat	3-Vector	With **location**, yields the direction and elevation angles
sky	3-Vector	Gives the camera's rotation angle, $\theta_r = \tan^{-1} \frac{sky_y}{sky_x}$
right	3-Vector	With the image's dimensions, yields the pixel's aspect ratio, κ
angle	Scalar	With the width of the image, yields f_y

Table A.1: Summary of the relationship between the parameters used in this report and those used by the 'POV-Ray' ray-tracer.

The **lookat** vector yields the camera's direction and elevation angles, θ_d and θ_e respectively, by:

$$\theta_d = -tan^{-1} \frac{\mathbf{lookat}_x - \mathbf{location}_x}{\mathbf{lookat}_z - \mathbf{location}_z} \tag{A.1}$$

$$\theta_e = \left| \begin{array}{cc} sin^{-1} \frac{\mathbf{lookat}_y - \mathbf{location}_y}{|\mathbf{lookat} - \mathbf{location}|} & \text{for } |\mathbf{lookat} - \mathbf{location}| \neq 0 \\ 0 & \text{otherwise} \end{array} \right. \tag{A.2}$$

The scalar, *angle*, is the horizontal angle of the field of view, and thus can be used to calculate the focal lengths f_x and f_y. However, the aspect ratio, which is described by the vector **right** and the image's dimensions, alters the focal length f_x but not f_y. Hence with the width, w, of the image, f_y, is given by:

$$f_y = \frac{w}{2\tan(0.5\,\mathrm{rad}(angle))} \tag{A.3}$$

The focal length in the x direction, f_x can be calculated using the height, h, of the image and the x:y ratio of the components in the vector **right** by:

$$f_x = \frac{\mathbf{right}_x/\mathbf{right}_y \cdot h}{2\tan(0.5\,\mathrm{rad}(angle))} \tag{A.4}$$

A.3 Real world camera arrangement

This camera arrangement was used for real world data capture and for some of the synthetic data examples.

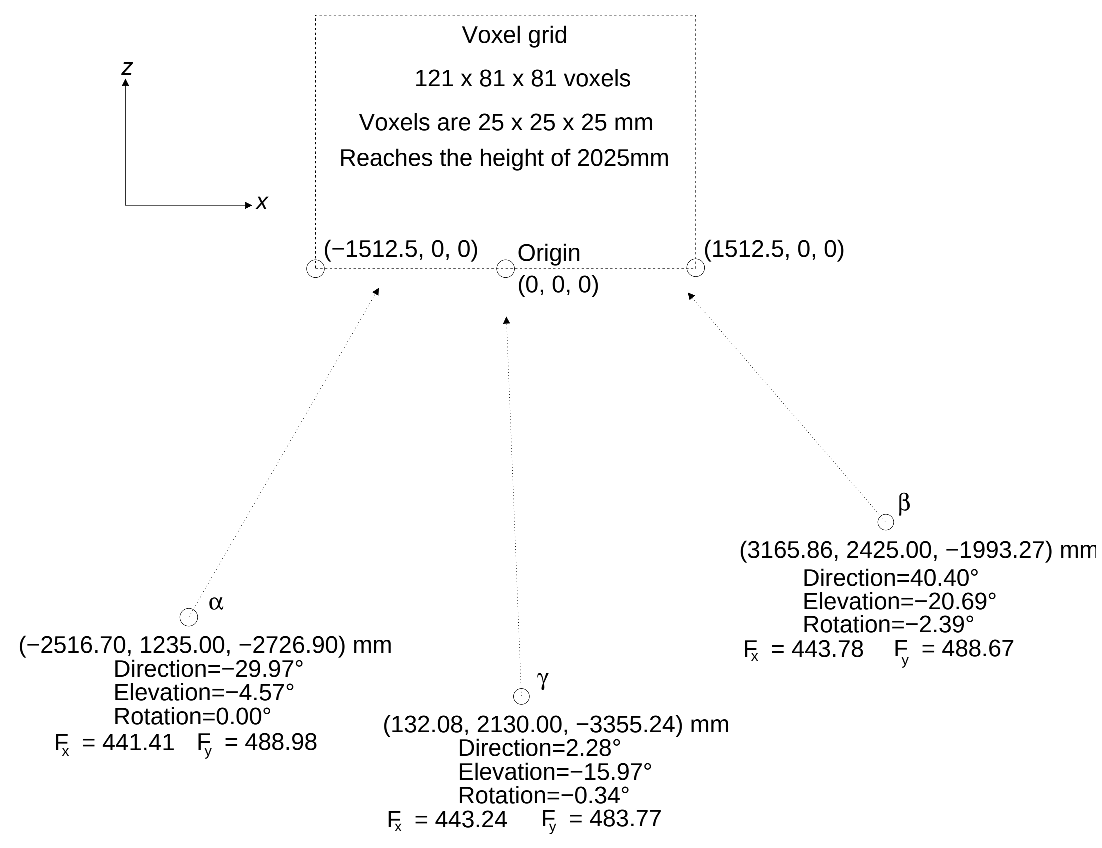


Figure A.1: Plan view of the three camera arrangement (α, β and γ), showing respective parameters, used in the experiments. All of the images have dimensions 348 × 280 pixels, with the principal point located at the centre; there is no skew factor.

By taking excess measurements, the cameras' positions were found, using trigonometry, to an accuracy of 5 mm. The cameras' three angles and two focal lengths were calibrated using the four front corners of the test board described in section 6.3.3; the calibration of each camera is independent of the others. As the four front corner positions were known, the respective four image coordinates for each camera provided eight values to solve for the five unknown parameters. A coarse grid search was performed on the entire parameter space followed by three further localised searches using successively greater resolution; the aim of the searches was to find a set of parameters that minimised the difference between the predicted and measured positions in the images of the four points. The final angle resolution of the search was $0.01°$, and the final focal length resolution was 0.02. The sum of the minimised distances of the four points was 4.25, 4.31, 1.66 for camera α, β and γ respectively. Thus each point was predicted to be approximately within a pixel of the measured value; the search with resolutions of $0.1°$ and 0.2 for the angles and focal lengths respectively produced only marginally different results. Camera γ has a smaller error because the radial distortion that is present affects the four corners in a consistent manner due to their symmetry in the picture. Note that the values shown are those for the half-sized images there were used, i.e., the focal lengths have been halved. These images have dimensions 348×280; the principal point is assumed to be at the centre of the image.

The analogue cameras used for the capture were the DFK 50H13 P, supplied by 'The Imaging Source'; the digital cameras used for the recording were the Sony DCR-TRV900E-PAL. A modified version of the Linux library 'LibDV', which is developed under the GPL, was used to decode the DV data.

A.4 Synthetic camera arrangement I

This camera arrangement was used for synthetic data creation only.

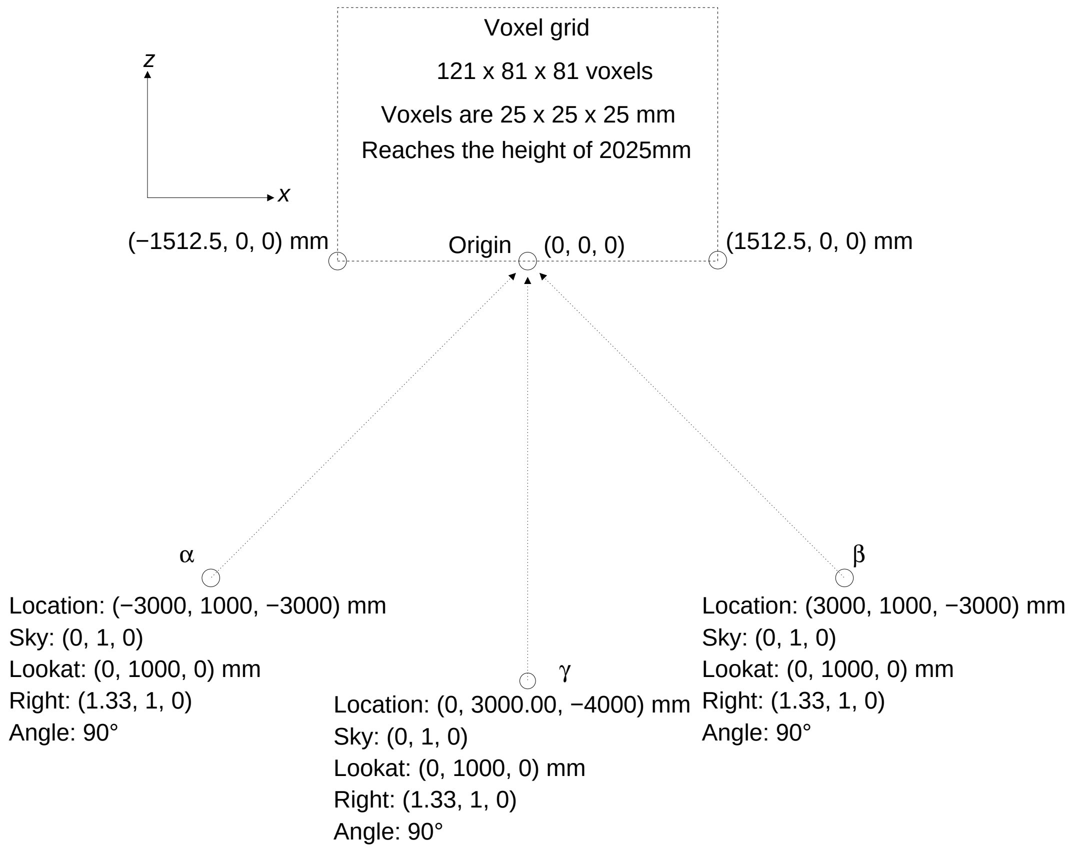


Figure A.2: Plan view of synthetic camera arrangement I (α, β and γ), showing respective parameters, used in the experiments. All of the images have dimensions 400×300 pixels, with the principal point located at the centre; there is no skew factor.

A.5 Synthetic camera arrangement II

This camera arrangement was used for synthetic data creation only.

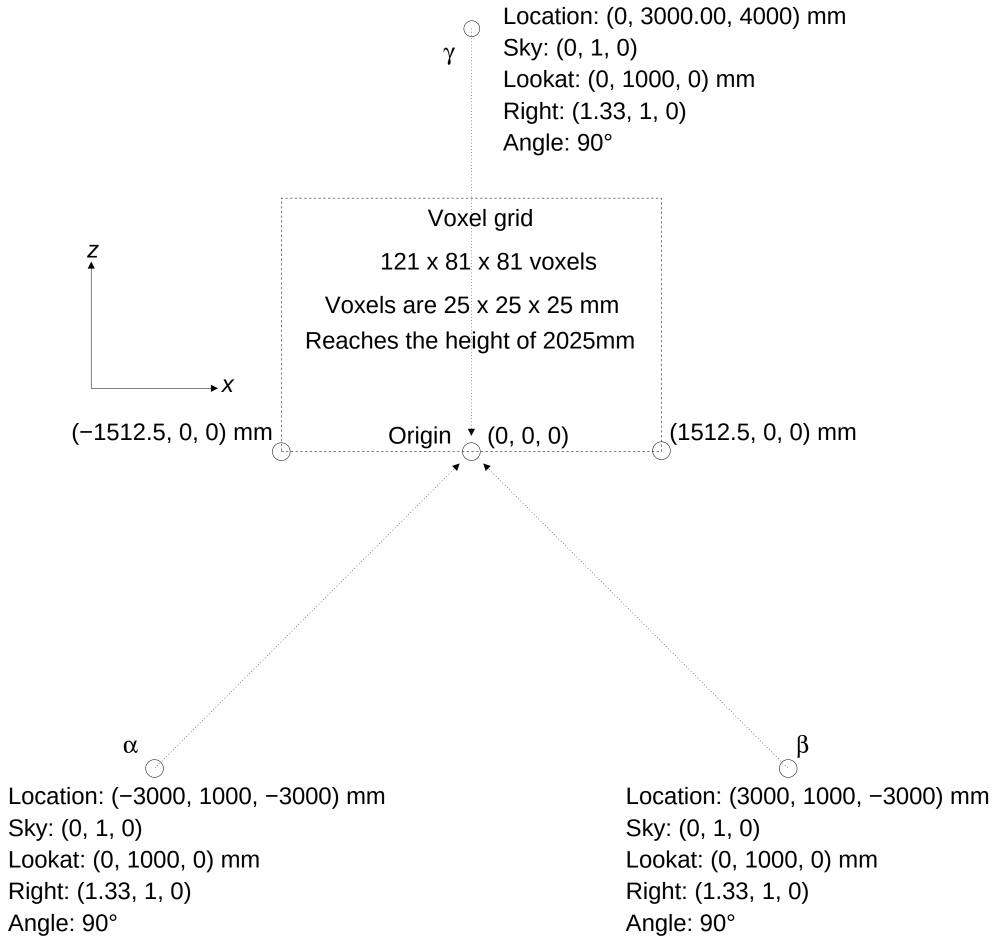


Figure A.3: Plan view of synthetic camera arrangement II (α, β and γ), showing respective parameters, used in the experiments. All of the images have dimensions 400×300 pixels, with the principal point located at the centre; there is no skew factor.

Appendix B

Further model extraction

B.1 Introduction

This appendix complements the work described in chapter 6, providing further illustration of the techniques described.

B.2 Synthetic example: analysis of a box moving around an arc

B.2.1 Setting the scene

Increasing the dimensions of the modelling, a further example is the extraction and description of a box, of unknown dimensions moving along the ground around an arc at a constant speed, with the motion described by the parameters listed in table B.1 and illustrated in figure B.1. The experiment was performed in a similar manner to that of the moving ball above, using two different camera arrangements, each with 100 trials in which each has a random set of parameters. Figure B.2 illustrates a typical sequence, showing an irregular floor texture and, once again, a complicated sky. Also present are small amounts of shadow and shading on the moving box itself; this is more clearly visible in the background removed images as shown in figure B.3. It is important to note the relative size of the box in the images which lies a moderate distance away from the cameras.

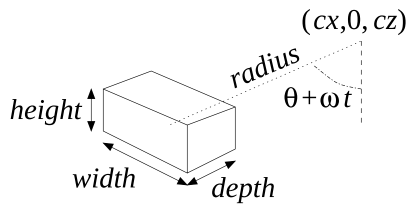


Figure B.1: The box-around-arc model.

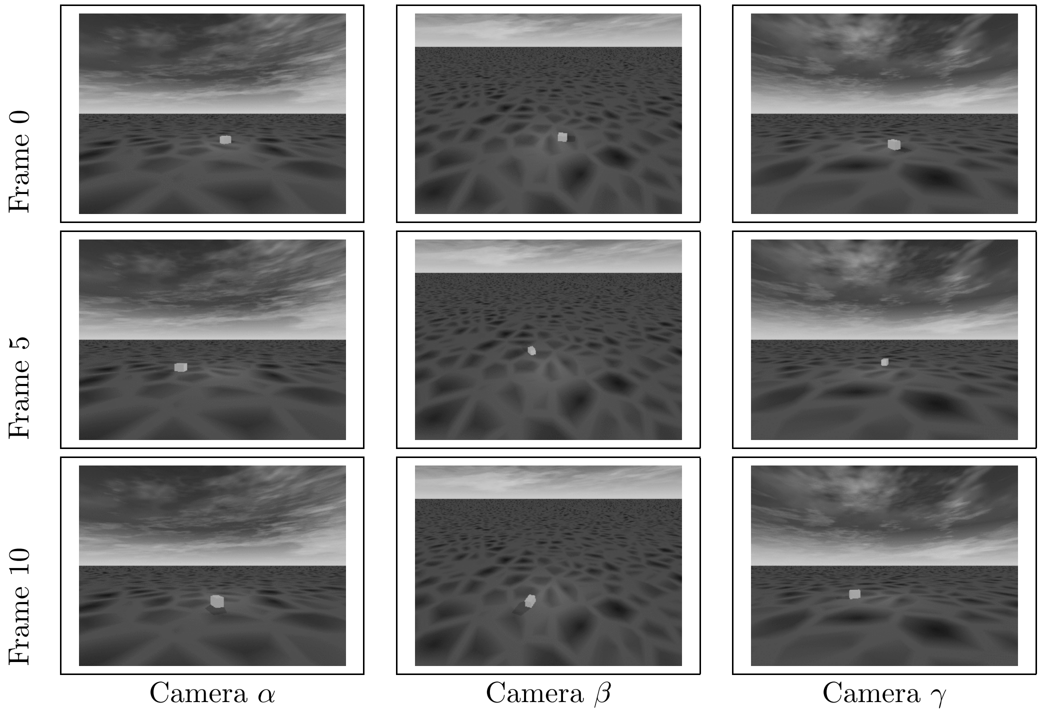


Figure B.2: Source moving ball ray-traced images.

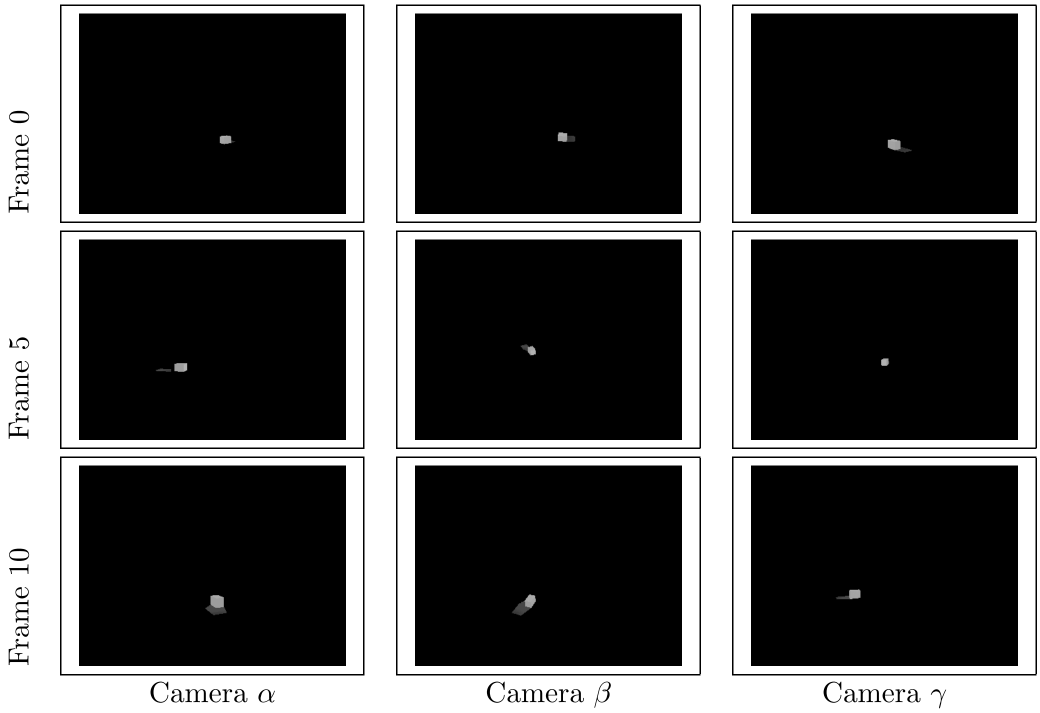


Figure B.3: 2D background removed images.

Parameter	Description
width, *height*, *depth*	The dimensions of the box
$(cx, 0, cz)$	The centre of rotation
radius	The radius of the arc
θ	The angle around the arc at time $t = 0$
ω	The angular velocity

Table B.1: The 8 parameters required for detecting a box moving around an arc.

B.2.2 Parameter extraction

The results shown in table B.2 are prepared in the same manner as those of the moving ball model above. There are no significant trends in the majority of the parameters, although the 2.75D and 3D algorithms commonly overestimate the size of the box; this effect was explained in section 5.3.2, the cause of it being the observable visual hull. The errors in the central arc position are slightly larger than the 25 mm resolution of the images at that depth, however, calculating it is extremely dependent on the angular velocity - for a slower speed, the central position is very much more affected by the noise of the reconstruction.

Therefore, as with the moving ball model, this more unusual motion was successfully captured. A possible use of this system could be the basis of monitoring cars on a road, after which a more complicated model could be applied that would be capable of identifying the type of vehicle.

				Error of the extracted parameters							
Analysis method	Camera set-up	Stat	Fitness	Dimensions w, h, d			Arc centre c_x, c_z		Radius	θ	ω
2D	# 1	μ	0.378	12.4	24.2	14.6	15.0	21.4	18.8	1.4	3.6
		σ	0.123	16.8	9.1	15.9	29.2	26.6	26.7	5.1	9.1
	# 2	μ	0.361	11.6	24.6	14.1	15.0	16.2	15.4	1.4	3.8
		σ	0.124	17.2	9.7	17.4	28.9	28.5	27.1	5.1	9.2
2.75D	# 1	μ	0.239	21.4	28.3	25.7	16.2	15.8	16.6	1.7	3.4
		σ	0.107	15.1	11.8	13.8	23.3	25.0	25.9	5.3	9.7
	# 2	μ	0.181	21.5	29.6	23.3	19.4	14.6	16.8	1.2	2.5
		σ	0.086	14.7	10.4	14.6	22.9	24.6	25.9	4.3	7.7
3D	# 1	μ	0.271	17.3	13.9	18.5	20.0	18.2	9.2	2.7	4.6
		σ	0.085	12.2	10.1	12.4	16.2	17.9	19.0	7.7	12.0
	# 2	μ	0.259	13.9	15.4	17.4	17.6	7.0	9.4	1.2	3.0
		σ	0.072	10.9	8.8	12.0	18.6	14.8	17.8	4.5	9.8

Table B.2: Extraction results of a moving sphere of unknown radius.

B.3 Gait intermediate processing examples

B.3.1 Introduction

The images presented below demonstrate the stages of reconstruction and background removal. They are comparable to the examples given in section 6.4, and as such the figures should be self-explanatory. Figure B.4 illustrates the original source data that is to be analysed.

Camera α Camera γ Camera β

Figure B.4: The human gait source data [sequence 11487-11546].

B.3.2 The reconstructed and filtered data

2D

Having removed the background, segmented images were produced. Those corresponding to the images in figure B.4 can be seen in figure B.5.

2.75D

The results of the reconstruction of just a single frame can be seen in figure B.6 showing the scene from the source camera angles. Having removed the background from the data, the data corresponding to the frames in figure B.4 can be seen in figure B.7, and those from new angles in figure B.8.

3D

The results of the reconstruction of just a single frame can be seen in figure B.9 showing the scene from the source camera angles. Having removed the background from the data, the data corresponding to the frames in figure B.4 can be seen in figure B.10, and those from new angles in figure B.11.

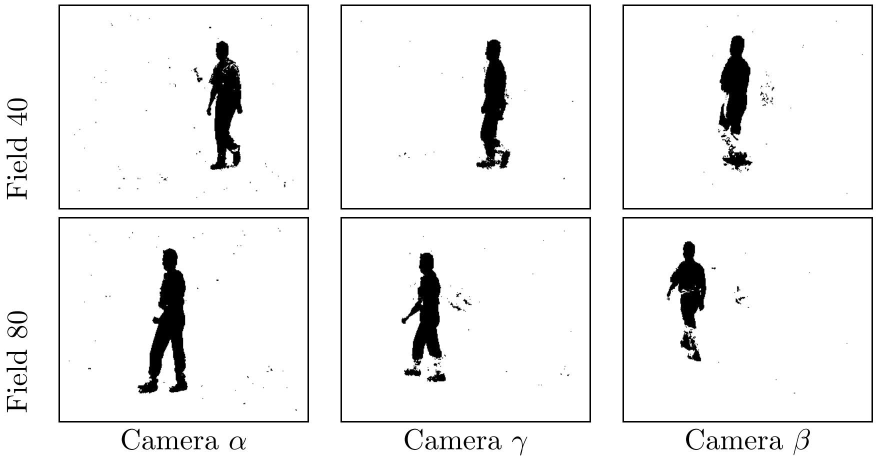


Figure B.5: The human gait 2D background removed images [sequence 11487-11546].

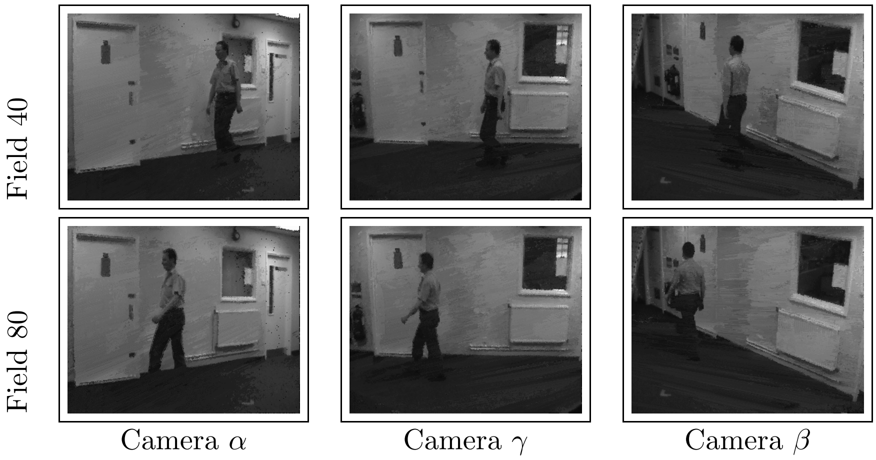


Figure B.6: The human gait 2.75D reconstructed scene from original camera angles.

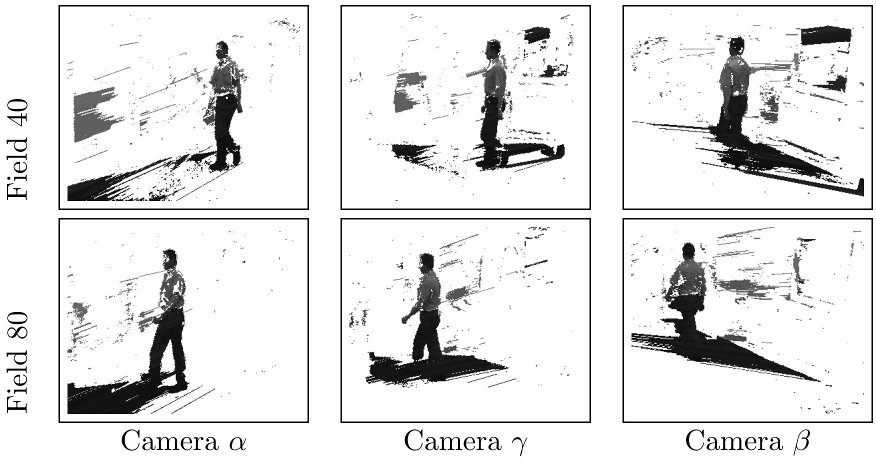


Figure B.7: The 2.75D filtered background removed scene from original camera angles.

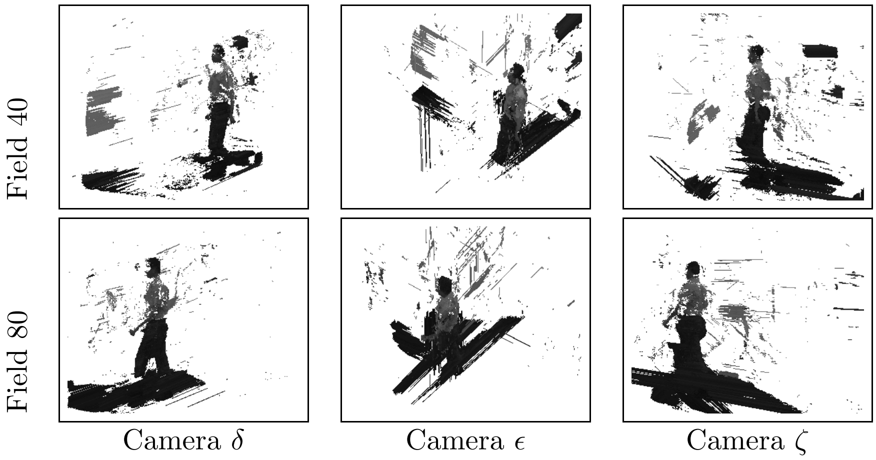


Figure B.8: The 2.75D filtered background removed scene from novel camera angles.

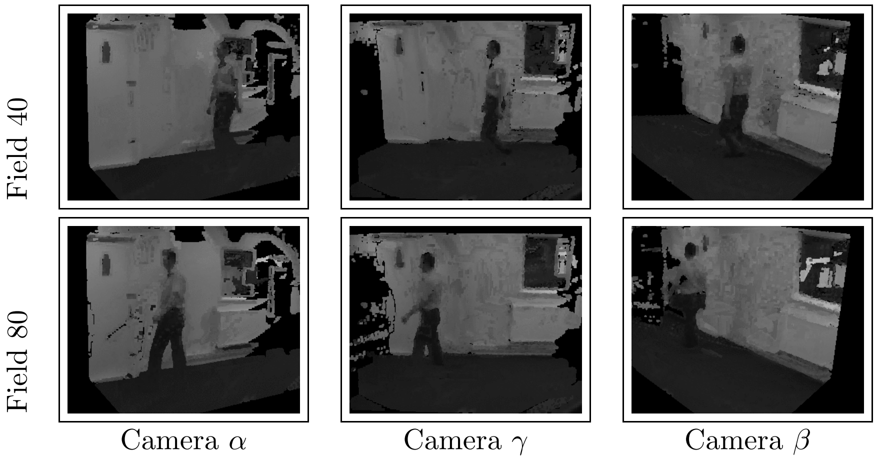


Figure B.9: The human gait 3D reconstructed scene from original camera angles.

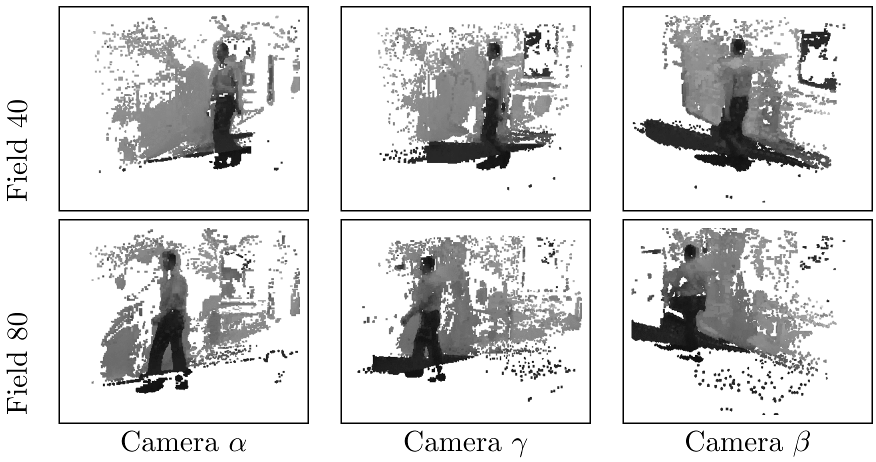


Figure B.10: The 3D filtered background removed scene from original camera angles.

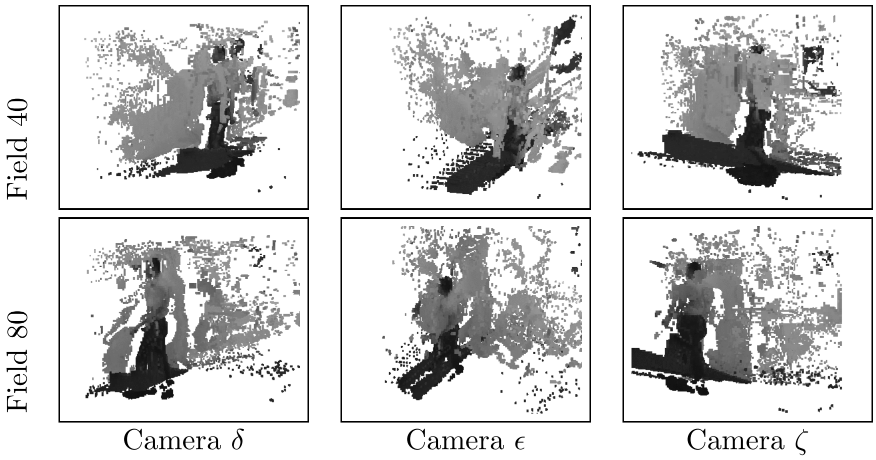


Figure B.11: The 3D filtered background removed scene from novel camera angles.

Appendix C

Pixel ray casting

C.1 Overview

When projecting a pixel through space, a locus in the form of a straight line is described. The line can be projected onto another view, and is also a straight line. The following sections analyse the rate at which the line must be projected away from the source camera's focal point, so that all pixels in the destination view's image line are stepped through once, and once only.

C.2 Theory

It is assumed, that the line can be mapped to the destination's intrinsic local geometry as defined in section 2.2.4, i.e., the destination camera lies at the origin of the coordinate system, and the image plane lies on the $z = 1$ plane, and pixels are square and centralised around the principal point.

Thus there is a line visible which follows the standard equation:

$$\begin{bmatrix} x_{3d} \\ y_{3d} \\ z_{3d} \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} + \lambda \begin{bmatrix} d \\ e \\ f \end{bmatrix} \tag{C.1}$$

i.e.:

$$\begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} = \underline{\underline{\mathbf{P}}}_1 \underline{\underline{\mathbf{P}}}_0^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{C.2}$$

where $\underline{\underline{\mathbf{P}}}_0$, $\underline{\underline{\mathbf{P}}}_0$ are the projection matrices for the source view and destination view respectively, as defined in equation 2.10, but having been turned in a 4×4 square matrix to enable the matrix manipulation, and:

$$\begin{bmatrix} (a+d) \\ (b+e) \\ (c+f) \\ 1 \end{bmatrix} = \underline{\underline{\mathbf{P}}}_1 \underline{\underline{\mathbf{P}}}_0^{-1} \begin{bmatrix} \mathbf{p} \\ 1 \\ 1 \end{bmatrix} \tag{C.3}$$

where $\mathbf{p}$ is the source pixel coordinate vector.

A 3D point on the line is mapped onto the destination image by the following equation:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{z_{3d}} \begin{bmatrix} x_{3d} \\ y_{3d} \end{bmatrix} \tag{C.4}$$

Noting that there is no f_x or f_y component as this is accounted for in the intrinsic geometry representation.

So, looking at just the equation for x:

$$x = \frac{a + \lambda d}{c + \lambda f} \tag{C.5}$$

Re-arranging, a function for λ is obtained:

$$\begin{aligned} xc + x\lambda f &= a + \lambda d \\ \lambda &= \frac{a - xc}{xf - d} \end{aligned} \tag{C.6}$$

It is also noted that

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}\lambda} &= \frac{\mathrm{d}}{\mathrm{d}\lambda} \frac{a + \lambda d}{c + \lambda f} \\ &= \frac{(c + \lambda f)d - (a + \lambda d)f}{(c + \lambda f)^2} \\ &= \frac{cd - af}{(c + \lambda f)^2} \end{aligned} \tag{C.7}$$

Hence $\frac{\mathrm{d}x}{\mathrm{d}\lambda}$ is positive for $cd - af > 0$, negative for $cd - af < 0$ and 0 for $cd - af = 0$.

As λ increases, the next λ (λ') must be calculated so that the line enters the next pixel. Since the next pixel depends on the sign of $\frac{\mathrm{d}x}{\mathrm{d}\lambda}$, there are three instances:

Instance 1:	$cd - af > 0$	The next pixel is at $x+1$
Instance 2:	$cd - af = 0$	The next pixel is at x
Instance 3:	$cd - af < 0$	The next pixel is at $x-1$

Instance 2 indicates a case where the ray is mapped to the same pixel in the destination image, and thus there is no dependency on the depth of the projection. In such a case, there is obviously no requirement to search for the next pixel that the line will cross.

C.2.1 The positive direction

First looking at the case where the next value of λ yields a point at $x+1$:

$$\lambda' = \frac{a-(x+1)c}{(x+1)f-d} = \frac{a-xc-c}{xf+f-d} \tag{C.8}$$

λ has thus changed by:

$$\delta\lambda = \lambda' - \lambda = \frac{a-xc-c}{xf+f-d} - \frac{a-xc}{xf-d} \tag{C.9}$$

Cross multiplying, and reducing the equation:

$$\begin{aligned}
\delta\lambda &= \frac{(a-xc-c)(xf-d)-(a-xc)(xf+f-d)}{(xf+f-d)(xf-d)} \\
&= \frac{-c(xf-d)-f(a-xc)}{(\frac{1}{z}x_{3d}f+f-d)(\frac{1}{z}x_{3d}f-d)} \\
&= \frac{(cd-af)z^2}{(x_{3d}f+fz-dz)(x_{3d}f-dz)} \\
&= \frac{(cd-af)(c+\lambda f)^2}{((a+\lambda d)f+f(c+\lambda f)-d(c+\lambda f))((a+\lambda d)f-d(c+\lambda f))} \\
&= \frac{(cd-af)(c+\lambda f)^2}{(af+\lambda df+f(c+\lambda f)-dc-d\lambda f)(af+\lambda df-dc-d\lambda f)} \\
&= \frac{(cd-af)(c+\lambda f)^2}{(af+f(c+\lambda f)-dc)(af-dc)} \\
&= \frac{(cd-af)(c+\lambda f)^2}{((af-cd)+f(c+\lambda f))(af-cd)}
\end{aligned} \tag{C.10}$$

Noting that the exception to the equation at $cd = af$ is invalid under the application of this equation,

$$\delta\lambda = -\frac{(c+\lambda f)^2}{(af-cd)+f(c+\lambda f)}$$

$$= \frac{(c+\lambda f)^2}{(cd-af)-f(c+\lambda f)} \tag{C.11}$$

Thus the step, $\delta\lambda$ is dependent on the current λ and on the four variables.

C.2.2 The negative direction

If, as λ increases, the image is swept in a negative-x manner, then it is the change in λ that would be needed to visit the point at $x-1$ that is required. So adapting the equations above:

$$\lambda' = \frac{a-(x-1)c}{(x-1)f-d} = \frac{a-xc+c}{xf-f-d} \tag{C.12}$$

$$\delta\lambda = \lambda' - \lambda = \frac{a-xc+c}{xf-f-d} - \frac{a-xc}{xf-d} \tag{C.13}$$

Cross multiplying, and reducing the equation we get:

$$\begin{aligned}
\delta\lambda &= \frac{(a-xc+c)(xf-d)-(a-xc)(xf-f-d)}{(xf-f-d)(xf-d)} \\
&= \frac{c(xf-d)+f(a-xc)}{(\frac{1}{z}x_{3d}f-f-d)(\frac{1}{z}x_{3d}f-d)} \\
&= \frac{(af-cd)z^2}{(x_{3d}f-fz-dz)(x_{3d}f-dz)} \\
&= \frac{(af-cd)(c+\lambda f)^2}{((a+\lambda d)f-f(c+\lambda f)-d(c+\lambda f))((a+\lambda d)f-d(c+\lambda f))} \\
&= \frac{(af-cd)(c+\lambda f)^2}{(af+\lambda df-f(c+\lambda f)-dc-d\lambda f)(af+\lambda df-dc-d\lambda f)} \\
&= \frac{(af-cd)(c+\lambda f)^2}{(af-f(c+\lambda f)-dc)(af-dc)} \\
&= \frac{(af-cd)(c+\lambda f)^2}{((af-cd)-f(c+\lambda f))(af-cd)} \\
&= \frac{(c+\lambda f)^2}{(af-cd)-f(c+\lambda f)}
\end{aligned} \tag{C.14}$$

A similar result to before, but with a change in sign of a component of the denominator.

C.2.3 Combining the two equations

The two equations differ only by a change of sign in a component of the denominator, but the different equations will only be used depending on $cd-af$ as this effects $\frac{dx}{d\lambda}$. Therefore, combining these results,

$$\delta\lambda = \begin{vmatrix} \frac{(c+\lambda f)^2}{|(af-cd)|-f(c+\lambda f)} & \text{for} & \begin{matrix} cd \neq af \\ cd = af \end{matrix} \\ \infty & & \end{vmatrix} \tag{C.15}$$

C.2.4 Confirming the result

To illustrate the result, consider a projected ray that moves horizontally across the destination view at a constant orthogonal distance. In this case, $f = 0$, and thus $\delta\lambda = \frac{c^2}{cd} = \frac{c}{d}$. This appears correct as it is dependent on the distance, c, of the line in 3D, and is inversely-proportional to the rate at which the points move in the positive direction along the x-axis.

For a ray that passes through the origin of the destination camera, i.e., one that moves directly along a projection line:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} + \lambda \begin{bmatrix} d \\ e \\ f \end{bmatrix} \tag{C.16}$$

for some λ. Therefore $a+\lambda d = 0$ and $c+\lambda f = 0$. Substituting λ, this yields $fa = cd$ which has already shown to produce an exception in the equation, indicating that it is not possible to traverse, no matter what change in λ, to the next pixel.

C.2.5 The y-axis and multiple views.

Similar equations can be formed for the y-axis. Then, to find the step for a given λ, the two equations are evaluated, and the minimum step is selected. It must be noted that the range of λ over a particular destination view is restricted so that it evaluates only pixels in the view and in front of the camera.

When there are several views, pixels may be visited more than once, although the combination of pixels from the different views will only be tested once. By selecting the minimum step from the various (relevant) images, this can be achieved.

Appendix D

Camera synchronisation device

D.1 Introduction

Figures D.1 & D.2 describe the circuit to produce a synchronising signal for the analogue cameras. There is actually a minor discrepancy between the cameras since one camera's signal is used to calculate the synchronisation signal for the others, however, this small temporal difference is insignificant, being approximately 0.2 ms.

If a driving camera is not present, then the respective LED will indicate an error (RED), and all other LEDs will turn off. If a driving camera is producing a suitable signal, this LED will turn GREEN, and the other two LEDs will turn on. These other LEDs will be GREEN if the difference between their field signal and that of the driving camera is relatively insignificant, and RED otherwise.

The electronics is concise, for example using a hybrid of analogue comparative filters with digital electronics to drive the LEDs; the emphasis was to quickly produce a workable solution, and thus, although this is not the recommended method, it has been found to be applicable in this situation.

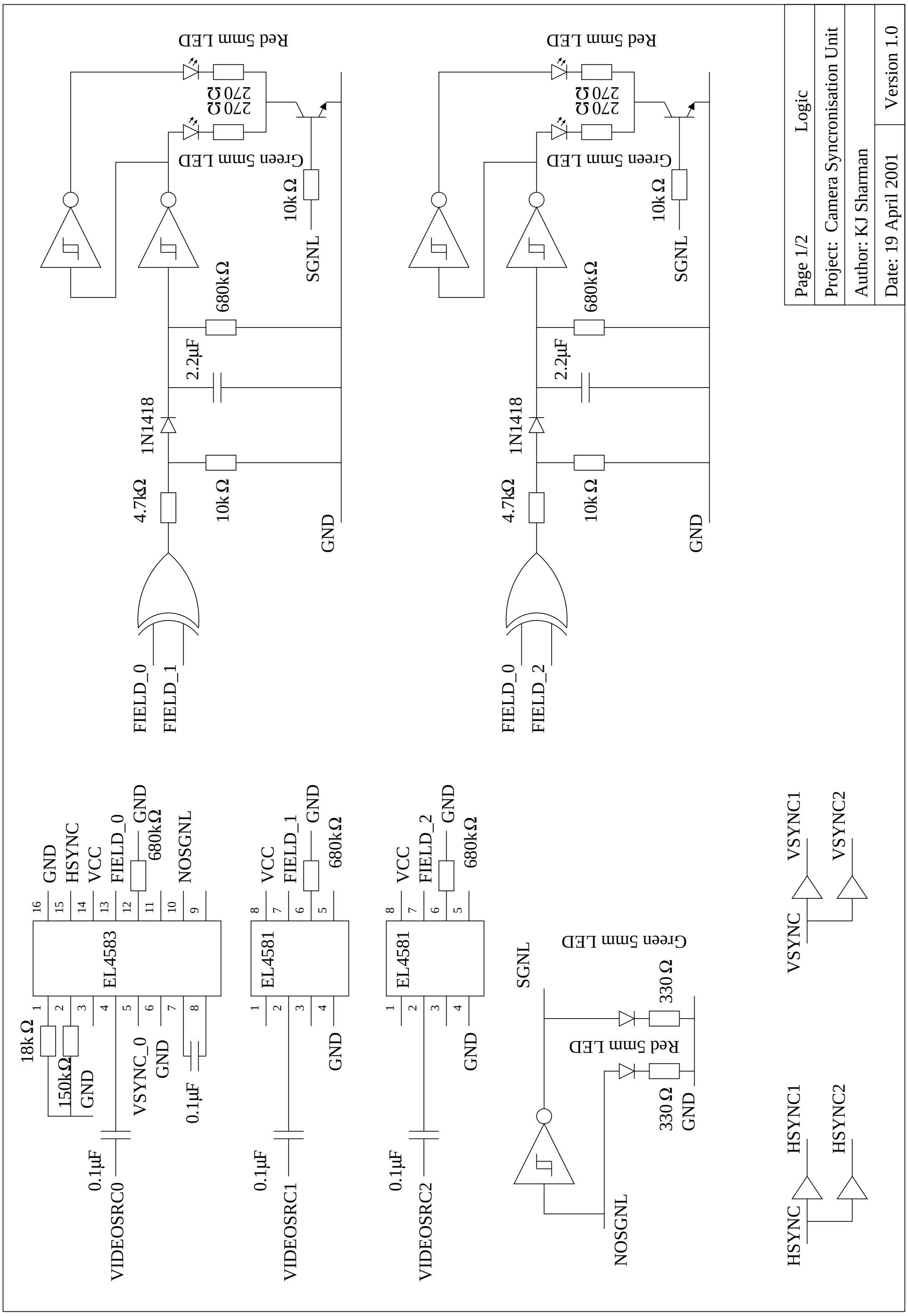


Figure D.1: Camera synchronisation device part I.

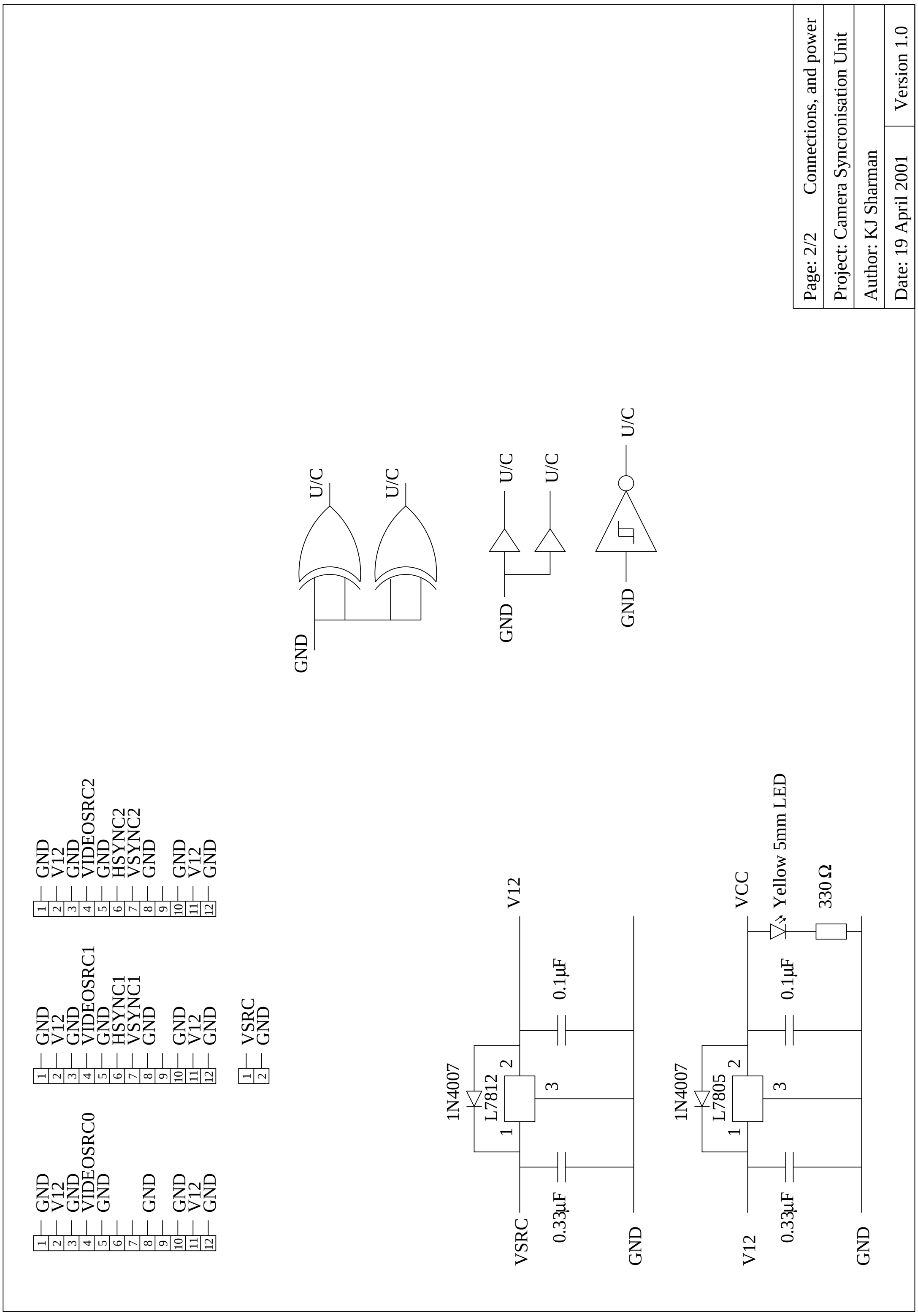


Figure D.2: Camera synchronisation device part II.

Appendix E

DV Codecs

E.1 Introduction

During the course of this research, it was found necessary to fully understand the DV encoding. The initial reason for this was so that tools to animate, split and convert the data to images could be produced. However, it was soon discovered that not just the codec used, but many others, misinterpreted the video format, producing extremely erroneous output.

This section does not intend to be a specification of the standard, but rather an overview of how the data is encoded in the standard and how this common mistake in codecs has been formed in order that others will not be plagued by it. The full standard is available from The Society of Motion Picture and Television Engineers (SMPTE 314M-1999).

E.2 Overview of the standard

DV encodes each frame separately and in a fixed number of bytes, therefore not utilising any temporal correlation between successive images. For the common 25 Mb/s standard, each frame, which includes audio information, fits into a 144,000 byte block for PAL or 120,000 bytes for NTSC. There is a different structure between these two main video standards, but the encoding is in essence the same, both requiring a compression method to reduce the video data to a much smaller amount.

Each frame is split into, for the NTSC format (known as 4:1:1), 32×8 pixel blocks which are represented by four blocks of 8×8 luminance information, but only two blocks of 8×8 chrominance information that will represent the red and the blue channels. Figure E.1 shows how the image data is sampled in such a representation, indicating that the chrominance fields are poorly represented from the outset.

The Discrete Cosine Transform is then applied to each of these six blocks; the DC term is encoded separately and without loss. However, in order to compress the

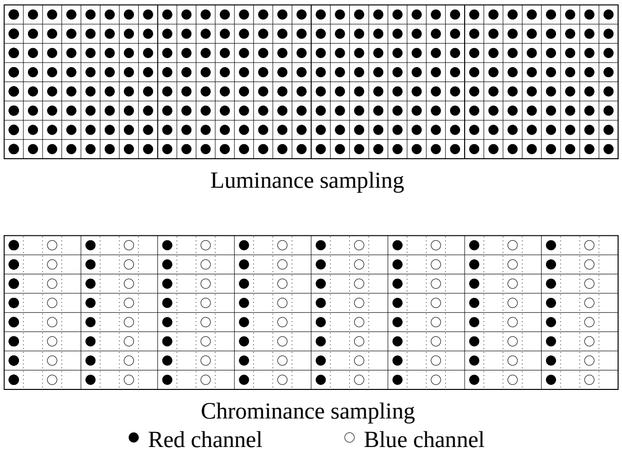


Figure E.1: DV sampling with the 4:1:1 representation. The luminance is sampled at every point in the original image, but the red and blue each sample one pixel in four. The solid lines indicate the block over which the chrominance is used for reconstructing the image.

data, it is essential that many of the other frequency components are reduced to zero. This is performed by scaling them by various amounts that result in favouring the luminance low frequency components and poorly representing the chrominance high frequency components, especially that of the blue channel. Having forced many of the components to zero, a specific Huffman encoding is applied to reduce the size of the represented data.

E.3 The 4:2:0 representation

The PAL standard is known as the 4:2:0 representation, where each frame is now split into 16×16 pixel blocks, again represented by four blocks of 8×8 luminance blocks and two blocks of chrominance, however, the sampling of the original image is performed in a different way. First, as seen in figure E.2a, the red chrominance channel is sampled on every other line, and the blue chrominance channel is sampled on the others. However, it is important to note that the standard is designed around the interlaced television standard, therefore this representation would seem to poorly sample the video as one field would be sampled for blue and the other for red—any bright moving object would be distorted. The standard, however, indicates that this is the method of sampling in a single field, not the frame, and in fact the true interlaced sampling structure is as appears in figure E.2b.

Codecs commonly do not interpret this interlaced structure correctly, resulting in images where, although the luminance is correct the colours appears to be interlaced in pairs of lines, not singly. There is no such problem with the 4:1:1 NTSC standard since the two different chrominance values are sampled on all field lines. In this research, the codec was corrected, however, although it will introduce slight errors, it is relatively simple to swap the chrominance after the RGB images have been produced.

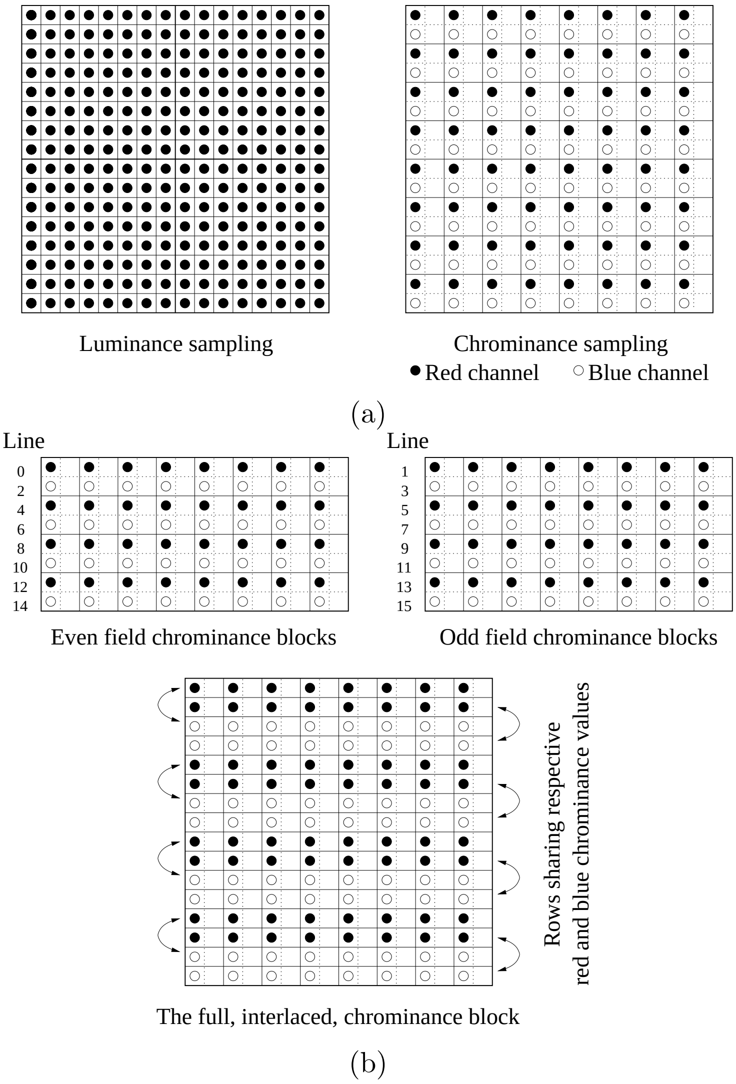


Figure E.2: DV sampling with the 4:2:0 representation. (a) shows the commonly misinterpreted format where the red and blue chrominances sample every other row. (b) shows that in the same field, the red and blue sample every other row, but not over the block of 16 × 16 pixels; To form the top left pixel, the respective luminance and red value is used along with the blue value from the next row in the field, i.e., the blue value from row 2 of the block.