

A GPU Parallel Approach Improving the Density of Patch Based Multi-View Stereo Reconstruction

Benjamin A. Haines

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy
July, 2016

To my beloved family and most treasured friends

Abstract

Multi-view stereo is the process of recreating three-dimensional data from a set of two or more images of a scene. The ability to acquire 3D data from 2D images is a core concept in computer vision with wide-ranging applications throughout areas such as 3D printing, robotics, recognition, navigation and a vast number of other fields. While 3D reconstruction has been increasingly well studied over the past decades, it is only with the recent evolution of CPU and GPU technologies that practical implementations, able to accurately, robustly and efficiently capture 3D data of photographed objects have begun to emerge. Whilst current research has been shown to perform well under specific circumstances and for a subset of objects, there are still many practical and implementary issues that remain an open problem for these techniques. Most notably, the ability to robustly reconstruct objects from sparse image sets or objects with low texture.

Alongside a review of algorithms within the multi-view field, the work proposed in this thesis outlines a massively parallel patch based multi-view stereo pipeline for static scene recovery. By utilising advances in GPU technology, a particle swarm algorithm implemented on the GPU forms the basis for improving the density of patch-based methods. The novelty of such an approach removes the reliance on feature matching and gradient descent to better account for the optimisation of patches within textureless regions, for which current methods struggle. An enhancement to the photo-consistency matching metric, which is used to evaluate the optimisation of each patch, is then defined. Specifically targeting the shortcomings of the photo-consistency metric when used inside a particle swarm optimisation, increasing its effectiveness over textureless areas. Finally, a multi-resolution reconstruction system based on a wavelet framework is presented to further improve upon the robustness of reconstruction over low textured regions.

Publications

Conference Proceedings

B. Haines and L. Bai. Towards a Robust Patch Based Multi-View Stereo Technique for Textureless and Occluded 3D Reconstruction. *Computational Vision and Medical Image Processing V: Proceedings of the 5th Eccomas Thematic Conference on Computational Vision and Medical Image Processing VipIMAGE, 2015*

J. Duan, B. Haines, W. O. C. Ward, and L. Bai. Surface Reconstruction From Point Clouds Using a Novel Variational Model, *Thirty-fifth SGAI International Conference on Artificial Intelligence, 2015*, Accepted for publication AI-2015.

Acknowledgements

First and foremost I would like to extend my heartfelt gratitude to all of my friends and family, who have supported me throughout a tough but rewarding experience. Their moral, emotional and financial support has been invaluable to me over the years and their loving guidance has shaped the person I am today.

I would like to gratefully thank my supervisor Bai Li, who has inspired me to undertake such a demanding yet rewarding project. Her encouragement and support has helped me focus my ideas and realise their potential, allowing me to find an area of research that has become more of a hobby than work.

Many thanks go to Buzz 3D who have provided my research with a distinct purpose and industrial guidance. Their keen engagement and support has kept me motivated and driven, even when a solution seemed unattainable.

Finally I would like to express my thanks to the University of Nottingham and all of my colleagues here, whom I have studied and socialised happily with for the past 7 years.

Contents

1 Introduction **1**
- 1.1 3D Reconstruction and Multi-View Stereo 2
- 1.2 Thesis Problem Statement 5
- 1.3 Aims and Contributions 7
- 1.4 Thesis Outline 9

2 Literature Review **10**
- 2.1 Organisation 11
- 2.2 Industrial Applications 11
- 2.3 No Surface Optimisation 12
  - 2.3.1 Visual Hull Models 13
  - 2.3.2 Photo Hull Models 18
  - 2.3.3 Summary of Methods Not Involving Optimisation . 22
- 2.4 Local Surface Optimisation 23
  - 2.4.1 Level Sets 23
  - 2.4.2 Active Contours 28
  - 2.4.3 Depth Map Fusion 30
  - 2.4.4 Patch Based 32
  - 2.4.5 Summary of Local Optimisation Methods 37
- 2.5 Global Surface Optimisation 38
  - 2.5.1 Graph Cuts 38
  - 2.5.2 Summary of Global Optimisation Methods 41
- 2.6 Discussion 42
- 2.7 Summary 47

3 Techniques for 3D Reconstruction **48**
- 3.1 The Camera Matrix and Calibration 49
- 3.2 Photo-Consistency and Similarity Metrics 53
  - 3.2.1 Normalised Cross Correlation 55
- 3.3 General Purpose Computing on Graphics Processing Units 57
  - 3.3.1 Execution Model 59
  - 3.3.2 Coalesced Memory Access 62
  - 3.3.3 Memory Model 64
- 3.4 Particle Swarm Optimisation 66

3.4.1 Standard Particle Swarm Optimisation 67
3.4.2 Hybrid Mutation 72
3.4.3 Global Local Nearest PSO 73
3.4.4 Synchronous, Asynchronous and Parallel PSO . . 74
3.5 Summary 77

4 Particle Swarm Optimisation for High Fidelity Reconstruction 78
4.1 PSO Patch Based Framework 80
4.1.1 Overview 81
4.1.2 Generating the Visual Hull 85
4.1.3 Patch Refinement by PSO 95
4.2 Datasets 102
4.3 Experimental Results and Analysis 106
4.3.1 Qualitative 108
4.3.2 Quantitative 120
4.3.3 Speed 129
4.4 Discussion 131
4.4.1 Technique Synopsis 131
4.4.2 Results and Limitations 132
4.4.3 Improvements 133
4.5 Summary 134

5 Enhancing the Similarity Metric: Photo-Detail-Consistency 135
5.1 The Photo-Detail-Consistency Metric 137
5.1.1 Defining Photo-Detail-Consistency 137
5.1.2 Non-Lambertian Reflection 142
5.2 Experimental Results and Analysis 144
5.2.1 Qualitative 145
5.2.2 Quantitative 158
5.2.3 Speed 170
5.3 Discussion 172
5.3.1 Technique Synopsis 172
5.3.2 Results and Limitations 172
5.3.3 Improvements 173
5.4 Summary 174

6 A Multi-Resolution Approach to 3D Reconstruction 175
6.1 Multi-Resolution Patch Refinement 177
6.1.1 PMVS Expansion and Filtering Stages 180
6.1.2 Multi-Resolution Study 183
6.1.3 Multi-Resolution Framework 192
6.2 Experimental Results and Analysis 197
6.2.1 Qualitative 198
6.2.2 Quantitative 212
6.2.3 Speed 226
6.3 Discussion 227
6.3.1 Technique Synopsis 227
6.3.2 Results and Limitations 228
6.3.3 Improvements 229

6.4 Summary 230

7 Conclusions and Future Work **231**
7.1 Problems in Multi-View Stereo 232
7.2 Contributions 234
7.3 Limitations 236
7.4 Future Work 238
7.4.1 GPU Implementation and Efficiency 238
7.4.2 Improving the Similarity Metric 239
7.4.3 Theoretical Concept 240

Appendices

A Datasets **243**

References **277**

List of Figures

2.1 Voxel carving 16
2.2 Implicit surface 24
2.3 PMVS feature matching and triangulation 33
2.4 PMVS Expansion and Refinement 34

3.1 Photo-consistency measures 54
3.2 Relations between work-items and work-groups 60
3.3 C code for serial implementation of squaring a vector . . . 61
3.4 Kernel code for parallel implementation of squaring a vector 61
3.5 Host code to invoke a GPU kernel 62
3.6 Memory access patterns 63
3.7 OpenCLs memory hierarchy 64
3.8 Optimised kernel code for squaring a vector 65
3.9 Traditional Particle Swarm Optimisation 68
3.10 Standard Particle Swarm Optimisation 72
3.11 Host code for synchronous Particle Swarm Optimisation . 75
3.12 Host code for asynchronous Particle Swarm Optimisation 76

4.1 Surface patches 81
4.2 Effect of sample size, μ, on patch position and orientation 83
4.3 Patch refinement 85
4.4 Example of assigning the half distance transform distance labels 86
4.5 Silhouette half distance transform 87
4.6 Horizontal Distance Transform pseudo-code 88
4.7 Vertical Distance Transform pseudo-code 89
4.8 Voxel classification kernel pseudo-code 91
4.9 Visual hull construction 93
4.10 Visual hull projection 94
4.11 PSO search space 96
4.12 The flow of GPU based PSO 98
4.13 The flow of our GPU based PSO initialisation kernel . . . 99
4.14 The flow of our GPU based PSO evaluation kernel 100
4.15 The flow of our GPU based PSO update kernel 101

4.16 Ground truth meshes 102
4.17 Synthetic datasets with various texture 104
4.18 Real life datasets 105
4.19 Monkey reconstructions via PMVS and PSO 112
4.20 Buddha reconstructions via PMVS and PSO 113
4.21 LynxCAD reconstructions via PMVS and PSO 114
4.22 Dino reconstructions via PMVS and PSO 115
4.23 Temple reconstructions via PMVS and PSO 116
4.24 LynxReal reconstructions via PMVS and PSO 117
4.25 Filtered ground truth reconstructions for PSO 118
4.26 Noise in real life reconstructions with PSO 119
4.27 Erroneous LynxReal reconstruction 119
4.28 PSO probability distributions for the Monkey datasets . . 123
4.29 PSO probability distributions for the Buddha datasets . . 125
4.30 PSO probability distributions for the LynxCAD dataset . . 127

5.1 Colour channels and the textureless effect 138
5.2 Non-Lambertian reflection affecting the NCC 142
5.3 Monkey reconstructions via PSO and PSOPD 148
5.4 Buddha reconstructions via PSO and PSOPD 149
5.5 LynxCAD reconstructions via PSO and PSOPD 150
5.6 Dino reconstructions via PSO and PSOPD 151
5.7 Temple reconstructions via PSO and PSOPD 152
5.8 LynxReal reconstructions via PSO and PSOPD 153
5.9 Filtered Monkey reconstructions for PSOPD 154
5.10 Filtered Buddha reconstructions for PSOPD 155
5.11 Filtered LynxCAD reconstructions for PSOPD 156
5.12 Noise in PSO vs PSOPD for the Dino dataset 156
5.13 Noise in PSO vs PSOPD for the Temple dataset 157
5.14 Noise in PSO vs PSOPD for the LynxReal dataset 157
5.15 PSOPD and PSO probability distributions for the Monkey datasets 161
5.16 PSOPD and PMVS probability distributions for the Monkey datasets 162
5.17 PSOPD and PSO probability distributions for the Buddha datasets 164
5.18 PSOPD and PMVS probability distributions for the Buddha datasets 165
5.19 PSOPD and PSO probability distributions for the LynxCAD dataset 167
5.20 PSOPD and PMVS probability distributions for the LynxCAD dataset 168
5.21 The flow of our GPU based PSO texture weight computation kernel 171

6.1 PMVS Patch Filtering 182
6.2 The effects of patch size on surface fidelity 184
6.3 MonkeyLow multi-resolution patch clouds 189
6.4 Errors over multiple image resolutions 191

6.5 Monkey reconstructions via PMVS and MRPSO 202
6.6 Buddha reconstructions via PMVS and MRPSO 203
6.7 LynxCAD reconstructions via PMVS and MRPSO 204
6.8 Dino reconstructions via PMVS and MRPSO 205
6.9 Temple reconstructions via PMVS and MRPSO 206
6.10 LynxReal reconstructions via PMVS and MRPSO 207
6.11 Filtered Monkey reconstructions for MRPSO 208
6.12 Filtered Buddha reconstructions for MRPSO 209
6.13 Filtered LynxCAD reconstructions for MRPSO 210
6.14 Noise in PSOPD vs MRPSO for the Dino dataset 210
6.15 Noise in PSOPD vs MRPSO for the Temple dataset . . . 211
6.16 Noise in PSOPD vs MRPSO for the LynxReal dataset . . 211
6.17 MRPSO and PSOPD probability distributions for the Monkey datasets 217
6.18 MRPSO and PMVS probability distributions for the Monkey datasets 218
6.19 MRPSO and PSOPD probability distributions for the Buddha datasets 220
6.20 MRPSO and PMVS probability distributions for the Buddha datasets 221
6.21 MRPSO and PSOPD probability distributions for the LynxCAD dataset 223
6.22 MRPSO and PMVS probability distributions for the LynxCAD dataset 224

A.1 Camera set up for Monkey datasets 244
A.2 Highly textured Monkey dataset 245
A.3 Medium textured Monkey dataset 246
A.4 Low textured Monkey dataset 247
A.5 Camera set up for Buddha datasets 248
A.6 Highly textured Buddha dataset 249
A.7 Medium textured Buddha dataset 250
A.8 Low textured Buddha dataset 251
A.9 Camera set up for LynxCAD dataset 252
A.10 The textured Lynx CAD Dataset 253
A.11 Camera set up for Dino dataset 254
A.12 The real life Middlebury Dino dataset 255
A.13 Camera set up for Temple dataset 256
A.14 The real life Middlebury Temple dataset 257
A.15 Camera set up for LynxReal dataset 258
A.16 The real life Lynx Can dataset 259

List of Tables

2.1 Reconstruction algorithm comparison 43

4.1 PSO statistics for the Monkey datasets 124
4.2 PSO statistics for the Buddha datasets 126
4.3 PSO statistics for the LynxCAD dataset 128
4.4 Refinement times for PMVS and PSO 129

5.1 PSOPD statistics for the Monkey datasets 163
5.2 PSOPD statistics for the Buddha datasets 166
5.3 PSOPD statistics for the LynxCAD dataset 169
5.4 Refinement times for PMVS, PSO and PSOPD 170

6.1 MRPSO statistics for the Monkey datasets 219
6.2 MRPSO statistics for the Buddha datasets 222
6.3 MRPSO statistics for the LynxCAD dataset 225
6.4 Refinement times for PMVS, PSO, PSOPD and MRPSO . 226

1

Introduction

Computer vision is a wide-ranging research discipline that attempts to model and simulate the abilities of the human vision system. It is the process of taking two-dimensional representations of the world, normally in the form of images, and applying computational analysis on these, in order to obtain numerical information or reasoned understanding of the data present. Nowadays, image data can take many different forms, from static 2D images, taken from either single or multiple viewpoints, to high definition video sequences. Computer vision even extends into medical imaging with the acquisition of multi-dimensional scan data.

3D reconstruction is a topic within this discipline concerned with inferring or recovering the three dimensional data which is lost in the imaging process. This can be thought of as the inverse problem to image formulation, where the physical properties of the three dimensional world are used to enable the production of realistic two-dimensional images. As with so much of computer science, the easier a problem is to define the harder it is to compute, 3D reconstruction being no exception to this rule.

1.1 3D Reconstruction and Multi-View Stereo

Recovering the entire geometry and topology of a 3D structure from a set of images is a more complex task than one would expect and finds its place among the fundamentals of computer vision. Historically, in the 1970's, Baumgart's work [11], first introduced the concept of finding the largest silhouette-consistent volume as a representation of a 3D object. This was later to become termed the *visual hull*, by Laurentini in his work in the 1990's, [95]. Since then the area of 3D reconstruction has been a dynamic and fast-paced field. This is mainly due to the vast number of applications of this knowledge throughout many aspects of every day life, from robotics, vehicle navigation, gaming systems, augmented reality, online shopping, surveillance, tracking, not to mention the emerging technologies of today such as 3D printing, medical imaging and many more.

Along side this vast array of applications comes an equally if not more extensive set of solutions for obtaining this type of 3D information. These processes can easily be classified according to the input data into the reconstruction pipeline, however, their efficiency is less well defined, being controlled by factors such as generality of the scene to recover, fidelity of reconstruction, run time and level of automation.

There are a number of ways in which to obtain appropriate data for input into a reconstruction algorithm, each of which being classed as either active or passive. The main set being:

- Range-based laser scanning,
- Video capture,
- Multi-view stereo

Laser scanners, also known as range-based scanners, tend to be used to construct high-resolution accurate depth maps and hence ground truth datasets of an object under reconstruction. They form an active method in that depth values are returned at the point of capture. However, this hardware is expensive and requires a specialist to operate. As well as this, they are particularly sensitive to capturing objects that have a high specular factor, due to the refraction of laser light. With the rapid development of technology, Microsoft recently released a more affordable

range scanner called the Kinect, [111]. This offers a consumer level scanning technology, however, is restricted in its accuracy and portability. Alternatively, a Structured Light approach, [135], can be used similarly to laser scanning, where known light patterns are projected into the scene. The distortion induced when this hits an objects surface can be used to recover its 3D shape. However, this approach distorts the surface texture and again does not scale well to highly specular objects. These factors limit the application of range scanning, making them less than ideal for a reconstruction system intended for general use.

A passive approach to reconstruction requires that the data first be collected, then algorithms that recover shape are run as a post-process. These days the majority of people have access to an acceptable quality video capturing system, whether it be via a mobile phone, digital camera or hand held video recorder. However, video resolution standards vary significantly over the range of devices available, as well as this video streams are highly susceptible to motion blur. These factors considered can cause many difficulties for a reconstruction algorithm. Taking this into account, video data tends to overcomplicate the task of reconstruction, compromising the robustness and accuracy of the solution, not to mention the fact that it relies heavily on the quality of user supplied data, [127].

Multi-view stereo (MVS) refers to the process of taking a collection of images from known camera viewpoints and using these to define the shape of the object within the frame, [138]. Such techniques make reconstruction accessible to anyone with a standard to high-end camera, allowing the end user flexibility and simplicity in the formation of 3D data. Another benefit of this is that MVS approaches can be applied to historic datasets, where no video capture or range scanning can be obtained. However, MVS techniques are limited in a number of ways, firstly they require a calibrated set of images, this calibration process can be difficult and erroneous, [36]. Secondly, their ability to accurately reconstruct an entire object diminishes when the object is of low texture or captured with a sparse number of input views, [138]. A sub-category of MVS problems is the binocular stereo case, where only two images, taken side by side, are used to obtain a depth map of the scene, [28]. Such approaches are limited to reconstructing only the partial section of the scene under view, rather than the whole object. Given the low cost and ease in capture of image datasets, this thesis focuses its efforts on purely the MVS case.

Typically MVS reconstruction algorithms would fall into two categories, those based on information from silhouettes known as *shape from silhouettes*, and those based on matching pixels throughout the image set, known as *shape from stereo*.

Shape from silhouette approaches aim to reconstruct the outer most approximation for the surface, the visual hull. These algorithms achieve their goal by calculating the intersection of viewing cones induced by the edges of silhouettes in 3D space. A drawback to such techniques is seen in the inability of the silhouettes to express concavities of an object; hence, these methods lead to reconstructions that are missing concave parts of the object. The merits of such methods can been seen in the overall robustness and speed of the algorithms when dealing with recovery of objects with low texture or high occlusion. Often these methods are used to form an initialisation, defining a 3D volume in which the actual surface is known to lie. This is used to reduce the size of the search space for more complex shape from stereo algorithms to run.

Shape from stereo forms the approach taken by the majority of 3D reconstruction algorithms. These methods are able to capture a surface by measuring the photo-consistency of an area in space. Under a Lambertian assumption [14], when this area of space correctly defines some part of the surface it should have a constant appearance throughout the image set, i.e. a high photo-consistency. Whereas, when this area of space does not correctly represent a surface it has an inconsistent appearance and thus a low photo-consistency. MVS reconstruction can then be cast into the problem of finding the area of space with highest photo-consistency score, which in tern defines the most 'likely' position of the real surface. This represents the necessity of MVS to match features in one image to equivalent features throughout the image set, deducing its 3D location and surface parameterisation by triangulation.

By working over surface space, shape from stereo has the ability to construct surfaces that include concavities. However, these methods do not cope well with low textured objects, where the local variance of the image is not significant. In such cases it becomes hard to take an accurate photo-consistency measure by matching pixels throughout the image set, as there is a low variational difference between opposing pixel positions. The issue plaguing reconstruction algorithms of this sort, is that the search space for this highest photo-consistent surface is not only

extremely large, but also extremely costly to evaluate. This is due to the metrics employed in matching surface patches. Large image sets with high resolutions, occlusion and lighting assumptions only serve to make such a matching process highly complex. Thus, high performance MVS algorithms usually fuse both shape from silhouette and shape from stereo techniques.

1.2 Thesis Problem Statement

In recent decades there has been significant interest in high performance recovery of three-dimensional data from passive two-dimensional images. This is mainly due to its importance and application throughout many areas of computer science. MVS techniques have received particular academic and industrial attention, largely thanks to its non-invasiveness, ease of capture, low cost and texture preserving properties. Jensen et al. [79] and Seitz et al. [138] have shown that, under well-controlled circumstances, MVS algorithms can achieve reconstruction accuracies paralleling that of prohibitively expensive active range scanners and without the need for texture distorting structured light.

The general public's ease of access to image capturing devices, offers the opportunity for MVS software approaches to penetrate every day life; however, for MVS to achieve high fidelity over a wide range of input imagery, high memory and processing times and at least moderately sized image sets are required. Reconstructing objects with low texture, non-Lambertian properties or within small datasets is still an open problem for these approaches, as such limiting their adoption into the public domain. Seitz et al. [138] showed that existing works vary in their ability to capture both high and low frequency surface details, most notably under low texture or sparsely viewed circumstances. In such situations, as will be discussed in the literature review in Chapter 2, current methods rely heavily on interpolation, as such the accuracy at which high frequency surface details can be recreated is lowered. In particular, most methods rely either on regularisation terms inside energy minimisation frameworks, which have a slow run time, high memory cost and often over smooth the surface, [31], or they neglect these areas and return no surface data, such as in depth map or patch based techniques, [53]. Consequently, low texture and sparse input views limit the application of MVS techniques in defining a general-purpose 3D reconstruction solution.

Since Seitz et al. [138] pioneering work on benchmarking MVS techniques, there has been a surge of activity within the MVS community. However, there is a considerable distance between state-of-the-art academic reconstruction systems and that in use in the public domain. The increasing complexity and narrowing speciality of academic systems in handling low texture and sparse input views has most notably caused this. For example, Autodesk have most recently released a MVS reconstruction tool named 123D Catch [9], for general public use. This system is capable of reconstructing well photographed objects to a reasonable accuracy, however, such reconstructions lack the high fidelity of systems reported by current state-of-the-art academic solutions, [79]. We hypothesis that the cause of this gap is that state-of-the-art academic research is converging on defining the absolute highest accuracy reconstruction algorithm possible, neglecting to account for its run time, memory requirements or generality to scene type. To suit the requirements of general public use a trade off between accuracy and robustness must take place. This has left a distinct gap and shortcomings when considering a robust general method.

Given the commercial value of the research in this thesis, this work was commissioned by industry, after their extensive research showed that there was no suitable solution, robust and general enough, for use in industrial circumstances. To date, there has been a great deal of publications on the MVS subject, yet there is still no high fidelity, robust and general industrial solution. This has raised the question as to how and why such methods of 3D reconstruction are limited.

With most documented MVS algorithms performing well under lab conditions, Jensen et al. [79] noted the limited scope of the tested reference scenes, as such the fact that performance deteriorates in industrial or real world situations. The scalability of MVS algorithms is further hindered by most publications overlooking or failing to define cases where reconstruction will not succeed, or the limits of each framework. Further to this, many current reconstruction techniques lack the ability to recreate the world to the desired level of accuracy in a sufficient run time and memory cost, an industrial solution demands such features. From our assessment of current state-of-the-art techniques, presented in Chapter 2, new novel algorithms are required in the field of high fidelity reconstruction. In particular, addressing the issue of reconstructing object of low texture or those captured in a sparse view set.

1.3 Aims and Contributions

3D reconstruction is a massively beneficial tool for many areas of computer vision, however, when an object's texture is low or the number of input views is sparse the ability to accurately reconstruct the true surface massively deteriorates. Depending on the MVS method, high levels of over smoothing occur or gaps and holes in the surface appear.

The purpose of this thesis is to introduce and outline research undertaken in the field of MVS algorithms. Its emphasis is on optimising MVS techniques towards robust, accurate and efficient reconstruction, accounting for factors for integration into industrial circumstances. Specifically this work will focus on improving the performance of a patch based MVS technique, traditionally such techniques produce gaps in the reconstructed patch cloud when under low texture or a sparse number of input views. Therefore, the main focus of this work will be to improve the density of patch clouds under these circumstances, intending to define an approach more suited to general-purpose reconstruction.

As well as this, this thesis will address some of the gaps and shortcomings in the 3D reconstruction literature in terms of clearly identifying current limitations and areas for improvement. It will serve as a way to identify and clearly define situations in which MVS algorithms performance degenerates and the effect this can have on the outcome of reconstruction. Although many techniques exist, the individual limitations of each with reference to the wider field is often not clear, therefore this thesis aims to aid future researchers who are undertaking a reconstruction task in making important decisions about their approach.

In order to fully utilise today's hardware technologies, this work formulates a flexible, unique and robust GPU approach to a multi-resolution patch based MVS reconstruction framework, taking a set of input images and reconstructing a highly accurate 3D patch cloud in a reasonable run time. To this end, the work of this PhD is focussed on overcoming and highlighting the issues identified within the current top performing approaches, providing reference and guidance for future work and development.

To validate and quantify the framework proposed, evaluation over ground truth datasets for both real life examples and synthetic datasets

will be performed. Analysing the effectiveness of our approach when compared against that of the current top-performing patch based technique. The purpose of which is to allow for the measurement of improvements throughout the course of this thesis by generation of a baseline performance.

The contributions of this thesis can be divided into five key sections:

1. An identification of open issues and limitations within the current top performing approaches, as well as an assessment of when such techniques are not applicable.
2. Experimentation and integration of a Particle Swarm Optimisation algorithm to remove feature based initialisation and expansion steps in patch based approaches. These steps currently limit the density of patches in low texture and sparse view sets. This method allows for the global optimisation of a patch's position and orientation parameters, rather than extracting the locally optimal parameters as used previously.
3. The unique adaptation of the evaluation procedure involved in Particle Swarm Optimisation onto the GPU. This work specifically designs the evaluation procedure to suit a MVS problem on the GPU, accounting for the sampling and computation involved in evaluating the photo-consistency function, utilising the benefits of modern hardware.
4. The definition of a new photo-detail-consistency function, to account for some of the limitations of the photo-consistency function, specifically those that arise under low texture or when sampling from sparse view sets.
5. Development of a novel multi-resolution patch based MVS reconstruction pipeline relying on Particle Swarm Optimisation and the propagation of restriction factors from lower resolution results. The purpose being to further increase density and accuracy of patches created in low textured regions or sparse view sets. Such circumstances arise often in general situations, pushing the state-of-the-art towards a more robust industrial solution.

Given that there are a plethora of techniques for recreating three-dimensional geometry providing many benefits and limitations depending on the use and application of the final reconstruction. Here, this thesis is restricted to algorithms that operate within the field of multi-view stereo, as those based on photometric stereo, model based approaches or active range scanners are not relevant to the PhD problem statement or aims.

1.4 Thesis Outline

In Chapter 2, a comprehensive literature review is provided. Primary consideration is given to the performance and limitations of state-of-the-art techniques, as well as their relevance to industrial applications.

Chapter 3 introduces some key concepts required to implement and understand the MVS framework presented. Particular attention is paid to the definition of the photo-consistency function, efficient GPU programming and the implementation of a Particle Swarm Optimisation.

Chapter 4 demonstrates our preliminary approach and experimentation into increasing the robustness of a patch based reconstruction framework for low textured datasets. Utilising a GPU Particle Swarm Optimisation, and its global optimisation strategy, reduces the need for accurate feature based initialisation inside gradient descent based schemes.

Chapter 5 defines an adapted approach to the photo-consistency metric, addressing the areas in which it can fail, as such detailing the introduction of our photo-detail-consistency function. The aim of which is to improve the noise in the resulting patch cloud.

Chapter 6 presents our full multi-resolution reconstruction framework with the aim of creating a more robust and efficient system for use in industrial circumstances. This technique incorporates the results from previous chapters, whilst defining new experimentation and evaluation.

Chapter 7 concludes this thesis, drawing together the results and methods from the previous chapter and relating them to the wider context of the thesis and the MVS problem tackled.

Appendix A contains each of the datasets used to evaluate our reconstruction framework.

2

Literature Review

This chapter provides a survey of techniques for 3D reconstruction relevant to this work. It includes a review of recent state-of-the-art MVS algorithms, assessing their strengths and weaknesses, as well as analysing how these are combined and the relationships between them. The methods presented vary in their ability to recover both high and low frequency surface details and differ in their performance over low textured regions or on sparse image sets. Particular attention is paid to the issues that arise under such circumstances and how these could be improved upon, inspiring the work in this thesis.

The benefits of MVS techniques have lead to a myriad of publications on the subject, all with wide-ranging methods and application. Surveying such an extensive range of publications would create a redundant literature review, instead we have chosen to concentrate our efforts on the top performing contributions to date and refer the interested reader to reviews available in [73, 79, 138, 146] to avoid re-covering well documented information. In the following, a taxonomy of algorithms in the multi-view reconstruction field is discussed including their individual components and how these relate to a wider more complex reconstruction system.

2.1 Organisation

In an effort to fully motivate the aims of this thesis, Section 2.2 presents publicly available industrial and open source methods to reconstruct 3D geometry from images. A brief discussion of their shortcomings and limitations show that the field of MVS is still far from solved. Section 2.3 defines fast and efficient methods to recover approximate geometry while discussing how these classical approaches are limited in their accuracy, therefore, requiring further surface optimisation techniques. In Section 2.4, the majority of literature regarding state-of-the-art MVS algorithms is discussed. These methods attempt to recreate high fidelity reconstructions by locally adjusting a surface model. Following this, methods to globally refine surfaces are presented in Section 2.5. Both sections will address the limitations of these approaches in terms of their ability to reconstruct high and low frequency details as well their performance over low texture or sparse input views. A discussion of open issues facing MVS reconstruction and its consequences can be found in Section 2.6. This chapter is concluded in Section 2.7 with a summary of our findings.

2.2 Industrial Applications

The advantages of obtaining accurate 3D models from images in a quick, easy and efficient manor has not gone unnoticed in industry. Galvanised by academic research, reconstruction technologies are evolving rapidly towards an acceptable level of quality. As such, both open source and commercial applications for extracting 3D models from multi-view images are beginning to emerge. Open source and commercial systems such as [1, 2, 5, 6, 8, 9, 75, 123, 125, 144, 152, 158, 159, 165] allow a user to upload a set of images and either a software or server side reconstruction process is run producing 3D reconstructions in various run times.

These systems, mostly in their infancy, require highly textured objects with very large input sets. Furthermore, image sets must be captured under stable lighting conditions to produce 3D data, thus are severely limited in their application. Through our experimentations, reconstructions produced by these systems often lack accuracy in order to retain

robustness, significantly over smoothing and simplifying the problem. We found that they also requiring excessive run times, ranging from several minutes to hours. Especially of note is that these systems tended to fail at producing any data for low textured objects or scenes containing an insufficient number of images.

Due to the black box nature of these industrial applications and their inability to perform under general circumstances, these systems are excluded from the rest of the thesis; they are however mentioned here for completeness and to inspire the following research.

2.3 No Surface Optimisation

In image formulation, objects within the scene reflect light into a camera lens, this is measured discretely at each pixel, forming an image. 3D reconstruction from images is the inverse problem to image formulation. Given images of the scene the problem becomes finding the surface that created these images. Such an inverse problem is inherently ill posed, since there are many different scene descriptions that match the data inside the images.

In order to alleviate some of these problems, MVS algorithms must make regularity constraints on the scene, assuming that the scene under view is represented by smooth geometry. A further simplification of the problem imposes colour consistency by assuming that the object under view obeys Lambertian properties [14]. This states that pixels on the surface of the object appear the same colour throughout the entire image set. In theory, if pixels with the same colour can be matched throughout the image set, then observing where these pixels intersect in world space allows for accurate reconstruction. However, this second assumption has been somewhat relaxed nowadays.

The interpretation of these assumptions have lead to various frameworks utilising different mathematical formulations. In the following section, classical frameworks for extracting geometry via silhouettes and colour matching are reviewed. These techniques are efficient for extracting the largest consistent surface, however, they do not rely on any form of optimisation, and thus accuracy and applicability can be an issue.

2.3.1 Visual Hull Models

In the following subsection we discuss shape from silhouette methods, forming techniques that extract the *visual hull* of the object. The visual hull is defined by Laurentini [95], as the maximal surface in which the object is consistent with the silhouettes.

The shape of the visual hull is an approximation of the true surface, it captures large object features contained along the boundaries of the silhouettes, however, lacks any small, high frequency details that lie within. Therefore, the visual hull will only ever express low frequency details, where silhouettes mask away data needed for high frequency reconstruction. Based solely on silhouettes, visual hull models are highly robust to low textured situations. However, by definition, the model returned over low textured regions tends to bulge from the true surface due to the fact they express the surface that is maximally consistent with the input silhouettes. Although enabling the robust extraction of low frequency geometry, as the image set is reduced, there are fewer silhouettes defining the contours of the object. Thus, as sparsity in the image set increases, the visual hull begins to deviate more from the true object under reconstruction, [95].

Given that robust reconstruction can be achieved even under low texture or sparse input views, the visual hull is normally used as an initialisation for higher accuracy energy minimisation techniques, attempting to further recover high frequency details. Alternatively, it can also be used for occlusion queries in depth map approaches defined later on, therefore, efficient extraction is required and will be discussed in the following text.

As described by Hartley and Zisserman [64], images display and define the projective mapping between 3D space and 2D image space as a linear mapping of homogeneous coordinates:

$$\mathbf{x} = P\mathbf{X} \tag{2.1}$$

In Equation 2.1, P represents the projection matrix that maps points in three dimensional world space, $\mathbf{X}$, onto two dimensional image space, $\mathbf{x}$.

This expresses the fact that each image pixel is coloured as the result of projecting an occluding object into the cameras lens along the ray from the camera centre to the object in view. This ray will be denoted as the *viewing ray*.

Images can be segmented to form silhouettes, using algorithms such as [23, 50, 71]. The set of pixels inside the silhouette now defines a set of viewing rays, termed the *viewing cone*. Each viewing ray, r, inside the viewing cone must meet the object at some distance, λ_r, along its extent. To be able to reconstruct the geometry contained in the images, the set of distances for each viewing cone related to each silhouette must be found.

This approach turns out to be infeasible as silhouettes do not define enough information to find an accurate value for λ_r along every viewing ray. Rather than finding λ_r values for every ray, the search is reduced to finding the distances for only the pixels that lie on the boundary between the inside and outside of the silhouettes. These boundaries are known as image based *occluding contours*. The occluding contour pixels have a special property that the projection of a viewing ray must meet and be tangent to surface of the object under reconstruction. Computing these distances is equivalent to computing the intersection of all the viewing cones, as shown in [11, 87], and this defines the visual hull. Therefore, shape from silhouette methods produce the outer-most approximation of the surface due to the silhouettes inability to express concavities on the object being constructed.

Given an input set of silhouettes the visual hull can be mathematically defined as in [47, 164] by:

$$VH(I,C) = \bigcap_{i \in I} V_i \tag{2.2}$$

Where $VH(I,C)$ is the visual hull, induced by the set of images, I, with occluding contours, C. This is the intersection of the viewing cones, V_i, created by the occluding contours, C, in each image, I. To date there are numerous algorithms that exist to compute this outer approximation to the surface, here the following methods are briefly defined: *Image Based Visual Hulls*, *EPVH*, *Voxel Carving*, *Convex Bricks* and *Carved Visual Hulls*.

Image Based Visual Hulls were proposed by Matusik et al. [109]. Their aim, rather than reconstructing a full polyhedral mesh, is to define the visual hull only for the current view direction. Hence, they only calculate depths for the viewing cone generated by the desired view. By exploiting epipolar constraints, the depth values for each ray can be computed quickly and efficiently. The epipole of the view direction is used to create a bin map of edges throughout the input silhouettes; calculating depth values for each ray comes down to calculating which bin the ray casts into in each silhouette, and the intersection of all these bins. Although this method does not build an exact 3D representation, the use of epipole binning displays a unique optimisation that has been employed in the following algorithm in order to increase run time.

EPVH, or Exact Polyhedral Visual Hulls were introduced by Franco et al. [48], inspired by [96, 109, 110, 164]. This algorithm extends the concept of image based visual hulls to define a full 3D polyhedral mesh of the visual hull. Occluding contours are cast into the volume and similar epipolar constraints to image based visual hulls or exploitation of epipolar geometry, proposed in [132], can be used to quickly define the section of the ray that consistently projects inside the silhouettes. This section of the viewing ray, now termed *viewing edge*, becomes an edge and two vertices in the reconstructed mesh. These intersection calculations are simple, stable and similar in nature to ray casting, [133]. Given that a good track of which contour created the viewing ray, and which contours defined the start and end points of the viewing edge is kept, a swept approach can be taken to step through the mesh and join the viewing edges with cone intersection edges to create a water tight mesh. This approach is both robust and efficient and due to the polyhedral nature of the reconstruction it does not suffer from aliasing introduced by volumetric approaches discussed later on. Such a technique makes it possible to create algorithms for visual hull extraction that run in real-time [48, 93, 110].

Visual Hull Voxel Carving, also termed volume or space carving, forms one of the most classical approaches to multi-view reconstruction. This robust and simple technique is defined in [42, 91, 105, 114, 119, 136, 172]. A voxel carving approach uses a volumetric representation of the surface. Volume based methods seek to express the geometry of the object by representing the area they fill. Dyer [42] expresses a volumetric

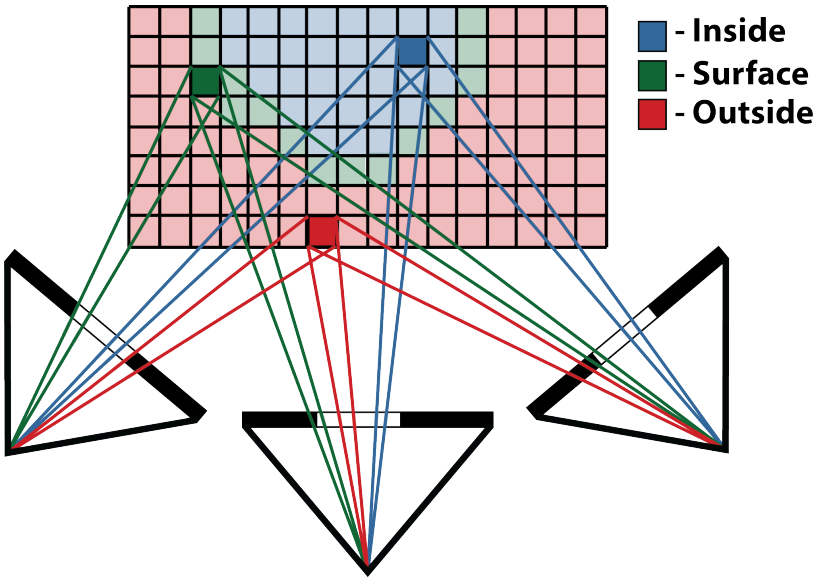


Figure 2.1: Two dimensional representation of the carving process labelling voxels by their intersection with the image based silhouettes

representation as a “bounded area in which the objects of interest lie”. It is most common for volumetric data to take the form of a bounded area filled with a regular tessellation of cubes, known as *voxels*, together forming a *voxel grid*. In a complete reconstruction voxels are classified as one of three states: *inside* the object, *outside* the object and on the object’s *surface*.

Voxel carving iteratively carves away inconsistent voxels until only those that are consistent with the input images remain, describing the surface as a set of discretely sampled voxels. Normally viewed as shape from stereo technique, to be discussed later on, the inconsistency check used to classify voxels can be simplified, removing photo-consistency and replacing this with silhouette-consistency. In the case of silhouette-consistency, voxel carving returns the visual hull of the object, whereas using photo-consistency returns the photo hull described in the following section.

Figure 2.1 demonstrates the voxel carving process generating the visual hull (green), voxels are projected into the image sets and labelled based on their intersection with the silhouettes. The most basic voxel carving technique sweeps through the volume to classify which voxels are inside the object, [150]. A voxel is discarded, or labelled as outside, if it does not consistently project inside all the silhouettes that define it, these are coloured red in the figure above. Voxels that project entirely inside are

classified as inside the surface, and those that project onto the occluding contours, i.e. partially inside and partially outside the silhouettes, are classified as surface voxels, coloured blue and green respectively in the figure above.

This algorithm is efficient and easily parallelised to the GPU, as in [105, 119, 172]. For further memory efficiency the volume can be cast into an octree as in [150]. Although this method is efficient to implement, the visual hull suffers from bricking artefacts cause by the voxel representation.

Convex Bricks, [25], are a volumetric approach designed to overcome the limitations of a voxels bricking artefacts. Rather than defining a set of static voxels, the volume is built with a set of convex bricks that can change shape throughout the reconstruction. Firstly the algorithm is initialised with the convex hull of the silhouettes, defined in [164]. This ensures that the shape of the first brick is built purely of convex polygons. To refine the bricks, each one is projected into each silhouette one at a time. The bricks are then 2D intersected with the non-convex silhouette, which causes convex bricks to change shape and possibly split to ensure convexity. Once the reconstruction has been intersected with all silhouettes it should define the same visual hull as voxel carving except for the fact bricking artefacts are removed. This approach is not as quick or simple as voxel carving or EPVH and removes the factors that make it ideal for GPU parallelisation. However, since this approach retains its volumetric representation whilst removing aliasing issues, it's ideal for situations were only the visual hull is required.

Carved Visual Hulls form a mesh based representation of the visual hull exploiting the fact that back projected viewing cones naturally encode all the information required to restrict the reconstruction. Techniques in [16, 51, 87, 97, 100] attempted to find or approximate *rim* and *frontier points* described by Furukawa and Ponce [51]. Rims, in world space, are the back projections of silhouettes occluding contours onto the model, whereas frontier points are special cases where rims intersect along the surface of the object. These types of intersection computations are extremely sensitive to numerical instabilities where frontier point are difficult to extract, this can cause meshing with holes or self-intersections, [97].

2.3.2 Photo Hull Models

Thus far, only methods to extract the visual hull from a set of silhouette images have been discussed. A well known problem of using silhouette data to reconstruct an object is that these can only encode convex shapes. Concave regions of an object cannot be captured by the silhouettes and hence can never be expressed by the visual hull. Shape from stereo techniques expand on such techniques by incorporating a scheme to reconstruct concavities on the object's surface. Whereas shape from silhouette approaches seek to define the maximal-outer surface, shape from stereo methods attempts to find a tighter minimal-inner surface. This is termed the *photo hull*, as such it express both high and low frequency information about the object's shape, whilst remaining consistent with the images that define it.

Classical methods to obtain the photo hull use a volumetric surface representation. Partitioning space with an octree approach as in [120, 150], or more recently a sparse voxel octree, as presented in [94], areas that contain large empty voxel clusters can be efficiently traversed. This strategy also allows for a neat method of level of detail (LOD) control by simply increasing or decreasing the resolution of the grid, implying that one voxel covers more / less area inside the volume. As the resolution decreases a voxel covers a larger volume and thus accuracy and resolution of the final reconstruction drops. The resolution of the volume is therefore tightly linked with the abilities of the model to express both high and low frequency details.

It has also been shown that voxel grid representations are robust to cameras that are not fully calibrated, where only the fundamental matrix is known. Works such as [85, 134] describe methods for using projective voxels to acquire shape representation, these methods only require a weak camera calibration. Discretisation of space into voxels also allows for robust and efficient methods to detect occlusion via simple occlusion queries and its structure is easily parallelised to suit GPU implementations such as in [70, 105, 119, 172].

Volume based representations are, however, limited regarding their memory efficiency and computational scalability. Large reconstructions fill a large volume and hence require a large voxel grid to be contained

in memory whilst the reconstruction is processing. In fact its storage is proportional to the bounding volume of the scene. Alongside this, for a quality representation of both high and low frequency details a large voxel grid at a high resolution, where every voxel will need to be processed, is required. Even with modern hardware this sort of resource demand and computational pressure stretches the limitations of the system.

A more significant issue in using discrete voxels as volume representation is their inability to express shapes other than cubes, and thus the introduction of bricking artefacts. By simply increasing the resolution of the grid bricking artefacts can be reduced, however this is not always feasible, for instance when there is a need to infinitely zoom into the object's surface or system resources are scarce. Thus volume based methods tend to have a poor time and resource complexity to precision trade-off.

Techniques extracting the photo hull of the object operate on the Lambertian assumption, therefore, can be seen as the multi-view equivalent to feature matching in binocular stereo. Thus an object's surface can be located by finding the set of voxels in space whose projection into the image set have the same, or at least similar, colour. The evaluation of similarity between the projection of a voxel into its image patches is known as the *photo-consistency* metric. The most notable description of a photo-consistency measure was introduced by Seitz and Dyer [136] in a *voxel colouring* process:

$$colour(p, I) = colour(\mathbf{v}, S) \tag{2.3}$$

A scene, S, is said to be photo-consistent if Equation 2.3 holds for all images, I, and pixels, $p \in I$. That is, for each pixel throughout the image set its colour matches that of a voxel, $\mathbf{v} \in S$, where the voxel is visible in I and projects to p via $\mathbf{v} = S(p)$. In reality this equality is reduced to a thresholding operation to allow for inaccuracies in image capture, thus enabling the segmentation of the surface by only including in S the voxels whose projection into the image set have a small difference to the expected colour and thus low photo-consistency score.

Because of the fact that the reconstruction is not limited to intersecting viewing cones, it can reconstruct arbitrary points on a surface by finding

the corresponding pixel matches throughout the image set; therefore, shape from stereo can reconstruct a much higher level of detail as well as reconstructing concave parts of an object's surface. This means that the final reconstruction will capture both high and low frequency details.

It is important to note here that a photo-consistency measure only defines the likelihood of a voxel sitting on the surface of the model, this is computed by comparing its projection onto image patches throughout the image set. Therefore, this method's fidelity degenerates when reconstructing a low textured or highly specular object. This is caused by the inaccuracies of the photo-consistency measure to match pixels to distinct surface voxels, [168]. Over low textured areas a large movement of a voxel results in a similar photo-consistency score, as the image patches created by projection of the voxel remain similar; hence there is a large amount of uncertainty as to where exactly the correct position for the surface voxel should be. To compensate for this, the methods below extract the outer most consistent voxel, and hence, akin to visual hull models, the surface tends to bulge over low textured regions, [91].

Similarly to the effects of low texture, sparsity in the view set also causes uncertainty in a surface voxels true location. When the size of the input image set is reduced, there are naturally fewer images for a voxel to project into, thus fewer image patches that can be compared. The smaller the number of image patches compared the less well defined the position of the minimal surface voxel will be. For example, a small motion of the surface voxel may result in a similar photo-consistency score throughout the image patches in the current image set, however, inclusion of a new image from a different angle may have meant the small motion of the surface voxel induced a large change in the image patch, and thus resulted in a much higher score. Therefore, under these circumstances, as the sparsity in the view set increases, the surface tends to bulge outwards similarly to the effects of low texture, [136].

There are many methods to define a photo-consistent reconstruction which will be discussed in the following sections, all of which range in complexity and run time, here *Voxel Colouring* and *Voxel Carving* are described, enabling the classical extraction of the photo hull.

Voxel Colouring, [30, 136], was one of the earliest introductions of a photo-consistency measure into the reconstruction pipeline. Using ordinal

visibility constraints, which define whether a point is visible in an image, a single sweep through the voxel grid marks which voxels obey the thresheld photo-consistency score, γ, in Equation 2.3. For perfectly Lambertian scenes only voxels with $\gamma(\mathbf{v}) = 0$, are accepted. However, relaxing the threshold allows for the reconstruction of scenes that do not obey this assumption. The set of labelled voxels are such that they define a volume that will always consistently project into the image set. However, such an approach is significantly vulnerable to aliasing artefacts in the presence of high occlusion and low repeating texture, or when there is insufficient input images. In such textureless regions the surface approximation will bulge, where it is no longer possible to remove inconsistent voxels. This is caused by the photo-consistency metric returning similar results over large displacements of the image based matching window, losing its discriminative abilities. As well as aliasing issues, voxel colouring can suffer from incomplete reconstructions, holes or missing sections that appear due to its liberal scheme allowing voxels on the true surface to fail the photo-consistency test.

Photo Hull Voxel Carving extends voxel colouring into voxel carving, [42, 70, 91, 105, 114, 119, 172], attempting to create a more robust method guaranteeing a complete reconstruction. Voxel carving to extract the visual hull from silhouettes was discussed in the previous subsection, here photo-consistency constraints are imposed to enable the creation of the photo hull.

Voxel colouring's method for checking photo-consistency forms the basic principles used by voxel carving. The algorithm differs from voxel colouring in its conservative approach to labelling voxels. Voxel colouring attempts to find only the voxels that are photo-consistent in a single sweep, and makes premature decisions on occlusion. Instead voxel carving makes multiple sweeps from all six faces of the volume to iteratively carve away voxels only if they are definitely not photo-consistent. This algorithm benefits from a more robust level of occlusion handling as only cameras oriented properly with the current surface are used to calculate the photo-consistency measure. This method can also take into account non-Lambertian reflection, [174]; however, this often requires prior knowledge of the position of the lights in the scene. Although this method is more robust than voxel colouring and has the ability to create surfaces without holes, it still suffers in low textured areas, where the surface stops carving or over carves depending on camera configurations.

2.3.3 Summary of Methods Not Involving Optimisation

So far this section has presented classical techniques for enforcing silhouette and photo-consistency constraints on the reconstruction of dense geometry. Section 2.3.1 showed that using silhouette data can provide an upper bound on the solution and allow for the robust extraction of the visual hull. This visual hull lacks fine surface details and concave regions, so requires further processing to obtain an accurate surface. Extracting mesh based visual hulls can suffer from numerical instabilities and issues with mutually disagreeing silhouettes, whereas voxel based surfaces are more stable, but accuracy depends upon the resolution of the volume. Extraction of such surfaces can be processed in real time, making them idea to use as initialisation for other more complex algorithms discussed in the following sections.

The extraction of the photo hull through voxel colouring or carving techniques, discussed in Section 2.3.2, defined the benefits of using photo-consistency measures to reconstruct fine surface details. However, the reliance of such techniques on a volumetric representation means that they are limited by memory and high computational demand, especially when fine voxel grids are used to extract high quality data. Further to this, the bricking artefacts induced by the shape of a voxel can be a limitation when high fidelity reconstruction is required. In practice these methods produce very low accuracy results on low textured areas, due to the lack of colour variation being modelled in the photo-consistency score, it was seen that such techniques bulge in regions of low texture.

As discussed, visual hull models can provide a highly robust approach to extracting the majority of low frequency surface data, unaffected by low texture but lacking in high frequency details. The lack of high frequency detail and bulging that occurs as the view set becomes sparse, means that further post-processing must occur to achieve high fidelity. The photo-consistency metric followed to allow for capture of the high frequency data not captured by the visual hull. However, the methods presented thus far suffer severe bulging under low texture or sparse view sets and are tightly linked to the resolution of the volume making them inappropriate for further enhancement.

2.4 Local Surface Optimisation

Thus far, efficient methods for extracting an approximation of the surface in terms of the visual hull or photo hull have been discussed. Lacking surface optimisation means that these techniques are incredibly quick to reconstruct a surface, which is highly desirable for real time applications. However, such methods are limited in their accuracy and often require further processing in order to extract a high fidelity reconstruction. In this section, techniques to optimise the resulting surface by making local stereo refinements are discussed.

2.4.1 Level Sets

Level set methods are a continuous technique iteratively minimising an energy functional, $E(S)$, defined over the surface, S, of a deformable model embedded into a volume. Level sets represent the object as an infinitely thin boundary between the inside and outside sections of a volume. This boundary moves and changes shape through each iteration such that the energy over the surface is lower than the previous iteration, once this has converged the surface can no longer move to a shape that results in a lower energy. Goldlucke et al. [59] defined the weighted minimal surface energy functional in Equation 2.4 below:

$$E(S) = \int_S w(\mathbf{x}, \mathbf{n}(\mathbf{x}))ds \tag{2.4}$$

A level set can be used to calculate the optimal surface in the sense that it minimises the integration of this energy functional over the surface, S. The energy over the surface is weighted by the function $w(\mathbf{x}, \mathbf{n}(\mathbf{x}))$, which depends on the current position, $\mathbf{x}$, and surface normal, $\mathbf{n}(\mathbf{x})$. By setting w to be the photo-consistency metric, measuring the discrepancy between the surface points throughout the image set, [24, 31, 41, 44, 46, 80, 89, 129, 130, 143, 171] casts this general level set framework into a MVS problem. The assumption being that the surface with maximal photo-consistency best represents the underlying object. Photo-consistency values range between -1 and 1, with 1 being optimal, thus can easily be cast to a minimisation problem by taking 1 minus the photo-consistency. A variety of photo-consistency metrics can be used for w, which are discussed in Chapter 3, for now we simplify the definition of w to a function

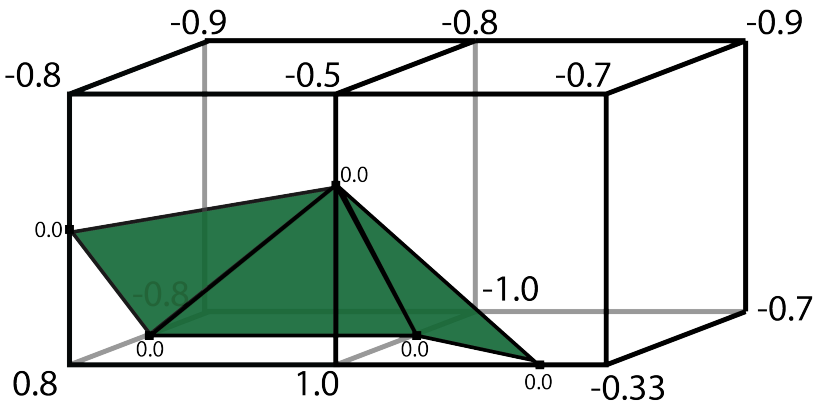


Figure 2.2: Implicit surface (green) embedded into two adjacent voxels. Each vertex of the voxels stores a value for ϕ

returning a scalar value relating to how photo-consistent a surface patch is at point $\mathbf{x}$, with normal $\mathbf{n}(\mathbf{x})$; the smaller w, the more likely the surface is.

Inside a level set framework the time evolving surface, $S(t)$, is represented implicitly at time t, by the zero level of a signed distance function, $\phi(S(t), t)$. Figure 2.2, displays the implicitly defined surface, $S(t)$, which sits inside the voxels, but is not explicitly stored. The voxel's vertices represent the value of the signed distance function, ϕ, and the surface is implicitly defined to sit at all distances for which $\phi(S(t), t) = 0$. Distance values inside the surface are positive, and distance values outside the surface are negative.

Minimisation of the energy functional inside a level set framework returns the surface that is locally optimal, [80]. This technique improves on the bricking and memory requirements of standard voxel classification approaches by no longer attempting to label entire voxels as inside, outside or on the surface. Instead level set based approaches assign weights to each of the voxel's vertices symbolizing the distance to the surface. Therefore although the volume is discretised into voxels, the underlying implicitly defined surface, ϕ, remains continuous. This fact means that level set methods have the ability to model the surface with sub-voxel precision, hence, level sets can produce a higher fidelity surface on a lower resolution grid.

Evaluating the level set at any point inside the volume returns a number indicating the signed distance of the point to the current surface. The deformation of the surface over time is controlled by steepest descent,

defined by calculating the derivative of the partial differential Equation 2.4 at every point on the surface. Goldlucke et al. [59] showed that minimisation of Equation 2.4 can be achieved via the Euler-Lagrange formulation in Equation 2.5. This derivative is used as a motion to drive (or evolve) the current surface towards the optimal solution over each iteration.

$$\frac{\partial \phi}{\partial t} = \left[-div(w \cdot \frac{\nabla \phi}{|\nabla \phi|}) + div_s(w_{\mathbf{n}}) \right] |\nabla \phi| \tag{2.5}$$

MVS is an inverse problem, therefore, another type of error that can be minimised is the error when comparing synthetic results, created by the model, against the real valued input data. This comparison can be done in image space. As shown in [38, 56, 99, 143], images created by the model can be compared in image space to the real data, thus minimisation of the energy functional is achieved by minimising error over the pixels:

$$\int_I w(H(p), \mathbf{n}(H(p)))dp \tag{2.6}$$

Expression 2.6 defines the *reprojection error functional* presented by Delaunoy and Prados [37]. I is the set of input pixels, $H(p)$ is the point on the surface corresponding to pixel p, and $\mathbf{n}(H(p))$ is its normal. A major issue in image formulation is that only visible (unoccluded) regions are displayed in the image. To rewrite Equation2.6 as an integral over the surface, the works of [130, 143, 169] propose incorporation of visibility constraints, $v_s(\mathbf{x})$, resulting in the following functional again presented by Delaunoy and Prados [37]:

$$\int_S w(\mathbf{x}, \mathbf{n}(\mathbf{x})) \cdot \mathbf{n}(\mathbf{x}) v_s(\mathbf{x}) ds \tag{2.7}$$

Thus minimisation of the energy function corresponds to finding the minimum smoothest surface with the lowest photo-consistency score. By adding silhouette terms into energy functional 2.7, such as Yezzi and Soatto [169], further improvements can be made such that the surface evolution is based on the most photo and silhouette consistent volume. Cremers and Kolev [31], proposed a convex formulation of this energy

functional, showing level sets have the ability to converge to the global optima. Given the convexity of the energy functional the local minima also corresponds to the global minima. However, achieving such a convex formulation in the context of multi-view stereo is a challenging exercise where over smoothing is difficult to handle.

Although minimising both energies functions 2.4 and 2.7 form a global minimisation, for each iteration the derivative of the surface, which drives the evolution, is calculated locally per voxel vertex. Given this form of optimisation, the surface will converge to the locally optimal solution, which is not necessarily globally optimal. Therefore a careful initialisation is required as level sets are particularly susceptible to local extrema. In order to help stop the surface becoming trapped in local minima the visual hull is normally utilised for initialisation and a regularisation term is added to the energy functional, [169], effectively smoothing the surface. The purpose of this regularisation term is to remove the noise in the photo-consistency function resulting in a smoother surface reconstruction. This regularisation term also helps the level set to evolve inside textureless regions; where the photo-consistency term returns little information, the regularisation term takes over and helps create a surface that is as smooth as possible whilst still minimising the total energy.

Cremers and Kolev [31], Li et al. [99] and Pons et al. [130] results show that far higher quality reconstructions than those presented in the previous section can be obtain by using a level set method, such surfaces containing both high and low frequency details. Although impressive reconstructions can be obtained, combining the effects of regularisation and lack of higher order derivatives, which control sharp surface deformation, reconstructions can lack surface details and suffer from over smoothing, therefore the absolute highest frequency details are still dropped. Through their results it can be seen that such methods find it difficult to reconstruct extremely high frequency surface details and sharp edges where the regularisation term prefers a smoother smaller surface over the larger area, higher frequency minimal photo-consistent surface.

Although the regularisation term and minimal surface bias make reconstructing the highest frequency details difficult they have their benefits in terms of dealing with textureless areas. In such circumstances, where the photo-consistency metric can no longer return an accurate location for the true surface, level sets rely more heavily on their regularisation term.

This shrinks the surface in favour of a smaller, smoother surface with equivalent photo-consistent score. Unlike the methods presented in the previous section, level sets tend not to bulge in such textureless regions, rather they shrink to the minimal surface. Whilst this over smooths the surface this actually tends to be a better fit than a surface that bulges. These results were shown and analysed by Seitz et al. [138].

As with all methods relying on a photo-consistency function, as the view set becomes sparse, so to does the number of image patches that can be compared to compute an accurate surface location. This creates an effect similar in nature to low texture, where the exact location of the surface can no longer be highly localised. In this situation the level set relies again on the regularisation term to extract the minimal smooth surface, so as the size of the image set drops, reconstructions become smoother, containing fewer high frequency details, and less accurate, a phenomena that can be seen in the results presented by Seitz et al. [138].

Given the implicit surface embedding, in order to visualise the actual surface, points in the volume whose distance function evaluates to zero must be found. The surface produced by extracting all zero valued distances of ϕ is termed the *zero layer* and is relatively easy to polygonise using robust algorithms such as marching cubes, [104].

An advantage of using level sets is that they are topologically adaptive, meaning that the topology of the surface is free to change throughout the evolution of the surface as the distance function updates. Achieving topology changes using methods such as graph cuts or active contours, discussed in the following sections, is a much harder process. Computation of the surface derivative on the implicit function allows for topological freedom, however, this corresponds to a significant computational burden. Narrow bands, [139], multi-resolution methods, [103], and GPU techniques, [92] have all been proposed to help reduce this demand. High computational complexity in calculating the surface derivative is still a considerable drawback to this approach, which leaves it hard to balance precision constraints to time complexity. Li et al. [99] run time table show that the large amount of computation can produce considerable time limitations.

2.4.2 Active Contours

Akin to level sets, active contour models form an iterative energy minimisation framework. Rather than employing an implicit surface representation, the deformable model is explicitly represented as a polyhedral shape, i.e. a deformable mesh. A polyhedral shape representation expresses the object's surfaces as a set of connected planar facets, such as a triangular mesh.

Meshing is robust in nature and irregular meshing is flexible enough to allow representation of any topology, including disconnected meshes. Akin to level sets, meshes benefit from the continuity of space they represent, avoiding any unwanted aliasing artefacts from a voxel based approach. Using polyhedral representations for the reconstructed object allows for a higher level of locality than level set approaches, as triangles do not conform to the voxel sampling. However, as a polygon has very little global knowledge, it becomes harder to enforce global restraints. Also without re-meshing, changing the surface topology during reconstruction cannot be achieved.

Active contour models must be initialised, this is normally achieved by extracting the mesh based visual hull as discussed by Hernández Esteban and Schmitt [66]. Once such a visual hull representation has been extracted, it is assumed that the topology of the object is correctly defined, thus further refinement of the mesh can be achieved using an active contour framework in order to create high quality reconstructions.

As with level set methods, the mesh is iteratively deformed obeying an energy minimisation framework. Works of [36, 37, 51, 53, 56, 66, 67, 68, 90, 173] describe methods to minimise either the weighted minimal surface energy functional, Equation 2.4, or the reprojection error functional, Equation 2.7, over the mesh in a MVS setting, thus iteratively defining the optimal photo and silhouette consistent surface.

Hernández Esteban and Schmitt [66] describe the use of snakes, which evolve based on gradient vector flow, [37]. Their work shows very impressive results at obtaining sharp features and both high and low frequency surface information. Similarly to level sets the deformation of the surface is controlled by local differential operators, which are taken over the vertices of the mesh rather than vertices of the voxels when

using an implicit function. Thus as the surface evolves only the vertices of the mesh are updated opposed to the entire implicit function in a level set framework. This gives a higher level of locality in terms of defining the derivative over higher frequency surfaces, where many vertices can be contained inside a single voxel when compared to using an equivalent level set. This means active contour energy minimisation tend to perform better at extracting high frequency surface details than an equivalent level set whilst having a similar performance in terms of dealing with low texture or sparse view set, as shown by Seitz et al. [138].

Throughout the evolution of mesh based surfaces, the deformation is unpredictable, thus re-meshing and edge collapse operators must be applied at regular intervals to avoid self-intersections and degenerate triangle configurations. This results in a slow, unruly process, discussed by Gargallo et al. [56]. Again these methods are very sensitive to the initialisation. Not only does the distance from the true surface depend on the initialisation, due to the local nature of the differential operator becoming trapped, but also in the number of vertices used and initial triangle configurations, this affects the contours ability to adapt to the level of detail. Large planar triangles cannot adapt well to small scale, high frequency details, however, highly tessellated meshes require large run times to update all vertices.

The results of top performing active contour methods, Delaunoy and Prados [37], Gargallo et al. [56] and Hernández Esteban and Schmitt [66], have shown impressive results defining high frequency details on small scale objects, however, the cost of such methods are that both surface and volumetric data must be stored for the surface to effectively evolve. These methods use hierarchical structures, such as octrees, to reduce the volume overhead, but the memory requirements still grow as the size of the object increases, thus scalability and computational pressure are still a major limitation.

Combining photo-consistency and silhouette constraints into an active contour model means that though surfaces achieve a high level of fidelity, they often do not back-project to the silhouettes that created them. This is caused by the difficulties of enforcing both constraints at once, where photo and silhouette-consistency compete to minimise the energy functional. Aside from this, mesh based optimisation is inherently hard to parallelise and so often to achieve high resolution results requires significant run time as reported in [36, 53, 66].

2.4.3 Depth Map Fusion

Depth maps are a concept in binocular stereo in which each pixel of an input image stores a depth value. These depth values correspond to the position of the underlying surface in 3D. A surprisingly simple extension into MVS can be devised from the creation of depth maps for each view. Since depth maps store reconstruction data per-pixel, they overcome the resolution and resampling limitations of level sets and active contour methods. They also benefit from a highly parallelisable and GPU friendly framework. As such these methods scale very well with the size of the reconstruction, number of input images and resolution of input imagery. While lacking the sub-pixel accuracies of energy minimisation approaches depth maps are extremely applicable to large scale reconstructions or high quality image sets, where efficiency in both memory and computation are key.

Stemming from key concepts in stereo vision, multiple methods for depth map reconstruction have been proposed, [19, 22, 49, 57, 58, 127, 140, 145, 153, 154]. Firstly, local depth maps are computed for each view using the surrounding group of images. Optimal depths are extracted per pixel using a standard photo-consistency measure or feature descriptor defined along viewing rays back projecting into the scene. The position along the ray where the photo-consistency measure is minimised or feature descriptors match is extracted as the optimal depth.

Goesele et al. [58] proposed a novel view selection scheme that runs ‘globally’ at image level to decide which images to compare and ‘locally’ at pixel level to decide a subset of these images which do not contain object occlusions. This view selection scheme shows impressive results for extracting depth on highly or spuriously occluded image sets. Analogously, Campbell et al. [22] proposed casting depth extraction into a Markov Random Field problem to perform outlier rejection on incorrectly labelled depth estimates. Such results showed promise in areas of repeating texture, occlusions or textureless regions, but at the cost of computational efficiency. Most recently, Shen [140] suggests creating depth maps by optimising for both depth and normal along the rays, borrowing the key concept of patch based stereo discussed in the next section, accounting for affine transformations of the matching window throughout the image set, improving the performance of the photo-consistency score.

Once local depth maps are found they are then merged or fused into a global surface by registration and regularisation. This removes inconsistent depth values by performing outlier rejection. Methods for computing the optimal local depth and best global description vary between methods. But largely the inconsistency in the matching metric and redundancy of depth data is used to filter out incorrect depths. This often leads to missing data or gaps in low textured areas, or sparsely viewed sections that are over filtered, as can be seen in Shen [140] results. Strecha et al. [145] cast the merging problem into an Expectation Maximisation (EM) framework in order to process crowded scenes. However, the complexity of the EM model grows exponentially in the number of images, so is typically limited to a maximum of three images for computational efficiency. Tola et al. [154] later showed that EM frameworks are unnecessary by exploiting image redundancy to stabilise the rejection process.

The global surface produced is represented as a point cloud and must be polygonised for use in most graphical situations. Standard techniques such as graph cuts, [22], or Poisson surface reconstruction, [81], can be used to fit the best surface to the point cloud. However, as these methods do not account for input imagery, the resulting surface can potentially be less photo-consistent than the original point cloud; Failing to faithfully reconstruct areas with missing data relying on interpolation to fill in gaps.

Depth map methods are ideal for situations where the image set is large and densely covers the object. In these situations there is significant redundancy created in the depth maps to drive the outlier rejection scheme in the merging stage above. When the image set is reduced, redundancy drops, and such schemes can then form holes in the reconstruction where depth values are not supported by enough input images in the outlier rejection scheme, Seitz et al. [138] shows this as a particular problem. Calculating redundant depth values, where depths for the same area of the surface will be computed multiple times throughout each pixel in each image, corresponds to wasted computation; However, this allows for the best possible reconstruction of the highest frequency details, where depth values are computed at the resolution of the input image set, not at the resolution of volume or mesh as with the energy minimisation frameworks above. Finally, depth maps require a full circumnavigation of the scene to work effectively at the boundaries of reconstructed objects, thus partially viewed or open surfaces tend to suffer reconstruction errors and holes, [57].

2.4.4 Patch Based

Patch based representations are a flexible approach to the MVS problem which output a point cloud of small oriented patches across the surface of the object, sometimes refereed to as *surfacels*. Patch based methods are a hybrid approach to depth map fusion that work over surface space rather than image space. Their aim is to solve redundancy and accuracy shortcomings in depth map techniques, rivalling energy minimisation methods without the significant computational burden or regularisation problems. A dense collection of patches are created by pixel-level correspondences that tightly cover the object's surface, except in textureless or occluded regions.

Patch based approaches have been consistently ranked as top performing by many reviews, [79, 138, 146]. Not only are they efficient to process, having a fairly fine granularity for parallelisation, but they also do not require the calculation of redundant depth values in depth map fusion. A distinct advantage of such an approach is the fact that these are the most adaptable to the scene type under reconstruction, [53]. Patch based representations are robust to multiple datasets such as single objects, standard scene reconstruction and crowded (or highly occluded) scene reconstruction.

Patch based methods, [26, 53, 62, 63, 76, 84, 98, 163], operate under a feature based initialisation and expansion paradigm. Where detected feature points are matched across the image set and used to robustly triangulate a feature's 3D position. This is used to create a patch, p, described by a world space position, $\mathbf{c}(p)$, and normal, $\mathbf{n}(p)$. The set of matched and triangulated feature points form a sparse patch cloud of robust and highly accurate patches. The density of the patch cloud is then increased by various region growing and expansion strategies.

Furukawa and Ponce [53], present a state-of-the-art patch based MVS technique, PMVS, for which the source code is available from [54]. First, sparse feature matches are obtained in a matching process using Difference-of-Gaussians and Harris operators. These features are matched throughout the image set, and are used to robustly triangulate a patch's world space position, $\mathbf{c}(p)$, and normal, $\mathbf{n}(p)$, forming an initial sparse patch cloud.

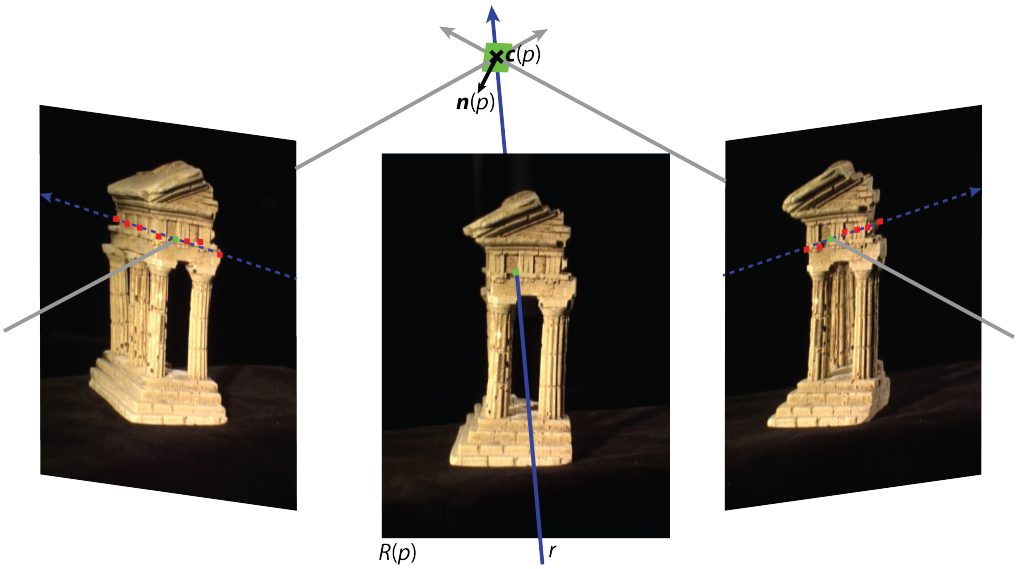


Figure 2.3: PMVS feature matching and triangulation

Figure 2.3 demonstrates the matching process for a single feature. First an unmatched feature is chosen in some reference view, $R(p)$, its viewing ray, r, is projected into space and then back projected into the surrounding images. A list of features that sit along the viewing ray in the surrounding images are collected and the matching feature is found by stepping along the list of features and comparing the reference feature with the ones in the list.

Once the matching feature is found a patch, p, can be created with centre, $\mathbf{c}(p)$, via triangulation, however, feature matching gives no information about the patch's normal, $\mathbf{n}(p)$. Therefore, to find the correct normal and accurate sub-pixel position, the patch is optimised in a gradient descent scheme using the photo-consistency function. In this case it's triangulated position and reference view's inverse view direction form the initialisation of the patch's centre and normal. The patch is confined to sit at some distance, λ_r, along the viewing ray, and its normal is cast into spherical coordinates, θ and ϕ. Therefore the optimisation takes place over the three parameters, λ_r, θ, and ϕ. Once this optimisation is complete the patch accurately sits on the surface aligning with the detected feature, with correct depth and orientation. Each patch in this initial patch cloud will project into a set of pixels throughout the input images, the image which views the patch best, i.e. the one whose view direction aligns most with the optimised patch normal, will become the patch's reference view, $R(p)$.

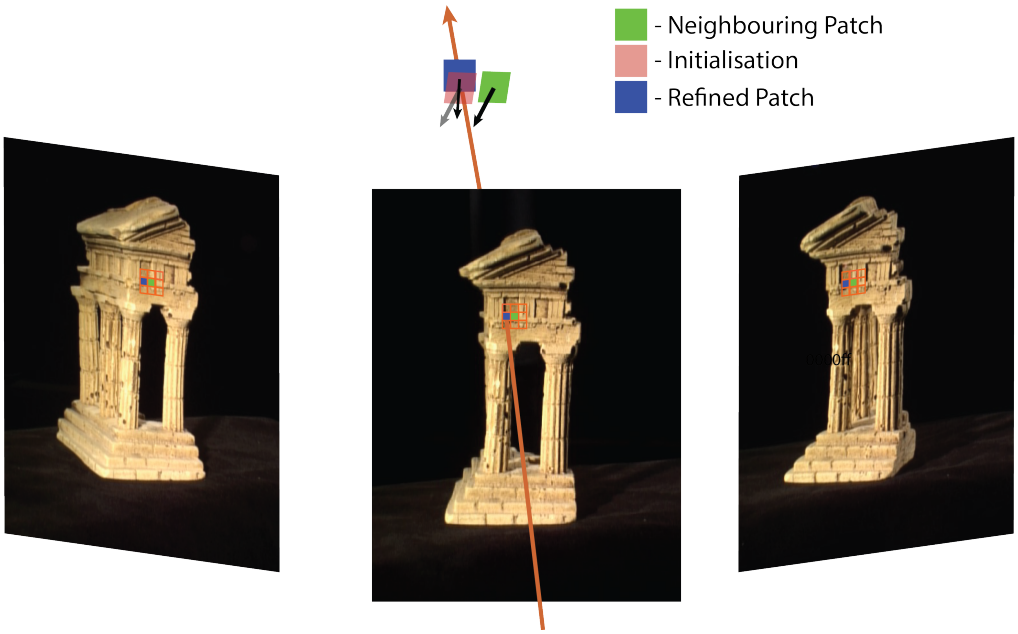


Figure 2.4: PMVS Expansion and Refinement

As the object under reconstruction will not be completely covered in features, especially in textureless regions, the initial patch cloud will contain holes. PMVS then runs an expansion step to increase the density of the patch cloud, this expansion is depicted for a single empty pixel in Figure 2.4. To perform expansion, all the empty pixels whose neighbouring pixel contain a patch are first gathered. Patches contained in the neighbouring pixels are used to initialise the position and orientation for new patches inside the empty pixels. The same gradient descent scheme, as described above, is run on each new patch to optimise its position and normal in accordance to the photo-consistency metric, such that the new patch 'snaps' to the underlying surface from its initial position. The photo-consistency is defined by sampling and comparing the surface of the patch, via bilinear interpolation, in the set of images for which the patch is visible. Given the assumption of surface smoothness, currently matched patches can be deemed 'close' to the newly created patches, thus form a good initialisation, avoiding issues with gradient descent becoming trapped in local minima.

The goal of refinement is to find the optimal position and normal for the newly created patch, in which the photo-consistency score is lowest. To optimise the patch's parameters Habbecke and Kobbelt [63] suggested using the Gauss-Newton algorithm to solve the non-linear least squares problem, whereas Furukawa and Ponce [53] use a conjugate gradient method. Both methods rely on derivative based optimisation hence are

computationally expensive and require a precise initialisation of the patch to avoid convergence to an incorrect local minima. Chen et al. [26] proposes the use of Particle Swarm Optimisation to alleviate some of the issues of initialisation within the gradient descent scheme. Their method significantly restricts the optimisation space, showing some improvement over standard gradient descent.

Surface smoothness assumptions are clearly violated across occlusion boundaries, where initialisation from neighbouring patches will be incorrect and gradient descent returns a local optima, or within textureless areas, where running gradient based optimisation results in inaccurate patch parameters. These situations result in a large number of incorrectly refined patches when expansion reaches a section of the object which joins to a textureless region or depth discontinuity; therefore, a filtering step is required. The purpose of the filtering step is to remove inconsistent patches that violate the visibility and depth consistency constraints according to the rest of the patch cloud. Patch expansion from any given feature point is stopped by the filtering stage when it grows into textureless regions or onto a depth discontinuity. The expansion and filtering stages are iterated, three times in the case of PMVS, to allow for the point cloud to grow into newly discovered regions. The issues relating to the expansion and filter stages employed by PMVS are discussed further in Chapter 6.

Feature based initialisation requires good levels of texture in order to match and triangulate features into patches. Using the filtering stage to account for errors in gradient based optimisation, patch expansion is limited to a fairly local neighbourhood of each feature. When the object is highly textured, feature points are close together, thus patch expansion is able to form a dense coverage of the entire surface. However, PMVS performs poorly in the presence of low texture or small input image sets, [53], where features are no longer close together and expansion only forms a sparse coverage of the surface. In such cases, patch based reconstructions create patch clouds with significant holes and missing data.

If accuracy is not an issue then Poisson surface reconstruction, [81], can be used to interpolate these holes and create a water tight mesh of the patch cloud. However, in order to fill these holes in the most accurate and photo-consistent manor Furukawa and Ponce [53], casts

the patch cloud into an active contour energy minimisation problem. Here, the visual hull is extracted as initialisation and is iteratively deformed towards the patches and most photo-consistent shape. In their results they shown incredibly high fidelity reconstructions for moderately well textured objects.

Whereas patch creation is relatively quick and accurate, from a few seconds to minutes, falling back to energy refinement procedures in order to fill gaps deteriorates the run time as documented by Furukawa and Ponce [53]. In most cases reconstructions can take hours, however for considerably sized data sets or large reconstructions the run time can take days. Thus, such patch based representations can be severely limited by the texture of the object, where failing to reconstruct dense patch clouds requires active contour energy minimisation approaches to fill these gaps rather than quick and simple Poisson surface reconstruction.

Patch based methods are not restricted in accuracy in the same way as voxel grid resolution affects volumetric representations or the regularisation term smooths surfaces in energy minimisation. Instead, the size of the patch and its corresponding matching window in image space can impact its ability to reconstruct high frequency surface details. Klowsky et al. [86] presented a study on how patch size affects the corresponding frequencies that can be reconstructed. They show that adapting the patch size to the surface frequency is incredibly important in defining surface details, whilst also noting their ability to accurately reconstruct both high and low frequency surface details. Smaller patches require fewer samples and computation, and can reconstruct higher frequency details, however they're less stable and susceptible to errors in refinement due to lack of texture variance over the samples of the patch. Increasing the patch size creates a more stable and less noisy patch cloud filling larger textureless regions, however, this comes at a cost of deteriorating the run time and over smoothing the patch cloud. Larger patches approximate a larger surface area with more samples and a planar assumption, thus reducing the patches ability to reconstruct high frequency surface details. Therefore using the correct patch size for accuracy and computational gains is a core problem in a patch based method, choosing an optimal patch size to achieve both robustness whilst reconstructing high frequency surface details is a challenging problem, and depends on the image set provided. Habbecke and Kobbelt [63] have suggested using the texture variance to somewhat adapt the patch size.

2.4.5 Summary of Local Optimisation Methods

So far in this section techniques for obtaining high fidelity reconstructions via local surface optimisation were presented. Local optimisation schemes such as level sets or active contours have the ability to accurately recreate high frequency surface data, however, they come with significant computational cost in the use of the derivative of the energy function and evolutionary nature of the optimisation. Whereas regularisation terms help these methods to intelligently fill areas of low texture or missing data, they inherently over smooth high frequency details and are susceptible to computing the minimal surface area in regions of low texture or when supplied with insufficient imagery. Minimal area surfaces do not necessarily reflect the true underlying surface. As well as this, the fidelity of the reconstruction is tightly linked to the volume resolution, which is hard to define over different object types.

Level sets can produce high quality surfaces at sub-voxel accuracy requiring lower resolution volumes, whereas active contour models can achieve high accuracy with high polygonisation. Polygonisation can be adapted over the object's surface controlling quality in different areas. Volumetric data structures always require large amount of memory to store, however this can be somewhat alleviated using octrees or narrow bands, regardless of this they are always limited in their scalability.

Depth map fusion and patch based approaches showed incredible promise for a faster more efficient reconstruction, however, these approaches require high redundancy in image space in order to form reliable surfaces. Textureless image regions still cause issues for these approaches, however rather than rely on regularisation terms to fill holes, they produce no data in these areas. Therefore, post processing of the scene is required to accurately fill gaps, deteriorating the run time. Such methods are robust to scene type and efficient to process, and have been shown to produce higher accuracy reconstruction and good scalability for most generally well textured objects. Open issues exist with patch based approaches in terms of correctly adapting the patch size to the underlying surface detail, as well as methods to accurately and robustly refine patches in textureless areas.

2.5 Global Surface Optimisation

So far local optimisation schemes for refining small areas or patches of the object have been discussed. These methods have shown good results in reconstructing high fidelity details over a small patch of the scene, however, their reliance on local derivative based optimisation means that these approaches are sensitive to initialisation and computationally expensive. In the following section, methods for global optimisation of energy functions are presented, alleviating the necessity for a good quality initialisation.

2.5.1 Graph Cuts

Graph cut methods, [17, 51, 69, 141, 160, 161], are concerned with computing the max-flow/min-cut of a weighted graph. Boykov and Kolmogorov [17] showed that weighted minimal surfaces in Equation 2.4, can be globally minimised when $w(\mathbf{x}, \mathbf{n}(\mathbf{x}))ds$ corresponds to a continuous Riemannian metric. The Riemannian metric is based on integral geometry, representing contour lengths computed from photo-consistency and silhouette-consistency measures. It defines a mapping on the surface that is constrained to vary smoothly from point to point.

Graph cuts can be viewed as the discrete equivalent to continuous level sets, both defining the minimal solution of the cost function over the surface. Unlike level sets whose roots lie in differential geometry, and thus require the derivative to drive the surface motion, graph cuts rely on integral geometry. As such they measure the distance of the current solution to the desired solution and thus compute globally the best step to the desired solution.

To obtain a graph structure of the space a voxel grid is used, voxel corners become nodes and voxel edges become weighted edges. Edges join nodes to each other and to independent source and sink nodes, forming a well connected graph. Edges now form an explicit representation of the surface, where edge weights approximate the length of the contour, and thus the surface is defined at some ideal weight, w_s, as described by Boykov and Kolmogorov [17].

Such a graph structure represents a simple binary Markov Random Field. Where voxel nodes are to be labelled as inside or outside the surface by defining the minimum cost of cutting edges to form this labelling. As such the problem is formulated as an s-t cut through the volume separating nodes connected to the source, s, from those connect to the sink, t. Given these facts a surface can be defined as the graph cut through the edges obeying and minimising the energy functional in Equation 2.4, [17].

Given the use of the Riemannian metric the edge weights represent the distance of the current edge to desired solution, thus allowing the globally optimum solution to be obtained by computing the graph cut where the sum of all edges results in the lowest overall energy. As graph cuts do not rely on the local derivative, Boykov and Kolmogorov [17] show that they can obtain a globally optimum surface, up to the resolution of the volume, in polynomial time. Boykov et al. [18] note that graph cut solutions only return the globally optimal solution to binary labelling problems, in the case of n-labels the globally optimum solution can only be guaranteed for convex functions.

Vogiatzis et al. [160] formulated MVS as a graph cut problem. Source nodes are voxels along the surface of the visual hull and sink nodes are voxels within some distance, D_{in}, inside the visual hull. The optimal surface is therefore assumed to lie within the source and sink surfaces. They showed the energy can be minimised effectively within a graph cut framework, however, a ballooning term needs be added to the energy functional, similarly to the regularisation term used in level sets. This is a side effect of minimising the Riemannian metric, as the empty set is the optimal solution having minimal geodesic length. At protrusions in the object's surface, a set of edges with high cost but low length would be preferred to the true set of edges extending into the protrusion with low cost but large length. Hence, the inclusion of a ballooning term is required to force the surface outwards into these areas. The limitations of adding such a ballooning term into the energy functional, means that graph cut approaches struggle to reconstruct thin surface structures or high frequency surface data. Seitz et al. [138] analysis showed that although graph cuts can achieve high fidelity, their level set counterparts out performed them in terms of accuracy.

As opposed to level sets, graph cuts lack sub-voxel accuracy, as the surface is no longer embedded into a signed distance function, rather it is defined by the set of edges forming the minimal cost surface. Vogiatzis et al. [161] explains the fact that by increasing the connectivity in the graph structure the accuracy of the approximation can be increased, allowing a higher number of connected nodes rather than the standard 6 neighbourhood connections. More edges increase the accuracy of the approximated continuous functional but comes at a high memory cost.

Given that a graph cut is only defined on the edges of the surface, it is not necessary to extract the zero-layer. Therefore, although graph cuts can be computed efficiently, they require a higher volume resolution in order to achieve the same level of accuracy as a level set. Their topology is also fixed from initialisation of the source and sink nodes via the visual hull. They have also been criticized for their 'blockiness' in relation to the volume resolution and the need to increase connectivity, meaning that graph cut methods often need to sacrifice precision due to memory availability, [161].

Zeng et al. [175] presents a fusion of voxel carving, patch based MVS and graph cuts in an effort to add scalability and flexibility to standard graph cut approaches. Allowing for the reconstruction of arbitrary large geometry without resource limitations and easier handling of complex topology. Here a coarse voxel carving approach is utilised to remove inconsistent voxels and partition the surface into chunks, or patches. To improve the accuracy of the surface chunks a graph cut is used to locally define the surface inside each voxel, significantly improving the resolution of the surface. The resulting patchwork is merged together using volumetric techniques inspired by Curless and Levoy [32]. Using this method means that arbitrarily large reconstructions can be made with any grid resolution, thus overcoming the limitations of voxel carving and standard graph cuts. Leaving only the run-time complexity of the system as a limitation. As the volume or its resolution increase in size, so does the number of patches required to complete the surface, thus the run-time grows linearly to the level of detail and size of reconstruction.

Such a technique shows promising results however Zeng et al. [175] noted the fact that it is unclear how to properly set the coarse voxel carving resolution against the fine graph cut size to achieve optimal reconstruction accuracy. Results have shown that getting this balance

wrong can produce a worse surface than graph cuts alone. Further to this, the resolution parameters required are also tightly dependant on the object under reconstruction, where using incorrect patch sizes can remove the ability to express complex topologies.

Akin to minimal surface bias of the level set and active contour methods presented in the previous section, graph cuts to benefit from the way they handle low texture and sparse input views. Rather than create a mesh with holes, as with depth map fusion or patch based approaches, graph cuts intelligently interpolate a surface, using the globally minimal surface requirement and incorporated ballooning term. This results in a slightly better surface over low textured regions when compared to level sets. In these circumstance, graph cuts default to the smallest length, smoothest surface that minimises the photo-consistency function, hence surfaces in low textured regions are smoothed out and flatted, [161].

2.5.2 Summary of Global Optimisation Methods

So far in this section techniques for obtaining high fidelity reconstructions via global surface optimisation were presented. Graph cuts define a framework for minimising energy functions with a lower time complexity than level sets, whilst also relieving the need for careful initialisation by guaranteeing the globally optimal solution is reached. However, graph cuts require high amounts of volumetric storage that cannot be compressed into octree structures or narrow band techniques. Although, Hornung and Kobbelt [69] have shown recent success in using a hierarchical approach for some memory benefits.

Though the nature of graph cuts guarantees a globally optimum solution, it is not clear whether this optimum is in fact closer to the true surface than a locally optimum solution. For example, when reconstructing a head, should the shape of the ear on the left side really affect the shape of the right. In the context of textureless data, graph cuts tend to perform better than level sets, with less over smoothing from the global nature of the optimisation. However, the introduction of a ballooning term means that, as with level sets, these methods still struggle to reconstruct the highest frequency details. As well as this, the assumption that the surface lies within D_{in} distance from the visual hull, limits the depth of concavities that can be reconstructed.

2.6 Discussion

The aims of this chapter were to highlight the areas and situations in which current state-of-the-art MVS algorithms excel, as well as to assess their shortcomings and limitations. Particular attention was paid to their ability to reconstruct both high and low frequency details as well as their performance over low textured regions and on sparse datasets. Such situations arise frequently in daily life and industrial circumstances, as such require robust and effective handling to recreate high fidelity geometry. From assessment of the current top performing solutions that have been presented in this chapter, it does not appear that any of these satisfactorily handle general situations. Over smoothing or incomplete reconstructions are still an open issue for most methods.

In order to provide standardised comparison between MVS algorithms, Seitz et al. [138] first presented a quantitative and qualitative evaluation of reconstruction techniques focussing on the accuracy and completeness of the resulting mesh. More recently this has been followed by reviews and benchmarks presented by Jensen et al. [79] and Strecha et al. [146]. These have shown that highly accurate reconstructions are possible for well textured objects with moderately sized data sets. However, as the texture of the object drops or the size of the input dataset is reduced, all MVS algorithms to date reduce in accuracy and robustness. This is caused by the photo-consistency metrics inability to locate the correct surface position to a high degree of accuracy. As the texture is reduced or the number of input views is lowered, the photo-consistency metric returns similar values over a larger section of space, therefore, the correct position and orientation of the surface becomes harder to localise and spurious matches can achieve a better photo-consistency score than the correct surface, [168]. For this reason dealing with reconstructions under such circumstances becomes a challenging problem.

Scalability in terms of speed and memory, in the reviews mentioned above, show that as the size of the scene grows, to achieve high accuracy over all surface details, their memory and computational demands must increase. The structure of different methods leads to significantly different results in terms of accuracy and scalability. Hence, the type of scene under reconstruction often defines the right choice of algorithm. Some algorithms will produce holes where reconstruction cannot be accurately

	Ability to reconstruct high & low frequency details	**Robustness to low texture**	**Robustness to sparse view sets**
Visual Hull	Accurately reconstructs low frequency detail contained along silhouette boundaries only	Surface sticks to silhouette boundaries, so bulges	Surface bulges and diverges further from true object the more sparse the view set becomes
Voxel Carving	High and low, limited to resolution of the voxel grid	Surface bulges and distorts	Surface bulges and distorts
Level Set	High and low, limited by resolution of the voxel grid and weighting of regularisation term	Minimal surface bias and regularisation term shrinks and smooths surface	More reliance on regularisation term which over smooths surfaces
Active Contour	High and low, limited by mesh polygonisation and weighting of regularisation term	Minimal surface bias and regularisation term shrinks and smooths surface	More reliance on regularisation term which over smooths surfaces
Depth Map Fusion	High and low, up to image resolution	Filtering creates holes in textureless regions where optimisation for the same surface inside different depth maps do not align	As the sparsity increases, filtering thins the depth maps creating holes where there's less support from neighbouring views
Patch Based	High and low, limited by size of patch	Filtering creates holes in textureless regions stopping expansion across depth discontinuities or into textureless areas	Fewer views make fewer feature matches, thus sparser initial and final patch clouds which contain more holes
Graph Cut	High and low, limited by resolution of graph and weighting of ballooning term	Minimal surface bias and ballooning term shrinks and smooths surface	More reliance on ballooning term and minimal cut which over smooths surfaces

Table 2.1: Reconstruction algorithm comparison

relied on, whereas others rely on regularisation/ballooning to predict where the correct surface could have been from that already been defined.

Table 2.1 provides a summary of the algorithms presented in the reviews and previous sections. Specifically, for each algorithm, a comment is made about its ability to reconstruct both high and low frequency surface details. For a quality reconstruction, it is desired that both high and low frequencies are accurately represented in the reconstructed mesh. Following this, the third and fourth columns of Table 2.1 describe the effects on the final surface in terms of how each method deals with low textured regions or increased sparsity in the view set, again noting that the purpose of this research is to improve MVS ability to accurately reconstruct data inside such low textured, sparse view sets.

Section 2.3.1 discussed methods for using silhouettes to extract the visual hull, these demonstrated an efficient and robust solution to textureless data. However, the visual hull lacks accuracy, given the silhouettes inability to express concave regions, and the fact that it will only express low frequency details contained along the silhouettes boundaries. Therefore a classical Voxel Carving approach to extract further details was described in Section 2.3.2. This is particularly well suited for fast GPU based reconstructions, combining photo and silhouette consistency into a single framework; However, this classical approach is limited in its ability to reconstruct textureless objects, causing bulging of the object's surface, and suffering from highly obtrusive bricking artefacts from the cube-like shape of a voxel.

Reconstructing an object by extracting its minimal surface using a level set, active contour or graph cut, were discussed in Sections 2.4.1, 2.4.2 and 2.5.1 respectively, and overcome many of the limitations of classical volumetric approaches. These surface representations do not attempt to discretize such large volumes of space and so can overcome significant artefacts caused by brick shape voxels. Minimal surfaces enjoy the freedom to make local and global changes to the model to produce the minimal cost surface. Whilst being able to construct high and low frequency details, the resolution of the volume and the weighting of the regularisation and ballooning forces in the energy functional define the surface fidelity.

These methods are distinctly limited by these regularisation or ballooning terms, enforcing smoothness constraints on the reconstructed surface. As such, this chapter discussed their tendency to over smooth the surface at locations containing thin structures. They also have a distinct bias towards surfaces with smallest area, removing the highest frequency details in favour of smaller area. Along textureless regions these approaches adopt the minimal surface, as such will tend to flatten out bumpy textureless surfaces. Depending on initialization, local methods may also suffer from bulging in these regions if caught in a local minima. Such circumstances are hard to predict and vary between different objects. As well as this, their scalability in terms of memory and run time is highly dependant on the resolution of the volume or mesh, [161]. To achieve a highly accurate reconstruction requires a large volume or mesh resolution, which then increases the computational demand and memory requirements, leading to an approach that is not feasible for general public use.

Seitz et al. [138] compares fidelity results between different energy minimisation techniques and shows that local surface optimisation tends to achieve a better quality reconstruction. I hypothesize that this is due to the derivative based nature of the deformation allowing the surface to be refined locally as opposed to globally. This would imply that locally optimal surfaces better reflect the true optimum than globally optimal surfaces. Alternatively, the need for the ballooning term in graph cuts, which stops the cut favouring smaller surface area to fine details, could enforce more smoothing than the local regularisation terms. Either way, the cost of a better surface approximation comes at a high computational price. Though achieving good results on small-scale reconstruction problems, in the context of large-scale reconstructions, such surface minimisation methods quickly become infeasible in their memory costs and computational demands, hindering their scalability.

Depth map fusion, which was presented in Section 2.4.3, provided a simple, principled and scalable approach to 3D reconstruction. Although this method is hindered by its heavy reliance on image redundancy to allow for quality fusion of multiple depth maps into a global surface. Such approaches lack the sub-voxel accuracies of energy minimisation, however, given modern day image resolutions, such accuracies would be negligible to the human eye, although to our knowledge no such quantitative proof has been performed.

Given that a pixel's depth is found by locating the minimal photo-consistency along its viewing ray, for textureless pixels depth values can be highly sporadic and erroneous, depending highly on the initialisation into depth optimisation and clarity of the underlying photo-consistency function, [57]. In such cases, the redundancy of the surface patch throughout the image set allows for the rejection of such outliers. This leaves these methods susceptible to holes in textureless regions, rather than surface flattening or bulging. The reliance on image redundancy, although scaling well to increased data sizes, does not scale down quite as elegantly, and so is not well suited to sparse image sets.

Patch based techniques, described in Section 2.4.4, show great promise for high accuracy 3D reconstruction, in particular PMVS was described as the top performing MVS algorithm across the board in benchmarks [79, 138, 146]. Forming a hybrid approach to depth map fusion, they to benefit from computational simplicity and low memory requirements,

whilst retaining a local nature to surface optimisation. In fact by removing redundancy in standard depth map approaches they afford better computational gains and flexibility to smaller image sets. The removal of regularisation or ballooning terms and the adaptability of patch sizes, allows for recovery of the highest fidelity reconstructions without over smoothing. However, care at small patch sizes, where texture variance is reduced, must be taken to avoid falling into noisy textureless refinement of patches, [86]. Patch based techniques also demonstrate the most scalability to scene size and type, [53].

As the expansion strategy of PMVS utilises gradient descent to refine a patch, a good initialisation from neighbouring patches is required. Across occlusion boundaries or in textureless regions this is not possible, resulting in a thinning of the patch cloud. Following this, filtering of the patches based on visibility and depth consistency further thins the patch cloud. Hence, growing patches into textureless areas is not possible and patch clouds produced, although highly accurate, contain holes over textureless regions. This effect is worsened when using sparse view sets, which result in fewer feature matches creating a thinner initial patch cloud on which to expand, for low textured object this was shown to create further larger holes, [53]. Poisson surface reconstruction can be used to fit a surface to the patch cloud and fill the holes, however, this may not be sufficient when there are large gaps in the surface. Thus, active contour refinement is sometimes used as the very last process, deteriorating patch based performance back to an energy minimisation framework. In theory, if patch based approaches could be adapted to produce moderately accurate patches within these textureless regions, Poisson surface reconstruction should be sufficient to reconstruct a surface in these situations. This would result in a reconstruction technique that is robust to low texture and sparse view sets, whilst most accurately extracting both high and low frequency surface details, that also runs in a much reduced computation time.

2.7 Summary

This chapter has extensively reviewed the state-of-the-art in MVS reconstruction. Whilst covering a wide range of the top performing approaches it has identified the following key principles:

- Visual hulls are robust, simple and efficient to create. However, they are insufficient for use in a general purpose setting as they lack high frequency surface details.
- High fidelity reconstruction is possible, but in most cases requires a large run time, high memory requirement and reasonably sized input dataset.
- Surface refinement via an energy minimisation has shown high fidelity results. However, these are susceptible to large amounts of over smoothing and require high memory and computation. In all cases, these are also sensitive to initialisation.
- Depth map and patch based techniques are incredibly efficient in terms of both memory and computation. However, such methods are less robust to textureless regions often producing sparse patch clouds in such circumstances. They also require larger image sets in order to perform efficient outlier rejection and higher quality feature matching.

3

Techniques for 3D Reconstruction

This chapter presents the fundamental techniques required for a complete understanding of the methods adopted by the MVS framework defined in the following chapters. Due to the nature of MVS, there are many different and wide ranging techniques employed in each system. To avoid over complicating this chapter, discussion is limited to only those techniques that are directly applicable to the MVS framework presented throughout this thesis.

In Section 3.1 the fundamental geometric principles of the camera matrix and projection are introduced, giving rise to the formation of multi-view camera set-ups. This section defines the method in which we can relate 3D world space to 2D image space and vice versa. A discussion on similarity metrics and the use of the photo-consistency function, used to match pixels, in image space, and patches, in world space, is defined

in Section 3.2. The photo-consistency metric is the primary tool used to accurately locate a surfaces location in world space, by comparing its projection throughout the image set. In Section 3.3 the principles of GPU programming are presented, defining the standard for implementing massively parallel algorithms, whilst also listing methods for achieving peak performance. A GPU applicable approach will be taken throughout this thesis, therefore a good understanding of parallel architecture is crucial. Following this, Section 3.4 presents the particle swarm algorithm and its GPU implementation, this can be used to find a functions optimal parameters. This optimisation technique will be used to refine a patch's parameters via the photo-consistency function in the MVS algorithm presented by this thesis. This chapter concludes with a summary of the techniques presented in Section 3.5.

3.1 The Camera Matrix and Calibration

This section presents the mathematical concepts related to 3D projection and camera calibration, both of which are fundamental tools for any reconstruction pipeline. It will present methods for relating 3D world space points to 2D image pixels. The ability to relate 3D world space points to 2D image space pixels allows for the evaluation of the photo-consistency metric, this is achieved by projecting a patch in world space to its corresponding image patches throughout the image set. In Chapter 4 our patch-based approach to MVS will be introduced, this relies extensively on the ability to project patches to pixels and the evaluation of the photo-consistency function. Discussing a complete formulation of the camera matrix and its calibration would simply reproduce work easily obtainable elsewhere, specifically the interested reader is referred to Hartley and Zisserman [64], and their comprehensive coverage of this subject matter. Here only the essential principles of the camera model are discussed, which directly affect the reconstruction system derived in this body of work.

Image formulation is the most fundamental concept in 3D Reconstruction. As previously discussed, this is the process of mapping three dimensional world space coordinates onto a two-dimension image plane. A three dimensional world space coordinate is mapped down onto a two-dimensional image plane by casting a ray from the centre of projection out to meet this point. This ray was, earlier, termed the viewing ray in Section 2.3.1. This viewing ray will then intersect a plane know as the *image plane*. The point at which the viewing ray intersects the image plane will become the image of the world space point. This follows the standard pinhole camera model where light is reflected into the lens and collected by the film, or the CCD for digital cameras.

$$\begin{bmatrix} x_{is} \\ y_{is} \\ z_{is} \end{bmatrix} = P \begin{bmatrix} x_{ws} \\ y_{ws} \\ z_{ws} \\ 1 \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} x_{is} \\ y_{is} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x_{is}}{z_{is}} \\ \frac{y_{is}}{z_{is}} \\ \frac{z_{is}}{z_{is}} \end{bmatrix} \tag{3.2}$$

Hartley and Zisserman [64] define this projective mapping between world space coordinates, $[x_{ws}, y_{ws}, z_{ws}]$, and image space coordinates, $[x_{is}, y_{is}, z_{is}]$, by the projection matrix, P, in Equation 3.1. Once P has been applied, a perspective divide of the image space coordinates by its final component, z_{is}, is required to produce the projected image space location in Equation 3.2. Therefore, any world space position can be projected into any camera's image space by applying Equation 3.1 followed by Equation 3.2, under the assumption that we know P for the specified camera.

The projection matrix, P, is a 3x4 matrix encoding the intrinsic and extrinsic properties of the camera. An intrinsic matrix, K, encodes the internal camera and lens setting which are the focal length, sensor size, skew and principle point. The extrinsic properties define the transformation of the world space coordinate system into the camera space system,

via a rotation matrix, R, and translation vector, T. Estimating the intrinsic and extrinsic parameters is the problem of camera calibration, however, once these are found, Hartley and Zisserman [64] showed that the projection matrix, P, can be expressed in Equation 3.3 as:

$$P = K[R|T] \tag{3.3}$$

Hartley and Zisserman [64] also show that recovering the camera's centre, C, in world space can be done using the camera's extrinsic properties, where R' is the transpose of R through Equation 3.4 below:

$$C = -R^{-1}T = -R'T \tag{3.4}$$

The problem of 3D reconstruction is, given a set of images alongside the calibrated projection matrices for each view, is it possible to define the inverse mapping from image space to world space? This is essentially the process of recovering the depth component, z_{is}, for each pixel of each image, that is lost by the perspective divide when dropping from three dimension to two in Equation 3.2.

Although camera calibration was not specifically studied, as it falls outside the scope of this project, calibrated input imagery into the reconstruction pipeline is required. To perform 3D reconstruction we require knowledge of a distinct projection matrix, P_v, for each input view, v, to allow for the comparison of the image of a world space position throughout the view set; therefore, issues related to obtaining accurate projection matrices through camera calibration are briefly discussed in order to allow for a full understanding of the 3D reconstruction process. To date there are two main approaches to calibration, these being *chart based calibration* (CBC) and *structure from motion* (SfM). CBC requires the insertion of an object with known geometry into the image set, normally a checkerboard, whereas SfM relies on the detection of sparse feature correspondences.

A typical SfM calibration starts by creating a set of sparse feature points to be matched throughout the image set. Variations in illumination and pose can cause a feature to look very different throughout the image set. Feature descriptors such as SIFT, Difference-of-Gaussian and Harris feature points, [113, 112], which are robust to these circumstances are

thus employed. These feature points are then robustly matched over pairs of images using RANSAC or patch based methods [52, 64, 126]. This simultaneously estimates a set of consistent feature points and the fundamental matrices linking them. The fundamental matrix defines the geometric relationships between two cameras. A full calibration of P, up to a constant scaling factor, is obtained by combining the results and running an auto-calibration process, [126]. SfM systems are limited in the scenes that they can calibrate due to their reliance on initial feature matching. Therefore, they tend to perform poorly in weakly textured situations, [52].

CBC requires the use of a calibration object, manually inserting texture into the scene. Due to this, such methods can guarantee calibration under a wide variety of different circumstances. Given that the geometry of the calibration object is known, feature detection and localisation is simplified, improving its robustness to lighting and view changes. Hence, these systems are very popular and Matlab provides a full camera calibration toolbox, [15]. Although CBC is highly accurate and fairly simple, adding known objects into photographs is not always feasible, for example, when using historic image sets or when the use of a calibration object distorts the true objects texture. Furthermore, designing a calibration object that is visible from all view directions is often not possible. Camera calibration is, therefore, not always a non-trivial procedure, some care must be taken to get good calibration results.

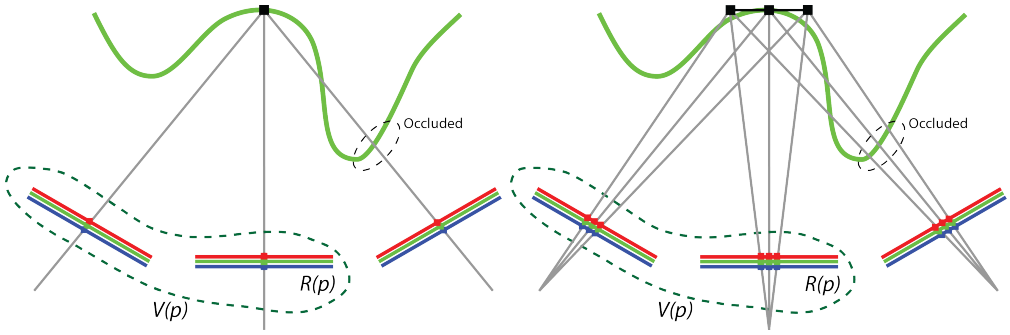
In order to achieve higher calibration accuracy, a global optimisation of all camera parameters and estimated sparse geometry can be undertaken. This process is known as *bundle adjustment*, [156]. The camera parameters are jointly optimised with the scene geometry to find the calibration and sparse point cloud that is maximally consistent with each other. Furukawa and Ponce [52] showed that the optimal camera parameters and precise localisation of each point in space can be refined by finding the local orientation of the point or camera which results in the most consistent distortion. Bundler, [142], is an open source SfM camera calibration toolkit with added bundle adjustment. This software was used to calibrate the LynxReal datasets, Section 4.2 and Appendix A, used in this project.

3.2 Photo-Consistency and Similarity Metrics

Photo-consistency is a fundamental tool in MVS. Using this metric, estimation of the discrepancy between image patches can be made. As previously mentioned, under the Lambertian assumption, pixels throughout multiple images are assumed to be similar when they express the same point on the surface in world space. This is a crucial tool for matching features and triangulating their 3D position. An object can be reconstructed by finding all the points in world space for which their projection into the image set results in a high photo-consistency, therefore separating the objects surface from the background. As presented in the literature review most MVS algorithms define a surface, either via level set, mesh, graph cut or patch, and check this surfaces photo-consistency value throughout the image set; thus ensuring that the surface description matches the image set with high photo-consistency. In practice, this is converted to a minimisation problem by taking one minus the photo-consistency, where the lowest photo-consistent surface results in the most optimal reconstruction.

In Chapter 4, our preliminary patch based MVS reconstruction algorithm will be presented, this method will rely heavily upon the photo-consistency metric to enable accurate extraction of high frequency surface details. In this case a patch's position and orientation will be refined by finding the optimal parameters in the sense that the projection of the patch, p, with position, $\mathbf{c}(p)$, and normal, $\mathbf{n}(p)$, results in the lowest photo-consistency score. Later on, this photo-consistency method will be built upon in Chapter 5 to define a more robust and accurate photo-detail-consistency measure.

Figure 3.1 displays the two main approaches to measuring the photo-consistency of a world space position; point based methods or regional based measures. Point based methods, Figure 3.1a, project a world space position into the set of images via Equations 3.1 and 3.2. The image space locations are then sampled throughout the set of images for which the position is visible, $V(p)$, this sampling occurs over each colour channel via bilinear interpolation. The variance in colour for each of the colour channels over every pixel that it projects into is used to define the photo-consistency value. One of the images will be labelled as the reference view, $R(p)$, and this will be used to compare with the



(a) Point Based (b) Regional Based

Figure 3.1: Photo-consistency measures

other images in the set. Absolute Difference (AD) is used to measure the variance between pixels, found by computing the sum of the differences between the reference pixel and all other pixels. Alternatively, the Squared Difference (SD) can be used, computing the sum of the squared differences between the reference pixel and all other pixels.

Although point based measures are computationally efficient, only extracting a pixel per view, these measures often require large image sets. For obvious reasons, these measures fail when the Lambertian assumption is broken, or when there is non-constant illumination throughout the view set. In these cases the difference in pixel intensities no longer corresponds to how well these reflect the same surface. The use of regional measures, which attempt to define a small patch of surface in world space and measure the image space photo-consistency of this, are more robust to these settings, [73].

Regional measures, Figure 3.1b, define a plane, centred over a world space position, which will projected into a window that is sampled in each visible image, $V(p)$. A reference view, $R(p)$, is chosen based on the view with the best angle to this plane. The plane is then scaled such that the projection of the plane into the reference image results in a μxμ pixel window. This plane is then projected into the rest of the view set and sampled via bilinear interpolation. Once the samples have been collected the Sum of the Absolute Differences (SAD), Sum of the Squared Difference (SSD) or Normalised Cross Correlation (NCC) can be taken over each of the colour channels, [46, 53, 66, 130]. These return a scalar value representing the similarity measure of the plane in space throughout the images that it projects into. Care must be taken when dealing with occlusion, as images in which the region of space is occluded in its view

will skew the matching score, making it seem a less feasible position. Hence, image sets are normally filtered for occlusion before applying the regional measure, [53].

The selection of the correlation metric, SAD, SSD or NCC, is independent of the rest of the reconstruction system. However the performance of these largely impacts on the effectiveness of the reconstruction. The general consensus is that NCC is the most popular and best suited to the multi-view scenario, [73, 130, 138]. For this reason the next section will present an in depth study of the NCC.

3.2.1 Normalised Cross Correlation

$$NCC_c(I_1, I_2, p) = \frac{\int_{\Omega(p)} (I_1(x) - \overline{I}_1)(I_2(x) - \overline{I}_2)dx}{\sqrt{\int_{\Omega(p)} (I_1(x) - \overline{I}_1)^2 dx \int_{\Omega(p)} (I_2(x) - \overline{I}_2)^2 dx}} \tag{3.5}$$

Hu and Mordohai [73] defined the NCC on colour channel, c, between two images, I_1 and I_2, for a patch, p, in Equation 3.5. Here, $I_n(x)$ is the colour at pixel x on the window, $\Omega(p)$, induced by the patch, p, projected into the view I_n; where, $\overline{I}_n$ is the average intensity of the μxμ window inside the image.

This metric returns a scalar value measuring the similarity between two image patches. Faugeras and Keriven [46] noted that the value returned by the NCC is the inner product between the two patch vectors, implying that high values (close to 1) are achieved when the image patches are similar, and low values (close to -1) when they are maximally different.

The NCC metric is an extremely popular method to compare image patches due to its invariance to both additive and multiplicative global illumination changes. This allows for 2 images that where taken at different times under different illuminations to be compared as if taken instantaneously under the same lighting conditions. This is an invaluable asset for MVS, where time spent moving the camera often allows for subtle illumination changes.

The subtraction of the average window intensity in the numerator means that any additive illumination changes in the form $I \mapsto I + \alpha$, have no effect on the matching score. I is the intensity of a pixel and α is the global added (or subtracted) illumination change. Division by the variance of the intensity over a patch, in the denominator of this score, ensures that the NCC remains invariant to multiplicative global illumination changes of the form, $I \mapsto \beta I$. Where β is the multiplicative illumination change. It is important to note that only the normalised cross correlation, with division by the patch variance, is invariant to multiplicative illumination changes.

As SSD is invariant to additive illumination changes, it can be seen that NCC and SSD will perform equally well when there are no multiplicative illumination changes present. However, in the case that there is, NCC has been shown to significantly outperform most techniques in dealing robustly with this change, [73].

$$\gamma(p) = 1 - \frac{1}{3\,|V(p)/R(p)|} \sum_{v \in V(p)/R(p)} \sum_{c \in \{r,g,b\}} NCC_c(I_{R(p)}, I_v, p) \tag{3.6}$$

The regional photo-consistency value, $\gamma(p)$, for a patch, p, which is visible in the set of images, $V(p)$, with $R(p)$ as the reference view, is defined by Furukawa and Ponce [53] and displayed in Equation 3.6. Here, $|V(p)/R(p)|$, is the number of views in which the patch is visible, excluding it reference view. $NCC_c(I_{R(p)}, I_v, p)$ denotes the NCC score between reference image, $I_{R(p)}$, and a second image, I_v, for the colour channel, c.

As depicted in Figure 3.1b, each colour channel computes a NCC score between 1 and -1, and each view computes a score with the reference view, this value is then averaged and converted to a minimisation problem to compute the final photo-consistency score, $\gamma(p)$. Both Furukawa and Ponce [53] and Hu and Mordohai [73] note that in order to reduce the sensitivity of the NCC to colour channels with small variance in any one channel, the standard deviation in the denominator of Equation 3.5 can be taken for the $3x\mu x\mu$ vector obtained by stacking the elements from all colour channels in the window after the mean values are removed.

3.2.1.1 Limitations to NCC

Given the NCCs invariance to illumination changes, one assumes that it performs very well against methods such as SSD, in ideal situations this does. By factoring out the illumination changes between images of different viewpoints, the NCC is able to return far stronger matching scores in well fitting areas and far lower scores where patches do not match, [73]. However, having illumination invariance also comes with certain shortcomings. In the situation of using the NCC to compare a continuous white patch against a continuous black, the NCC is undefined. Given the mean intensity subtracted from each pixel's intensity will result in zero and so the numerator and denominator will be zero, the NCC cannot return any information about their similarity. After all a white patch is nothing but the black patch that has under gone a significant illumination change. The NCC looses the any ability to distinguish between constantly coloured patches. Such constant patches do not arise very often in computer vision problems, mainly due to sensor noise, however, it is very likely that textureless surfaces will have approximately constant colouration hence causing the NCC results to 'float' around 0. This is one of the main reasons why 3D reconstruction algorithms fail in textureless regions, where it is no longer possible to take an accurate measure of photo-consistency exactly aligning with the correct surface.

3.3 General Purpose Computing on Graphics Processing Units

Nowadays parallel computing is an increasingly important aspect of computer science. Algorithms to solve today's complex problems are often very computationally demanding, strongly limiting their applicability. Taking the NCC across a range of input imagery and then minimising this score based on either derivative or stochastic methods, is one such computationally demanding procedure encountered in the MVS domain. As such, GPU methods are explored in this work to reduce the computation time of such complex metrics. Chapter's 4 to 6 make extensive use of a GPU approach to Particle Swarm Optimisation and taking the NCC for a patch, therefore, this section covers the principle concepts of GPU programming, providing the reader with the necessary information to appreciate the following work.

Fuelled by the need for more processing power, there has been a rapid increase in the performance and programmability of the graphical processing unit (GPU). As such GPUs have become highly parallel programmable processor as well as extremely powerful graphics engines. Given their peak arithmetic and memory bandwidth, they are quickly becoming a compelling alternative to traditional parallel CPU techniques. As such the academic world is shifting towards General Purpose GPU (GPGPU) computing in order to solve these complex problems more efficiently and effectively. Owens et al. [124], presents a survey of some of the more classical GPGPU techniques and the impact of such approaches. In this section the techniques involved in adopting a GPGPU based algorithm are discussed.

Historically, OpenGL and DirectX provided limited graphics APIs for interaction with the GPU. This interaction was achieved through a fixed function pipeline, where programmers would use the API to send a description of their scene from the CPU to the GPU. The GPU would then transform this description, rasterize it and finally light and texture it to form a 2D image. This fixed functionality quickly became a limitation, not only in the speed of rendering but also the types of rendering possible. Therefore a programmable pipeline was introduced, where programmers could override the fixed functionality in their own code, called shaders.

Originally shaders could only be written in assembly, making them tricky to understand and time consuming to maintain. In response higher level programming languages, OpenGL shading Language (GLSL), and DirectX's High-Level Shading Language (HLSL) were developed. Analogously, NVIDIA developed C for Graphics (Cg) this was compatible with both OpenGL and DirectX APIs. Given the simplicity of these newer shader languages, interest was sparked in the academic community for using this graphics pipeline in a more generalised way to solve more scientific problems. This turned out to be highly effective and hence the revolution in GPGPU computing. However, the inflexibility of the graphics pipeline meant that not all problems could be suitably mapped to the device.

In order to overcome the restrictions of a strict pipeline and programming style, NVIDIA release a new API for interacting with the GPU, free of shaders and the graphics pipeline. They called this API Compute Unified Device Architecture (Cuda). Requiring NVIDIA GPUs and drivers to

enable Cuda to run, motivated the development and release of the Open Compute Language (OpenCL) by Khronos Group. OpenCL is not only for GPUs, rather can be used to interact with any underlying hardware compatible with the OpenCL compute model. Whereas Cuda is based on Cg and a long established development, OpenCL is comparatively newer but provides a vendor independent API for GPGPU.

The establishment of OpenCL and Cuda APIs for interacting with the GPU has had a wide impact on compute performance over many application domains. Most notably medical imaging, image processing, computer vision, pattern and signal analysis, physical simulations, and financial modelling, are all among a wide range of areas benefiting from GPGPU technologies, [124]. Due to the independent development of each of these APIs, there are some technical differences between them, however, they are fundamentally similar in their massively parallel design. Code written for one platform is easily portable to the other. In this body of work, due to its open source, cross platform nature, OpenCL was chosen as the platform on which to develop the multi-view framework. Hence, the following sections will discuss the OpenCL compute model and factors related to this that must be taken into account when writing GPGPU based algorithms.

3.3.1 Execution Model

GPUs achieve high levels of performance through data and task parallelism, breaking large problems into smaller sub-problems that are executed in parallel by a high number of GPU cores. These sub-problems should be broken down further into sub-tasks, which again can be solved in parallel only this time at thread level on each of the individual cores. The smaller or simpler the sub-problems and sub-tasks the faster it can be computed.

Thus to design an effective parallel algorithm requires the programmer to first break the problem into sub-problems and sub-tasks that can be solved in parallel. The ‘coarsity’ at which the problem is broken down into sub-problems and sub-tasks can have a major impact on performance. Sub-tasks should require minimal processing, so as not to block processing on a single core. Whereas sub-problems shouldn’t require too many sub-task to solve, limiting the throughput of sub-problems. Balan-

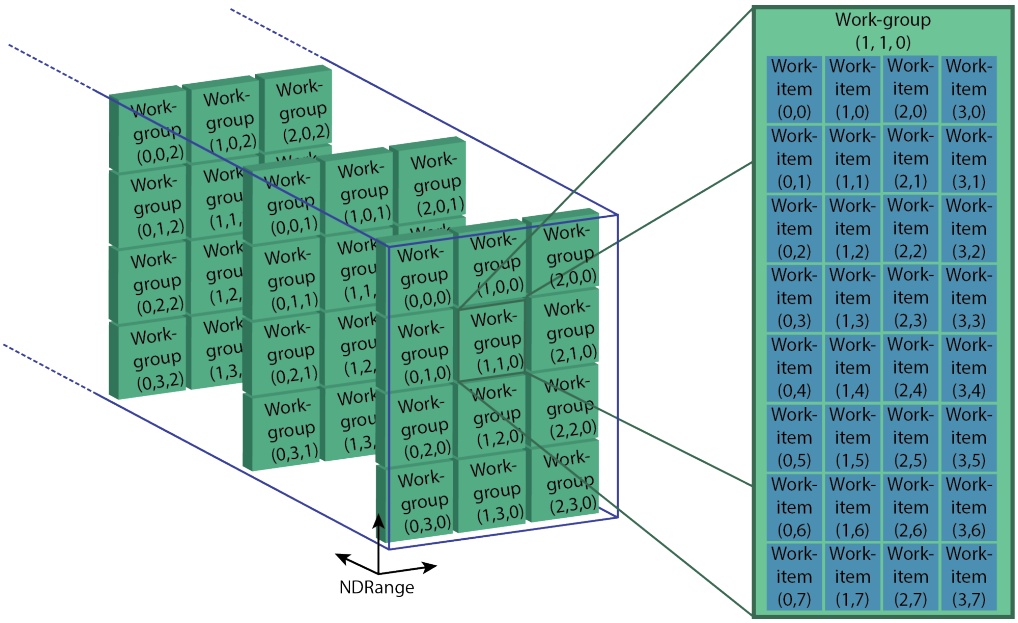


Figure 3.2: Work-items are ordered inside work-groups creating an NDRange

cing the break down of a parallel algorithm is always problem dependant. As different GPUs have many different configurations of cores and thread processors, this problem is also device dependant.

In terms of GPGPU computation, this represents two different levels of parallelism. The first, most simple level is fine-grained data or execution parallelism. In OpenCL this is termed a work-item, in the analogy above this was called a sub-task. A work-item is the finest level of parallelism in any GPGPU algorithm and as such each of these work-items executes a kernel. This kernel is a simple, C-like, function run in parallel by all threads, jointly solving the problem. The second, coarse-grained, level of parallelism is seen as the sub-problem in the analogy above. In OpenCL this is termed a work-group, where each work-group consists of multiple work-items. All work-items belonging to the same work-group are computed effectively in parallel on one of the many GPU cores. The grid of work-groups, constituting the problem to be solved, is called an NDRange in OpenCL, and is equal to the total number of work-items in each dimension. The NDRange is spread over all of the cores on the GPU to achieve the highest possible parallelism. This also allows scalability of the problem over the number of physical cores available. Therefore, a GPU with more cores can process more work-groups in parallel, solving the problem faster. Figure 3.2, displays the relationship of work-groups to work-items over a three dimensional NDRange.

To solve a problem the programmer invokes the kernel function over a one, two or three-dimensional (“ND") range of work-groups. These work-groups are processed in parallel as are their work-items. The work-items inside a work-group can cooperate efficiently between themselves through a local cache and synchronization primitives. However, synchronisation between work-groups is not possible. To illustrate the difference between kernels/work-items, work-groups and NDRange clearly we will look at a simple example of squaring a one-dimensional vector. Figure 3.3 gives the standard C implementation of this problem below:

```
void square(int n, int* a)
{
        for(int i=0; i<n; i++)
                a[i] = a[i] * a[i];
}
```

Figure 3.3: C code for serial implementation of squaring a vector

In this example a is a vector which is n elements long, for simplicity we will assume $n = 2048$ elements and ignore any problems with integer overflow. In order to transfer this implementation to the GPU we will need to define a kernel and work-group size; since we know the size of the array, the one dimensional NDRange must be 2048. Remembering that the kernel operates on each work-item and this work-item must be of finest coarsity, we define each work-item to be one element in the resulting array; Therefore the kernel function must take an element from the array and compute the squared value and write it back. Figure 3.4 defines the OpenCL kernel code that solves this problem:

```
kernel void square( global int* a)
{
        int indx = get_global_id(0);
        a[indx] = a[indx] * a[indx];
}
```

Figure 3.4: Kernel code for parallel implementation of squaring a vector

Here the keyword $kernel$ is required to distinguish the difference between a function that runs on the host device (CPU) and one that will be run over the work-items on the GPU. The keyword $global$, is used to mark the fact that vector a is stored in global memory space as discussed in the following section. The function $get_global_id(0)$ returns the index of the current work-item that the kernel is processing, [60].

The remaining problem is to define the size of a work-group, OpenCL specifies that this must be a multiple of the NDRange. In this example, imagine the GPU on which we wish to run this code contains 64 cores. To ensure maximum speed, computation should be spread over each of theses cores. To achieve maximum occupancy, the work-group size should thus be set at 32. In this instance, each work-group will contain 32 work-items and there will be 64 work-groups in total. This information is transferred to the GPU when invoking the kernel in Figure 3.5. For more information on interfacing with the GPU please refer the to the OpenCL specifications, [60]. The C code that must be called on the host device (CPU) to invoke the kernel to run over all work-items with the given work-group and NDRange size is shown in Figure 3.5 below:

```
int errNum;                 // error code
size_t global = 2048;       // global NDRange size
size_t local = 32;          // local work-group size
cl_command_queue cq;        // handle to OpenCL command queue
cl_kernel squareKernel;     // handle to OpenCL kernel

errNum = clEnqueueNDRange(cq, squareKernel, 1, NULL,
                          &global, &local, 0, NULL, NULL);
```

Figure 3.5: Host code to invoke a GPU kernel

3.3.2 Coalesced Memory Access

To allow for complex address mapping for each work-item an indexing scheme is used, as in the kernel presented in Figure 3.4. Each work-item can be identified within a kernel by a unique local work-item ID, this being its index into the work-group containing it (function $get_local_id(n)$ in the OpenCL specifications). Its global work-item ID and work-group ID can be queried as the work-items index into the entire batch of work-items and which work-group it belongs to respectively (functions $get_global_id(n)$ and $get_group_id(n)$ respectively in the OpenCL specifications). This indexing structure makes it possible to assign each work-item a unique memory access location. This is useful, for example, when simulating the assignment of a work-item to pixel in image processing applications, or work-items to array indices when working with vectors.

This indexing scheme can lead to a memory access pattern known as coalesced. A coalesced access of memory occurs when each work-item within a work-group simultaneously reads an address within the same

neighbourhood as the others. Coalesced memory reads are related to the way GPUs store and access memory held in the global space. As reading from the global memory space is relatively slow, when a kernel requests memory, the GPU will automatically read in a large chunk of memory, which will contain multiple elements. If all items in a work group access elements sequentially, the GPU can deliver all the required memory to the kernels with the minimum number of memory reads. In this situation locality and alignment criteria are met, meaning that memory reads can be broadcast by the distribute bus.

As slow memory reads can become a bottleneck in many applications, coalesced memory reads define the fastest possible way to read memory inside a GPGPU algorithm. Ensuring that coalesced memory criteria are met can have a large impact on the over all execution time of kernels, [115, 155]. However, if the structure of the problem is data dependant or in some way unstructured, it may not be possible to define a coalesced memory access, in which case, memory access is highly likely to become a bottleneck. Figure 3.6 shows different forms of memory access that could have been used in the kernel defined in Figure 3.4, displaying the fact that different access patterns require a different number of chunks to be read.

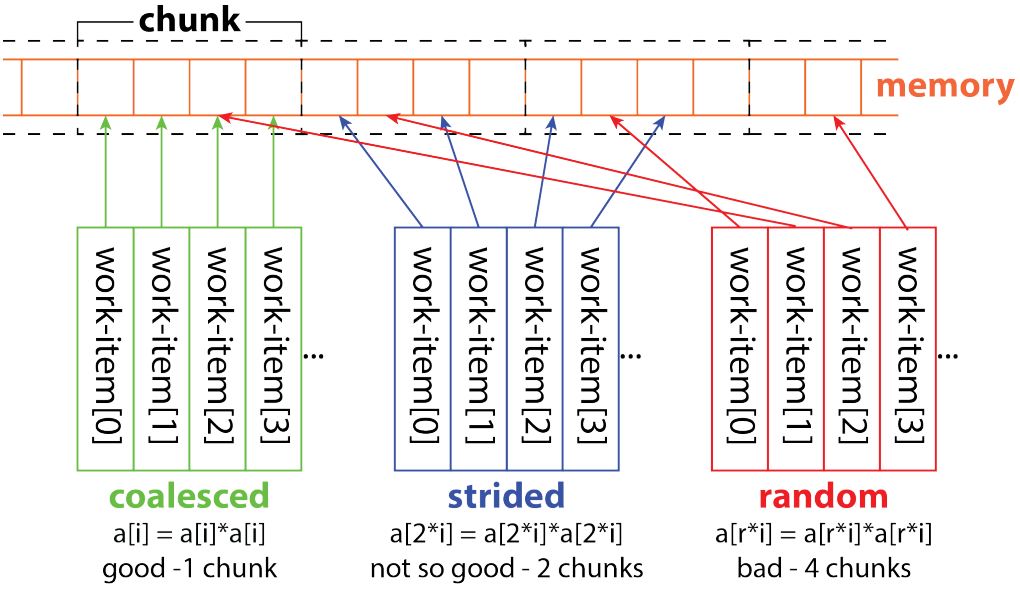


Figure 3.6: Memory access patterns

3.3.3 Memory Model

As well as coalescing memory access through the indexing scheme, to achieve the best performance, memory structure must be accounted for. Figure 3.7 displays OpenCLs memory hierarchy.

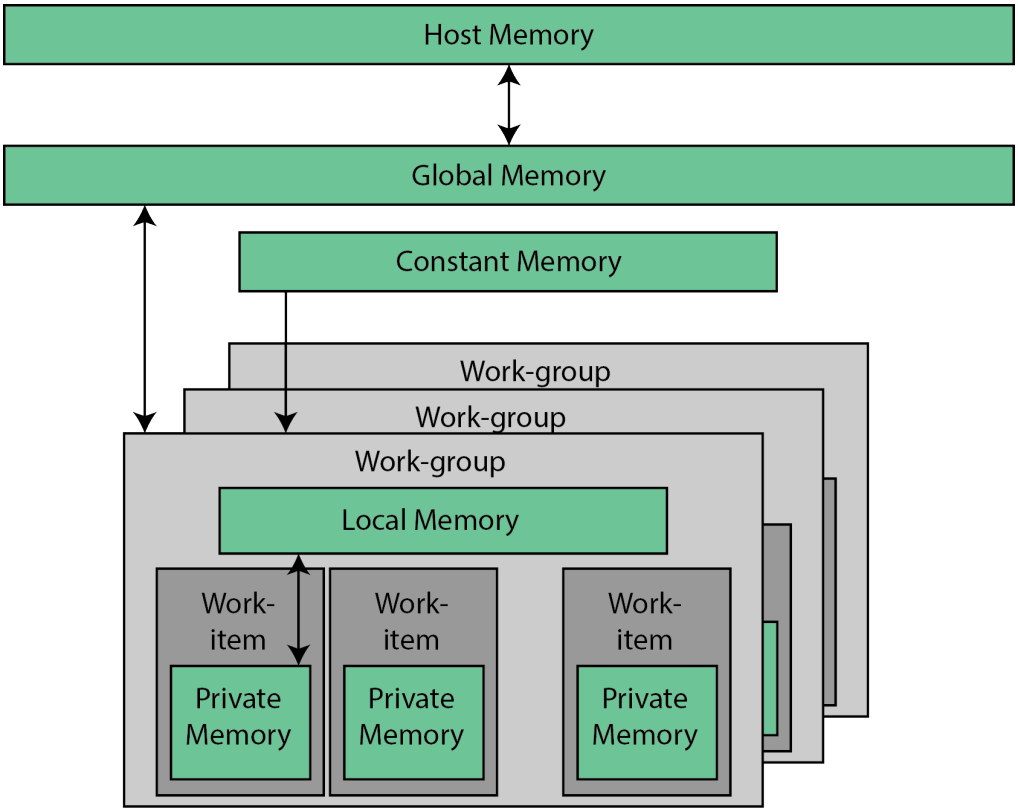


Figure 3.7: OpenCLs memory hierarchy

Figure 3.7 shows that each work-item has access to a limited amount of private memory, these are the lowest latency registers usually used to store each work-items variables. All work-items within the same work group can communicate efficiently through a small amount of low latency local memory. Additionally, every work-item has access to constant as well as the highest latency global memory. Constant memory is a small amount of memory intended only for broadcasting and cannot be altered by kernel calls. Global memory persists over multiple kernel calls, as such defines the only method of transferring data from one kernel call to the next. Global memory has the highest latency so reading and writing to this memory space should be kept to a minimum for peak performance.

A common optimisation of global memory usage is to read sections of the global memory into private or local memory when each work-group first starts, [115]. The work-items inside the work-group will then operate on private and local memory rather than global, writing results back to global memory only if necessary. However, this optimisation is not always possible, especially when work-items read and write locations are unpredictable or not linked to the structure of the problem. To illustrate this form of memory optimisation, we choose to update the implementation of our vector-squaring kernel defined in Figure 3.4. It should be note that this kernel requires two global memory reads, followed by a global memory write. To minimise the use of global memory we will use private memory to read and store the global memory and write the result back, using only one global memory read and one global memory write, this updated kernel is displayed in Figure 3.8 below:

```
kernel void square( global int* a)
{
        int indx = get_global_id(0);
        int privateA = a[indx];
        a[indx] = privateA * privateA;
}
```

Figure 3.8: Optimised kernel code for squaring a vector

Finally host memory resides CPU side and is used to copy memory to and from global memory, held GPU side. Uploading and downloading memory from the graphics card requires transfer across the high latency memory bus, thus should be kept to a minimum. Ideally, data should be uploaded once to the GPU as initialisation, then only downloaded once the computation is complete. Intermediate results should be stored in global memory between kernel calls, avoiding the necessity to downloaded intermediate results back to the CPU.

Given the memory architecture of modern GPUs and the execution model defined by OpenCL, to achieve peak performance the following programming guidelines have been suggested, [115]:

1. Minimise memory transfer between the host device and the GPU.
2. Minimise the use of global memory.
3. Ensure global memory access is coalesced if possible.

3.4 Particle Swarm Optimisation

In this section the optimisation of a candidate/cost function through the use of Particle Swarm Optimisation (PSO) is discussed, overcoming some of the limitations of derivative methods. In Chapter 4 this work will apply the PSO procedure in a patch based MVS framework, setting the candidate function to be a locally defined regional photo-consistency function. PSO will be used to refine a patch, p, in terms of finding its optimal centre, $\mathbf{c}(p)$, and normal, $\mathbf{n}(p)$, resulting in a patch with the best photo-consistency score.

Optimising a set of parameters based on the value of a fitness function is an incredibly broad and complex field. In most MVS cases, especially patch based techniques, the best position and normal for a patch, corresponding to the optimal photo-consistency value, must be found. Finding the parameters that give the optimal photo-consistent result, either locally or globally, is a very sensitive problem; depending not only on the complexity of the function but also the technique used in obtaining the optimal result. With many different techniques existing, the most effective one to use is normally application dependant.

Iterative methods such as gradient or conjugate gradient descent, Newton and Quasi-Newton methods are popular techniques for accurately detecting a functions optimum via its derivatives. Level set methods use such techniques to iteratively define the motion of the surface towards the optimal result, [59]. They are highly effective when correctly initialised, in most cases requiring only a small number of iterations to converge. However, in cases of poor initialisation or a noisy cost function such methods suffer from a tendency to get stuck in local minima, where the derivatives of the cost function become zero, [46]. As such, a careful and costly initialisation or the need to run the method over multiple seed points is often required, reducing their accuracy and efficiency.

To avoid these limitations and guarantee convergence to the global optimum, it is required that the fitness function be convex, [31]. In this case there is only one optimum, this being the global optimum. As discussed, Cremers and Kolev [31] and Kolev and Cremers [89], introduced a convex photo and silhouette consistency function for level set frameworks, but such convexity results in high levels of smoothing. To achieve the most ac-

curate results requires a non-convex functional, hence, refinement using derivative based methods requires at least a semi-accurate initialisation of the surface in order to avoid local minima.

Heuristic methods such as the Genetic Algorithm (GA) or PSO have the ability to provide approximate solutions to the optimisation that are far less affected by initialisation or local extrema, [83]. Heuristic methods are derivative free, meaning that they overcome local minima and inherit good global search properties. However, they are limited in their local search resolution, losing the ability to step along the gradient exactly onto the global minimiser. Hence, heuristic methods perform well at detecting global optima, but without a significant number of iterations, they often lack exacting precision.

Particle Swarm Optimisation is a population based stochastic optimisation method for continuous non-linear functions introduced by Kennedy [82] and Kennedy et al. [83]. Inspired by the social behaviours of animals such as birds flocking or fish schooling, it can be used to iteratively minimise or maximise the solution to a given candidate function. This is achieved via a population based search strategy with non-deterministic and probabilistic rules. Compared with other stochastic optimisation techniques such as GAs, PSO has been shown to have comparable or in some cases superior search performance as well as faster more stable convergence rates, [65]. It has classically been employed in situations where the search space is tricky to explore, such as in large data sets with high dimensionality. In these cases derivative based methods would be computationally expensive and sensitive to the large number of local minima.

3.4.1 Standard Particle Swarm Optimisation

PSO consists of a set of particles making up a swarm or population. The goal of each of the particles is to traverse the search space to find the globally optimal solution. In the most classical approach, named Traditional Particle Swarm Optimisation (TPSO), [82, 83], each particle, p, evaluates the fitness function at it's current position. It then keeps track of the position it has visited that had the best score, its personal best position, $pbest_p$. It also tracks the global best position of the entire swarm, $gbest$. Any particle can update $gbest$ if its current position results in a better

score than the position held by $gbest$. Based on the personal and global best locations, each particle can define a velocity vector, the direction in which it will explore. Over each time step, or iteration, the particles move along their velocity vector and converge to the global best-found solution, ensuring a quality sampling of the entire search space.

Algorithm 1 Traditional Particle Swarm Optimisation

procedure TPSO($X(t)$)
  $t \leftarrow 0$
  for $p \in X(t)$ **do** ▷ Initialise the swarm
    $Initialise\ \mathbf{x}_p(t),\ \mathbf{v}_p(t)\ and\ pbest_p$
    $Evaluate\ f(\mathbf{x}_p(t))$
  end for
  $Update\ gbest$
  while $t < tmax$ **do** ▷ Iterative refinement
    $t \leftarrow t + 1$
    for $p \in X(t)$ **do**
      $Calculate\ \mathbf{v}_p(t)$
      if $\mathbf{v}_p(t) > v_{max}$ **then** $\mathbf{v}_p(t) \leftarrow v_{max}$ **end**
      if $\mathbf{v}_p(t) < -v_{max}$ **then** $\mathbf{v}_p(t) \leftarrow -v_{max}$ **end**
      $Update\ \mathbf{x}_p(t)$
      if $\mathbf{x}_p(t) > \mathbf{X}_{max}$ **then** $\mathbf{x}_p(t) \leftarrow \mathbf{X}_{max}$ **end**
      if $\mathbf{x}_p(t) < \mathbf{X}_{min}$ **then** $\mathbf{x}_p(t) \leftarrow \mathbf{X}_{min}$ **end**
      $Evaluate\ f(\mathbf{x}_p(t))$
      $Update\ pbest_p$
    end for
    $Update\ gbest$
  end while
  return $gbest$ ▷ return the optimal solution
end procedure

Figure 3.9: Traditional Particle Swarm Optimisation

Figure 3.9 shows the pseudo-code for TPSO defined in [82, 83]. $X(t)$ denotes the swarm at the t-th iteration confined to the search space $[\mathbf{X}_{min}, \mathbf{X}_{max}]$. Each particle, $p \in X$, has a position, $\mathbf{x}_p(t)$, and velocity, $\mathbf{v}_p(t)$. Where $x_p^i \in \mathbf{x}_p$ and $v_p^i \in \mathbf{v}_p$ denote the i-th components of the position and velocity respectively. $f(\mathbf{x}_p(t))$ denotes the value of the fitness function for the particle, p, at position $\mathbf{x}_p(t)$.

The total number of iterations before termination is represented by t_{max}, controlling how long the swarm has to converge. Setting this to low means that the global optimum may not be found, setting this to

high corresponds to wasted computation. v_{max} represents a velocity dampening parameter defining the resolution within which a particle can move. In effect, v_{max} enhances the local exploration of a particle. If v_{max} is too high the particle could skip over good solutions, if v_{max} is too low particles may not explore beyond their local neighbourhood. In practice, Eberhart and Shi [43] showed that v_{max} should be limited to the dynamic range of $[\mathbf{X}_{min}, \mathbf{X}_{max}]$.

As previously stated the velocity of a particle is based on two social factors, its personal best position, $pbest_p$, and also the entire swarms global best position, $gbest$. At each iteration within TPSO, Kennedy [82] first updates a particles velocity by Equation 3.7, and then translates its position by Equation 3.8. In the evolution equations below, $rand()$ is a randomly generated number between 0 and 1. Kennedy [82] multiplies each random factor by 2 so that its mean is 1, the rational being that the particle would "overfly" its target roughly half of the time.

$$\begin{aligned} v_p^i(t) = v_p^i(t-1) &+ 2 \cdot rand() \cdot (pbest_p^i - x_p^i(t-1)) \\ &+ 2 \cdot rand() \cdot (gbest^i - x_p^i(t-1)) \end{aligned} \tag{3.7}$$

$$x_p^i(t) = x_p^i(t-1) + v_p^i(t) \tag{3.8}$$

Equation 3.7, used to update the velocity of a particle, can suffer from a process called *stagnation*, [102]. When a particle's position coincides with the personal and global best the particle's velocity will become zero, all other particles will be drawn towards it. If no other particle improves $gbest$ on the way towards the current particle this may lead to premature convergence, where no further improvement can be made. Eberhart and Shi [43], experiment with the use of constriction factor, k, and inertia weight, ω, to help overcome this limitation. Inclusion of theses factors formalises the up-to-date PSO approach, named Standard Particle Swarm Optimisation (SPSO), [20, 27]. SPSO defined an improved velocity update function formulated by Clerc [27] in Equation 3.9 below:

$$\begin{aligned} v_p^i(t) = k \cdot [\omega \cdot v_p^i(t-1) &+ \vartheta_p \cdot rand() \cdot (pbest_p^i - x_p^i(t-1)) \\ &+ \vartheta_l \cdot rand() \cdot (lbest^i - x_p^i(t-1))] \end{aligned} \tag{3.9}$$

Initially, Clerc [27] replaces the reliance on $gbest$ in TPSOs velocity update strategy, by the particles local best position, $lbest$. This local best is drawn from the n neighbours of each particle in terms of its index within the swarm, and will be discussed further in the following sections. Then the multiplication by a factor of 2 is replaced by two acceleration constraints, ϑ_p and ϑ_l. These acceleration constants are used to balance the learning factors, influencing the ability of the swarm to draw particles towards their personal or local best. Setting these values along with the values for the constriction factor, k, and inertia weight, ω, will be discussed in the following sections.

3.4.1.1 Calculating Inertial Weight, ω, for SPSO

The inertia weight, ω, in SPSO was added to the velocity update in order to replace the need to clamp particles velocities by v_{max}, used in TPSO. Typically defining a good v_{max} for all dimensions is crucial for performance. Large v_{max} is required to allow proper exploration of large dimensions, however, a small v_{max} is required to prevent explosion-like behaviours in smaller dimensions, [82] . By dynamically adapting the value of ω, the swarms velocity can be constrained correctly for both situations.

Small values of ω result in more rapid convergence, but to a sub-optimal local minima, whereas higher values prevent divergence, [43]. While running the SPSO, ω is adapted over time. Initially higher values are used to encourage exploration and prevent divergence, whereas lower values are used towards the final iterations to increase the rate of convergence, only when the swarm is more confident of its optima, therefore avoiding local minima. In SPSO, ω decreases linearly, [102]. Equation 3.10 defines ω between two limits, ω_{min} and ω_{max}, from the total number of iteration, T, and current iteration, t, [102]. The choice of ω_{min} and ω_{max} are some what arbitrary, however, Bratton and Kennedy [20] recommended setting $\omega_{max} > 1$ and $\omega_{min} < 1$. In this work we set $\omega_{max} = 1.2$ and $\omega_{min} = 0.4$.

$$\omega = \omega_{max} - \frac{\omega_{max} - \omega_{min}}{T} \cdot t \tag{3.10}$$

3.4.1.2 Calculating Constriction Factor, k, and Acceleration Constants, ϑ_p and ϑ_l, For SPSO

Now we define how to set the constriction factor, k, in Equation 3.9. This constriction factor is used to dampen the rate of change in velocity, calculated in Equation 3.11 below, [43]:

$$k = \frac{2}{|2 - \vartheta - \sqrt{\vartheta^2 - 4\vartheta}|} \tag{3.11}$$

where

$$\vartheta = \vartheta_p + \vartheta_l \tag{3.12}$$

Bratton and Kennedy [20] explain that SPSO would spiral around a solution with no guarantee of convergence when $\vartheta < 4$. Whereas, for $\vartheta > 4$, convergence is quick and guaranteed. This condition on k, restricts the feasible values for ϑ_p and ϑ_l, in Equation 3.12. Whilst it is possible to weight the influence of personal and local acceleration constants differently, for simplicity these are kept equal. If these factors are equal it is not obvious what ratio their weighting should take, this likely being problem specific, [20]. For instance giving more weight to the personal factor may encourage premature convergence in certain cases. Eberhart and Shi [43] suggest using $\vartheta = 4.1$, which implies, via Equation 3.12, that $\vartheta_p, \vartheta_l = 2.05$ and $k \approx 0.72984$, for good quality convergence in SPSO.

3.4.1.3 Wrapping up SPSO

Finally, the notion of "Letting the particle fly" was introduced in SPSO, [20, 43], removing boundary conditions on the volume. This was designed to remove the tendency of TPSO to perform better when a global optimum is close to the centre of the search space. Rather than clipping particles to the search space boundary and artificially adapting their velocity vectors, a particle is allowed to leave the search space, in the knowledge that it will eventually be drawn back inside by the swarm. To facilitate this behaviour, when a particle leaves the search space it's fitness value, $f(\mathbf{x}_p(t))$, is no longer evaluated, instead a large value is given to discourage further exploration. SPSO forms a simpler algorithm structure than TPSO, as is outlined in Figure 3.10. Zhou and Tan [176] explain that SPSO has been shown to return better results than TPSO, whilst retaining much of the simplicity of TPSO.

Algorithm 2 Standard Particle Swarm Optimisation

procedure SPSO($X(t)$)
  $t \leftarrow 0$
  for $p \in X(t)$ **do** ▷ Initialise the swarm
    $Initialise\ \mathbf{x}_p(t),\ \mathbf{v}_p(t)\ and\ pbest_p$
    $Evaluate\ f(\mathbf{x}_p(t))$
  end for
  while $t < tmax$ **do** ▷ Iterative refinement
    $t \leftarrow t + 1$
    for $p \in X(t)$ **do**
      $Find\ lbest$
      $Calculate\ \mathbf{v}_p(t)$
      $Update\ \mathbf{x}_p(t)$
      $Evaluate\ f(\mathbf{x}_p(t))$
      $Update\ pbest_p$
    end for
  end while
  return $gbest$ ▷ find and return the optimal solution
end procedure

Figure 3.10: Standard Particle Swarm Optimisation

3.4.2 Hybrid Mutation

Although SPSO, improves the initial performance of the swarm, it can still suffer from stagnation in the later stages when reaching the near-optimal solution. Observe the case when a particle's position aligns with the local best, this particle can only move away from its current position if its inertia factor and velocity are different from zero. All other particles will be drawn towards this point and if they all have low velocity they will stop when they catch up, causing premature convergence. This fact means that SPSO cannot guarantee convergence to any minima, only convergence to the best-found solution of the swarm so far. In practice using sufficient particles and iterations alleviates this problem and results are normally close to the optimal solution, all be it that they may not be exact via this method, [162].

In order to overcome this stagnation phenomenon, hybrid methods of Esmin et al. [45], Wang et al. [162] and Zhou and Tan [177] have suggested the addition of a *mutation* operation into SPSO, the likes of which are commonly found in the GA. This operation allows the particles

to avoid local minima and stagnation by randomly choosing a particle and moving it to a different position in the search space thus 're-energising' the search. The mutation operation defined by Ling et al. [102] in Equation 3.13, is applied to a randomly selected particle's position, $\mathbf{x}_p(t)$.

$$\mathbf{x}_p(t) = \mathbf{x}_p(t) + \eta \tag{3.13}$$

Where η, defined in Equation 3.14, is a randomly generated translation that is confined to the search space bounds $[\mathbf{X}_{min}, \mathbf{X}_{max}]$:

$$\eta \in [-0.1 \cdot (\mathbf{X}_{max} - \mathbf{X}_{min}), 0.1 \cdot (\mathbf{X}_{max} - \mathbf{X}_{min})] \tag{3.14}$$

3.4.3 Global Local Nearest PSO

Pongchairerks and Kachitvichyanukul [128], highlight the importance of the social structures on the performance of PSO. In SPSO, personal and local forces control the velocity of a particle, Equation 3.9. Personal forces enhance the local exploration of the particle, pulling it in the direction of its own personal best. Whereas local forces enhance the particles global exploration property, pulling particles out of local minima towards the true global optimum. A natural extension of the velocity update equation is to include multiple social structures namely; personal, global, local and nearest neighbour forces. This leads to a straightforward extension of the velocity update equation, proposed by Pongchairerks and Kachitvichyanukul [128], in Equation 3.15 below:

$$\begin{aligned} v_p^i(t) = \omega \cdot v_p^i(t-1) &+ \vartheta_p \cdot rand() \cdot (pbest_p^i - x_p^i(t-1)) \\ &+ \vartheta_g \cdot rand() \cdot (gbest^i - x_p^i(t-1)) \\ &+ \vartheta_l \cdot rand() \cdot (lbest^i - x_p^i(t-1)) \\ &+ \vartheta_n \cdot rand() \cdot (nbest^i - x_p^i(t-1)) \end{aligned} \tag{3.15}$$

Analogously to TPSO, $gbest$ is defined as the global best position out of all particles within the swarm. Local force, $lbest$, is computed by finding the particles best position within its n immediate neighbours. In this instance neighbours are not related by distance inside the search space, but by their indices within the swarm. Hence, for the i^{th} particle, $lbest$ is the best position of its n neighbours, $p_{n-i/2}, p_{n+1-i/2} \ldots p_{n+i/2}$.

The nearest neighbour force, $nbest$, is the equivalent to the local force, accept this time based on the n neighbours that are spatially closest to the particle. These new forces follow a ring topology, as opposed to the star structure of the global best topology, therefore, local and nearest neighbourhood forces slowly propagate to the surrounding particles.

In Equation 3.15, ϑ_p, ϑ_g, ϑ_l and ϑ_n are the personal, global, local and nearest neighbour acceleration constants, balancing the influence of learning factors. Setting $\vartheta_p, \vartheta_g = 2$ and $\vartheta_l, \vartheta_n = 0$, results in the TPSO presented in Equation 3.7. Setting $\vartheta_p, \vartheta_l = 2.05$ and $\vartheta_g, \vartheta_n = 0$, defines the SPSO presented in Equation 3.9. Setting $\vartheta_p, \vartheta_g, \vartheta_l, \vartheta_n = 1$, results in Pongchairerks and Kachitvichyanukul [128], global local nearest PSO (GLN-PSO) structure. Results presented in their work shows that the inclusion these extra forces within GLN-PSO outperforms TPSO and SPSO.

Choosing the optimal acceleration constants for the best results is still an open problem. It is not clear the influence of each learning factor on the effectiveness of the swarm. Pongchairerks and Kachitvichyanukul [128], speculate that the inclusion of the local and neighbourhood terms allow particles higher opportunity to explore spaces around $gbest$, $lbest$ and $nbest$ separately, before converging rapidly towards the end of the optimisation when these forces align.

3.4.4 Synchronous, Asynchronous and Parallel PSO

Despite PSO having the advantage of being derivative free, it can still require large amounts of computation to solve. Each particle must evaluate the fitness function multiple times in order to find the optimal solution, with larger swarm sizes or high dimensionality further increasing the computational demand. Fortunately, PSOs structure allows for an embarrassingly parallel algorithm, highly suited to a GPU implementation. In the most naive implementation each particle can be processed independently. Since the evaluation of the particles fitness functions tends to be the most costly part of optimisation, implementing this within a multi-threaded or GPU based environment can dramatically increase the speed of optimisation. Making parallel PSO a powerful and practical tool for search and optimisation. Synchronous and asynchronous PSO are two different strategies for parallelising SPSO previously presented.

In synchronous SPSO, [116, 176], first all the fitness values are computed in parallel for the particles current positions within the swarm. Following this $lbest$ is found by accessing global memory where the particles are stored, finally all particles velocities and positions are updated, again in parallel. Hence, there are specific synchronisation points within the algorithm where the swarm waits on all the particles to finish and complete a certain process. This form of iteration, being perhaps the most simple to parallelize, requires each particle be represented by a work-group, where each work-item corresponds to one dimension of the problem. Therefore, each work-item will compute the value of the fitness function or velocity for a single dimension of its work-groups assigned particle. Two kernels can then be iterated; an evaluation kernel, computing the fitness function for each particle, and an update kernel finding $lbest$ and computing the velocity vector. These two kernels must be iteratively applied to complete the optimisation. Each kernel must load and store data in global memory at the end of its execution to allow for inter-kernel data sharing. While being completely scalable to any swarm size, frequent reads and writes to global memory limit the performance of this approach. For more information on implementing SPSO in a synchronous manner on the GPU please refer to Mussi et al. [116] comprehensive coverage of the subject, they include ways to achieve optimal performance by coalesced memory access within each of the kernels. Figure 3.11 displays the pseudo-code for the host device to execute in order to run SPSO in a synchronous manner, note here that after each kernel call the host must wait on its completion before calling the next.

Algorithm 3 Synchronous Particle Swarm Optimisation

procedure SYNCHRONOUSSPSO($X(t)$)
  Initialise $X(t)$, *set the positions of each particle*
  Transfer $X(t)$ *to the GPU via a buffer*
  while $t < tmax$ **do** ▷ Iterative refinement
    $t \leftarrow t + 1$
    Run Evaluation Kernel
    Run Update Kernel
  end while
  return $gbest$ ▷ find and return the optimal solution
end procedure

Figure 3.11: Host code for synchronous Particle Swarm Optimisation

Asynchronous PSO, [21, 39, 88, 117], removes the need for specific synchronisation points, allowing the velocity forces to be updated immediately after each particles fitness function is evaluated. Without these synchronisation points each particle is free to evolve independently of the others, without waiting for the most up-to-date results. Therefore, PSO can be implemented inside a single kernel where all of its personal data can be stored in fast access local and private memory, avoiding the use of the global memory space. In practice, global memory does need to be used to store current particles personal best positions, $pbest$, allowing neighbouring particles to calculate $lbest$. However, $pbest$ is only updated in global memory when a new personal best is found.

As opposed to synchronous PSO, all particles in the swarm execute simultaneously, requiring only a single kernel to be scheduled. This kernel will evaluate the particle's fitness function, then immediately find $lbest$, no matter if the particles surrounding it have finished computing and updating their $pbest$, finally the kernel will then update the velocity vector. Mussi et al. [117], argue that independent evolution of the particles is more "biologically plausible", making the swarm more reactive the newly discovered minima. Their results have shown asynchronous PSO has the ability to achieve faster execution times, while retaining the same levels of numerical accuracy. However, the use of local memory limits the total swarm size, so asynchronous PSO can only obtain highly accurate results for problems of smaller dimensions, [39]. We refer the interest reader to Cagnoni et al. [21] comprehensive description of implementing asynchronous SPSO in OpenCL, for the full definition of the asynchronous kernel. Figure 3.12 displays the pseudo-code for the host device to execute in order to run SPSO in an asynchronous manner.

Algorithm 4 Asynchronous Particle Swarm Optimisation

procedure ASYNCHRONOUSSPSO($X(t)$)
  Initialise $X(t)$, *set the positions of each particle*
  Transfer $X(t)$ *to the GPU via a buffer*
  while $t < tmax$ **do** ▷ Iterative refinement
    $t \leftarrow t + 1$
    Run Asynchronous Kernel
  end while
  return $gbest$ ▷ find and return the optimal solution
end procedure

Figure 3.12: Host code for asynchronous Particle Swarm Optimisation

3.5 Summary

This chapter has introduced the fundamental techniques that are employed throughout the MVS framework presented in the following chapters. It has outlined the following key principles to our approach:

- Classically, the camera projection matrix, P, can be used to project 3D geometry into 2D imagery.
- Defining the surface in 3D space can be done through the use of a regional based NCC measure forming the photo-consistency metric. This is robust to illumination changes, but can fail in textureless areas.
- PSO is a derivative free algorithm for optimisation of an energy function, less likely to get stuck in local minima. Its particle-based sub-problems are ideal for GPGPU computation. This work intends to use PSO to overcome some of the limitations of patch-based methods relying on derivative based approaches.

4

Particle Swarm Optimisation for High Fidelity Reconstruction

This chapter presents work on our preliminary MVS 3D reconstruction framework. Aspects of its design are discussed at length followed by a full analysis of both its quality and speed. The framework adopts a similar strategy to the state of the art patch based system, PMVS [53], described in Section 2.4.4. However, is designed to overcome the limitations imposed on PMVS over sparse or low textured regions due to its reliance on feature based initialisation and gradient descent schemes. Feature based initialisation is required by PMVS to enable gradient descent optimisation of a patch's parameters that does not get stuck in local optima. Whilst achieving high accuracy on both high and low frequency details, PMVS can fail to adequately fill large textureless regions or sparsely viewed surfaces where feature matching and gradient descent fail. To account for these shortcomings a novel optimisation of the photo-consistency function via PSO is introduced. Allowing for feature free patch initialisation and global refinement of its parameters, whilst retaining the ability to achieve high fidelity patch optimisation.

Our adapted patch based PSO approach is presented and qualitative and quantitative results show its viability against the gradient descent scheme employed by PMVS. It is shown that optimisation of a patch's position and normal parameters via our PSO based algorithm returns patches that are of comparable accuracy whilst moving toward denser reconstructions. Whilst this initial method produces good results for textured data, it proves to be susceptible to errors in textureless regions. As discussed in Section 3.2, these types of errors arise from the use of the NCC as a photo-consistency measure. Whereby the optimal parameters achieving the smallest NCC score no longer relate to the optimal patch parameters aligning with the surface. As discussed in Section 2.4.4, PMVS removes erroneous patches, whose optimal NCC score does not align with the true surface, in a filtering step leading to sparser patch clouds and holes in textureless regions. However, our preliminary approach chooses to leave these erroneous patches, forming the basis on which the system is later refined to better deal with low textured objects.

Whist our preliminary PSO results show promise in terms of a feature free initialisation and highly accurate patch refinement, results show it to be limited to fairly well textured regions, patches refined in low textured regions are often erroneous. These limitations are discussed at length, motivating Chapter 5, in which the photo-consistency measure is updated to account for texture detail and improve the quality of the metric. This allows for a more robust matching and guiding constraint increasing the effectiveness of the photo-consistency score in textureless areas. In Chapter 6 the entire framework is adapted into a coarse-to-fine, multi-resolution approach based on the approximation and detail coefficients of the wavelet transform. Using a coarse-to-fine technique improves the fidelity of reconstruction and allows creation and filtering of denser patch clouds in textureless regions.

This chapter begins, in Section 4.1, with an overview of the proposed PSO framework, following a patch based surface reconstruction paradigm. Detailed descriptions of the key components of this approach are presented. Section 4.2 then describes synthetic and real world datasets used to evaluate our algorithm's performance. Section 4.3, compares our PSO scheme against PMVS, [53], presenting both qualitative and quantitative reconstruction results. Its quality and limitations against PMVS are analysed to assess its potential for improved MVS. A discussion of these

results, presenting theory on the causes and how to overcome limitations of the current system are presented in Section 4.4. Finally, Section 4.5 concludes with a summary of methods and results presented thus far.

4.1 PSO Patch Based Framework

Patch based reconstruction methods have been and still are one of the top performing approaches to 3D reconstruction to date, [79]. Their local approach to refining the position and orientation of a patch, free of global surface minimisation and independent of smoothness or regularisation, make patch based methods extremely powerful in the recovery of fine grained detail over an object's surface. However, these approaches still fail to densely and faithfully reconstruct textureless or highly occluded surface regions; thus, though performing well under lab conditions, they deteriorate in industrial or real world situations.

As discussed in Section 2.4.4, current patch based methods, such as PMVS, generate high quality patch clouds, however, they produce holes or sparse coverage in textureless and occluded regions, where both feature matching and gradient descent fail. These then require expensive active contour energy minimisation techniques to interpolate and fill a dense high fidelity surface. Being designed primarily to work on reasonable size image sets, this phenomenon is worsened when the size of the input image set is reduced, akin to the textureless effect this decreases the number of feature matches possible, therefore lowing the number of seed points that patch expansion can grow from, in turn reducing surface coverage, further deteriorating reconstruction results and producing holes.

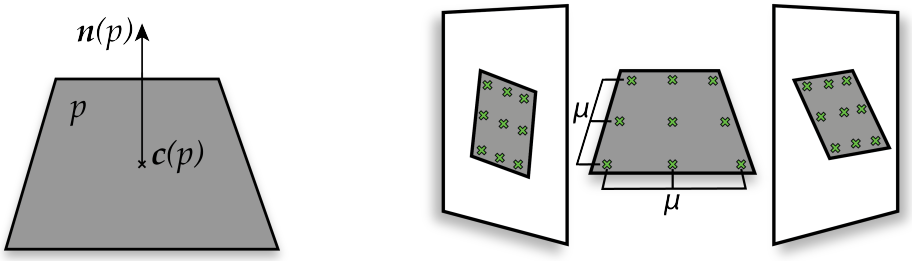
Such shortcomings hinder the adaptation of these methods to general purpose or industrial applications, where object surfaces are often captured in sparse image sets and contain low texture. The aim of this work is to adapt such patch based methods to better fill textureless regions whilst producing overall a high fidelity, dense patch cloud, expressing both high and low frequency surface details to the best of its ability. Therefore, reducing the need to revert to expensive energy minimisation techniques, requiring high levels of computation and over smoothing the highest frequency surface details. This section will define our preliminary methodology to overcome such limitations, removing the necessity for feature based matching and gradient descent.

4.1.1 Overview

As input into the reconstruction process our algorithm requires a calibrated set of input images, where both intrinsic and extrinsic camera settings are known (Section 3.1). Once supplied with camera parameters our approach then proceeds in the patch based manner below, this process will be described fully throughout the following sections:

1. Extract silhouettes throughout the image set, where foreground pixels represents the object to be reconstructed (Section 4.1.2).
2. Extract the Visual Hull (Section 4.1.2).
3. For each foreground pixel, check that it's valid for this pixel to create a patch. Validity is checked by observing the viewing angle the pixel makes with the visual hull (Section 4.1.2).
4. Each valid pixel will create a patch, before doing so initialise its corresponding patch parameter search space. This is done by intersecting the pixel's viewing ray with the visual hull (Section 4.1.3).
5. Run PSO on the position and normal parameters for each pixel's patch, refining the patch to sit accurately on the object's surface (Section 4.1.3).
6. Combine all patches into a single patch cloud.

As with all patch based approaches, a patch, p, consists of a centre, $\mathbf{c}(p)$, and normal, $\mathbf{n}(p)$, as shown in Figure 4.1a. It is assumed that, given the segmentation of silhouettes and a visual hull (described in Section 4.1.2), each view will create a set of patches that must successfully back project onto the object's surface. That is to say, for each view, each foreground pixel inside the silhouette should create a patch, where the patch reflects the subsection of the object's surface behind that pixel.



(a) A patch, p, with centre, $\mathbf{c}(p)$, and normal, $\mathbf{n}(p)$

(b) μxμ samples are overlaid on patch, p, and sampled via their projection into the image set

Figure 4.1: The basic patch set-up and sampling

In order to create a dense patch cloud, in this work we follow a novel patch-per-pixel approach. Similarly to depth map approaches [140], every foreground pixel, inside the segmented silhouettes, is assumed to create and view a surface patch. Hence each silhouette pixel is used to initialise a patch and construct a search space on which that patch can be refined. Refinement is necessary to move the patch from its initialisation to a new position and orientation that better reflects the object's surface underneath each pixel. In fact, before creating a patch underneath each silhouette pixel, we first check whether the visual hull creates too sharp a viewing angle with the pixel. Patches are only created for silhouette pixels whose viewing angle to the visual hull is within fifty degrees. The image whose pixel created the patch will become its reference view, $R(p)$. The combination of each of the view's patches should densely define a patch cloud covering all visible surfaces of the object.

The objective of our patch-based system is to find the patch cloud that is most consistent with the input views, therefore, each patch undergoes a refinement procedure on its position and normal. The purpose of this refinement is to allow the patch to more accurately represent a planar region of the object's surface, underneath its corresponding pixel, through the improvement of its photo-consistent matching score, Equation 3.6.

Refining a patch's position and normal can lead to an extremely large and costly search space. In order to reduce this space a patch's centre, $\mathbf{c}(p)$, is confined to sit at some positive distance, λ, along the viewing ray generated by its pixel. Its normal, $\mathbf{n}(p)$, is then cast into spherical coordinates, θ and ϕ, and confined to a direction such that it will not become back facing to the reference view that generated the patch.

To test if the current patch, with parameters λ, θ and ϕ, successfully back projects onto the object's surface, the NCC score, Equation 3.5, is used as a photo-consistency measure. Before taking the NCC measure, a set of un-occluded views of the patch, $V(p)$, are created by observing its current position and orientation relative to its visibility throughout the image set, this was depicted in Figure 3.1b, and is simplified to basic depth map testing as in [53]. The photo-consistency measure is then computed by averaging the NCC score of the reference view with every other view that the patch is visible in, $|V(p)/R(p)|$. In this chapter we use the classical NCC based photo-consistency score previously defined in Section 3.2 Equation 3.6.

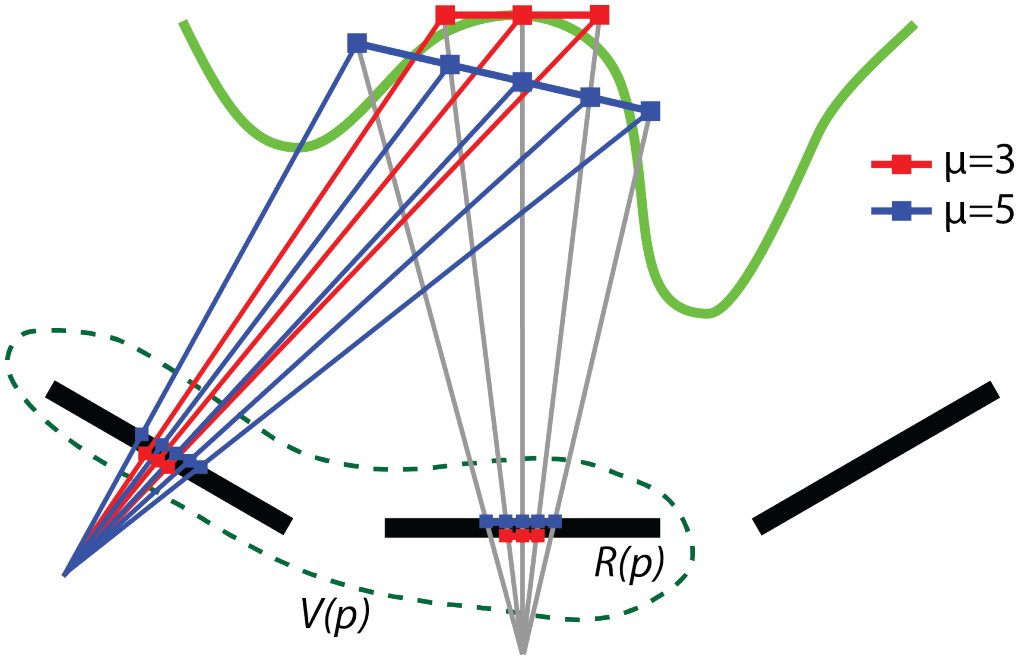


Figure 4.2: Effect of sample size, μ, on patch position and orientation

Given the set of views that observe a patch, its projection, Section 3.1, into each view is sampled, via bilinear interpolation, at regular locations over its surface, see Figure 4.1b and 4.2. Akin to Furukawa and Ponce [53] the patch is scaled such that it produces a μxμ pixel window in its reference view, $R(p)$, which is the image in which the pixel created the patch, this is shown in Figure 4.2. As discussed earlier the choice of μ determines the quality of reconstruction. Large μ are robust in textureless regions, while producing a smoothing effect on the patch cloud as the patch size increases. This is illustrated in Figure 4.2, where two refined patches, created for the same pixel in the reference view, $R(p)$, but with different sample sizes, have different position and orientation. Larger patches are also more expensive to compute, requiring more samples over the patch's surface. Small μ are capable of reconstructing high frequency surface details at lower cost. These express a smaller surface area and sampling size, however, small patches are unstable in textureless regions. Akin to Furukawa and Ponce [53] findings, through experimentation, we found that using a 5x5 matching window offered the best accuracy to robustness and time trade off.

If a patch has a high photo-consistency score then it can be classed as the correct position and orientation for the patch, as the sampled colours are consistent for the surface it describes. If the photo-consistency score is low, its sampled colour is inconsistent throughout the viewing set, and

the patch must be repositioned and reoriented to better fit the surface. The task of patch refinement is to find the optimal parameters λ, θ and ϕ, for the centre, $\mathbf{c}(p)$ and normal, $\mathbf{n}(p)$, respectively, such that it maximises the photo-consistency score. In practice this is cast to a minimisation problem by taking $1 - NCC(p)$, as in Equation 3.6.

Rather than initialise a patch-per-pixel as in our approach, PMVS [53], relies on feature matching to accurately triangulate a sparse set of patch positions, and expansion from a robustly determined neighbour in order to fill the patch cloud, see Section 2.4.4. This in turn defines a patch's reference view, $R(p)$, and centre, $\mathbf{c}(p)$, at depth, λ, for all newly generated feature matched patches. In the case of feature based initialisation, its normal, $\mathbf{n}(p)$, is still unknown, so is initialised to the inverse of the reference view's optical axis. Gradient descent on the photo-consistency function is then employed to jointly extract the locally optimum parameters λ, θ and ϕ. The assumption being that the initialisation was 'close to optimal' to avoid gradient descent becoming trapped in local minima. This means seed patches, which will later be used to expand, can only be created via well matched features, which do not occur often in low textured or sparsely viewed circumstances.

This work proposes the removal of feature based matching, which can only be used in well-textured regions, and the replacement of gradient descent with PSO. PSO can be utilised to find the globally optimal photo-consistent solution for any patch, without requiring a robust feature triangulation as in PMVS. PSO only requires the search space for $\mathbf{c}(p)$ and $\mathbf{n}(p)$ be defined. The optimal parameters will then be extracted without the need for robust initialisation or the risk of getting stuck in local minima. PSO is implemented on the GPU to offset its computational costs, this will be described in Section 4.1.3.

In order to define a sensible depth range for PSO to explore, the visual hull is extracted, see Section 4.1.2, and the intersection of the viewing ray on which the patch is confined to sit is found. This intersection defines a subsection of the viewing ray for which PSO will search. Figure 4.3 displays a two dimensional representation of our optimisation process, generating the optimal patch for one foreground pixel of an input view. The intersection (dark blue) of the viewing ray (light blue) with the visual hull (green) is used to define the depth search range. Inside PSO, each particle represents different patch configurations, searching for the op-

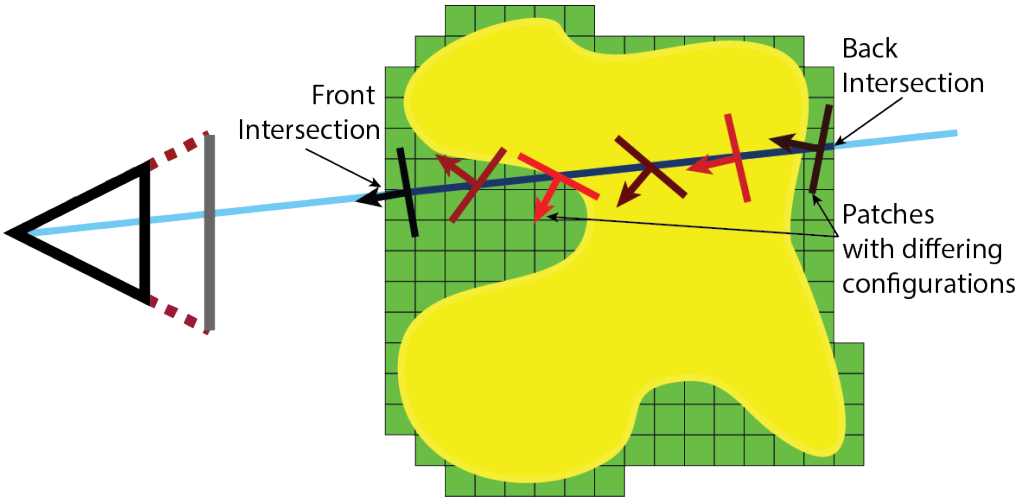


Figure 4.3: Particles in PSO represent different patch configurations searching for the optimal parameters along the viewing ray

timal solution. In the diagram above, the lighter red the patch the smaller the photo-consistency value for that configuration. Once PSO has run, the particle with minimal personal best score is extracted as the optimal patch configuration and its parameters are used to construct the patch for this pixel. PSO is run, creating a patch, for each silhouette pixel that passed the viewing angle test, thus once all pixels are processed the most photo-consistent patch cloud is returned.

Due to noise in the photo-consistency function and spurious matching, small valued patches do not necessarily sit close to the correct parameters for the patch, however, as with all other MVS systems it is assumed that the smallest value will always align properly with the true surface, [138], which is yellow in the diagram above.

4.1.2 Generating the Visual Hull

Our method utilises the visual hull to define a depth interval for PSO to search for each patch's viewing ray. Extracting silhouettes in the input views allows for the generation of the visual hull. Often silhouette extraction is seen as a drawback to reconstruction techniques, requiring an accurate segmentation to allow for its inclusion as an energy term in minimisation frameworks, [138]. Accurate segmentation is required as it is assumed that the occluding contours of the silhouette must project exactly onto the surface of the object thus serving as a strict driving force, deforming the surface to match the occluding contours of the silhouettes.

In our approach, silhouettes are used to form the visual hull, which in turn is only used to restrict the search space of the problem. Therefore the accuracy of this segmentation is significantly less important. Specifying a bounding volume, using a simplex algorithm finding the convex hull of the cameras, is equally valid, [10]. However, it is seen worthwhile to generate silhouettes and thus the visual hull, as this can be computed efficiently and further reduces the search space as opposed to the bounding volume.

Extraction of silhouettes, binary labelling pixels as foreground and background, in our case, is done via simple thresholding and histogram back projection, [7]. As object datasets are normally taken against a solid coloured background, this approach proved to work effectively. In cases where the background is not a solid colour, standard geodesic active contour methods can be utilised or Campbell et al. [23] presents an effective method for silhouette extraction specifically designed for MVS problems, although this method is more computationally demanding.

Once the silhouettes have been extracted they are passed through a GPU pipeline proposed in this work, generating a volumetric octree representation of the visual hull. Each voxel will have a label corresponding to its surface classification, as discussed in Section 2.3.1. The pipeline created in this body of work is based on the classical approach presented by Szeliski [150], however we extend this to a GPU algorithm below. Whereby the silhouettes are transformed via the half distance transform, mapping pixels to distance values. These distance values represent the largest square that can be drawn from the current pixel, such that the square does not leave or enter the silhouette (depending on whether the pixel was originally inside or outside the silhouette). The pixel is always assumed to sit in the top right corner of this square, and an example of this can be seen in Figure 4.4.

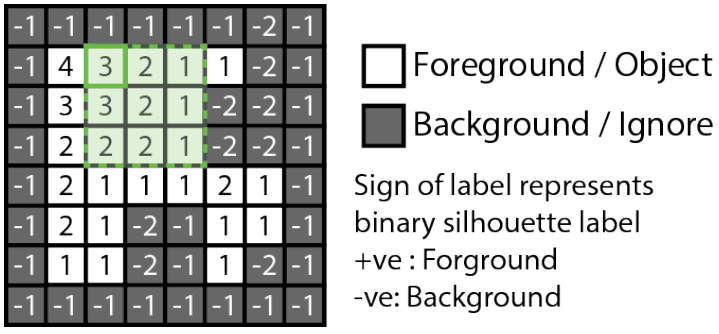


Figure 4.4: Example of assigning the half distance transform distance labels

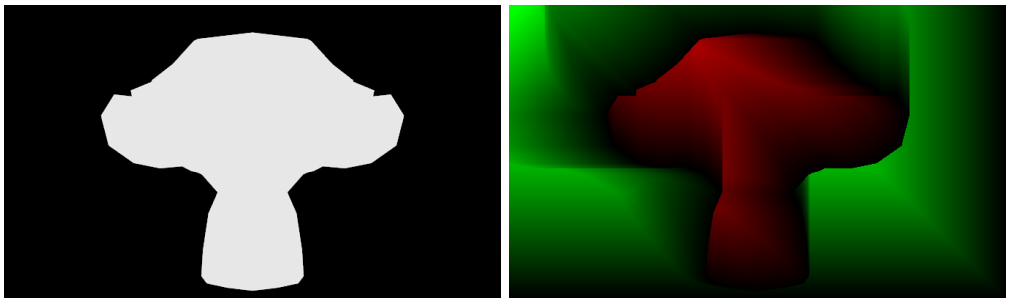


Figure 4.5: The half distance transform (right) applied to an input silhouette (left). Red colours define distances inside the silhouette and green colours define external background distances

An example of the half distance transform on an input silhouette is shown in Figure 4.5. Red colours represent positive distances inside the silhouette, green represents negative distances outside the surface. The darker the colour, the smaller the distance to the inversion of the foreground/background labelling of the silhouette.

Our implementation computes the half distance transform using a GPU approach via two directional sweeping kernels, one horizontal and one vertical. Rather than using two-dimensional buffers to store images, OpenCL allows images to be stored and accessed more efficiently in $image2d$ objects, [60]. The limitation to using $image2d$ objects is that they currently can be either read only or write only, reading and writing to the same $image2d$ object is not supported. Therefore, a double buffered approach to the distance transform is utilised, using a source $image2d$ object containing the un-transformed image, and a destination $image2d$ object, where the transformed image will be written to.

First, the image is broken into work-groups, assigning one work-group per row of the input image, as described in Section 3.3.1. This means the NDRange of the problem will be one dimensional and equal to the number of rows in each image to be transformed. Then the horizontal distance transform kernel, which runs per work-item, sweeps backwards along the row, counting and assigning the distance value based on the distance to the previous pixel with opposing silhouette label.

As the distance values for each pixel relies on knowledge of its neighbour the right, it is hard to completely parallelise this algorithm. Therefore, whilst we can break the image into coarse sub-problems, computing the distance transform for each row independently, we cannot break the transform along a single row into a fine-grained sub-problem per pixel.

Therefore, for each sub-problem, which determines the distance transform of a row, we use only the first work-item in the work-group to perform this scan, whilst the others lay idle.

For increased memory performance each work-group loads a row of the image into local memory, as discussed in Section 3.3.3. The more work-items there are the faster the memory can be copied into the work-group, however, the more idle work-items there will be during the distance transform, as only the first one of the group will do any computation. Balancing the number of work-items is therefore dependant on the number of work-items each core of the GPU can evaluate in parallel.

To compute the distance transform the first work-item in the work-group scans over the row in local memory, writing the result back to the same local memory and finally to the global destination memory when complete. The pseudo-code for this kernel is given in Figure 4.6 below:

Algorithm 5 Horizontal Distance Transform Kernel

procedure HORZDT($image2D\ src$, $image2D\ dst$, $local\ int\ *localRow$)

$row = get_global_id(0)$ ▷ Find the row this work-group works on

$CopyGlobalToLocalMem(row, src, localRow)$
$Barrier()$ ▷ Wait for all work-items to finish copying

if $get_local_id(0) = 0$ **then** ▷ Select first work-item in work-group
  $previousBackground = src.width$
  $previousForeground = src.width$
  for $i = src.width - 1$ *to* 0 **do** ▷ Scan row from right to left
    if $localRow[i] < 0$ **then** ▷ At background pixel
      $previousBackground = i$ ▷ Count dist since foreground
      $localRow[i] = -(previousForeground - i)$
    else ▷ At foreground pixel
      $previousForeground = i$ ▷ Count dist since background
      $localRow[i] = previousBackground - i$
    end if
  end for
end if

$Barrier()$ ▷ Wait for first work-item to finish transform
$CopyLocalToGlobalMem(row, localRow, dst)$
end procedure

Figure 4.6: Horizontal Distance Transform pseudo-code

Algorithm 6 Vertical Distance Transform Kernel

```
procedure VERTDT(image2D src, image2D dst, local int *localCol)

    col = get_global_id(0)                    ▷ Find the column used by work-group

    CopyGlobalToLocalMem(col, src, localCol)
    Barrier()                                 ▷ Wait for all work-items to finish copying

    if get_local_id(0) = 0 then               ▷ Select first work-item in work-group
        if localCol[src.height − 1] < 0 then
            localCol[src.height − 1] = −1
        else
            localCol[src.height − 1] = 1
        end if
        for i = src.height − 2 to 0 do        ▷ Scan column from bottom to top
            if localCol[i] < 0 then                                  ▷ At background pixel
                if localCol[i + 1] ≥ 0 then   ▷ If previous col was foreground
                    localCol[i] = −1
                else                          ▷ Previous col was background, must check row
                    colStep = localCol[i + 1]
                    if abs(localCol[i]) ≥ abs(colStep) + 1 then
                        if src[col − colStep, i − colStep] > 0 then
                            localCol[i] = colStep
                        else
                            localCol[i] = colStep − 1
                        end if
                    end if
                end if
            else                                                     ▷ At foreground pixel
                if localCol[i + 1] < 0 then   ▷ If previous col was background
                    localCol[i] = 1
                else                          ▷ Previous col was foreground, must check row
                    colStep = localCol[i + 1]
                    if localCol[i] ≥ colStep + 1 then
                        if src[col + colStep, i + colStep] > 0 then
                            localCol[i] = colStep
                        else
                            localCol[i] = colStep + 1
                        end if
                    end if
                end if
            end if
        end for
    end if

    Barrier()                                 ▷ Wait for first work-item to finish transform
    CopyLocalToGlobalMem(col, localCol, dst)
end procedure
```

Figure 4.7: Vertical Distance Transform pseudo-code

To complete the transform a second sweep is run in the vertical direction with a second kernel, this time breaking the image columns into work-groups. This gives an NDRange equal to the number of columns of the image. Similarly to the horizontal kernel, the vertical kernel sweeps up the columns combining the previously computed horizontal distance, at the current pixel, with the vertical distance transform, again using local memory. Figure 4.7, on the previous page, defines the pseudo-code for such a vertical kernel. This time src will be the result of applying the horizontal distance transform, written to as dst in Figure 4.6.

Once the half distance transform of each of the input silhouettes is found, the labelling of each voxel in the visual hull can be defined. Akin to Szeliski [150] a voxel is projected into each view and the image space bounding box of its projection is found. The distance value for the pixel at the top left of the bounding box is checked against the box's maximum dimension, defining whether the voxel projects inside, outside or partially onto the silhouette. Voxels that lie completely inside the silhouette for all views are labelled as interior nodes and no longer decomposed. Voxels lying completely outside the silhouettes in one or more views are labelled as exterior nodes and are no longer decomposed. Those voxels that lie over both silhouette and background are then labelled as surface voxels and are further refined in the next level of the octree.

Given the speed and compute power of this GPU approach, defining the visual hull in a volumetric way is only limited by the resolution of the volume. As the depth of the octree grows, so does the number of voxels required to represent this level. Increasing the resolution of the volume results in a quadratic increase in the amount of memory required to store the tree. We utilise Laine and Karras [94], sparse voxel octree arrangement specifically designed for GPU memory requirements, allowing for deeper octrees without storing superfluous voxels.

To compute the entire visual hull down to the finest level of the octree, a kernel is invoked for each voxel at the current level. This implies a work-item to voxel mapping, Section 3.3.1. Each kernel traverses the sparse voxel octree from the root down to the current voxel. If it hits a voxel at a higher level that was labelled as entirely inside or outside the volume then the kernel will stop processing. If it does not hit a labelled voxel the kernel will project this voxel into all views, computing its label given the distance maps as above, adding it to the sparse octree structure.

The pseudo-code for this voxel classification kernel is presented in Figure 4.8. The octree object is stored in an unsigned long vector, which can be traversed efficiently with ray casting algorithms, as described by Laine and Karras [94]. The distance transformed silhouettes can be transferred to the kernel via an $image2d_array$ object, a special data structure provided by OpenCL for efficient access to image data within arrays, [60]. The octree position and dimension are stored in a floating-point array with 8 components, OpenCL offers a $float8$ object for this purpose, [60].

Algorithm 7 Voxel Classification Kernel

procedure CLASSIFY($image2D_array\ sils, ulong\ *octree, float8\ octDim$)

$voxel = (get_global_id(0), get_global_id(1), get_global_id(2))$
▷ Collect the voxel index for this work-item
$hitInOrOutsideVoxel = TraverseOctree(octree, voxel)$
▷ Traverse the octree from root to voxel

if $hitInOrOutsideVoxel = true$ **then**
  $return$ ▷ Do not compute voxels the are entirely inside/outside object
else
  $c = inside$ ▷ Start with an inside classification
  for $i = 0\ to\ N^o\ silhouettes$ **do**
    $ibb = ProjectVoxelIntoSilhoutte(voxel, sils[i])$
    ▷ Find image space bound box
    if $ibb.projectsOutsideImage$ **then**
      $c = outside$ ▷ Projects entirely into background for this view
      $break$
    else if $ibb.cornerInsideSilhouette$ **then**
      if $ibb.maxDim > ibb.cornerSilDistValue$ **then**
        $c = surface$ ▷ Partially inside & outside silhouette
      end if
    else if !$ibb.cornerInsideSilhouette$ **then**
      if $ibb.maxDim \leq abs(ibb.cornerSilDistValue)$ **then**
        $c = outside$ ▷ Projects entirely into background for this view
        $break$
      else
        $c = surface$ ▷ Partially inside & outside silhouette
      end if
    end if
  end for
  WriteVoxelToOctree(voxel, c, octree)
end if
end procedure

Figure 4.8: Voxel classification kernel pseudo-code

Figure 4.9 demonstrates the process of refining the visual hull for a dataset presented later in this chapter. First, the image space bounding boxes of the silhouettes are back projected into world space for each input view. Using a simplex algorithm, [10], the convex hull of the intersection of these back projected bounding boxes is found. This defines the bounding volume used as the root of the octree, seen in the top left corner of the figure. Once the distance maps for each silhouette are computed, the sparse voxel octree is refined over multiple levels starting at the root.

We found that 8 levels were sufficient to obtain good quality visual hull representations regardless of the dataset. The level of the octree and size of the object is tightly related to the resolution at which the object is represented. For instance, the size of a voxel in the third level of an octree is significantly smaller for a smaller object than it is for a large object, as these naturally encode different volumes of space. However, when reconstructing larger objects, smaller surface details become less visible and less important, hence by the eighth octree level most low frequency surface information is well recovered regardless of object size, similar to the finding of Szeliski [150].

The reconstruction of the visual hull including the computation of distance maps could be computed, on average, in 1.5 seconds via our GPU algorithm. However, this run time is likely to vary depending on GPU, image resolution and level of octree required. No doubt more optimised kernel functions taking into account better memory coalescing, described in Section 3.3.2, would further reduce the run time; however, as visual hull computation is not the main focus of this work, we leave this open to further analysis.

Once the visual hull has been defined, in order to remove erroneously labelled pixels within the silhouettes, it is then projected into the view set. To do this, the ray-casting algorithm presented by Laine and Karras [94] is used to project a ray through each silhouette pixel and intersect it with the visual hull inside the sparse octree structure. This allows re-labelling of silhouette pixels and creation of depth maps for efficiently looking up the intersection of a pixel's viewing ray with the visual hull. This process ensures that pixels incorrectly labelled as silhouette, via our thresholding method, which do not intersect the visual hull are accounted for and removed producing overall higher quality segmentation.

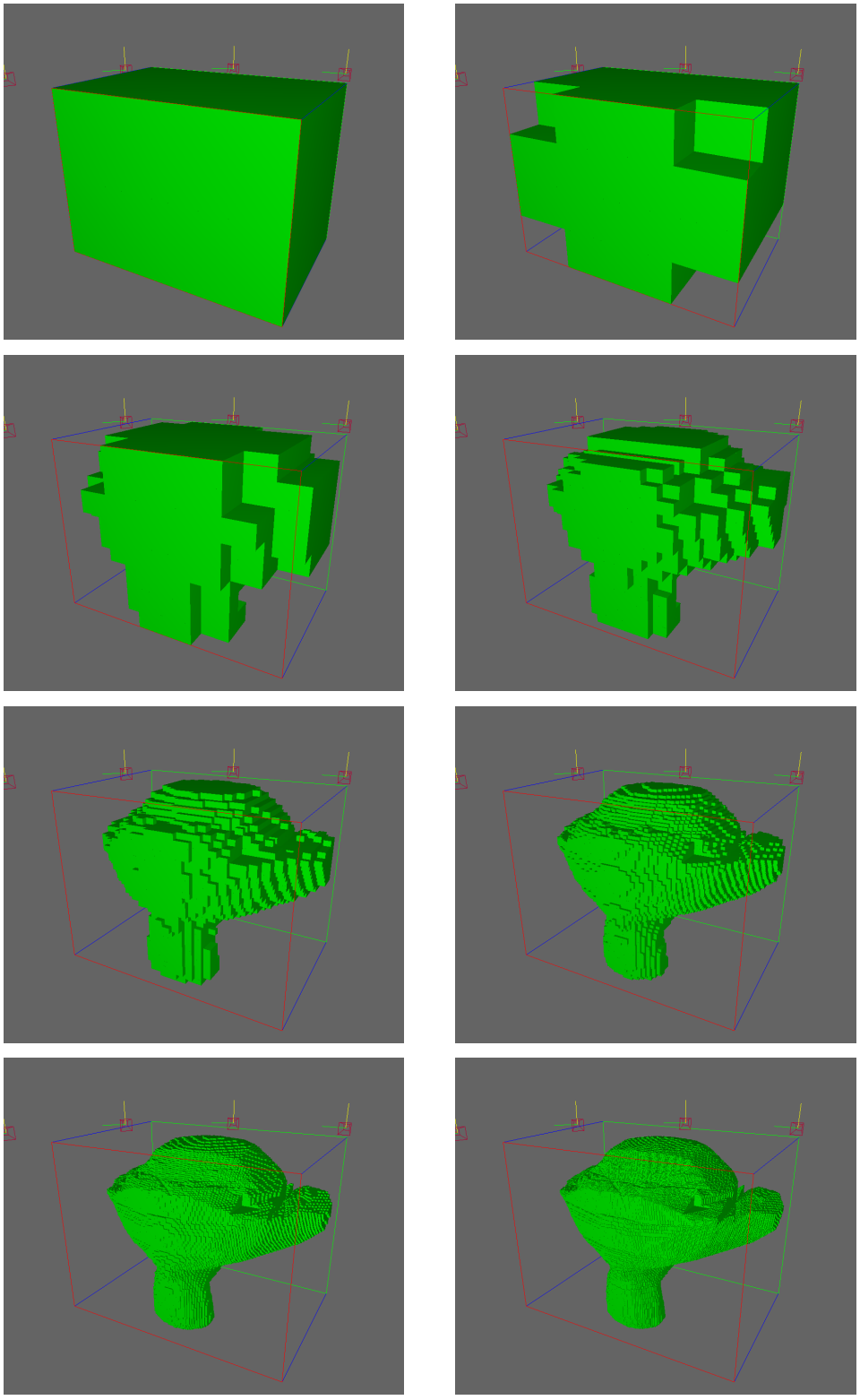


Figure 4.9: The progressive refinement of the visual hull over multiple levels of the octree. The top left figure shows the first level down to the bottom right displaying the eighth level

Front and back depth maps can be created when intersecting each silhouette pixel's viewing ray with the visual hull, in the process described in the previous paragraph. Depth maps are then used to approximate a surface normal for each pixel using the gradient operator. An example of the depth and normal maps created by projecting the visual hull into one of the views is shown in Figure 4.10.

Our adapted PSO algorithm uses the surface normal to calculate the viewing angle of the image to the current surface approximation. The angle created by the viewing ray, sent out through each silhouette pixel's centre, and the normal to the visual hull, stored in the normal map, can be found using the dot product. Only pixels with a viewing angle less than fifty degrees to the surface are allowed to create patches, as pixels with higher viewing angles are likely to have sharp grazing angle to the surface which would be better refined in other views.

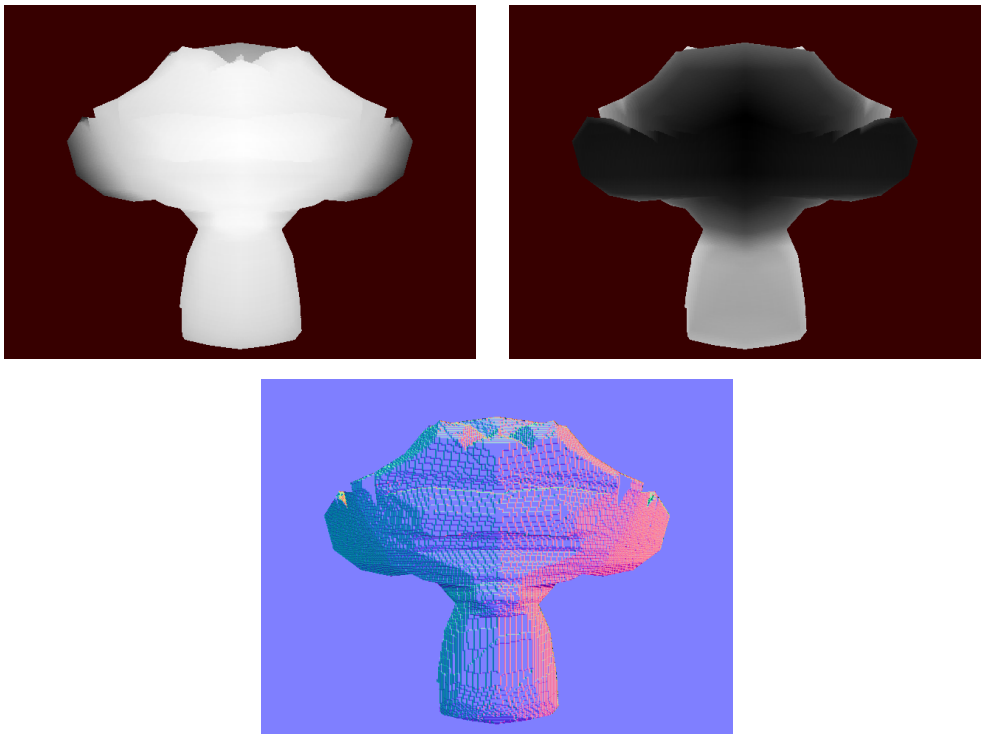


Figure 4.10: The projection of the visual hull into an input view creating the front and back intersection depth maps and normal map respectively. Lighter white colours represent a shallow distance, and darker black colours represent a larger, deeper distance

4.1.3 Patch Refinement by PSO

In the context of MVS the cost function evaluated by each particle inside the PSO will become the per-patch photo-consistency measure, Equation 3.6. As this is computationally expensive to evaluate we implement our PSO algorithm on the GPU, using OpenCL to achieve a faster run time performance. Whilst GPU based PSO is not new, [177], the novelty of our work comes in the adaptation of its implementation to specifically suit a MVS setting. A notable benefit of using PSO over gradient descent is that the computation of the photo-consistency functions derivatives are not required. There is no natural way to express the derivative of the NCC, therefore, gradient-based schemes must revert to central difference to compute these. This in turn is computationally expensive and sensitive to sampling. Instead, as discussed in Section 3.4, PSO offers a stochastic optimisation technique where the swarm evolves based on the local and global knowledge at each particle, adapting well to the large search space and extracting the global, rather than local, optimal patch parameters.

Chen et al. [26] originally proposed the replacement of traditional gradient descent with GLN-PSO, Section 3.4.3, for the optimisation of a patch's parameters inside the framework of PMVS. By using the sparsely matched features as initialisation they were able to restrict the optimisation search space to nearby depth and normal values, thus performing optimisation similarly to gradient descent; extracting the locally optimal patch, based on the feature matched initialisation. Their results proved effective against gradient descent alone, but due to their reliance on feature detection, expansion and filtering steps their approach is limited in the extent it can expand into textureless regions.

We look to build upon Chen et al. [26] work by removing the reliance on feature matching. Instead of using feature matching to create seed patches and expand outwards, for each silhouette pixel we assume an appropriate search space can be defined by finding its intersection with the visual hull. Thus allowing initialisation of a patch's search space inside textureless regions without relying on feature matching and expansion reaching such areas. This widens the search space for PSO to explore, but allows texture independent initialisation via the visual hull. The intention being to achieve a similar result to PMVS, however, accounting for patch optimisation inside textureless regions allowing for denser patch clouds, without the requirements of feature matching or expansion.

Although GLN-PSO can have superior performance in terms of finding the optimal solution in fewer iterations, Section 3.4.3, its reliance on the global best particle and neighbouring best particles positions are not easily adapted to a GPU based algorithm. Knowledge of the global best particle would require running a reduction operation, [115], on all the particles; reducing the vector of photo-consistency values for each particle to the one with the single best solution per iteration of the swarm. Then each particle would require a slow global memory read to access this global best particle over each kernel, Section 3.3.3. Furthermore, calculating a particle's spacial neighbours also requires high levels of computation that are not easily parallelized to the GPU. Therefore, we opt to exclude these high computational force operations, favouring instead personal and local forces in the form of SPSO, Section 3.4.1. Mutation operations are also randomly applied to add further robustness to stagnation, Section 3.4.2.

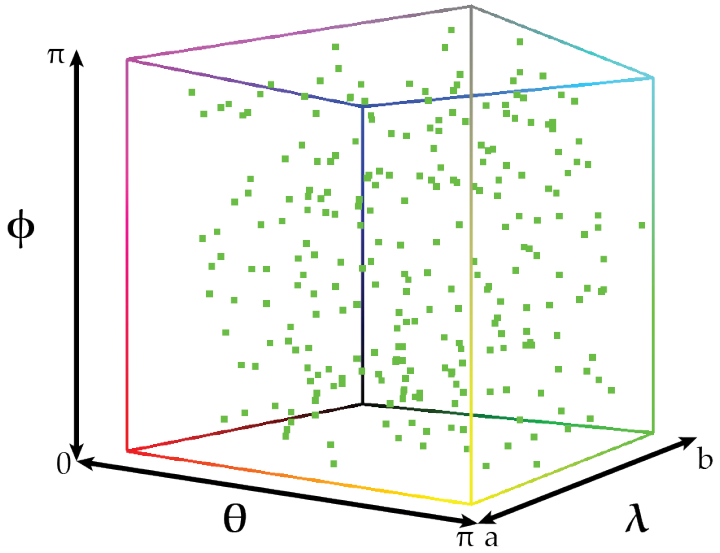


Figure 4.11: The PSO's search space is defined over 3 dimensions. λ, is constrained to lie between the front intersection, a, and back intersection, b, with the visual hull. The normal parameters, θ and ϕ, are clamped between 0 and π. The green points represent the random initialisation of the particles that explore the search space

The search space for patch refinement can be relatively large, but it is of low dimensionality, one for the patch depth, λ, along its viewing ray, and two for the patch normal expressed in spherical coordinates, θ and ϕ. Figure 4.11 displays the set up of the PSO's search space, in which each particle represents a different patch configuration. The job of PSO will be to find the optimal patch configuration to use as the refined patch for the current pixel.

In order to constrain the search space without the requirement of feature matching, the depth range is set to the patch's intersection of its viewing ray with the scenes bounding volume. Because of the availability of silhouettes, the visual hull can be obtained and the depth range can be further restricted to lie within the front and back intersection of the viewing ray to the visual hull. Obtaining the front and back intersection depth maps was discussed in Section 4.1.2. This information can be extracted for every pixel, regardless of whether it contains a feature match. This guarantees that the search space must contain the correct depth position, regardless of how deep any surface concavity may be. Whilst also guaranteeing that any patch positioned outside this range will sit outside of the visual hull, hence, reflecting an invalid depth configuration for the patch.

The patch normal, expressed in spherical coordinates relative to its reference view, $R(p)$, is restricted such that $\theta, \phi \in [0, \pi]$. This ensures that the patch does not become back facing in the reference view. Having such wide search space on the patch normal allows for sharp grazing angles of the patch to the reference view. Scaling up the patch to a μxμ matching window in the reference view, should go some way towards account for this effect.

Evaluation of the photo-consistency function represents the highest computational demand and bottleneck for all reconstruction systems. Therefore, we choose to implement the evaluation of a particles fitness value, the photo-consistency score, on the GPU. In practice the entire PSO algorithm is implemented GPU side avoiding redundant data transfer between host and device memory, this was discussed in Section 3.3.3. The CPU only sends the search space constraints and reads back the optimisation results when complete. The intention of this is to offset the computational burden, improving the speed of patch reconstruction. We chose to implement PSO in a synchronous manor, as the asynchronous version offers little in terms of computational gains when the evaluation of the cost function is significantly higher than the cost of computing the evolution of the swarm itself, Section 3.4.4. We would like to optimise, as much as possible, the speed at which the sampling and NCC for each patch can be computed. Therefore, we adapt Zhou and Tan [176], synchronous GPU implementation of SPSO into a MVS setting for patch optimisation as follows.

Firstly, all images are transferred to the GPU using one of OpenCL's image arrays. OpenCL efficiently caches image objects inside the array to allow for the most effective texture addressing and sampling operations, [60]. OpenCL does not cache or arrange normal global memory buffers in the same way, so it is seen as worthwhile to use these features for improved performance when accessing image data; global memory buffers, Section 3.3.3, are then created GPU side to hold each particle's current position and value, $X(t)$, personal best position and value, $PB(t)$, local best position and value, $LB(t)$, and finally, the velocities of each particle, $V(t)$. This should be done only once when the program first starts, as these buffers will then be re-used for each invocation of PSO.

Algorithm 8 GPU-PSO

procedure GPUPSO($X(t), PB(t), LB(t), V(t)$)

Initialise $X(t)$, $PB(t)$, $LB(t)$ *and* $V(t)$ ▷ in parallel

for $t \leq Number\ of\ Iterations$ **do**
Evaluate $X(t)$ ▷ in parallel
Update $PB(t)$, $LB(t)$, $V(t)$ *and* $X(t)$ ▷ in parallel
$t \leftarrow t + 1$
end for

Reduce $LB(t)$ ▷ in parallel
Read back optimal parameters
end procedure

Figure 4.12: The flow of GPU based PSO

Figure 4.12 shows the flow of our GPU based synchronous PSO algorithm. When PSO is first started, an initialisation kernel is run per particle of the swarm. This defines a particle to work-item mapping, which can be run extremely efficiently. The kernel reads the position and velocities from the global buffers and writes back new, randomly initialised results, this happens in parallel over all particles of the swarm.

Unfortunately, OpenCL does not yet have any random number generation routines, as such, in this work we introduce the use of the fourth components of each velocity vector storing a randomly generated number. These random numbers, between 0 and 1, are initialised when the buffers are first created CPU side. When random numbers are required, the old random number in the fourth component of the velocity is replaced by

a new one on the GPU, without any host to device communication. We borrow Java.Random, [40], seeded random number algorithm to generate and replace these random numbers. Zhou and Tan [176] have suggested solving the random number generation problem by first uploading a large buffer of randomly generated numbers which PSO will move through as and when a new random number is required. However, as PSO requires many random number this buffer must be very large, corresponding to lots of wasted space. Therefore this approach is not feasible especially when we require large amounts of free memory to store all the image and reconstruction data. The pseudo-code for the initialisation kernel is presented in Figure 4.13 below, where the NDRange must be one dimensional and equal to the number of particles of the swarm, Section 3.3.1. The size of each work-group is arbitrary and set depending on how many work-items a single core of the GPU can process in parallel.

Algorithm 9 Initialisation Kernel

procedure INITIALISE($X(0), PB(0), LB(0), V(0)$)

$p = get_global_id(0)$ ▷ Get particles index for work-item
$r = ReadRandomNumber(V(0)[p])$

$RandomlyInitialisePosition(X(0)[p], r)$
$r = MakeNextRandomNumber(r)$
$SetEqual(PB(0)[p], X(0)[p])$ ▷ Store the initial patch as personal best
$SetEqual(LB(0)[p], X(0)[p])$ ▷ Store the initial patch as local best

$WriteRandomNumber(r, V(0)[p])$ ▷ Store a new random number

end procedure

Figure 4.13: The flow of our GPU based PSO initialisation kernel

After the swarm is randomly initialised it is now free to refine the optimal solution by iterating an evaluation and update kernel. The purpose of the evaluation kernel is to compute, as quickly as possible, each particles photo-consistency score. The update kernel is then responsible for defining the velocity and updating the position of each particle in accordance to the SPSO update routines, Equations 3.9 and 3.8 respectively. As evaluation of the photo-consistency score is computationally demanding, we carefully design the evaluation kernel to run as efficiently as possible in a MVS setting. Figure 4.14 displays our coarse-grained approach to the evaluation kernel.

Algorithm 10 Evaluate Kernel

procedure EVALUATE($X(t)$)

$p = get_group_id(0)$ ▷ Particles index is work-group id
$v = get_local_id(0)$ ▷ View index is work-items id in work-group

Check if $X(t)[p]$ *is visible in image* v
if *visible* **then**
  for $c \in R, G, B$ **do**
    Compute $NCC_c(I_{R(p)}, I_v, X(t)[p])$
  end for
end if

$Barrier()$ ▷ Wait for all work-items to finish
if $v = 0$ **then** ▷ Use only first work-item to sum all views NCC_c
  Sum NCC results for visible images
end if
end procedure

Figure 4.14: The flow of our GPU based PSO evaluation kernel

Each evaluation kernel should be as fine grained as possible, therefore, when evaluating the photo-consistency score for a particle (representing a 3D patch configuration), each kernel will evaluate the NCC score between a single view and that particle's/patch's reference view, $R(p)$. This means that for each particle in the swarm, a work-group containing a number of work-items equal to the number of input views is created. Each work-item, within the work-group, will work out its NCC score with the reference view if it is valid to do so, i.e. if the patch is front facing in the current view and within the valid depth range. Once each work-item in the work-group has computed its NCC score, synchronisation barriers are then used to enforce one work-item in the work-group to sum the average NCC over the entire work-group. To evaluate all the patches in parallel then requires the total number of work-groups to equal the total number of particles in the swarm, Section 3.3.1.

This coarse-grained approach, computing the NCC score between two views inside one work-item and summing over the work-group for the full photo-consistency score, proved to be more efficient than computing the total photo-consistency score inside only one work-item. In theory, this approach spreads more work in terms of sampling and NCC computation over the available cores, increasing occupancy and thus run time, Section

3.3.1. However, for any given patch, not all views will be able to see the patch, hence, not all work-items inside a work-group will do any processing, causing divergence, and some kernel invocations are wasted. Therefore, we see scope to improve this approach, however, leave this as future work.

Once the photo-consistency score for each particle in the swarm is computed, the personal best, local best, velocities and current positions must then be updated via the update kernel. Similarly to the initialisation kernel, a particle to work-item mapping is used, this time taking advantage of local memory. Each work-group loads the personal best, local best, velocities and current positions from global to local memory. Each kernel then operates on local memory to perform the update operation, and writes the result back to global memory once complete. This avoids multiple global reads to local neighbouring particle values, as well as allowing for coalesced reads and writes to the global memory, Section 3.3.2. Figure 4.15 below gives the pseudo-code for the velocity update kernel defined in this work.

Algorithm 11 Update Kernel

procedure UPDATE($X(t), PB(t), LB(t), V(t)$)

$p = get_global_id(0)$ ▷ Get particles index for work-item

$CopyGlobalToLocalMem(p, V(t), V)$
$CopyGlobalToLocalMem(p, X(t), X)$
$Barrier()$ ▷ Wait for copy to finish

$UpdatePersonalBest(PB(t)[p], X[p])$

$Barrier()$ ▷ Wait for all particles to update personal best
$CopyGlobalToLocalMem(p, PB(t), PB)$
$CopyGlobalToLocalMem(p, LB(t), LB)$
$Barrier()$ ▷ Wait for copy to finish

$LB[p] = CheckForLocalBest(PB, LB)$
$V[p] = UpdateVelocity(X[p], PB[p], LB[p], V[p])$
$X[p] = UpdatePosition(X, V[p])$

$CopyLocalToGlobal(p, LB[p], LB(t)[p])$
end procedure

Figure 4.15: The flow of our GPU based PSO update kernel

Finally, in order to find the best position found amongst the entire swarm, a standard reduction operation on the local best position and values buffer is run, [115]. This sweeps through the buffer searching for the particle with the minimum photo-consistency score, this will correspond the optimal patch parameters. Once this minimal particle is found its parameters are read back to the CPU, and used as the patch's refined parameters. Our GPU based PSO algorithm is then run again, until all the patches that pass the viewing angle test are refined.

4.2 Datasets

We will be evaluating our PSO based approach by quantitative and qualitative comparison with the current state of the art patch based technique, PMVS. To ensure a fair and comprehensive evaluation of both techniques, a range of different datasets, displaying differing surface characteristics, were selected for evaluation. The full set of images from each datasets as well as an illustration of the camera settings and visual hull can be found in Appendix A.

To enable the production of quantitative results, Blender [13] was used to create computer generated multi-view images of ground truth objects, whilst also procedurally generating different levels of texture over them. Figure 4.16 displays the ground truth meshes used to create the synthetic datasets.

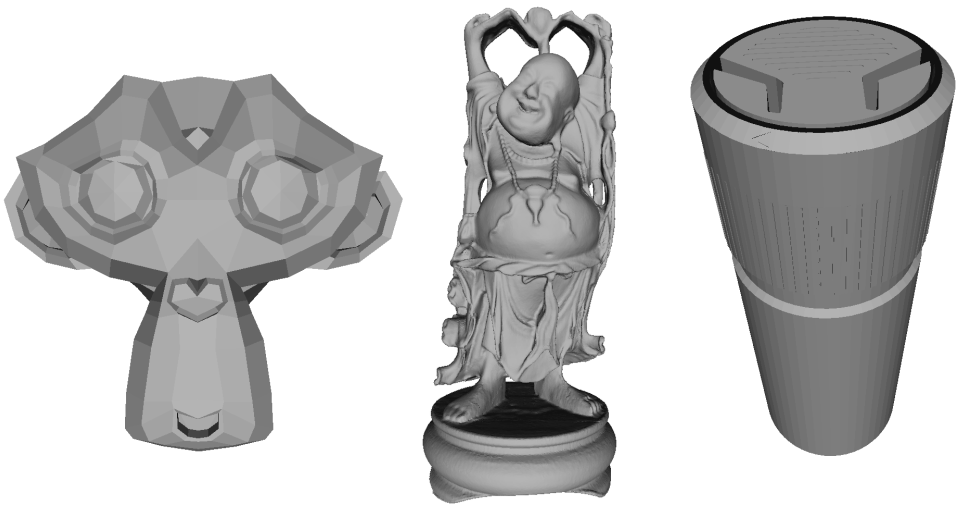


Figure 4.16: Ground truth meshes for the Monkey, Buddha and CAD datasets respectively

A Monkey mesh was chosen as this has large planar regions, which should benefit PMVS's expansion procedure. Local expansion and refinement should easily capture the correct surface properties regardless of the texture quality. In contrast, the Buddha mesh represents a far more complex object, with lots of high frequency surface variations and non-planar regions, a challenging example to reconstruct densely with high fidelity. Finally, the last mesh represents a typical product created using Computer Aided Design (CAD), this offers smooth surfaces with areas of high frequency details, note in particular the fine fins on the top sections of the can. This is also a good example of real life application, where many products are designed through CAD based systems.

In order to evaluate the effectiveness of both PMVS and our PSO approach to varying levels of texture, three datasets for both the Monkey and Buddha ground truths were created. Each dataset contains a different level of texture, from high, medium to low. From here on, MonkeyHigh and BuddhaHigh will be used to refer to the Monkey and Buddha datasets created with high levels of texture. Analogously, we will use MonkeyMed, BuddhaMed, MonkeyLow and BuddhaLow to refer to the medium and low textured version respectively. To simulate a real example, whilst also removing issues with image noise, invalid lighting conditions and camera calibration errors, the CAD mesh was used to generate a dataset with a standard product texture applied. This contains large textureless areas, whist also containing regions of varied high texture. This dataset will be referred to as LynxCAD from now on. An image from each of these datasets can be seen in Figure 4.17.

As well as using computer generated imagery for quantitative assessment, real life photography is also used to test the system and create qualitative results. We have used the sparse 'Temple' and 'Dino' benchmarks publicly available from Seitz et al. [137], as accurate camera calibration is included with these datasets. These complement our synthetic datasets fairly well, as the Dino dataset contains planar regions over its fins, whereas the Temple dataset exhibits lots of high frequency surface detail. These datasets also show a variety of features not present in our ground truth datasets, such as the self-shadowing on the Dino model and large, highly occluded regions such as the columns of the Temple model.

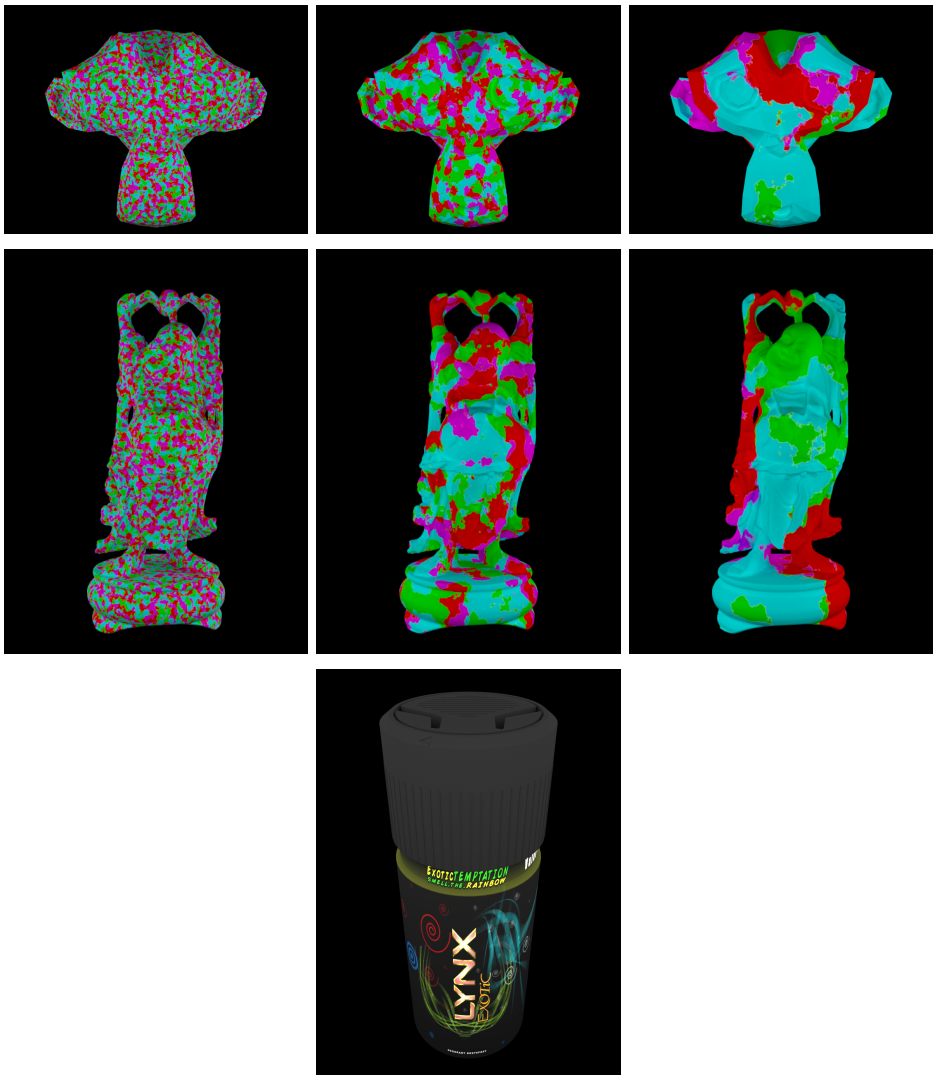


Figure 4.17: An image from each of the high, medium and low textured datasets for the Monkey and Buddha ground truth mesh, and a standard texture applied to the LynxCAD model

We have also obtained an equivalent real life alternative to the LynxCAD dataset, by photographing an actual Lynx can and calibrating the images via Bundler. This dataset shows extremely challenging features, in the fact that it has highly textureless regions and the photographs are fairly noisy. Also its glossy, specular material properties clearly cause a violation of the Lambertian reflection assumption.

Unfortunately, we were unable to obtain ground truth meshes for these datasets, however, felt it was worthwhile including these in analysis for the extra features they exhibit and fair assessment of both algorithms. Dino, Temple and LynxReal, will be used to refer to these datasets from now on. Images from each of these datasets can be seen in Figure 4.18.



Figure 4.18: An image from each of the real life datasets: Dino, Temple and LynxReal respectively

4.3 Experimental Results and Analysis

In this section the results of running both our PSO algorithm and the top performing PMVS algorithm, [53], are evaluated. We chose to test against PMVS as this has been shown to be the top performing approach to obtaining high quality reconstruction of both high a low frequency surface details, [138, 146, 79], as well as following a similar approach to our system. This section will analyse the effectiveness of our PSO technique for 3D patch reconstruction, when feature based initialisation and expansion steps are removed. In particular, we assess how the quality of reconstruction is affected when the texture of an object is reduced and the number of input views is low. In all experiments, 16 views of each object are used to perform reconstruction, representing a typical sparse input set easily obtained by any user. Results of reconstructing both computer generated and real life imagery, from the datasets presented in the previous section, are given; therefore, allowing for both quantitative evaluations, from ground truth data, and also qualitative results, from real life examples, where no ground truth was available.

Our PSO algorithm was implemented in C++, using the OpenCL API, [60], to interface with the GPU. A benefit of our approach is that very few parameters must be set to achieve reasonable reconstruction within a given time frame. As previously discussed in Section 4.1.1, we fix the patch size, μ, to 5; larger patch sizes stabilise the errors, but in turn smooth out high frequency surface details, reducing the accuracy and increasing the run time. Aside from setting the search space bounds for each particle within a PSO optimisation, which is automatically handled per patch as described in Section 4.1.3, there are only two free parameters for PSO that must be set; the swarm size and number of iterations.

Bratton and Kennedy [20] note that differing the swarm size can often result in slightly improved results, however, this is problem dependant thus changes for every new function. They note that there is no definitive swarm size that is optimal across all functions. They conclude their findings remarking that whilst tuning the swarm sizes can result in better optimisation, generally speaking it is of minor importance. They suggest swarm sizes between 20-100 particles produce results in which none are clearly superior or inferior as the problem function changes. Since our PSO is implemented on the GPU, we chose a swarm size of 64 particles, which scales nicely to the number of cores on modern day GPU's.

The number of iterations PSO operates over controls the accuracy to which a solution can be found, see Section 3.4.1. Given the fairly large swarm size for the low dimensionality of the problem, through experimentation we found that 10 iterations allowed PSO sufficient time and coverage to converge on an accurate solution in all cases that were tested. However, a larger number of iterations would no doubt further improve the accuracy, at the cost of a longer run time.

In order to fairly compare our results against PMVS, we set PMVS's parameters to allow for the densest, most accurate results possible as reported by Furukawa and Ponce [53]. Firstly, the image resolution for which the reconstruction should operate on is set to 0, therefore, obtaining the highest resolution patch cloud possible. The cell size is set to 1, allowing PMVS to reconstruct a patch in every pixel, if possible. The patch size is set to 5, equivalently to ours, allowing good acquisition of high frequency details, avoiding smoothing effects that can effect patch accuracy. The minimum number of views a patch is required to match in is reduced to 2, accounting for wider baselines inherent in sparser view sets. The source code for the PMVS algorithm, which was used to generate the results in this thesis, is publicly available from Furukawa and Ponce [54].

Due to the availability of ground truth the synthetic datasets have been used to produce quantitative results. Since the camera parameters are precisely controlled, noise introduced by imprecise camera calibration or faulty pixel sensors can be ruled out. If the algorithm is fed with poor camera initialisation, when projecting world space patch samples into image coordinates, Section 3.1, these may no longer project to their true image space pixels; resulting in an incorrect sampling for that patch configuration. If the sampling is incorrect this distorts the photo-consistency metric that could mean that the best photo-consistency score no longer relates to the correct surface parameters. In turn meaning that patch refinement returns less accurate results introducing unwanted error into our quantitative analysis.

Given that these images were generated from ground truth meshes, the distance between reconstructed patches and the ground truth mesh can be found, defining a measure of accuracy on the resulting patches. For this, the one-sided Hausdorff distance metric, [74], is used, computing the distance of a patch to its closest point on the surface of the object. For

a full Hausdorff metric, the closest patch to every point on the surface must also be found, and the maximum distance between them is calculated. However, as we are only interested in the quality of reconstruction for an individual patch, so that we can measure it against those produced by PMVS, not the patch set as a whole, we choose to use only the first half of the transform. Therefore, the larger the distance of the patch to the closest point on the surface, the larger the error on correctly refining that patch. If we used the second half of the transform, we would be measuring the distance of every random point on the surface to the closest patch, as patches are confined to sit only along the view ray generated by the reference view's pixel, it would be unfair to measure the distance for all random points on the surface, as the algorithm is only designed to optimise the patch to best reflect the surface that is intersected by the viewing ray.

4.3.1 Qualitative

Figures 4.19 to 4.24 display the results of running PMVS and our PSO based algorithm on each of the datasets presented in the previous section. Recalling the goal of this research is to improve the density of patch reconstruction in textureless regions whilst achieving a similar accuracy to PMVS in high textured regions, this section will assess the results presented in terms of four factors: The effect on sparsity and density as texture is reduced, the effect on high frequency details, accuracy and smoothness as the texture is reduced, the roll of image resolution, noise or erroneous patches produced in each method.

Sparsity and Density

Reconstructions via our PSO based patch system are consistently denser than that of the corresponding PMVS solution, albeit PMVS has removed incorrect patches, it was unable to expand into and define surfaces in low textured regions. Notice that our PSO approach was able to recover the chin and head sections in the Monkey datasets, Figure 4.19, as well as the head, neck, belly and base inside the Buddha datasets, Figure 4.20; whereas PMVS left large holes in these regions. Increasing the patch size for PMVS would go some way towards filling these holes, however, as discussed in Section 4.1.1, this would have the undesirable effect of decreasing the amount of high frequency details PMVS could extract.

PMVS did better than expected in dealing with textureless areas inside the BuddhaMed and BuddhaLow datasets, Figure 4.20. This was most likely due to the image resolution and the smoothly changing surface properties. Although, it is still clear that as the texture is reduced the density and accuracy of PMVS is also reduced, whereas only the accuracy is affected for PSO. Note how the base and belly regions of the Buddha become sparse and inaccurate as the texture is lowered. It is worth noting that PMVS also dealt fairly well with low texture over the Dino model, however, the features created by the shading and self-shadowing no doubt helped start patch expansion. Notice the sporadic holes in the dinosaur's surface far from the shadow features, there are also large portion of the dinosaurs back and base missing, which are recovered in our approach.

Unexpectedly, PMVS didn't deal well with the reduction in texture for the Monkey datasets, Figure 4.19. Large planar regions should allow for easy condition for patch expansion to deal with, expanding one robustly matched patch exactly onto its correct position. This should guarantee that gradient descent finds the correct patch parameters. However, there seems to be some issue with the expansion and filtering stages that have removed such patches, meaning that as the texture is reduced large wholes begin to appear.

High Frequency Detail, Accuracy and Smoothness

Although PMVS has performed incredibly well, in terms of accurately creating a subset of patches over the surface of the objects, it suffers greatly in low textured areas. The additional density of our approach has allowed us to fully capture and reconstruct the high frequency details of the corresponding models. Notice the facial features and base of the Buddha model are fully recovered in our PSO approach as opposed to PMVS, Figure 4.20. The corresponding PMVS patch cloud smooths over these features or creates holes depending on the texture resolution, again caused by the filtering methodology removing inconsistent patches and expansion not reaching all areas. As the texture resolution drops, the accuracy at which both algorithms can reconstruct patches also drops. Notice the base of the Buddha statue, Figure 4.20, becomes smoothed out over both approaches when texture detail is reduced. This reflects the difficulty in refining patches inside textureless areas, where the photo-

consistency matching score is less well defined and doesn't align exactly with the underlying surface properties, Section 3.2. All results show that smoothness increase as the texture decreases, seen where the edges of two planar regions become smoothed out.

PMVS will naturally work very well for the temple dataset, Figure 4.23, given that it's highly textured and has good diffuse material properties over the entire surface, however, note our algorithm's ability to equally capture the high frequency details on the temple dataset. It must be noted that whilst PSO captured large quantities of these details, it was also hindered greatly by noisy, incorrectly refined patches, whereas PMVS was not. When assessing the Dino model, Figure 4.22, PSO was able to densely recover all parts of the surface, whereas the equivalent PMVS approach suffered holes. Although PSO was able to capture the fine details such as those along the fins of the model, the patch cloud for PSO is clearly significantly more noisy than that produced by PMVS.

Image Resolution

Both the real life and CAD models of the Lynx can, Figures 4.24 and 4.21 respectively, contain large textureless regions, where their higher image resolution increases the textureless effect. Larger image resolution implies that details that may not have been textureless at a lower resolution begin to appear textureless. For example, a 2 pixel thick line at a low resolution could become a 10 to 20 pixel thick line at a higher image resolution, where the centre of the line now appears to be textureless.

Within these large resolution textureless areas, PMVS has either failed to create patches, as in the LynxCAD example, or incorrectly refined their position, as in the LynxReal dataset. Figure 4.27 more clearly shows how PMVS erroneously created patches inside the can, particularly in the low textured, top section of the can, most likely caused by the violation of the Lambertian assumption. PSO also struggled in this area, however its increased density allowed for some patches to be correctly refined whilst the majority were noisy, this can be seen by the sporadic coverage in Figure 4.27 or Figures 4.24 and 4.21.

Again as discussed in Section 4.1.1, increasing the patch size would go some way towards filling holes at the expense of fidelity. However, with larger image resolutions allowing for a larger number of patches to

be created, expressing more high frequency surface details, using larger patch sizes quickly degenerates the run time.

Noise and Error

Throughout the reconstructions, PMVS performed very well, whilst our PSO approach consistently demonstrates density improvements, it was highly affected by incorrectly refined patches, most notably, in textureless regions. This in turn, leads our preliminary PSO approach to create noisy patch clouds. Figures 4.25 shows the patch clouds created by PSO, except this time all patches whose Hausdorff distance is larger than 0.005 units are removed. In the next section it will be shown that the majority of patches created by PMVS lie within 0.005 units of the ground truth. The similarity between our filtered PSO clouds and that of PMVS is striking, showing that in well texture regions we are performing equivalently. As Hausdorff distances aren't available for the real life datasets, Figure 4.26 displays the patch clouds created by only one of the views for each of the datasets, noisy or incorrect patches can be seen more clearly.

By assessment of these figures it can be seen that when a patch corresponds to a region of high texture, extracting the globally optimal photo-consistent patch parameters corresponds to correctly refining the patch, this is shown by the fact that patches in well textured regions are kept in Figure 4.25. However, when this pixel has a low texture, the globally optimal solution to the photo-consistency function may no longer align with the correct patch configuration, thus patch refinement is noisy, as is seen in Figure 4.26 where not all patches align with the surface. In this preliminary framework, such patches are left unfiltered creating noise in the final reconstruction. It is clear to see that as the level of texture drops more invalid patches are created resulting in increased sparsity in the filtering of Figure 4.25.

LynxReal and LynxCAD display perhaps the worst affected reconstructions, Figures 4.24 and 4.21, this is due to their higher image resolution inducing more textureless regions than the other datasets. This phenomenon resulted in a sharp point at the base the LynxCAD reconstruction, where patches sit correctly inside the visual hull but incorrectly in relation to the surface.

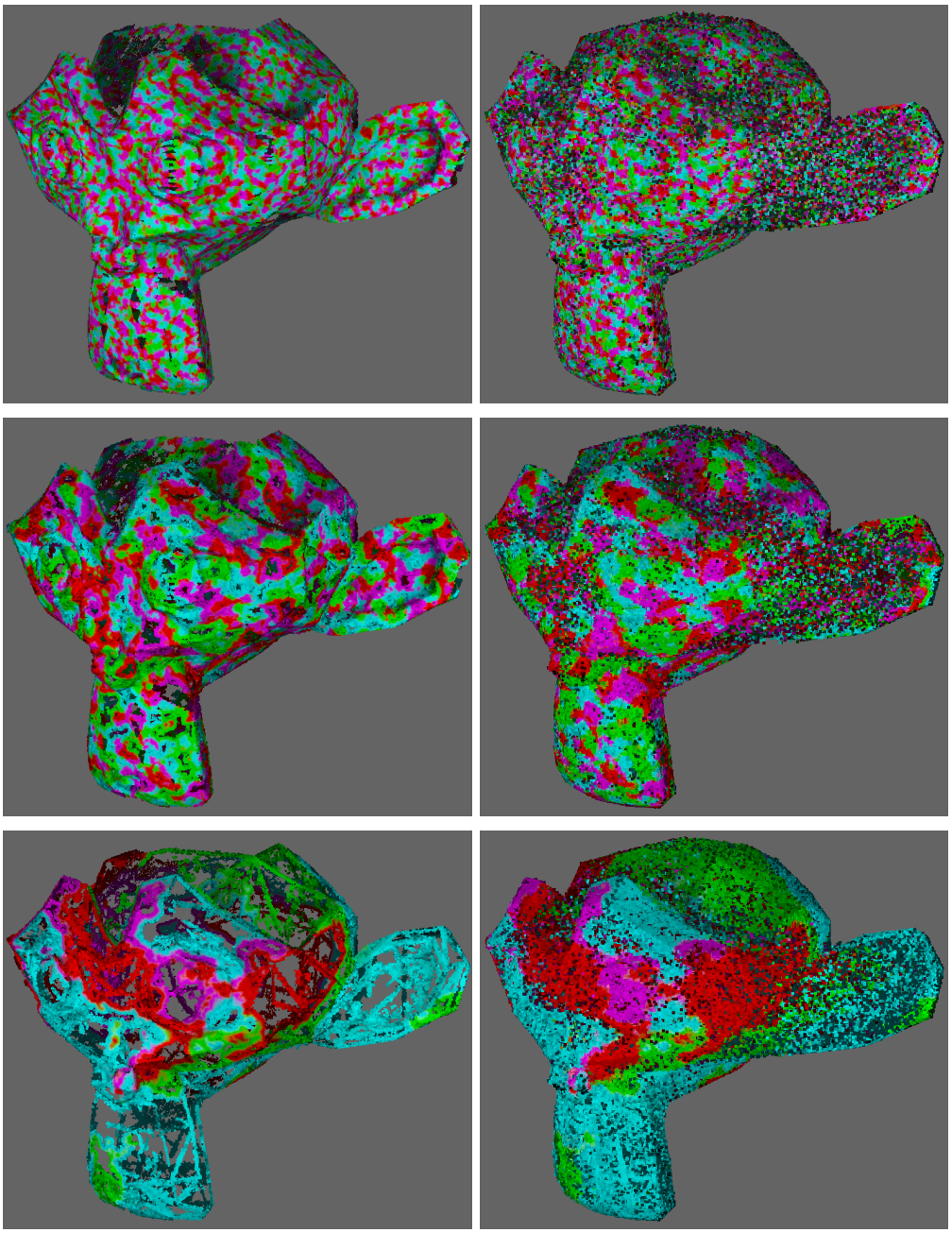


Figure 4.19: Reconstruction results for the Monkey dataset with varying levels of texture.
Left Column: PMVS results. **Right Column**: PSO results

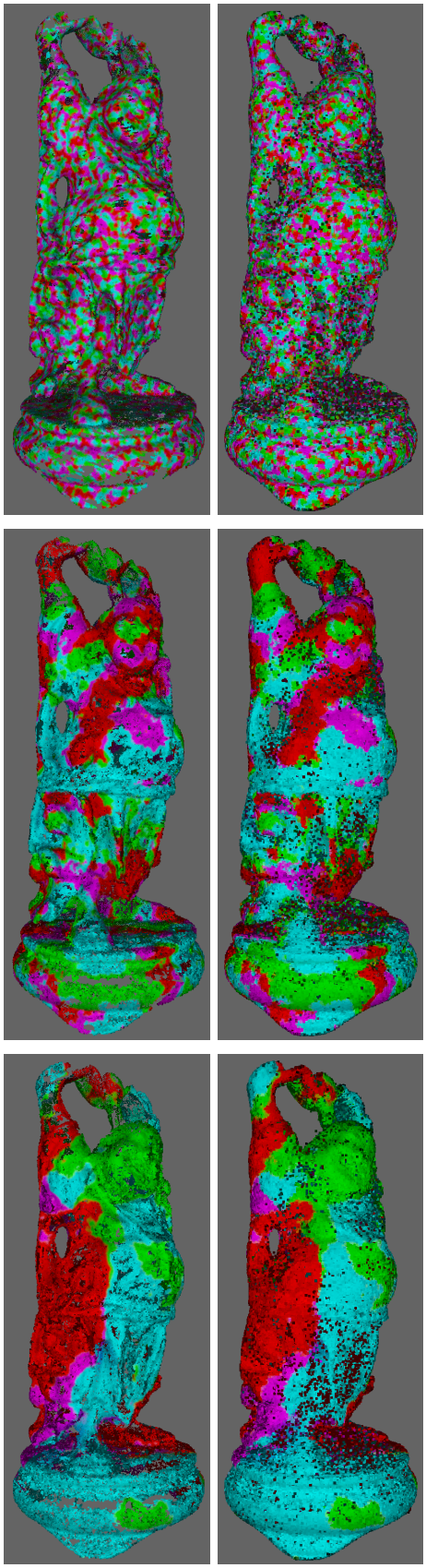


Figure 4.20: Reconstruction results for the Buddha dataset with varying levels of texture.
Left Column: PMVS results. **Right Column**: PSO results



Figure 4.21: Reconstruction results for the LynxCAD dataset
Left Column: PMVS results. **Right Column**: PSO results

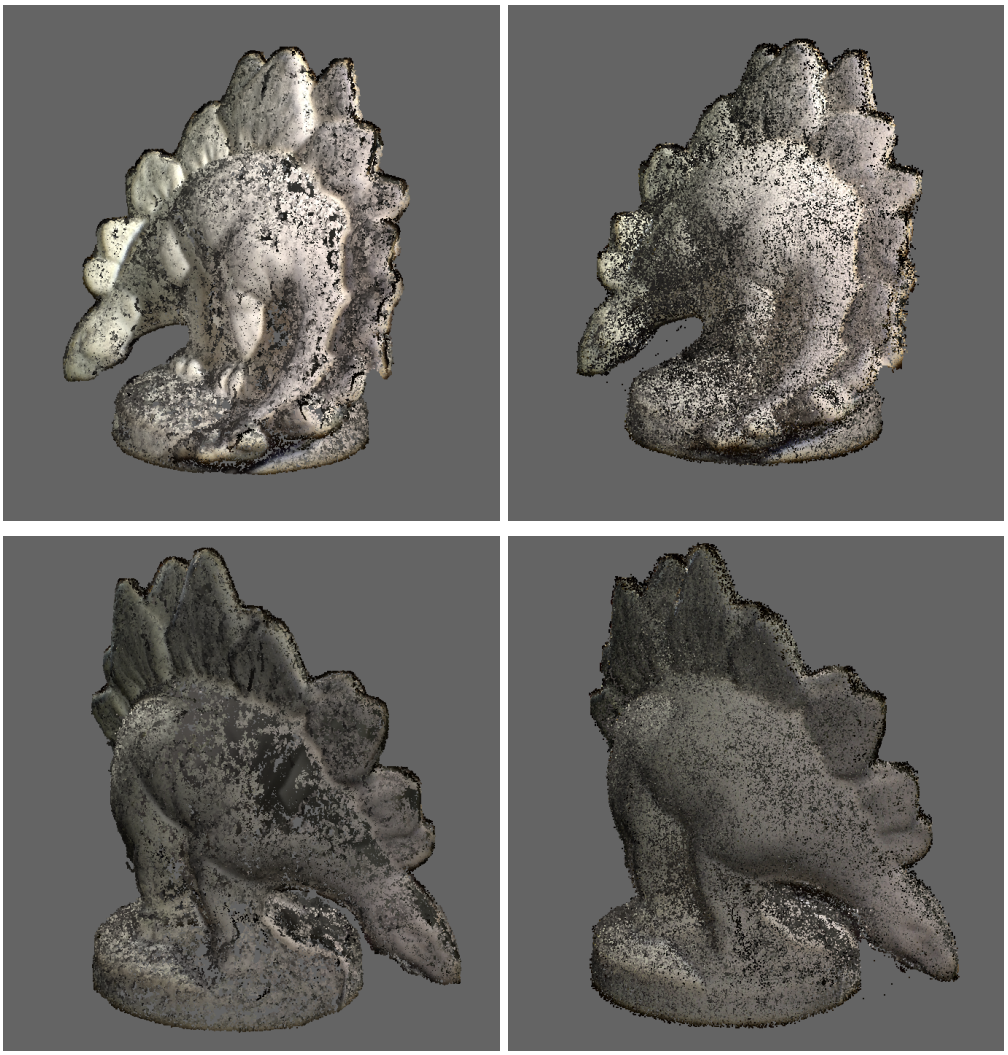


Figure 4.22: Two views of the reconstruction results for the Dino dataset **Left Column**: PMVS results. **Right Column**: PSO results

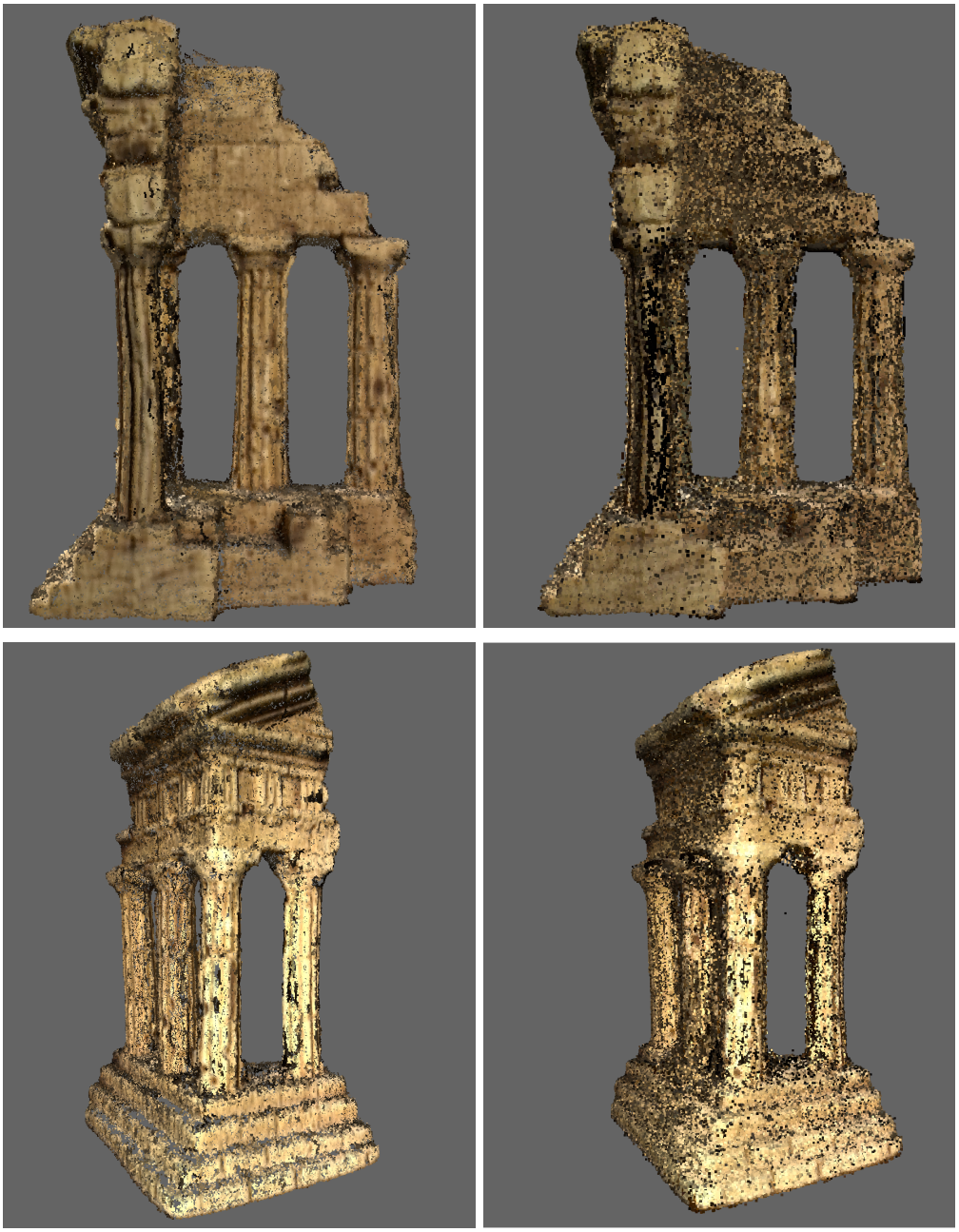


Figure 4.23: Two views of the reconstruction results for the Temple dataset
Left Column: PMVS results. **Right Column**: PSO results

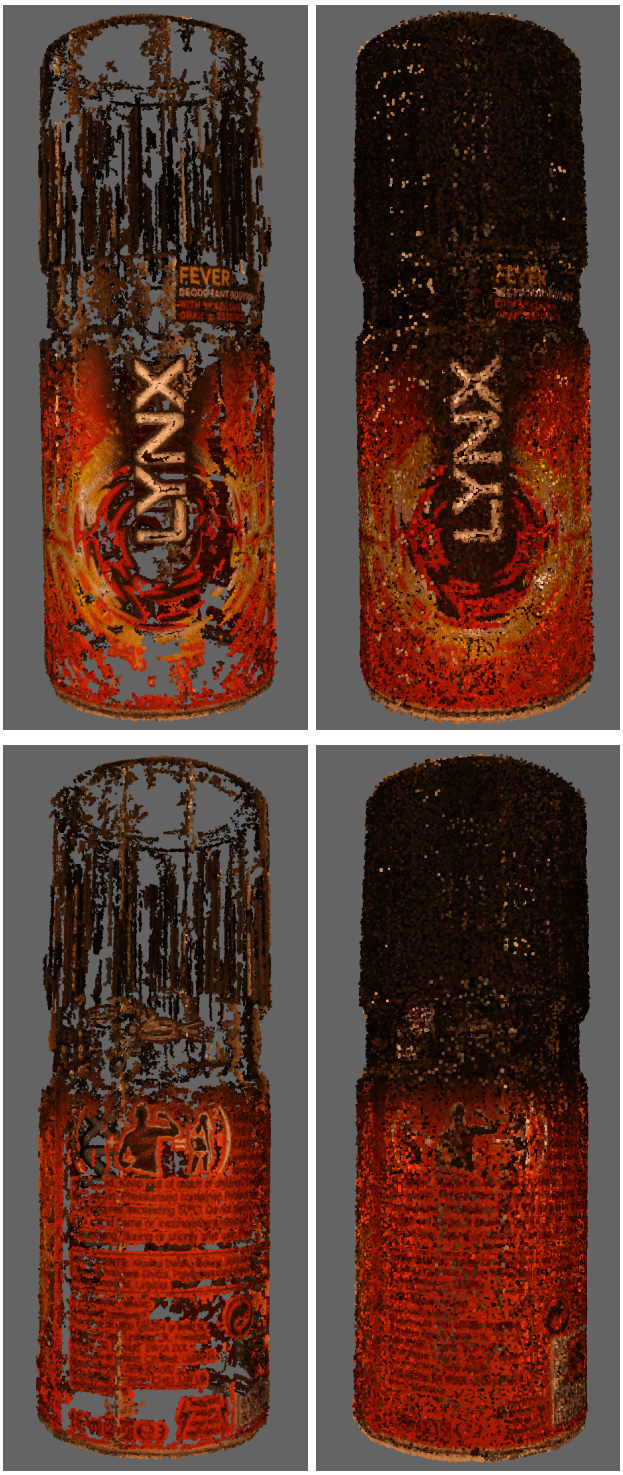


Figure 4.24: Two views of the reconstruction results for the LynxReal dataset
Left Column: PMVS results. **Right Column**: PSO results

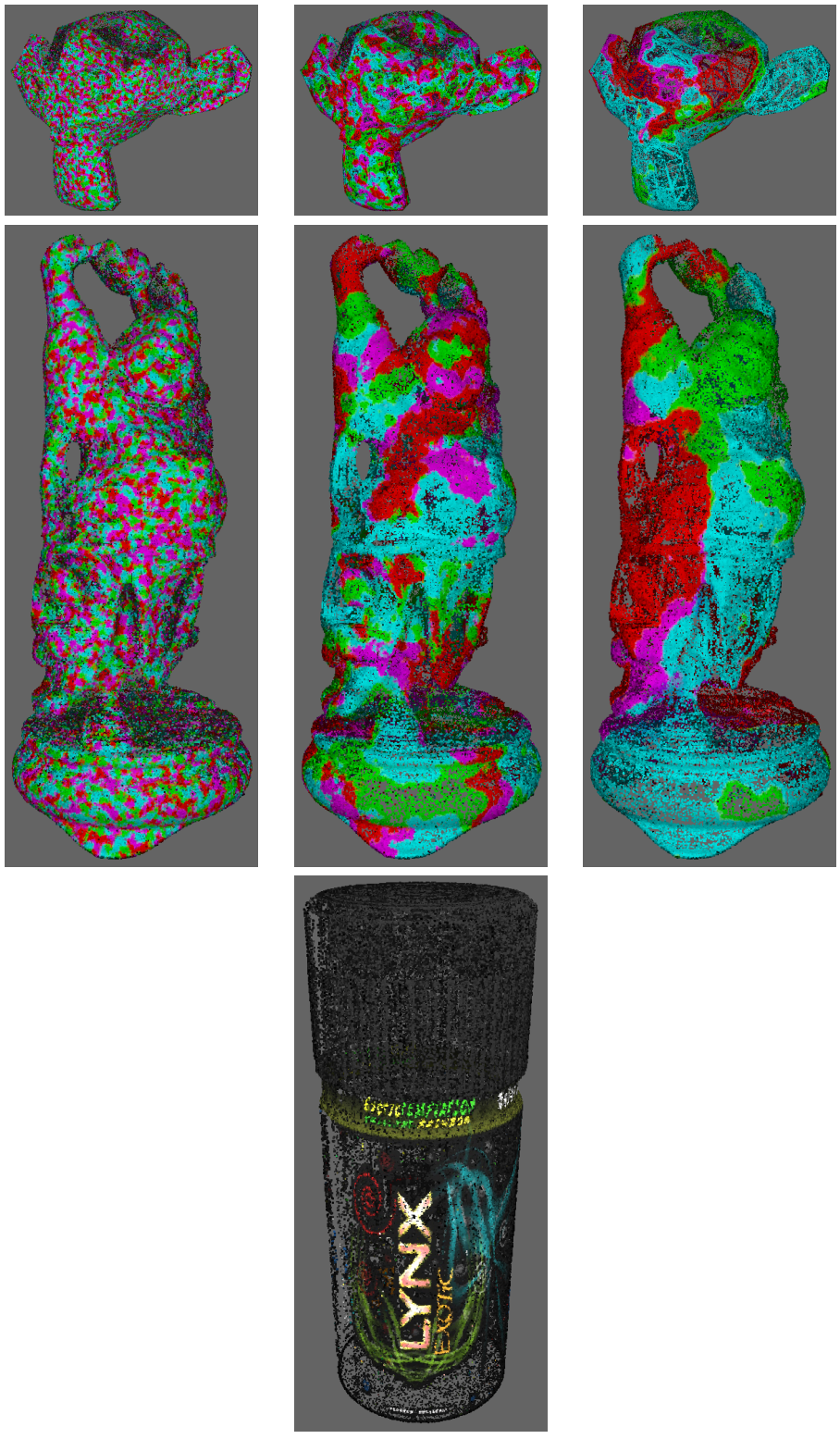


Figure 4.25: Noise reduced patch clouds created by PSO, these are filtered by removing all patches in the cloud whose distance is more then 0.005 units from the surface

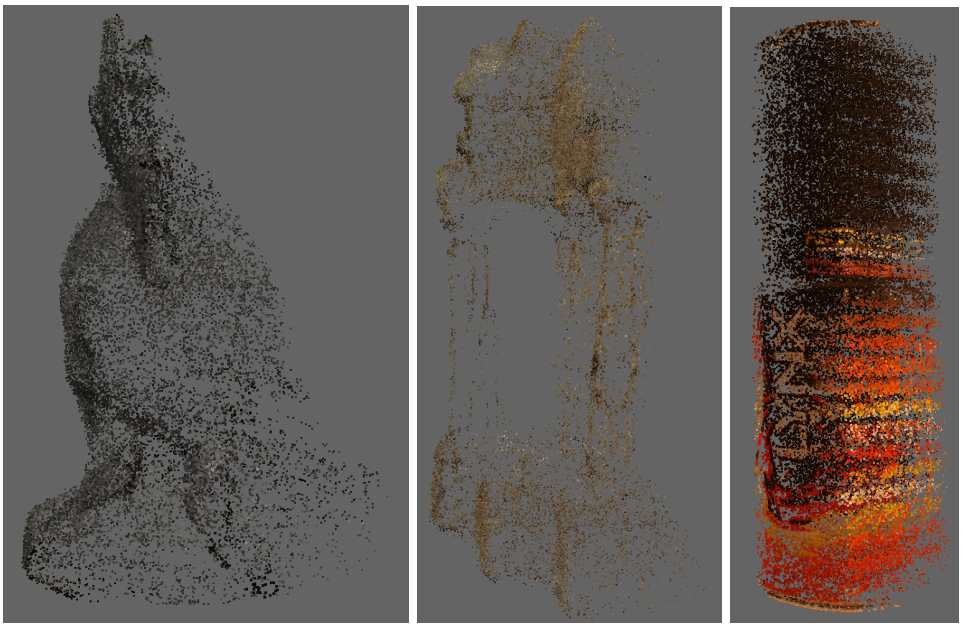
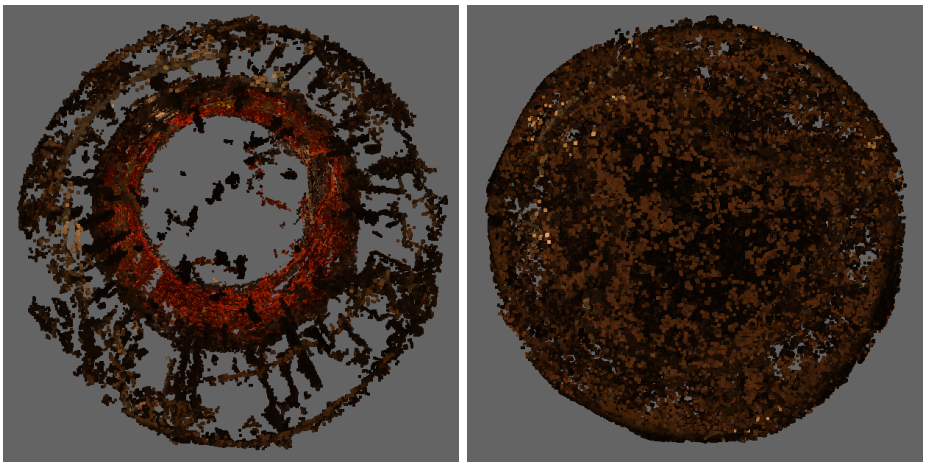


Figure 4.26: Patches created by PSO for a single view for each of the real life datasets, noise can be seen when refining a patch inside a low textured area



(a) PMVS (b) PSO

Figure 4.27: Erroneous reconstruction around the lid of LynxReal

4.3.2 Quantitative

Through the qualitative results presented thus far, it is clear that our PSO technique is a step towards achieving equivalent patch accuracy in well-textured areas whilst allowing for density improvements in textureless regions, although the accuracy of textureless patches is currently questionable. Qualitative results seem to show it is limited by the noise produced, raising the question as to the validity of PSO for patch refinement. Therefore, in this section, quantitative results are presented, assessing its accuracy and effectiveness, in comparison with PMVS and our overall goal of refining patches inside textureless regions. Specifically we will assess 2 key areas: the general performance of PSO and PMVS as well as the effect on accuracy and noise as texture is reduced.

Figures 4.28 to 4.30 and Tables 4.1 to 4.3 display the relative probability distributions for the patch refinement error and accompanying statics for each of the ground truth datasets. The x-axis of the probability distributions represents the relative error, as the one-sided Hausdorff distance, and the y-axis represents the relative density, or probability of that error. All distributions are normalised such that their area sums to one. The mean, median, mode and max of each distribution are then supplied in a following table. After which a table representing the inverse cumulative frequency, the percentage of patches in the patch cloud whose error is greater than the given distance, is displayed. PMVS High, PMVS Medium and PMVS Low correspond to the error distribution for patches created by running PMVS on the high, medium or low textured datasets respectively. Analogously, PSO High, PSO Medium and PSO Low correspond to the errors when using our PSO based algorithm.

General Performance of PMVS and PSO

The probability distribution functions show the effects of inaccuracy or noisy patch refinement. Regardless of texture, PMVS extracts only accurate patches with a high probability of low error, filtering away bad ones, whereas PSO produces both accurate and inaccurate patches, without filtration. The max error for PMVS over the Monkey and Buddha models is 0.18, whereas, PSO suffers inaccuracies up to 0.45. As such the tail of the distributions are considerably longer for PSO than PMVS. The

median and mean is also higher for PSO, due to the distortion introduced by poorly refined patches.

The inverse cumulative frequencies for all the dataset, show that PMVS had a smaller percentage of patches with an error greater than 0.005 when compared with PSO. This was expected due to the filtration employed inside PMVS. Again, PMVS has the advantage of filtering, so the percentage of error past 0.005 drops more rapidly.

In terms of the overall accuracy, comparing the mode and alignment of the peaks for all the distributions, shows that both PMVS and PSO are equally capable of extracting highly accurate solutions. Both methods produce accurately refined patches in the presence of low texture, as the modes and peaks are always at low errors, regardless of texture. The mode in most cases is slightly lower for PMVS than PSO, and the spread around the peak is noticeably narrower, showing that patches left by PMVS were consistently more accurate than those produced by PSO. Intuitively, noisy patches distorting the spread cause this, however, this also reflects PSO's stochastic nature, and the fact that PSO cannot step exactly onto the global minima. As mentioned previously, in Section 3.4.1, increasing the number of iterations of the swarm is likely to improve the accuracy, at the cost of run time, however, given the current quality of refinement this seems unnecessary. More iterations allows longer for the patches to converge to a more accurate solution, however there is no guarantee this is the global best as it could potentially reach a local minimum.

Given that all the distributions are right-skewed shows that, even though erroneous patch refinement occurs heavily inside our PSO algorithm, it's not severe enough to impinge upon the overall quality with peak performance still sitting at low error. The large amount of erroneous patches in the LynxCAD dataset, clearly degrades the quality when compared to PMVS, although, interestingly PSO achieves a better peak performance, lower mean, Figure 4.30.

When measured by reconstruction errors alone, not accounting for the density of PSO, adding are large number of less accurate patches into the patch cloud, PMVS appears to perform better. Both the median and mean for all datasets, are lower for PMVS as opposed to the corresponding PSO reconstructions, showing that PMVS is more effective in terms of its

ratio of well to poorly refined patches. A large cause of this is the inclusion of less accurate or badly refined patches within textureless regions in our PSO approach, as discussed in the qualitative results of Section 4.3.1.

Accuracy and Noise

As the texture is reduced PSO and PMVS both create more badly refined patches. For all datasets, and both methods, the median grows as the texture is reduced, showing evidence that as the texture drops, it becomes harder to accurately refine a patch, regardless of the technique used.

In terms of accuracy corresponding to the level of texture, all the distributions remain right skewed, with the sharpness of the skew depending on the level of texture. Reconstructions of highly textured objects are constantly more right skewed than that of their equivalent low textured versions. This shows that both PMVS and PSO are limited in their accuracy depending on the texture of the object. Lower textured objects create more patches with higher reconstruction errors, spreading the high quality results into the right half of the distribution, lowering the peak and reducing the skew. PSO's performance degenerates more rapidly than PMVS when the texture is reduced, as without filtration more badly refined patches are produced. The effect of this can be seen in the enlarged tails of the distributions, low textured objects have a larger volume of patches inside the tail than that of their higher textured counterparts.

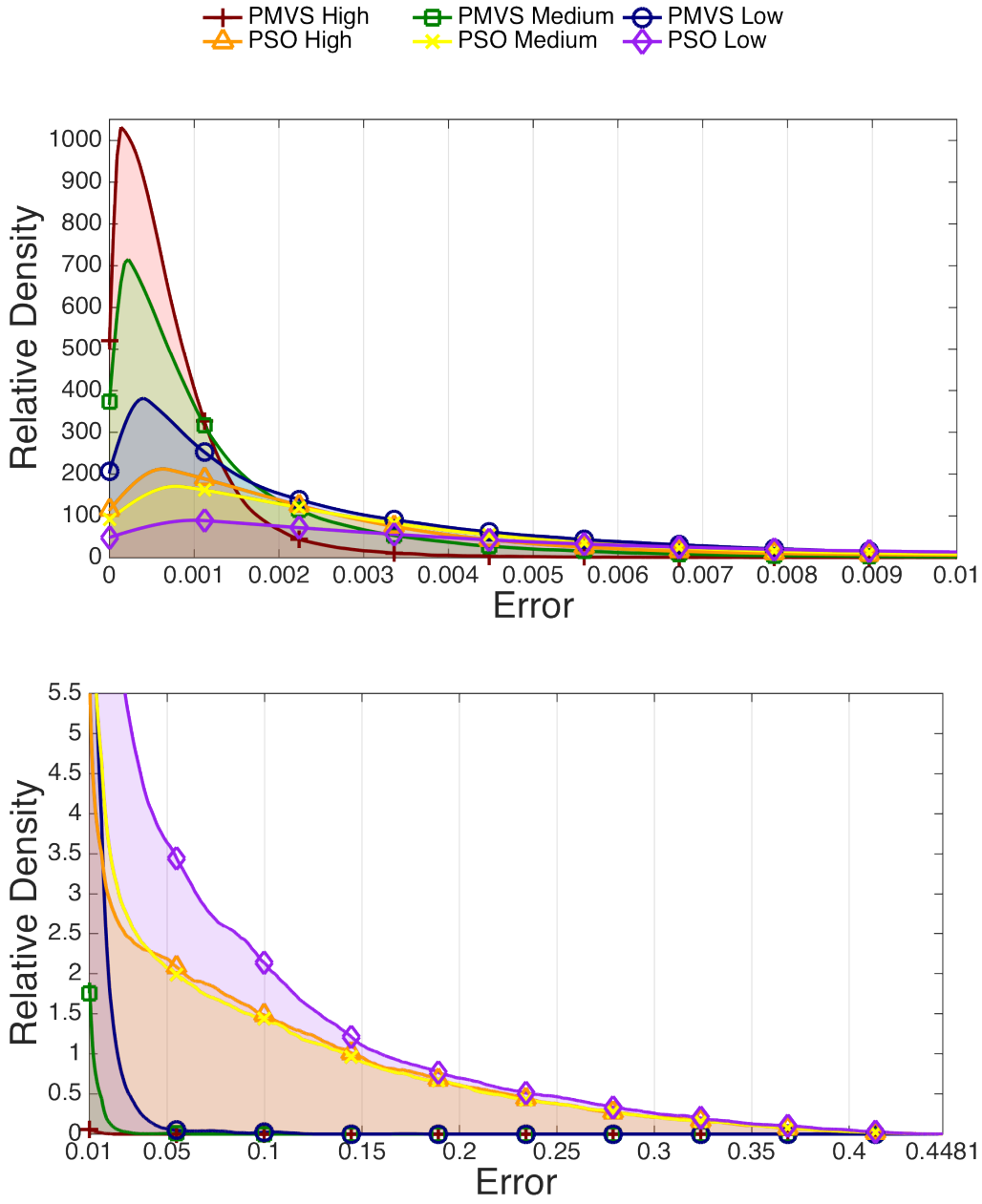


Figure 4.28: Probability distributions for the Monkey datasets
Top: Normalised probability distributions for the errors in patch refinement for running PMVS and PSO on each of the three different textured datasets.
Bottom: Zoomed in tail end of the distribution up to the largest error

		Mean	Median	Mode	Max
PMVS	High	0.0007	0.0005	0.0001	0.0180
	Med	0.0014	0.0008	0.0002	0.0580
	Low	0.0033	0.017	0.0004	0.1751
PSO	High	0.0377	0.0032	0.0006	0.4446
	Medium	0.0371	0.0039	0.0008	0.4439
	Low	0.0505	0.0148	0.0010	0.4481

		Residual Error							
		0.0	0.005	0.01	0.02	0.03	0.1	0.2	0.3
PMVS	High	100.00	0.34	0.02	0.00	0.00	0.00	0.00	0.00
	Med	100.00	4.43	0.61	0.06	0.02	0.00	0.00	0.00
	Low	100.00	19.49	6.07	1.48	0.59	0.04	0.00	0.00
PSO	High	100.00	41.94	34.91	31.23	28.61	14.98	5.09	1.21
	Medium	100.00	46.07	35.65	30.63	27.62	14.48	5.02	1.21
	Low	100.00	67.66	56.20	46.84	40.80	18.42	6.14	1.52

Table 4.1: Statistics for the Monkey datasets
Top: Different averages of the reconstruction error on each dataset.
Bottom: The inverse cumulative frequency of errors as a percentage of entire patch cloud

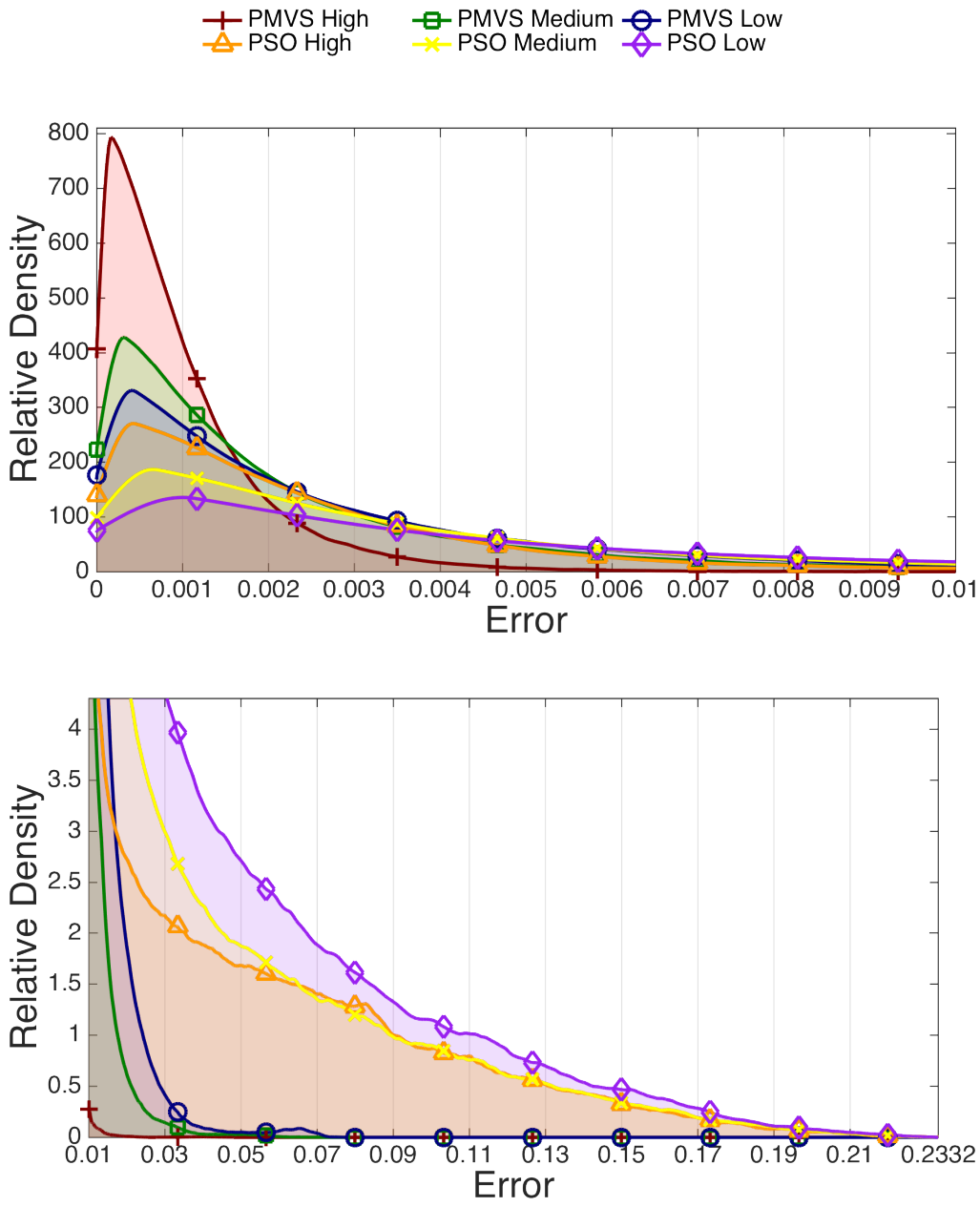


Figure 4.29: Probability distributions for the Buddha datasets
Top: Normalised probability distributions for the errors in patch refinement for running PMVS and PSO on each of the three different textured datasets.
Bottom: Zoomed in tail end of the distribution up to the largest error

		Mean	Median	Mode	Max
PMVS	High	0.0010	0.0007	0.0002	0.0449
	Med	0.0024	0.0014	0.0003	0.0678
	Low	0.0024	0.0019	0.0004	0.0737
PSO	High	0.0140	0.0023	0.0004	0.2327
	Medium	0.0156	0.0035	0.0007	0.2331
	Low	0.0204	0.0052	0.0010	0.2323

		Residual Error							
		0.0	0.005	0.01	0.02	0.03	0.05	0.1	0.2
PMVS	High	100.00	0.82	0.06	0.01	0.00	0.00	0.00	0.00
	Med	100.00	12.08	2.61	0.41	0.13	0.02	0.00	0.00
	Low	100.00	20.01	6.27	1.31	0.39	0.13	0.00	0.00
PSO	High	100.00	28.54	20.18	16.57	14.19	10.42	3.86	0.08
	Medium	100.00	40.55	26.04	18.73	15.12	10.53	3.95	0.08
	Low	100.00	52.72	36.97	26.33	21.00	14.25	5.23	0.12

Table 4.2: Statistics for the Buddha datasets
Top: Different averages of the reconstruction error on each dataset.
Bottom: The inverse cumulative frequency of errors as a percentage of entire patch cloud

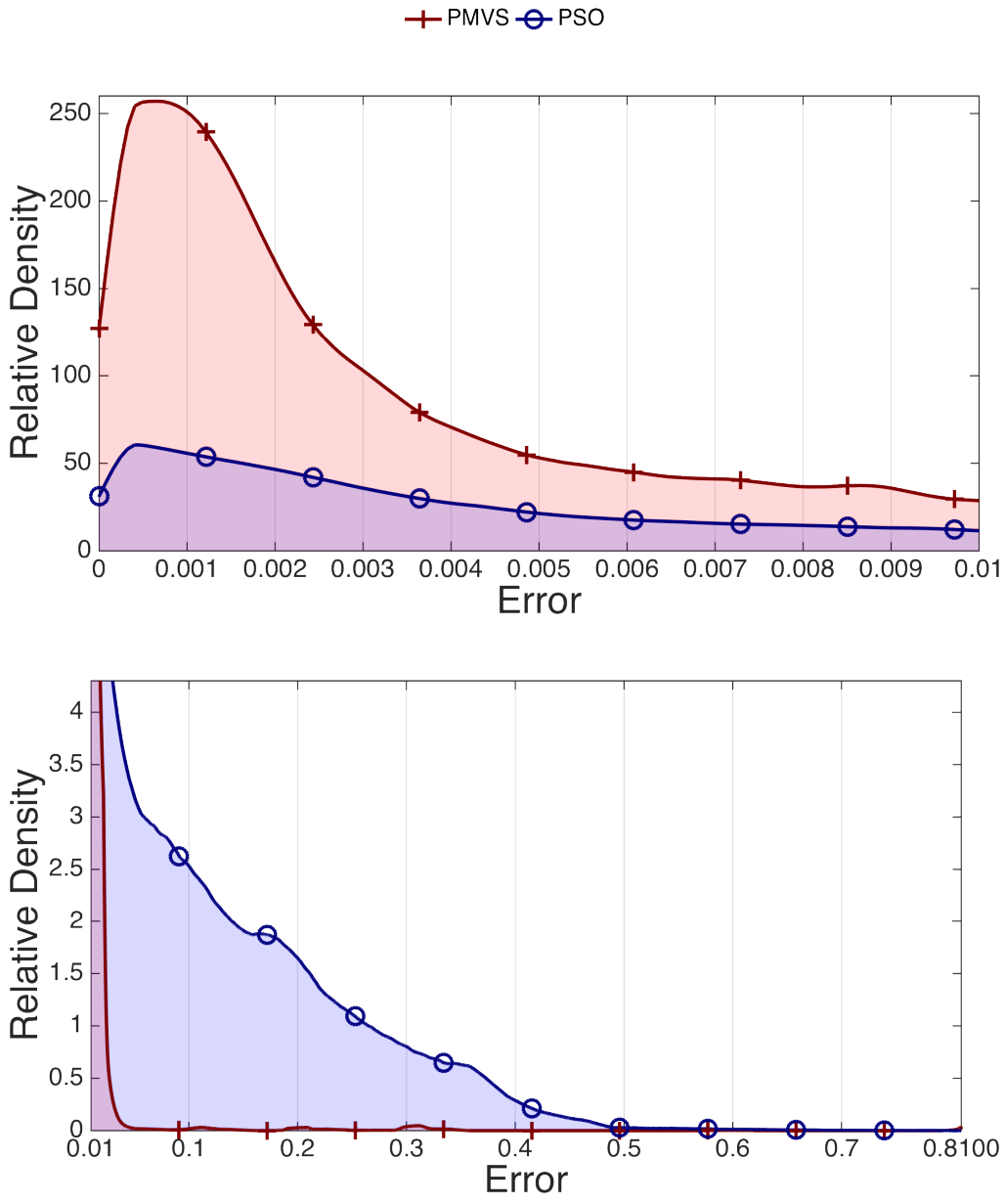


Figure 4.30: Probability distributions for the LynxCAD dataset
Top: Normalised probability distributions for the errors in patch refinement when running PMVS and PSO on the dataset.
Bottom: Zoomed in tail end of the distribution up to the largest error

	Mean	Median	Mode	Max
PMVS	0.0046	0.0023	0.0007	0.8100
PSO	0.1007	0.0592	0.0004	0.8060

	Residual Error							
	0.0	0.005	0.01	0.02	0.03	0.1	0.3	0.5
PMVS	100.00	27.72	7.80	0.96	0.53	0.38	0.16	0.01
PSO	100.00	79.58	71.90	64.90	60.24	38.95	7.42	0.30

Table 4.3: Statistics for the LynxCAD dataset
Top: Different averages of the reconstruction error.
Bottom: The inverse cumulative frequency of errors as a percentage of entire patch cloud

4.3.3 Speed

		PMVS Run Time (sec)	Total Patches inside PMVS Cloud	PSO Run Time (sec)	Total Patches inside PSO Cloud
Monkey	High	280	356120	883	461921
	Med	298	328743	1008	461921
	Low	257	193085	1013	461921
Buddha	High	221	250264	732	399178
	Medium	284	239817	879	399178
	Low	356	217628	879	399178
LynxCAD		1323	692776	4091	1805718
Dino		731	338430	971	525354
Temple		390	239161	578	327818
LynxReal		1670	494138	2550	1393262

Table 4.4: Refinement times for PMVS and PSO

Table 4.4 displays the run times and total number of patches created by running PMVS and PSO on each of the datasets respectively. All experiments were run under a 64-bit Mac OS X platform, on an Intel i7 Quad Core 3.4GHz Processor with 32GB RAM; running OpenCL on a mid-range GeForce GTX 680MX GPU with 2GB of main memory and 1536 Cuda cores. PMVS is capable of running patch refinement in parallel over multiple CPU cores, however, there is currently no GPU extension for this. Therefore, PMVS was run using 8 threads, essentially refining 8 patches via gradient descent in parallel, spreading work equally over the 4 hyper-threaded CPU cores, simulating our approach spreading work over the GPU.

The current implementation of the algorithm proposed in this chapter allows for one PSO optimisation to be run at a time, where the evaluation of each particle's fitness function is executed in parallel on the GPU. This allows for the quick refinement of one patch before moving onto the next, as such we do not optimise patches in parallel. Our GPU based approach is scalable in terms of running multiple PSOs in parallel, therefore refining more than one patch at a time, however, for simplicity this optimisation was not implemented. Running multiple PSOs in parallel via a GPU approach is easily achievable but optimal performance is highly dependant on the number of cores, memory, and clock speed of the GPU, thus to avoid overcomplicating the assessment presented here we chose not to do this.

Using our implementation of GPU based PSO, described in Section 4.1.3, we are able to refine a single 5x5 patch in roughly 2ms. The run time depends on the number of iterations of the swarm and the size of the patch. No doubt with a more highly optimised evaluation kernel which takes into account memory coalescing, Section 3.3.2, for the computation of the photo-consistency measure, these times could be further improved. Currently the kernel described in Section 4.1.3 evaluates the photo-consistency function for a single particle inside one work-group, therefore, each work-item computes the NCC with the reference view and one other view. This requires each work-item to read from a different image. Perhaps if this strategy was modified such that all particles who need to compute their NCC with a single view could be grouped into a work-group where each work-item would then read from the same image, local memory and memory coalescing could be used to potentially speed up the sampling and computation involved in computing the NCC.

Analogously to further optimisation of the PSO's evaluation kernel, the mid-range GPU used in this evaluation contains only 1536 cores, so is in fact sub-optimal for the current task. Nowadays top of the range GPUs such as the NVidia QTX Titan Z, have up to 5760 cores in total, [121]. Running our highly parallel algorithm on these types of graphics cards would allow for more evaluations to be run in parallel, and the possibility of refining many more patches at once, this would undoubtedly achieve significant speed improvements. The current trend in CPU architecture development is moving towards more processing cores but with lower clock speeds and efficient branching operations. Whereas GPU architectures are rapidly evolving towards more cores at higher clock speeds but less efficient branching instructions. This demonstrates the future proofed scalability of our swarm optimisation, and shows promise that speed increases will unquestionably increase over time.

Finally, as our PSO based approach creates significantly more patches than PMVS, therefore, its total run time is scaled respectively. Granted that many of the patches produced by PSO are erroneous, rather than remove or not compute these, the following chapters focus on improving their accuracy. The run times are presented here to provide a baseline to measure further algorithm speeds in the following chapters.

4.4 Discussion

4.4.1 Technique Synopsis

Given that PSO refinement of a patch only requires knowledge of an approximate depth range, there is no need to perform the matching and expansion procedures required for precise initialisation of the derivative based methods inside PMVS. Instead, for each silhouette pixel that passes the viewing angle test, a search space is automatically generated and a patch refined via GPU based PSO. The purpose of this refinement is to find the patch's optimal position and normal parameters.

The search space for each patch is defined by first extracting the visual hull, this was detailed in Section 4.1.2. The visual hull can be extracted by thresholding each input image to create approximate silhouettes. By defining a sparse voxel octree, the visual hull is created by projecting each voxel in the octree into the approximate silhouettes, using this to define each voxels surface label. Once the visual hull is defined it is projected into the silhouettes to remove incorrectly labelled silhouette pixels, as well as to define the front and back intersections and normal values for the visual hull surface. The normal is used to test which silhouette pixels pass the viewing angle test.

Section 4.1.3 details how PSO is implemented and specifically tailored to a MVS approach on the GPU, as well as the methods used to define its search space parameters. The intersection of the viewing ray to the visual hull is used to set the valid depth search range for the patch. Its normal search range is set such that the patch cannot become back facing in its reference image. Once the search space is defined there is enough information available to optimise the patch. Evaluation of this technique in Section 4.3 showed that the success of patch refinement is dependant on the levels of texture present.

Although our method shares similarities to depth map techniques, in that we attempted to find a depth value for each pixel, our search is not just limited to depth but also the normal. The main difference between the two approaches is that our technique is patch based, attempting to find the valid depth and normal parameters by observing all views at once, rather than defining the depth using binocular stereo and then merging this with depth maps from multiple views.

4.4.2 Results and Limitations

Quantitative and Qualitative results in Section 4.3 have shown that, from only a coarse search space initialisation, PSO has the ability to extract highly accurate and dense patch clouds, however, patch refinement is highly dependant on the texture resolution and currently corrupted by noise. Comparing the entire patch cloud to PMVS, demonstrates its ability to fully reconstruct high frequency surface details to a comparative accuracy, whilst moving towards an approach that improves the density of reconstruction. Our PSO method has produced denser patch clouds, with a large numbers of correct patches of compatible accuracy, validating its use as an optimisation technique, however, it also produces large amounts of noise in textureless regions demonstrating an area for further improvements.

Section 4.3.1 showed that when refining a patch in a well textured region, PSO has the ability to perform comparatively to the derivative based methods of PMVS, without requiring feature based initialisation and expansion. When stripping away noise in the PSO patch cloud returns a cloud remarkably similar to that obtained by PMVS. However, it also showed the large amount of noise and error PSO introduces in textureless regions, which PMVS is unable to move into. It showed how both algorithms accuracy diminishes as the texture reduces, reflecting the difficulties in refinement over textureless regions.

Section 4.3.2 showed that the peak accuracy of both approaches were remarkably similar, again demonstrating the potential of PSO to perform equally to PMVS. It was seen that the peak performance of PSO was only slightly smaller than PMVS, and we postulated that this reflected the stochastic nature of PSO not stepping exactly onto the global minimiser. However, PMVS can be seen to perform much better than PSO, in terms of the mean and median being smaller for PMVS than PSO, implying that PMVS consistently produced patches of lower error and noise. We argue that this is in some way expected, PMVS only captures well textured regions to a high accuracy and is unable to move into and refine patches in lower textured areas; whereas our PSO approach attempts to refine patches in both high and low textured regions, where refining patches in low textured regions is of lower accuracy and introduces large amounts of noise.

Our PSO method, finding the globally optimum patch parameters, has provided high quality reconstruction in textured areas. However, PSO alone starts to produce erroneous patches in the presence of low texture, where the globally optimum solution may no longer align with the correct patch parameters. Currently, these erroneous patches are left inside the patch cloud, creating noisy reconstructions. In order to improve this technique, these noisy patch configurations must either be improved or removed, moving towards a method equally as effective as PMVS whilst retaining density in low textured regions.

4.4.3 Improvements

Local patch expansion and refinement inside the framework of PMVS showed promise at better dealing with the boundaries of textureless regions. Expansion from neighbouring patches has a high probability of aligning correctly, such that the local optimum found by gradient descent is also the optimal surface representation. However, there is a strong likelihood of making small errors and over smoothing the finer details, this error is then propagated throughout the remaining expansion. Relying on local refinement in the expansion stages is one of the main reasons why holes appear at textureless region, here, gradient descent can fail and patches are removed. This effect has been seen predominantly throughout all the datasets presented.

Using expansion to restrict the search space in our PSO based technique, corresponds to clipping the search space such that the locally optimum solution has a higher chance of being globally optimal within the clipped range. Compared to PSO's larger search space this is more likely to succeed in these cases. However, there is also and a strong possibility that the correct parameters no longer lie inside the restricted search space, and there is still no guarantee that the global optimum now relates to the correct surface.

Our experimentation has shown that regardless of technique, patch refinement produces the best results when computed with textured patches, not when computed by restricting the search space. Therefore, to help resolve incorrect refinement, first well-textured regions should be defined, allowing the photo-consistency function to properly adapt to these regions, improving the clarity of the metric and the probability that the global

optimum remains correct. This should allow for a reduction in the noise of the patch cloud. Then a multi-resolution approach can be used to induce texture inside textureless regions. PSO can then be run at a coarser level, with potentially more texture, in these situations patch refinement should prove more accurate than relying on clipping from expansion alone. Results of refining patches at a coarser resolution should then be propagated upwards filling in holes and allowing for further refinement of high frequency data, not present in the coarser levels.

4.5 Summary

This chapter has introduced our preliminary PSO patch based MVS algorithm. It has outlined the following key principles and limitations of the approach:

- The visual hull can be extracted efficiently through our GPU based algorithm, the accuracy of which is not a problem. The intersection of each pixel's viewing ray with the visual hull can be used to initialise the search space of PSO.
- PSO proved to be a fast and effective technique for finding the globally optimal parameters for each patch, providing a feature free initialisation for each patch's search space and removing the need for patch expansion procedures.
- Our PSO based approach performs equally to PMVS in textured regions allowing for accurate and dense reconstruction of high frequency details. However, it is limited in its performance inside textureless regions, where our approach creates noisy reconstructions

5

Enhancing the Similarity Metric: Photo-Detail-Consistency

Previously, in Chapter 4, the use of PSO to globally optimise a patch's parameters based on the photo-consistency function was proposed. This proved to be highly effective at creating accurate patch clouds that were significantly denser when compared to PMVS, however, suffered from noisy patches where refinement failed. Refinement fails when the global optimum, found by PSO, no longer aligns properly with the correct position and orientation of the surface. In these cases local refinement offers a more suitable solution than global, at the cost of the overall accuracy of reconstruction.

To develop this technique further, two stages of enhancement were proposed, Section 4.4.3: first, the improvement of the photo-consistency score, in order to increase the amount of correctly refined patches; secondly, the inclusion of a multi-resolution filtering step, in order to

further increase the robustness of refinement and remove poorly refined patches. This chapter focuses on the former, the improvement of the per patch photo-consistency function.

In this chapter, a photo-detail-consistency function is defined which provides an adaptive approach to the standard photo-consistency function, Equation 3.6. The purpose of using this adaptive photo-detail-consistency metric when optimising a patch's parameters is to improve the likelihood of the global optimum, along the entire viewing ray, aligning correctly with the underlying surface position and orientation properties. Therefore, patch optimisation, via PSO, should succeed more often, reducing the amount of incorrectly refined patches. This shares similarities with patch expansion in PMVS, where using a neighbouring patch as initialisation improves the likelihood of local refinement succeeding.

The photo-detail-consistency measure is defined by taking into account two different factors that impact the effectiveness of the photo-consistency function. These two factors being the texture over the patch, and the angle at which the patch is viewed, defined by the angle between the patch normal and reference view. In this chapter the improvements to the clarity of the photo-detail-consistency function are described in detail, again providing qualitative and quantitative results against both PMVS and the provisional PSO algorithm. Results have shown a notable improvement in the number of correctly refined patches and a further marked increase in the overall accuracy of patch refinement. Unfortunately, the adapted photo-detail-consistency function does not overcome incorrect refinement in all cases, where some noisy patches are still produced under textureless circumstances. Further filtering steps are still required, and are presented in the following chapter.

This chapter begins in Section 5.1, by defining our novel photo-detail-consistency function, describing both factors that are considered and reasons why they are effective. Brief analyses of surfaces that break the Lambertian assumption are also discussed and reasons why no current photo-consistency metric can deal with such situations are described. After presenting the details of our similarity metric, Section 5.2 presents qualitative and quantitative results, analysing the improvements this adaptive technique makes to our preliminary PSO algorithm and when compared to PMVS. Section 5.3 presents a brief discussion of these results, concluding the chapter with a summary in Section 5.4.

5.1 The Photo-Detail-Consistency Metric

5.1.1 Defining Photo-Detail-Consistency

The NCC, defined in Section 3.2, takes into account additive and multiplicative illumination changes by normalising away the effects of such changes over the patch. It is generally accepted as the optimal similarity measure for use in multi-view environments, [73, 130, 138]. However, as previously discussed the NCC fails to return information about a patch's similarity when the texture over that patch is low. In such low textured cases, the colour variation over the patch is approximately constant, causing the denominator in the NCC to become zero, hence no information is returned. PMVS accepts that this limitation occurs and uses a filtering step to remove patches that were incorrectly refined when the texture is low and the measure fails to produce accurate results. In this section, a more active approach to dealing with cases where the NCC can fail is taken to improve its performance.

The limitations of the NCC, have been a long-standing problem within the single view, binocular stereo community. Attempts at improving the NCC score for the stereo case have been explored in [72, 166, 170], however, these are yet to be fully extended into a MVS setting. Chen et al. [26] borrowed some of the geodesic support theory to adaptively weight each pixel inside their consistency function, however, as with the stereo case, this is more complex and requires a high level of computation. To continue to achieve high accuracy with low computational cost, we assess further the limitations of NCC to enhance the score without deteriorating the speed of reconstruction.

The photo-consistency measure, previously defined in Equation 3.6, can suffer similar anomalous results when comparing highly textured patches as it does in the textureless case. If the NCC is computed per image channel, and one or more of those image channels have low texture, these channels will return incorrect results or a poor measure of the patch fit. This effectively skews the similarity score when summed and averaged with the results from the highly textured channels.

An example of this anomaly is displayed in Figure 5.1, where an image from the MonkeyMed dataset is taken and split into its red, green and

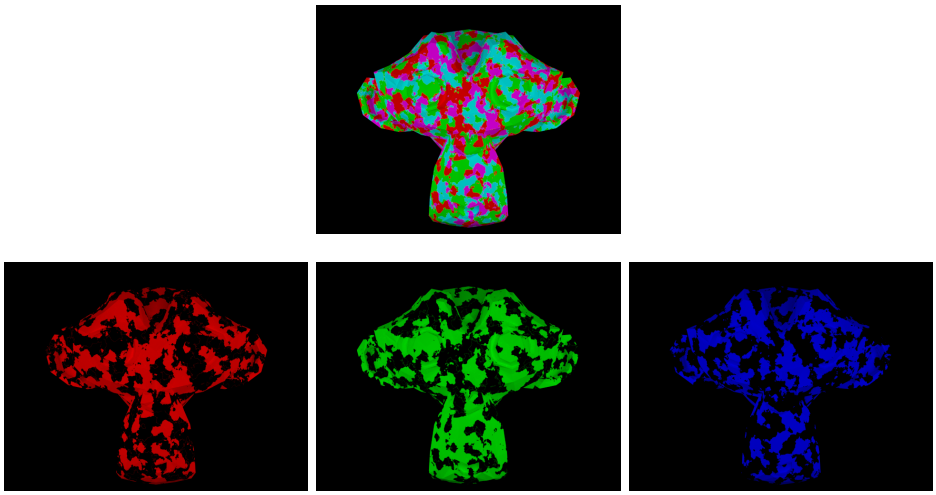


Figure 5.1: An example the textureless effect seen across image channels

blue colour channels respectively. For many points over the image, where there is texture on the red channel, there is none on the green and blue channel, or vice versa. At these locations, although the red channel would return accurate NCC results, the green and blue channels would return smaller or even zero valued results. When these are averaged to compute the total photo-consistency score, the true global optimum is less well defined, with shorter and smoother peaks. In the worst-case scenario, the correct position and orientation are potentially no longer globally optimal. This can happen when moving the location of the patch results in the inclusion of some texture on the green and blue channels, producing a better average score than that of the score at the correct location with the red channel alone.

Further to this, when a patch is projected into an image where the angle between the patch and image plane is high, bilinear interpolation is used to sample the patch, Figure 4.1b. In cases where the patch axes fail to cover sufficiently many pixels in the image based matching window, bilinear interpolation returns similar values, creating a textureless effect. Similarly to the averaging of good colour channels with poorly textured ones, averaging the results over multiple views, where some views have a poor viewing angle to the patch, further distorts and offsets the photo-consistency function. The naive solution to this is to simply ignore all views whose viewing angle to the patch is below some threshold; however, this does not scale well to sparse view sets where viewing angles are likely to be increasingly shallow as the sparsity is raised.

$$\gamma(p) = 1 - \frac{1}{|V(p)/R(p)|} \sum_{v \in V(p)/R(p)} \mathbf{w}_{a,v} \sum_{c \in \{r,g,b\}} \frac{\mathbf{w}_{t,c}}{||\mathbf{w}_t||} NCC_c(I_{R(p)}, I_v, p) \quad (5.1)$$

Equation 5.1 proposes adaptively weighting the NCC score for each patch based on texture and viewing angle weights $\mathbf{w}_t$ and $\mathbf{w}_a$. This new similarity metric is defined as the *photo-detail-consistency* metric.

The amount of texture over a patch, projected into its reference view, can be computed and used to weight, via $\mathbf{w}_t$, the likelihood of each colour channel returning a bad result. The weighting is then normalised across the three colour channels of the patch using, $||\mathbf{w}_t||$. Where $\mathbf{w}_{t,c}$ is the c^{th} component of $\mathbf{w}_t$, relating to colour channel c. The normalisation of $\mathbf{w}_t$ avoids the weighting converging to zero in the presence of low texture.

Once a patch's parameters are defined the set of views in which the patch, p, is visible, $V(p)$, can be found, as described in Section 4.1.1. By observing the angle each image plane in the visible image set, $V(p)$, makes with the patch normal, cases where bilinear interpolation fails to accurately extract sufficient patch samples can be accounted for by weight $\mathbf{w}_a$. The viewing angle weight, $\mathbf{w}_a$, is not normalised as the number of views a patch is visible in changes per patch configuration. This can cause instabilities in such a weighting, when small changes to the patch orientation can result in a large change to the size of its visible view set and thus in it's weighting. In the equation above $\mathbf{w}_{a,v}$ is the v^{th} component of $\mathbf{w}_a$, relating to the weight for view v.

Texture Weight

Texture weight $\mathbf{w}_t$ prevents textureless colour channels contributing to the similarity metric. Thus it is close to zero when the texture on an image channel is low and high when there is sufficient texture. A simpler approach, without weighting colour channels, would be to normalise the concatenation of each colour channels sampled points, into a single sampled vector. The NCC would then take the inner product of the two larger patch vectors, producing a value between -1 and 1, rather than averaging them to form the final photo-consistency score. However, this does not take into account the performance benefits of well-textured colour channels against low textured channels, or the fact that texture further from the patch centre is less reliable than that at the centre of the patch.

$$\mathbf{w}_t = 1 - e^{-\tau^2/2\sigma_t^2} \tag{5.2}$$

where

$$\tau = \sum_{(x,y)\in\Omega(p)} d\|\nabla I_{R(p),c}(x,y)\|$$

Texture weight, $\mathbf{w}_t$, in Equation 5.2, is computed by observing the texture, over the patch, p, projected into its sample window, $\Omega(p)$, in its reference view, $R(p)$, for each of its colour channels, c. These texture samples are then weighted by distance, d, and summed to form a scalar value, τ, quantizing the texture over the patch, which is then mapped to the texture weight, $\mathbf{w}_t$. In all our experiments, σ_t was set to 0.2, as this produced good results across the board. It is possible to tune this parameter on a per-patch basis; however, this would require a large amount of computation, heavily increasing the run time.

Under the Lambertian assumption, the texture over the patch can be represented by the change in gradient of the input image. The gradient of the reference image at any point (x,y), for the colour channel, c, is denoted $\nabla I_{R(p),c}(x,y)$ in Equation 5.2. In practice, for the highest resolution, the response of passing a Sobel edge detection kernel, [106], over the image is used to approximate the image gradient. For lower resolution images, the detail coefficients of the Haar wavelet transform, inside a multi-resolution approach, presented in the following chapter, are used as gradient approximation as these are readily available and reduce the computation requirements.

The unknown shape of the object and viewing angle to the surface largely defines the shape and visibility of the texture over the patch. Image gradient further from the centre of the patch is likely to change throughout the view set, as the patch refines its position and orientation; however, texture close to the centre of the patch should remain consistent as this reflects the exact position of the patch sitting on the surface. To account for this factor, image gradient closer to the centre of the patch is given preference in the patch's total texture response, τ. Therefore each sample over the patch computes its gradient which is weighted by its spatial distance, d, from the centre of the patch, (x_c, y_c). Akin to the geodesic weighting proposed by Chen et al. [26], Equation 5.3 defines

how to compute d, where μ is the patch size and (u, v) are the image space sample locations.

$$d = \frac{e^{-\lfloor \mu/2 \rfloor^{-1}\sqrt{2((x-x_c)^2+(y-y_c)^2)}}}{\sum_{(u,v)\in\Omega(p)} e^{-\lfloor \mu/2 \rfloor^{-1}\sqrt{2((u-x_c)^2+(v-y_c)^2)}}} \tag{5.3}$$

Viewing Angle Weight

To help reduce the influence of views with sharper viewing angles to the patch, as such situations where bilinear interpolation produces textureless samples, the weighting factor, $\mathbf{w}_a$, is computed. Each component of $\mathbf{w}_a$ is calculated based on a single view, v, from the set of visible images, $V(p)$, using the patch's normal, $\mathbf{n}(p)$, and the current view's principle view direction, $\mathbf{vd}(v)$. Therefore, $\mathbf{w}_{a,v}$, the v^{th} component of $\mathbf{w}_a$, represents the weighting for a single view.

$$\mathbf{w}_{a,v} = \begin{cases} 0 & \text{if } \mathbf{n}(p)\cdot\mathbf{vd}(v) > 0 \\ 1 - e^{-(\mathbf{n}(p)\cdot\mathbf{vd}(v))^2/2\sigma_a^2} & \text{if } \mathbf{n}(p)\cdot\mathbf{vd}(v) \leq 0 \end{cases} \tag{5.4}$$

The equation above reflects the fact that a patch is observed, and sampled properly by a view, when the patch axis aligns adequately with the view's own axis. This will occur when the patch's normal and view's principle view direction point in opposite directions. The closer these two vectors are to aligning, such that they point in exactly opposite directions, the better the quality of patch sampling that can be obtained. When the two vectors point in the same direction, such that the angle between them is less than ninety degrees, the inner product will be positive. Under these circumstances the patch will be back facing in the current view, therefore, should not be sampled or included in the photo-detail-consistency score, so is given a weight of zero. When the patch is not back facing in the current view, the weight is calculated based on the inner product. When the inner product is close to minus one, the axis alignment is close to being optimal, hence this view is given a larger weight. When the inner product is close to zero, there is a shallow viewing angle to the patch, increasing the probability that bilinear interpolation will fail, hence these views are given a lower weight. Like σ_t, factor σ_a is a tuneable parameter controlling the strength of the filtering; smaller values represent stricter filtering. In our experiment σ_t and σ_a are both fixed at 0.2.

5.1.2 Non-Lambertian Reflection

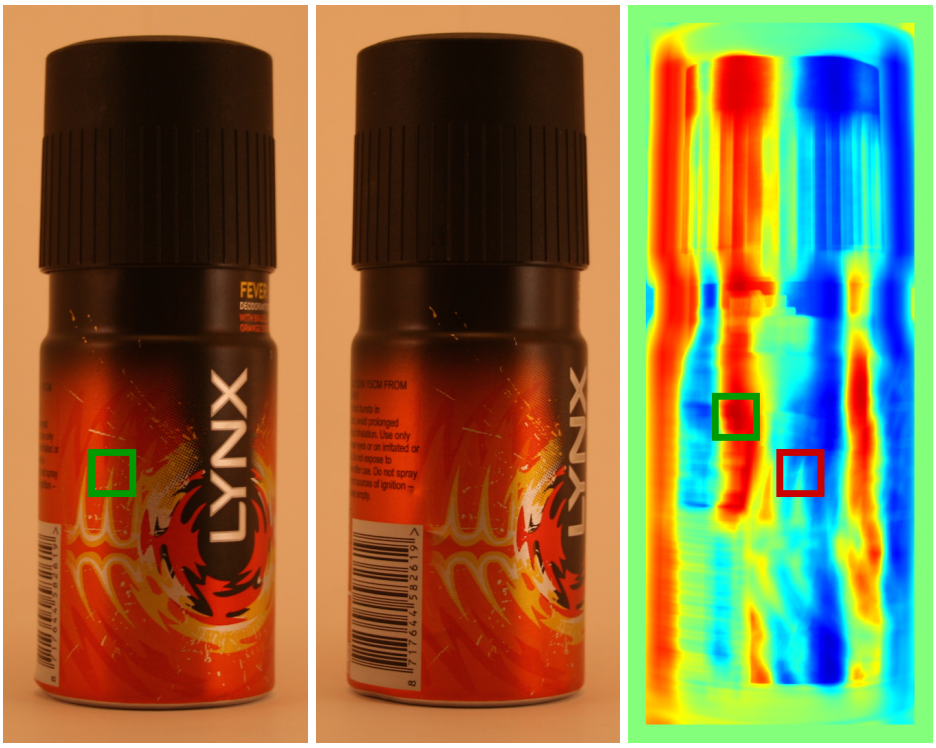


Figure 5.2: Non-Lambertian reflection distorting the true similarity metric
Left: The reference image with patch marked by a green box.
Middle: The comparison image
Right: The NCC score, red is strong matches, blue is weak matches.
The green box marks the highest NCC score, whereas the red box marks the true solution

A limitation of our photo-detail-consistency metric, or any other metric relying on NCC, is that reflection models that do not cause additive or multiplicative illumination changes throughout the image set are not accounted for, such models being highly complex to deal with. Currently, triangulating features or patches throughout an image set relies on matching textured patches with each other. However, non-Lambertian reflection causes illumination changes that can be seen as a textual change throughout the image set. These illumination changes cause step edges and gradient changes in the corresponding texture of the image. Stripping away such artefacts would be simpler given the underlying shape, however, since this is what's being refined the shape is often highly distorted by the false matching induced around these areas.

The LynxReal dataset is a prime example of the problems caused by surfaces that do not obey the Lambertian reflection assumption. Its glossy, highly specular surface properties cause two thick white strips

and a darker section to appear based on the angle that the Can is viewed from. This is worsened by the fact the position of the light moves with the view. This can clearly be seen in Figure 5.2, where although the two views of the Can are taken from different orientations, the thick white and black strips remain constant. The change in illumination across the Can's surface clearly induces false step edges; as well as this there is gradient induced as the reflection drops off. This adds extra features, not related to the shape of the Can, which will be matched as if they were.

Figure 5.2 displays the result of taking a patch in a reference view and computing the NCC score at every pixel in the comparison view. The assumption being that the strongest NCC score relates to the position in the comparison image where the equivalent surface patch is located. Clearly the stronger matches, indicated in bright red, lie not around the correct surface position, but rather the position in the image that has the most similar illumination properties. Of course this is a rather contrived example, as the search space for the optimal NCC would not normally cover the entire image, however, this demonstrates the issues more clearly. Our PSO approach would only search the projection of the viewing ray in the comparison image, and PMVS would only search a very local subsection of this viewing ray. Therefore, patch refinement in these circumstance is still possible, all be it only if the viewing ray does not intersect the areas in the image where illumination changes match more highly than the underlying surface texture.

In particular, the top black section of the Can lid contains noticeable illumination issues. Figure 4.27 clearly showed how PMVS feel victim to these non-Lambertian effects. This led to patches being created inside the Can, where the texture induced by the illumination matched with a high accuracy. However, this position did not relate to the underlying surface properties. Both PSO and PMVS were able to extract the correct position for patches over the bottom section of the Can. For PMVS, its feature matching scheme and local expansion meant that the local optimal position was still correct, despite the global optimum being far from correct. For PSO, in many areas over the Can the true texture of the object is more prominent than that of the texture induced by its illumination, hence, the NCC can still succeed in these areas.

5.2 Experimental Results and Analysis

In this section, the improvements made by optimising a patch's parameters via our adaptive photo-detail-consistency function are analysed. The aim of this improved photo-detail-consistency function is to increase the number of correctly refined patches, reducing the noise in the final PSO patch cloud. Analysis is formed for both the synthetic and real life datasets, equivalently to that presented in the previous chapter. The results have shown that using our improved photo-detail-consistency metric over standard photo-consistency, not only significantly increases the number of correctly refined patches, but also greatly enhances the fidelity of the entire reconstruction.

From this section onwards, the use of the photo-detail-consistency metric inside our PSO based framework will be abbreviated to PSOPD. Whereas PSO alone implies using the same framework, however, optimising the standard photo-consistency function in place of the photo-detail-consistency. PSO generated the results in the previous chapter. All parameters for the swarm optimisation in PSOPD are kept consistent with those used for PSO, Section 4.3.

As in the past chapter, qualitative results are presented, these highlight the improvements made by PSOPD over the preliminary PSO algorithm already defined. Patch clouds generated are visibly less noisy and clearly of higher accuracy than the patch clouds generated using photo-consistency alone. A quantitative analysis confirms that our improved similarity metric is capable of achieving a higher level of accuracy, when compared to the standard metric used in PSO. These quantitative results further validate the PSOPD in comparison to PMVS, showing that, given its added density, they are in fact equally as accurate, especially when lowering the texture. Although incorrect patch refinement still occurs over large textureless areas, it can be seen that PSOPD allows for patches to better adapt to the underlying surface properties, extracting high frequency data not present in either our preliminary PSO algorithm or, more significantly, when compared to the patch clouds generated by PMVS. Analysis of the speed of reconstruction, shows that even though there is more computation involved in calculating the photo-detail-consistency metric, this still produces run times within an acceptable quality to PMVS and PSO.

5.2.1 Qualitative

Figures 5.3 to 5.8 show the patch cloud reconstructions for each of the synthetic and real life datasets, comparing the results of using PSOPD to the preliminary PSO framework. Chapter 4 determined that as texture is reduced density and sparseness are not affected in our PSO approach, only that more noisy and erroneous patches are introduced; this also occurs when increasing the image resolution. Therefore, in this section we will assess the improvements achieved by using PSOPD in terms of the following factors: The reduction in patch error and noisy refinement, as well as the effect on high frequency detail, accuracy and smoothness when texture is reduced.

Noise and Error

The photo-detail-consistency function used in PSOPD has clearly outperformed the standard photo-consistency metric used by PSO, showing much higher quality results. The focus of this photo-consistency metric is on improving the quality of reconstruction by reducing the frequency of noise or badly refined patches. It is, however, infeasible for any metric to ensure that patch refinement will succeed in every case, especially in low textured circumstances. Filtering the noise, removing patches that are inevitably poorly refined, was not a criterion of this chapter; therefore, some areas of the patch clouds still remain affected by noise. Akin to the results of running PSO against PMVS, our PSOPD reconstructions remain dense, largely thanks to our patch-per-pixel approach. No gaps appear in the head or chin sections of the monkey, Figure 5.3, and an explicitly denser reconstruction is obtained when compared to the results of PMVS in the previous chapter.

Results presented in the following figures clearly show significant improvements in terms of noise reduction. There are visibly much fewer badly refined patches in our PSOPD approach, than compared to PSO. In particular, observe the reduction in noise inside the ears of each of the Monkey datasets, or the legs and side sections of the Buddha, Figures 5.3 and 5.4 respectively. The Lynx texture and symbols over both the LynxCAD and LynxReal datasets, Figures 5.5 and 5.8, show a marked improvement in noise corruption, where the word “fever" and text on the LynxReal dataset is distinctly of higher quality.

Akin to Chapter 4, Figures 5.9 to 5.11, display the patch clouds created for the synthetic datasets, where ground truth is available, filtered such that all patches whose distance is greater than 0.005 units from the ground truth are removed. As no ground truth is available for the real life datasets, Figures 5.12 to 5.14 display images of the patch clouds generated by only a single view from each of the datasets.

These figures better highlight the improvements in terms of the reduction in noise when using PSOPD against PSO. For all datasets, patch clouds generated by PSOPD are consistently denser after filtering than that of PSO, showing that more patches are correctly refined when using the photo-detail-consistency metric. This can be seen most clearly in the base of the Buddha dataset, Figure 5.10, in particular notice how the large green areas are densely reconstructed as opposed to PSO alone. Also the single view of the temple, Figure 5.13, has significantly fewer badly refined patches, seen in the increased sparsity of patches that do not align with the correct surface. Both the Dino and Temple datasets, Figures 5.12 and 5.13, responded well to the improved photo-detail-consistency metric, greatly reducing erroneous patches, seen most predominantly around the planar regions of the Temple or base sections of the Dino model.

It is harder to see the noise reduction improvement in the LynxCAD or LynxReal datasets, Figures 5.11 and 5.14, mostly caused by the significant amount of textureless patches induced by the high image resolution. Nevertheless, large improvements are made, for example over the blue textured section and top black section of the LynxCAD dataset, which are now much more densely recreated. Although less improvement has been made in terms of increasing the density of highly accurate patch reconstruction for the Monkey datasets, Figure 5.14, nonetheless, both the high and medium textured versions remain densely reconstructed. The low textured Monkey dataset is challenging to reconstruct given that large areas are both planar and low textured, however, some good improvements in terms of density have been made which can be seen in the green and red sections on top of the monkeys head.

High Frequency Details, Accuracy and Smoothness

It is clear to observe, that at the same time as reducing noise in the reconstruction, the accuracy of previously well-acquired surfaces is visibly improved. Use of the photo-detail-consistency metric means patches have the ability to better adapt to the detail frequency of the underlying surface. Note the more accurate acquisition of the Buddha's face and chest sections, most noticeable in low textured circumstances inside Figure 5.4. Comparing the PSOPD Dino reconstruction with its equivalent PSO reconstruction, Figure 5.6, the high frequency variation over the surface of its fins are clearly better defined, although noise still distorts some of this information. Looking at the temple dataset, Figure 5.7, the panels and step sections are again of much higher fidelity, far surpassing the levels of detail acquired by PSO alone.

The Monkey and Buddha datasets, Figures 5.3 and 5.4, demonstrate the effect of reducing the level of texture on the ability to correctly refine patches. It can clearly be seen that, analogously to PSO, as the texture over the surface is reduced the number of incorrectly refined patches increases. The level of texture still clearly impacts the effectiveness of our similarity metric; however, the amount of noise is far less severe when using our adaptive photo-detail-consistency metric, as opposed to standard photo-consistency. The top section of the Monkey's head and the facial details of the Buddha, shows that even in low textured settings, accurate patch refinement in most circumstances is still possible, and of better quality than PSO.

The LynxCAD and LynxReal datasets, Figures 5.11 and 5.14, still pose a significant challenge for reconstruction techniques, mainly due to their higher resolution and largely textureless regions. With a small, 5x5, patch size and high image resolution, textureless areas in the Lynx Can examples are worsened. As such, both our PSO and PSOPD methods are affected by a large number of incorrectly refined patches. PSOPD can be seen to offer a more robust solution when compared to PSO alone, better capturing surface details. Although, without increasing the patch size or filtering erroneous patches, large errors still persist. The incorrectly reconstructed point on the base of the LynxCAD dataset shows this.

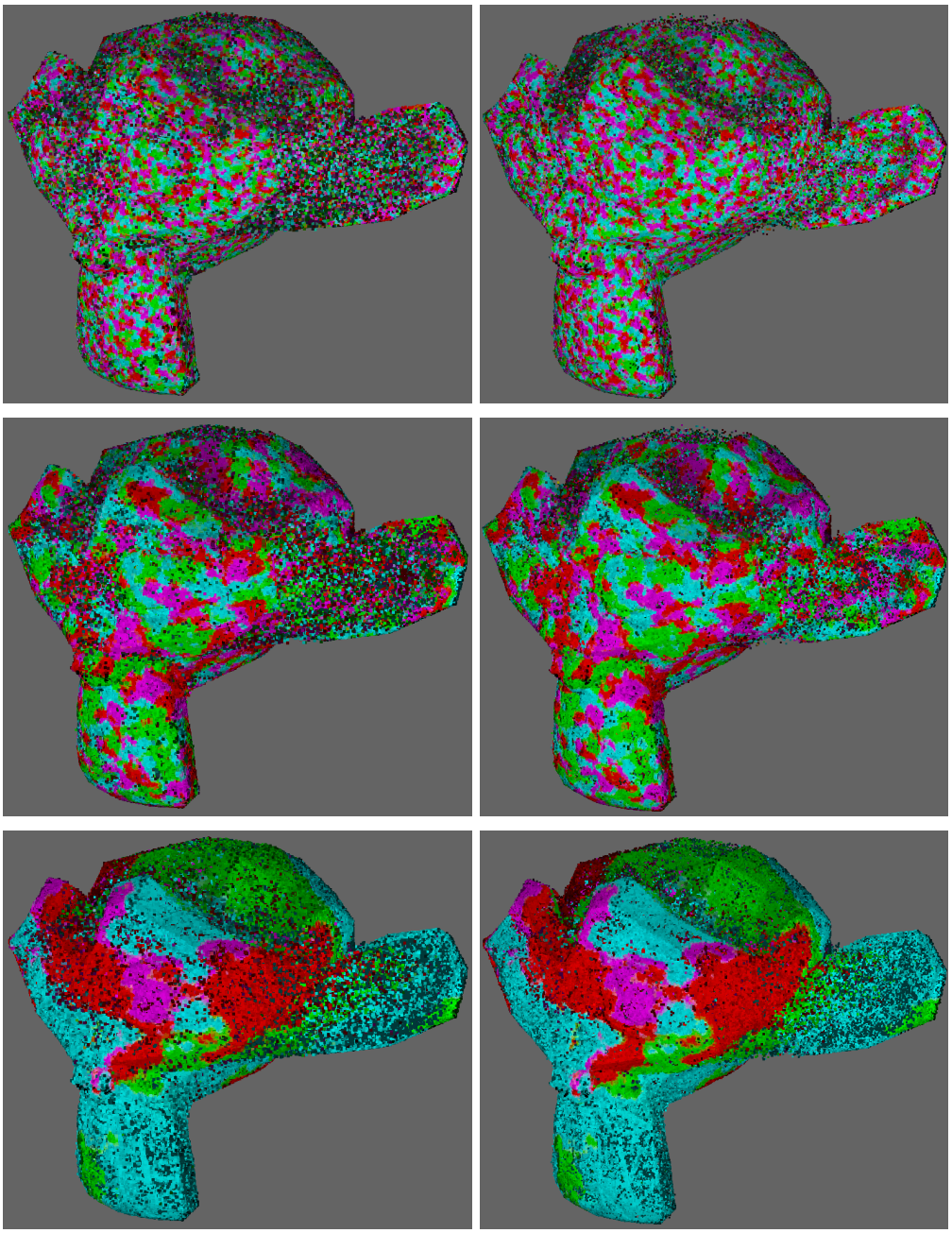


Figure 5.3: Reconstruction results for the Monkey dataset
with varying levels of texture.
Left Column: PSO results. **Right Column**: PSOPD results

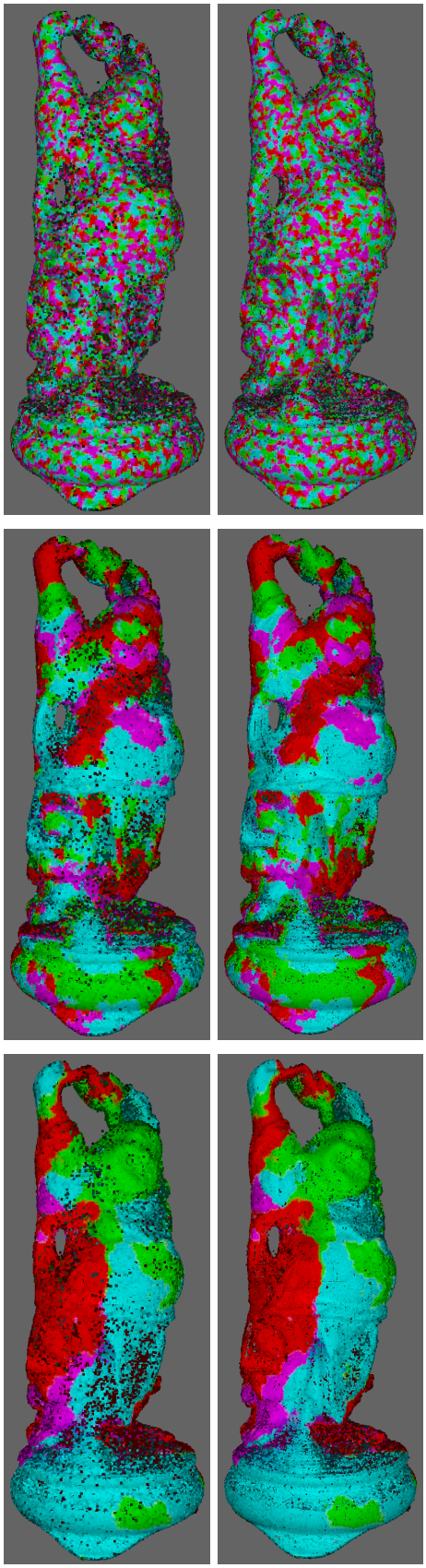


Figure 5.4: Reconstruction results for the Buddha dataset
with varying levels of texture.
Left Column: PSO results. **Right Column**: PSOPD results



Figure 5.5: Reconstruction results for the LynxCAD dataset
Left Column: PSO results. **Right Column**: PSOPD results

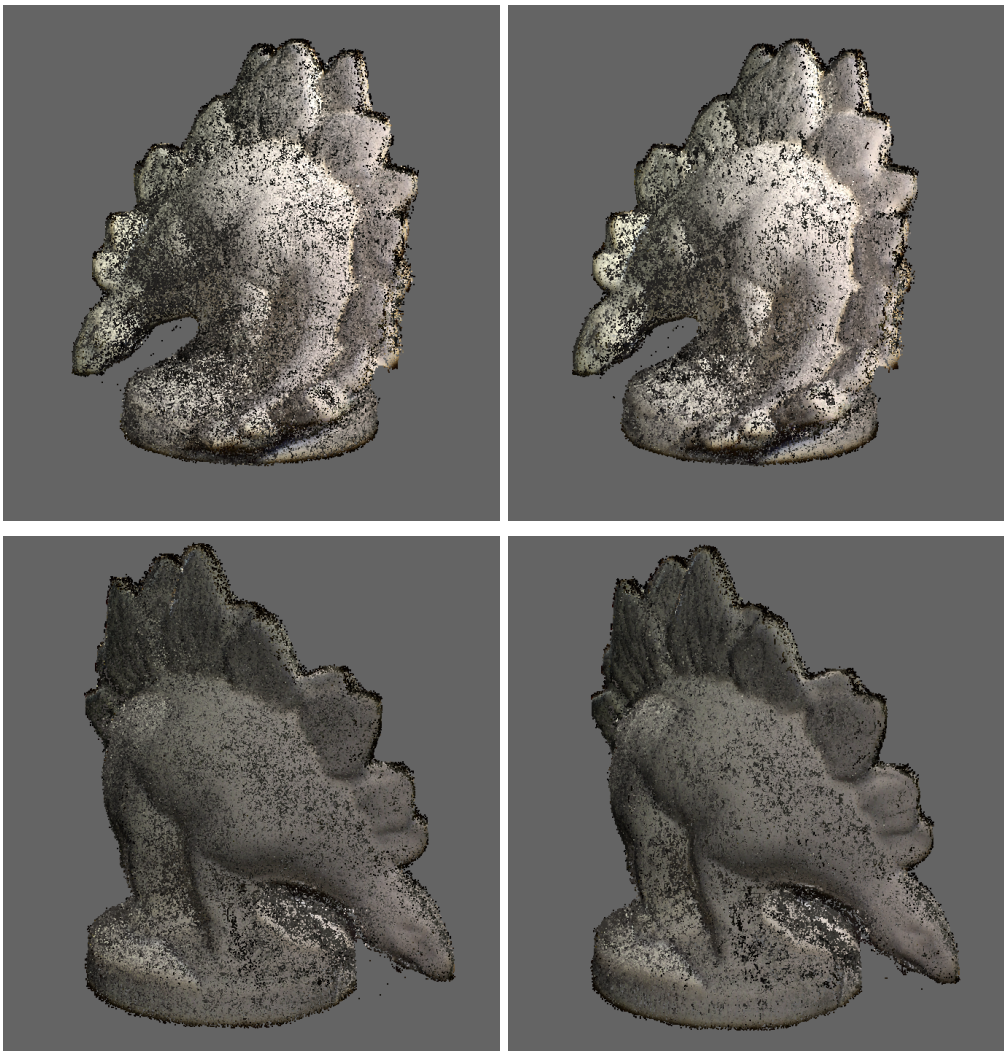


Figure 5.6: Two views of the reconstruction results for the Dino dataset
Left Column: PSO results. **Right Column**: PSOPD results

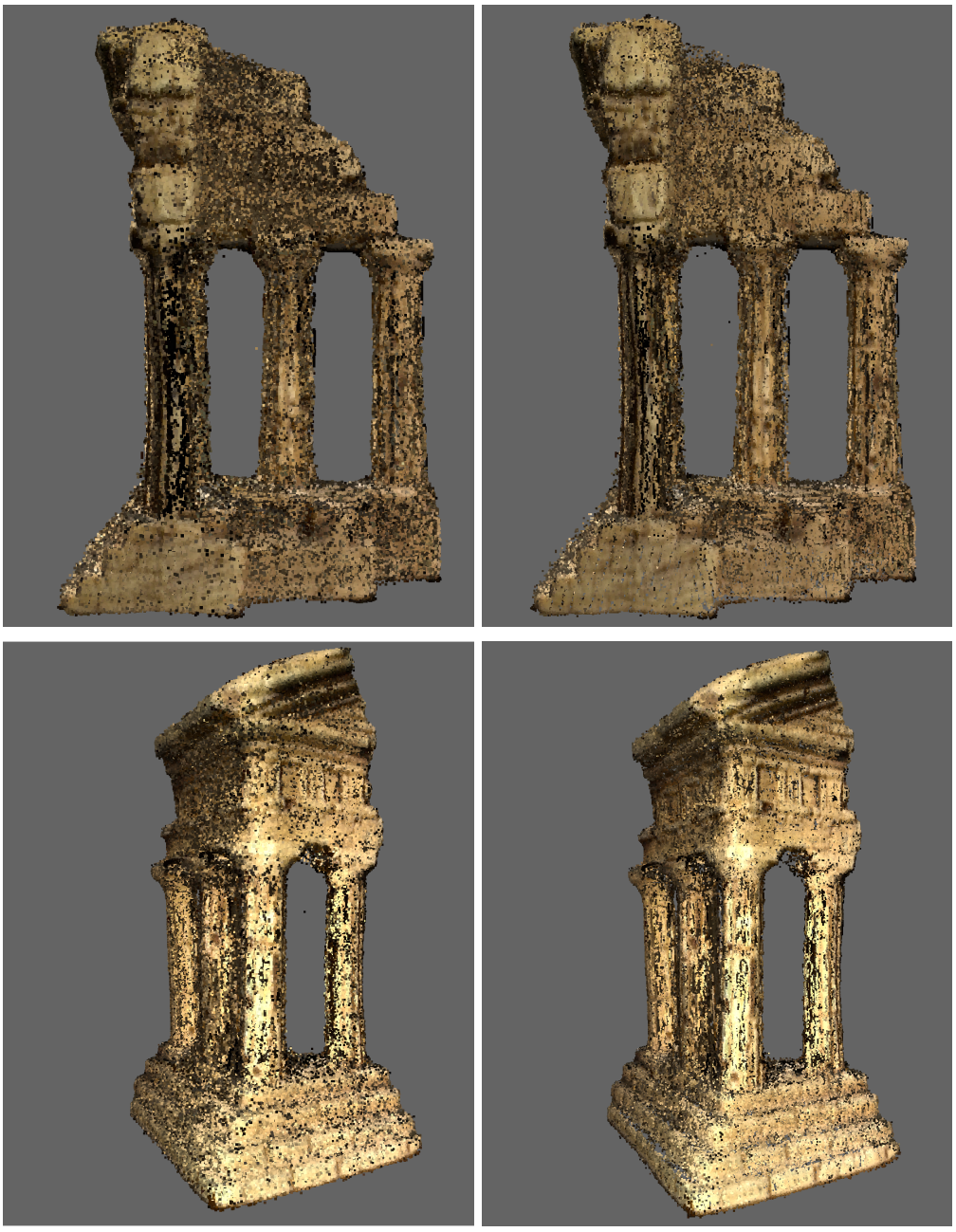


Figure 5.7: Two views of the reconstruction results for the Temple dataset **Left Column**: PSO results. **Right Column**: PSOPD results



Figure 5.8: Two views of the reconstruction results for the LynxReal dataset
Left Column: PSO results. **Right Column**: PSOPD results

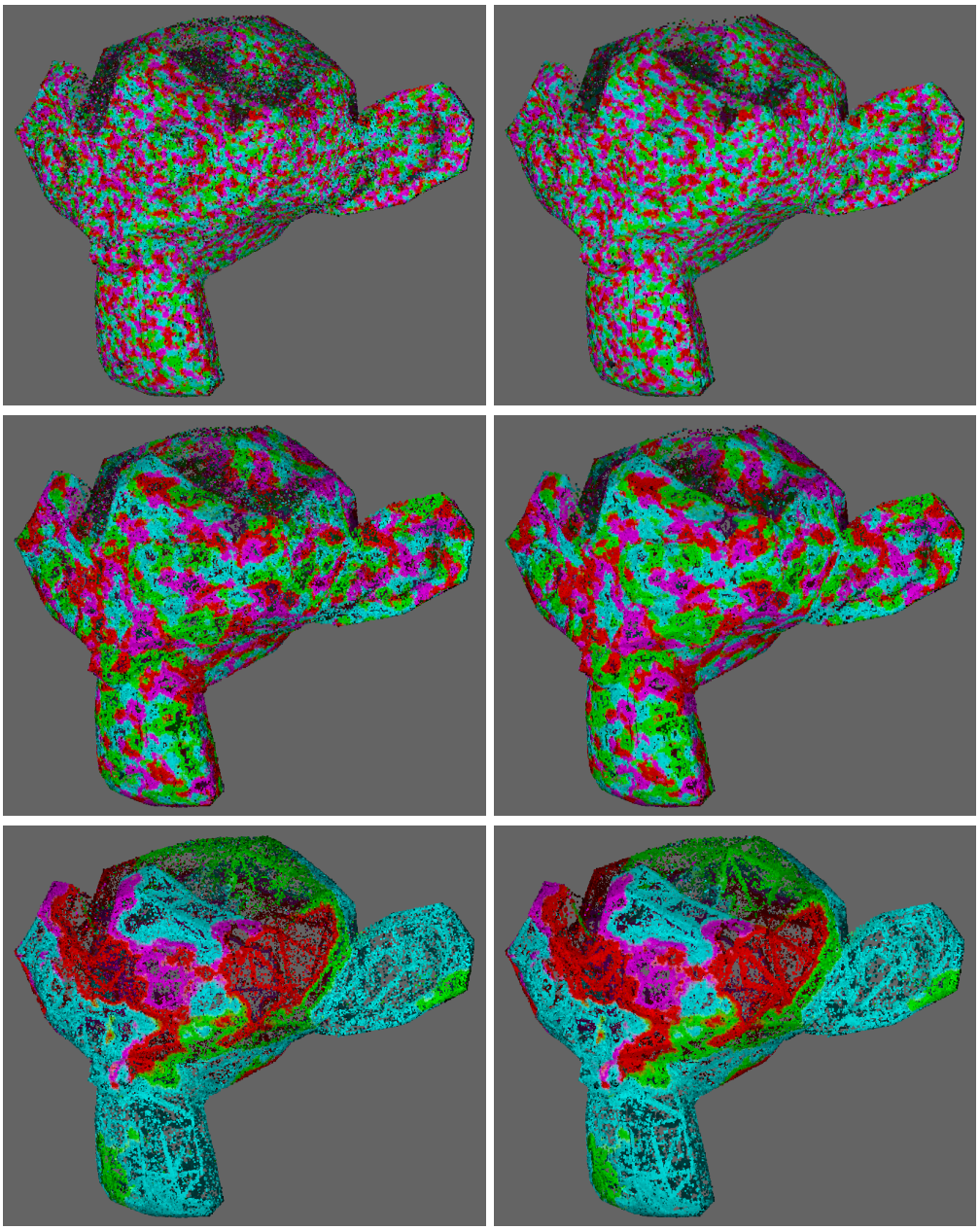


Figure 5.9: Noise reduced patch clouds created for the Monkey datasets by PSO (left) and PSOPD (right), these are filtered by removing all patches in the cloud whose distance is more then 0.005 units from the surface

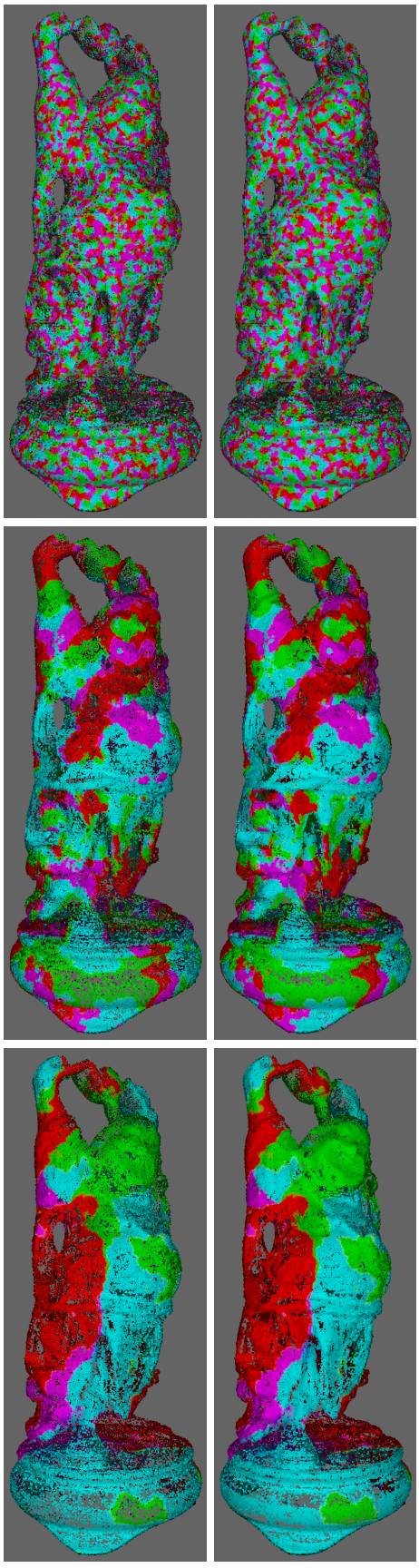


Figure 5.10: Noise reduced patch clouds created for the Buddha datasets by PSO (left) and PSOPD (right), these are filtered by removing all patches in the cloud whose distance is more then 0.005 units from the surface



Figure 5.11: Noise reduced patch clouds created for the LynxCAD dataset by PSO (left) and PSOPD (right), these are filtered by removing all patches in the cloud whose distance is more then 0.005 units from the surface

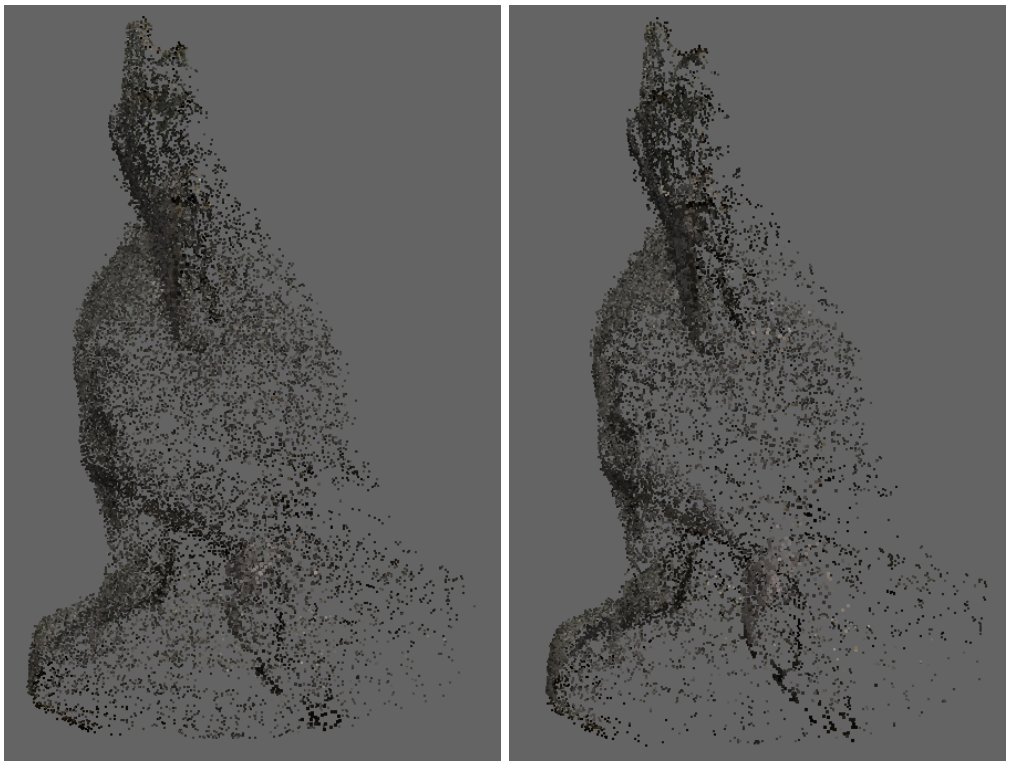


Figure 5.12: Patches created by PSO (left) and PSOPD (right) for a single view from the Dino dataset

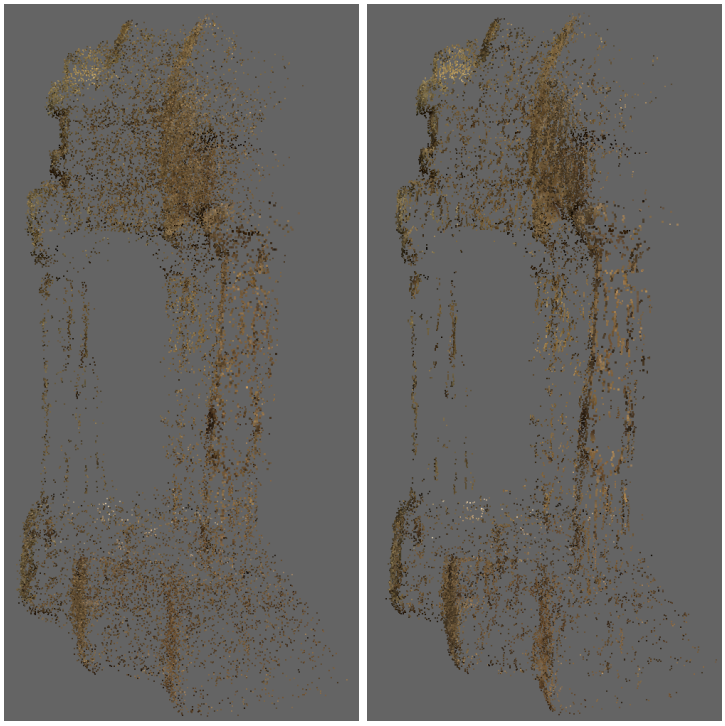


Figure 5.13: Patches created by PSO (left) and PSOPD (right) for a single view from the Temple dataset

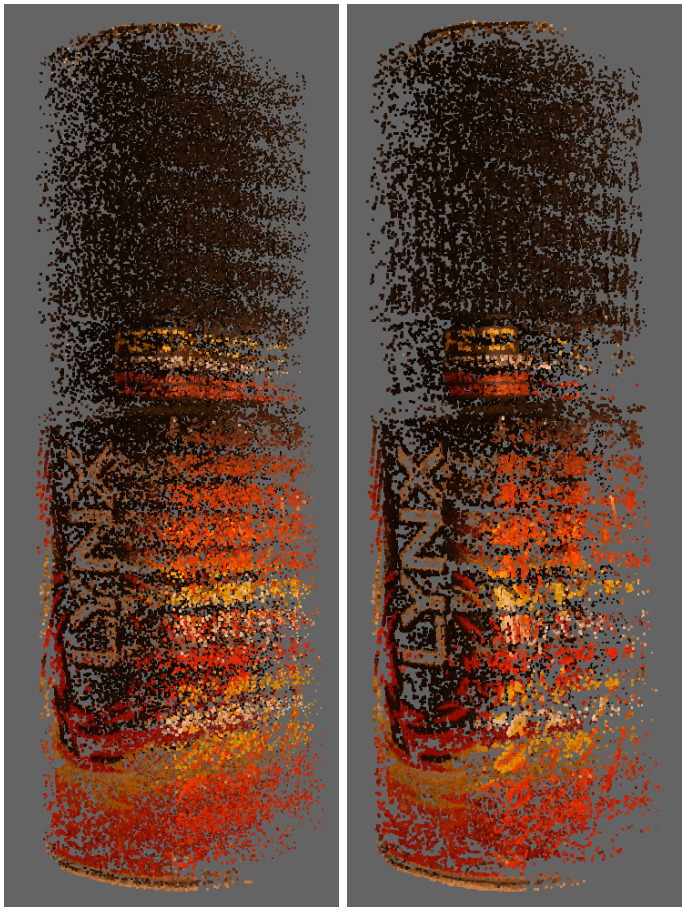


Figure 5.14: Patches created by PSO (left) and PSOPD (right) for a single view from the LynxReal dataset

5.2.2 Quantitative

From the qualitative results presented thus far, it is visibly obvious that the levels of noise have been significantly reduced and that patch refinement is potentially more accurate than that achieved with PSO. The purpose of these quantitative results is to provide numerical means to further analyse such improvements and the effectiveness of PSOPD. Analogously to the previous chapter, Figures 5.15 to 5.20 and Tables 5.1 to 5.3 display the probability distributions of the error on patch refinement, measured by the one-sided Hausdorff distance metric. First, the probability distributions for PSOPD are plotted against PSO alone, showing the improvements made by switching to the photo-detail-consistency metric. Secondly, the probability distributions for PSOPD are plotted against the equivalent errors in PMVS, showing the validity of our PSOPD approach in terms of PMVS. Finally, statistics such as the mean, median, mode and max are given, followed by the inverse cumulative frequencies, providing results in terms of numerical values. Akin to the previous chapter, PSOPD High, PSOPD Med and PSOPD Low, will refer to the error distributions created by the patch clouds for the high, medium or low textured datasets respectively. This section will discuss the improvements introduced in PSOPD via the following factors: The improvement made when using PSOPD as opposed to PSO, and PSOPD's performance in comparison to PMVS.

PSOPD Performance Comparison with PSO

Comparing the error distributions for PSOPD against PSO shows distinctly improved results. The tails of each distribution show that significantly fewer patches reside in positions with larger errors. The inverse cumulative frequency tables consistently show that the percentages of patches with an error greater than 0.005 is considerably lower when using PSOPD over PSO. Unfortunately, both measures can still suffer the same degree of error, induced in the extremely low textured areas. This accounts for the fact that the max values are approximately equal. Nevertheless, PSOPD has far fewer of these high error patches. The reduction in the mean, when comparing PSO to PSOPD, further justifies the improvements in the number of noisy patches, showing that there are indeed more low error patches. The heights of the corresponding peaks visually define the fact that there are more patches tightly packed under the PSOPD peaks than there are for PSO.

In terms of texture, the difficulties in refining textureless patches is reflected in the fact that when texture is reduced the volume of patches inside the tail of each distribution increase as well as the values of the mean. Both PSO and PSOPD have increasing difficulty in refining patches when the texture is low, however, PSOPD is notably more robust at dealing with these cases than PSO, with a far small percentage of patches lying at errors greater than 0.005 for all datasets.

Interestingly, the mode and alignment of peaks sit at constantly lower error values for PSOPD, proving not only that the number of erroneous patches has been reduced, but also that the accuracy to which the majority of patches can be refined has improved. In fact, across all datasets, the mode shows that the accuracy of refining low textured datasets, with PSOPD, is equivalent to the accuracy obtained refining highly textured datasets via PSO. This improvement in accuracy is also visible in the right skew of the distributions, where the PSOPD peaks are significantly higher and have sharper points towards lower error values than PSO.

PSOPD Performance Comparison with PMVS

Analysing the results of PSOPD, has clearly shown its improvements in relation to PSO, however, to form a comprehensive analysis of this technique, its validity in terms of the state of the art PMVS algorithm must also be assessed. Previously, PSO proved its effectiveness in terms of consistently producing highly accurate patch refinement, all be it suffering from high levels of noise in the output. We now re-assess its validity when using the improved photo-detail-consistency metric.

The overall effectiveness of our PSOPD method compared to PMVS, can be seen in the alignment of the mode and peaks of the distributions, these have persistently approximately equal errors with their PMVS equivalent. This shows that although PSOPD lacks the ability to step exactly onto the global minimiser, without a large number of iterations, by improving the photo-detail-consistency function these peaks are now easier to optimise for.

Although PSOPD is still affected by noise, its patch refinement shows consistency with PMVS. Both distributions have a narrow spread around their peaks, which is only slightly widened by noise in the case of PSOPD.

Given image noise and the fact bilinear interpolation is used to sample patches, it seems unlikely that any further significant improvement in the accuracy of refining well textured patches is possible, without increasing the number of iterations. Our global approach to patch refinement is theoretically capable of extracting patches to a high degree of accuracy, potentially surpassing PMVS, if the number of iterations is increased. However, the accuracy relating to the underlying surface position is still limited, most notably by the 5x5 patch size. With the exception of the LynxCAD dataset, which is highly affected by noise, the mean and median results for PSOPD are closer to that of PMVS than PSO. This shows that, as a whole, when patch optimisation succeeds it will be of consistently high quality, like that of PMVS.

In terms of accuracy relating to the level of texture, as the level of detail drops more poorly refined patches are created. PMVS filters and removes these patches, seen by the shorter tail of the distributions and smaller max values. PSOPD does not remove these erroneous results, seen by the increase in the volume of patches within the tail of the distributions, which also increases the mean and median values when compared to PMVS.

For the highly textured datasets, the inverse cumulative frequency shows that the percentage of patches with an error higher than 0.005 is now closer to the filtered patches of PMVS, showing that without filtration our dense patch based techniques is pretty close to that produced by PMVS, even though it contains noise. As texture is reduced, since there is no filtering, this then increases. Note that the performance on all of the Buddha datasets, Figure 5.18, is remarkably close to PMVS, however, as can be seen in Figure 5.4, we are consistently denser, regardless of the level of texture.

Finally, the LynxCAD dataset is a challenging example to reconstruct without filtering steps, but by assessing the peak performance of PSOPD, we again see improvements compared to PMVS, Figure 5.20. Albeit it that there is a large percentage of unfiltered noise clearly affecting the reconstruction quality, the peak accuracy has still improved.

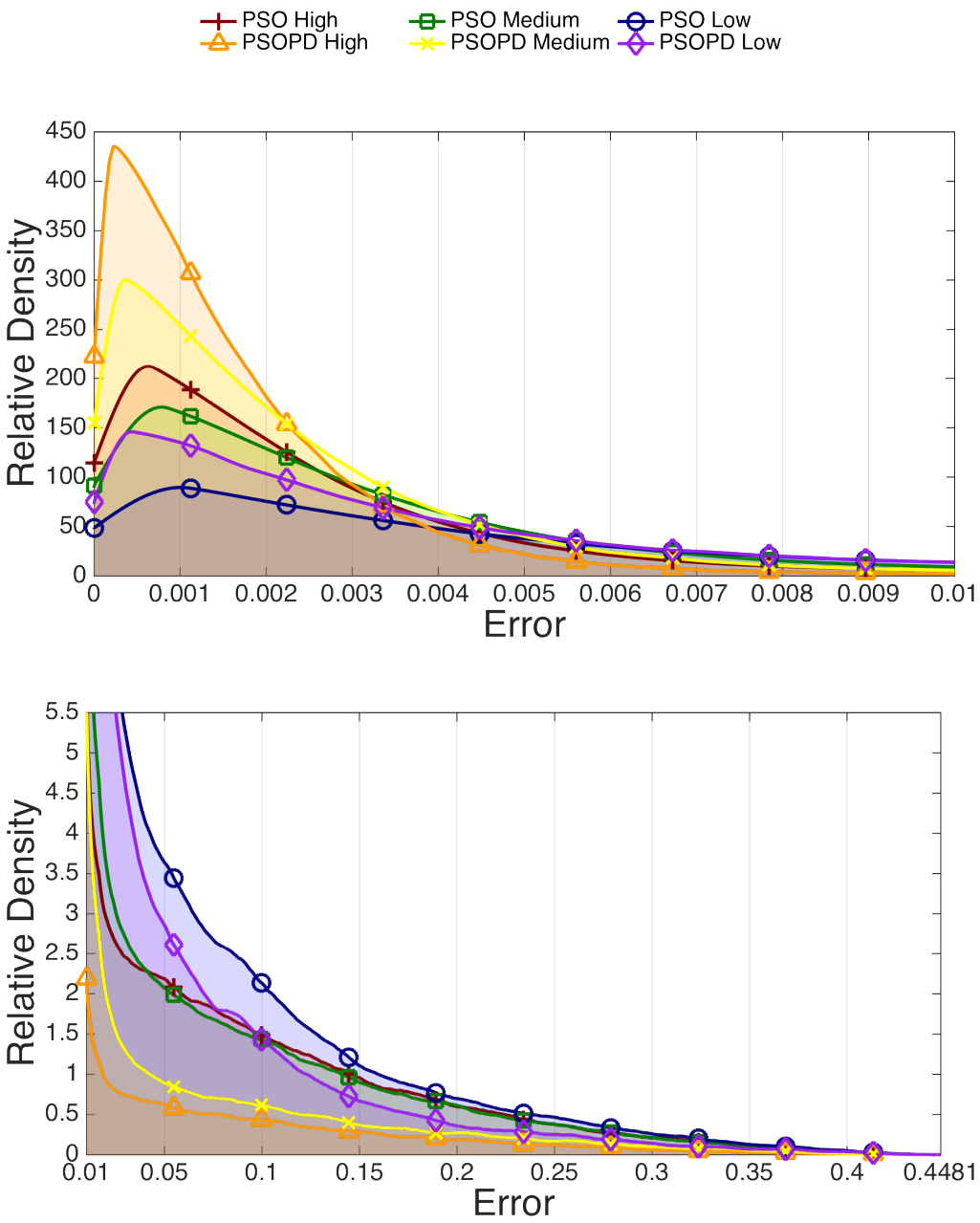


Figure 5.15: Probability distributions for the Monkey datasets
Top: Normalised probability distributions for the errors in patch refinement for running PSO and PSOPD on each of the three different textured datasets.
Bottom: Zoomed in tail end of the distribution up to the largest error

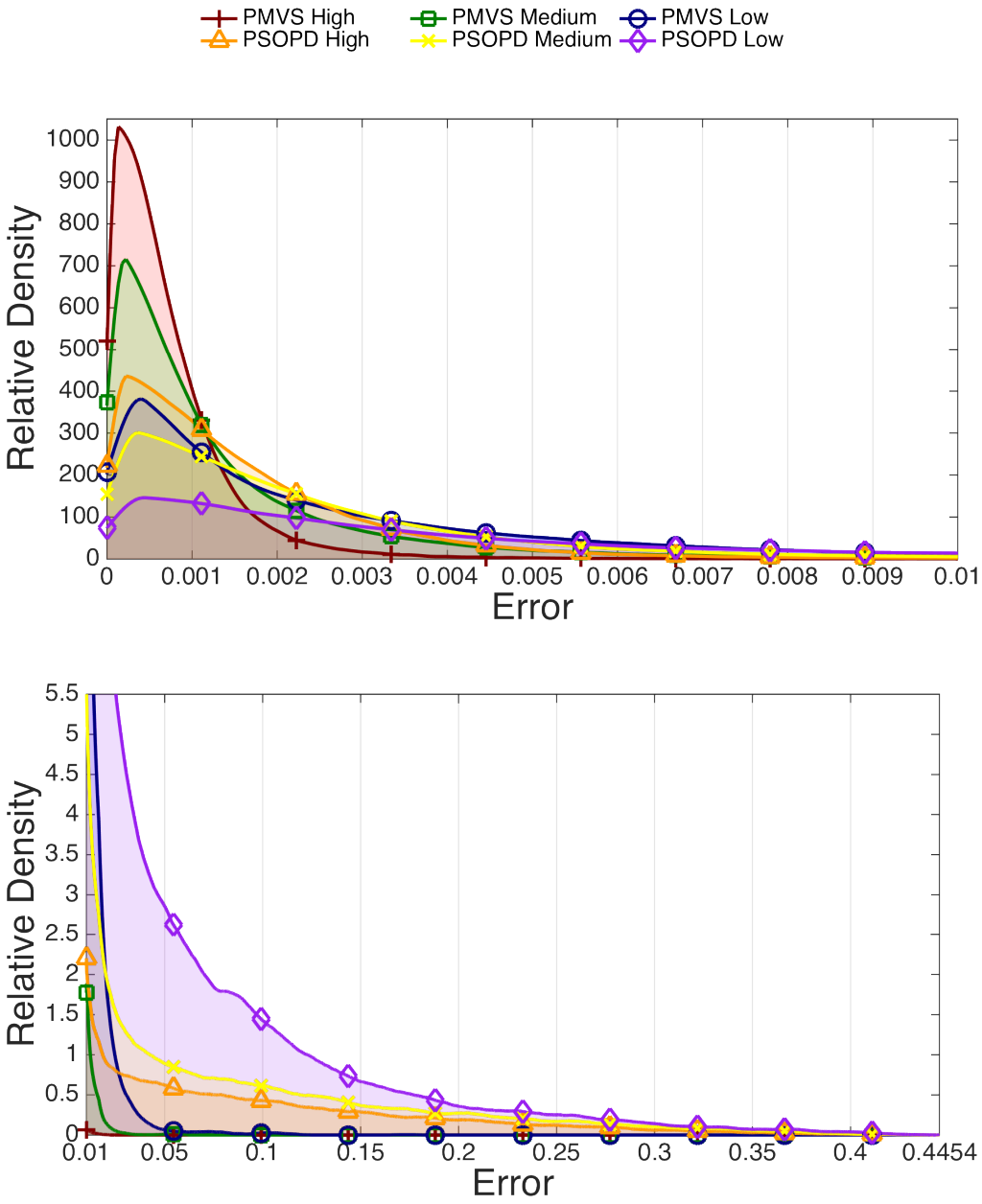


Figure 5.16: Probability distributions for the Monkey datasets
Top: Normalised probability distributions for the errors in patch refinement for running PMVS and PSOPD on each of the three different textured datasets.
Bottom: Zoomed in tail end of the distribution up to the largest error

		Mean	Median	Mode	Max
PMVS	High	0.0007	0.0005	0.0001	0.0180
	Med	0.0014	0.0008	0.0002	0.0580
	Low	0.0033	0.017	0.0004	0.1751
PSO	High	0.0377	0.0032	0.0006	0.4446
	Medium	0.0371	0.0039	0.0008	0.4439
	Low	0.0505	0.0148	0.0010	0.4481
PSOPD	High	0.0123	0.0014	0.0002	0.4437
	Medium	0.0179	0.0021	0.0004	0.4391
	Low	0.0335	0.0060	0.0005	0.4454

		Residual Error							
		0.0	0.005	0.01	0.02	0.03	0.1	0.2	0.3
PMVS	High	100.00	0.34	0.02	0.00	0.00	0.00	0.00	0.00
	Med	100.00	4.43	0.61	0.06	0.02	0.00	0.00	0.00
	Low	100.00	19.49	6.07	1.48	0.59	0.04	0.00	0.00
PSO	High	100.00	41.94	34.91	31.23	28.61	14.98	5.09	1.21
	Medium	100.00	46.07	35.65	30.63	27.62	14.48	5.02	1.21
	Low	100.00	67.66	56.20	46.84	40.80	18.42	6.14	1.52
PSOPD	High	100.00	13.85	10.38	9.14	8.34	4.48	1.59	0.37
	Medium	100.00	24.51	16.68	13.65	12.13	6.43	2.42	0.62
	Low	100.00	54.24	42.38	33.31	27.82	10.89	3.36	0.87

Table 5.1: Statistics for the Monkey datasets
Top: Different averages of the reconstruction error on each dataset.
Bottom: The inverse cumulative frequency of errors as a percentage of entire patch cloud

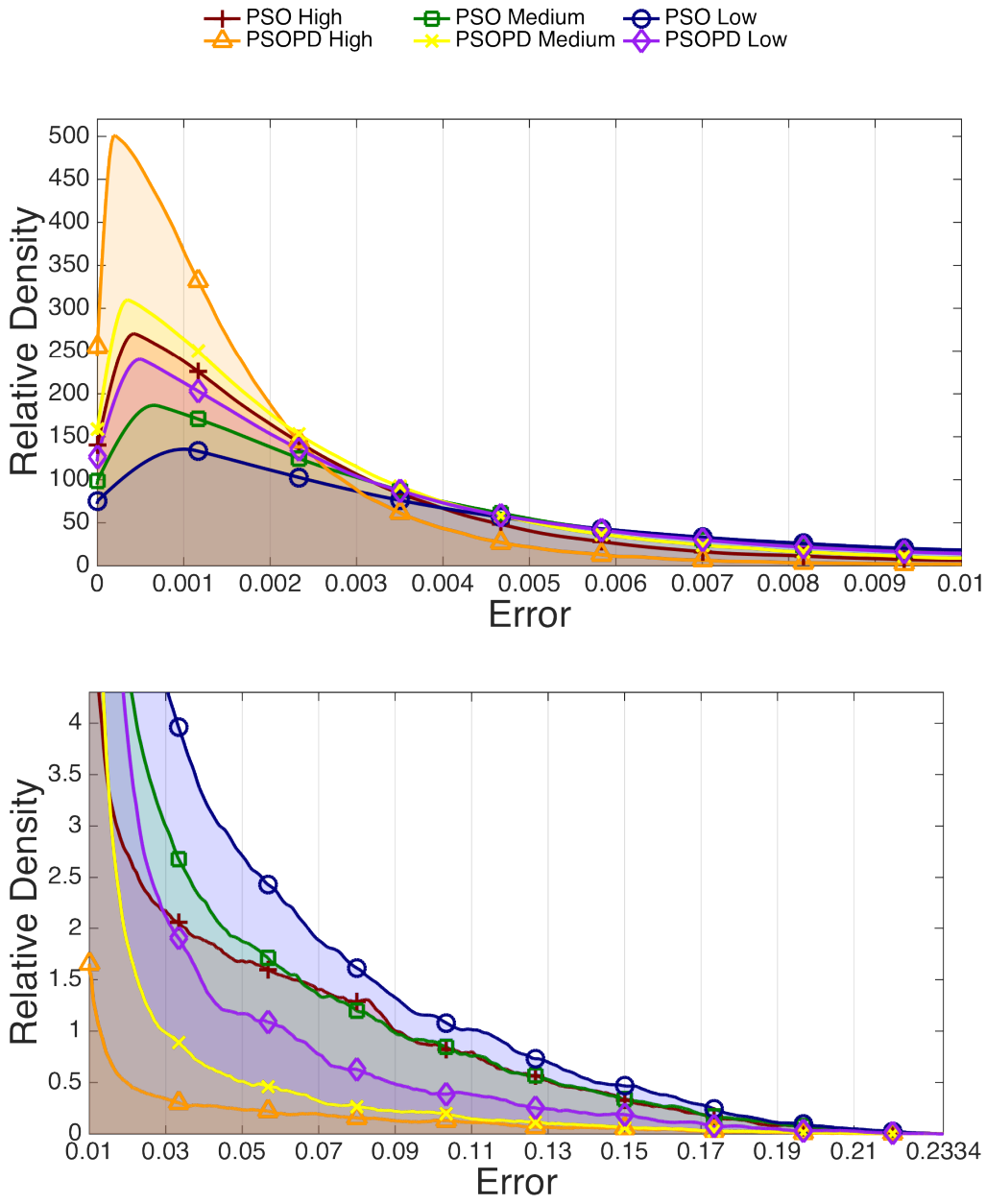


Figure 5.17: Probability distributions for the Buddha datasets
Top: Normalised probability distributions for the errors in patch refinement for running PSO and PSOPD on each of the three different textured datasets.
Bottom: Zoomed in tail end of the distribution up to the largest error

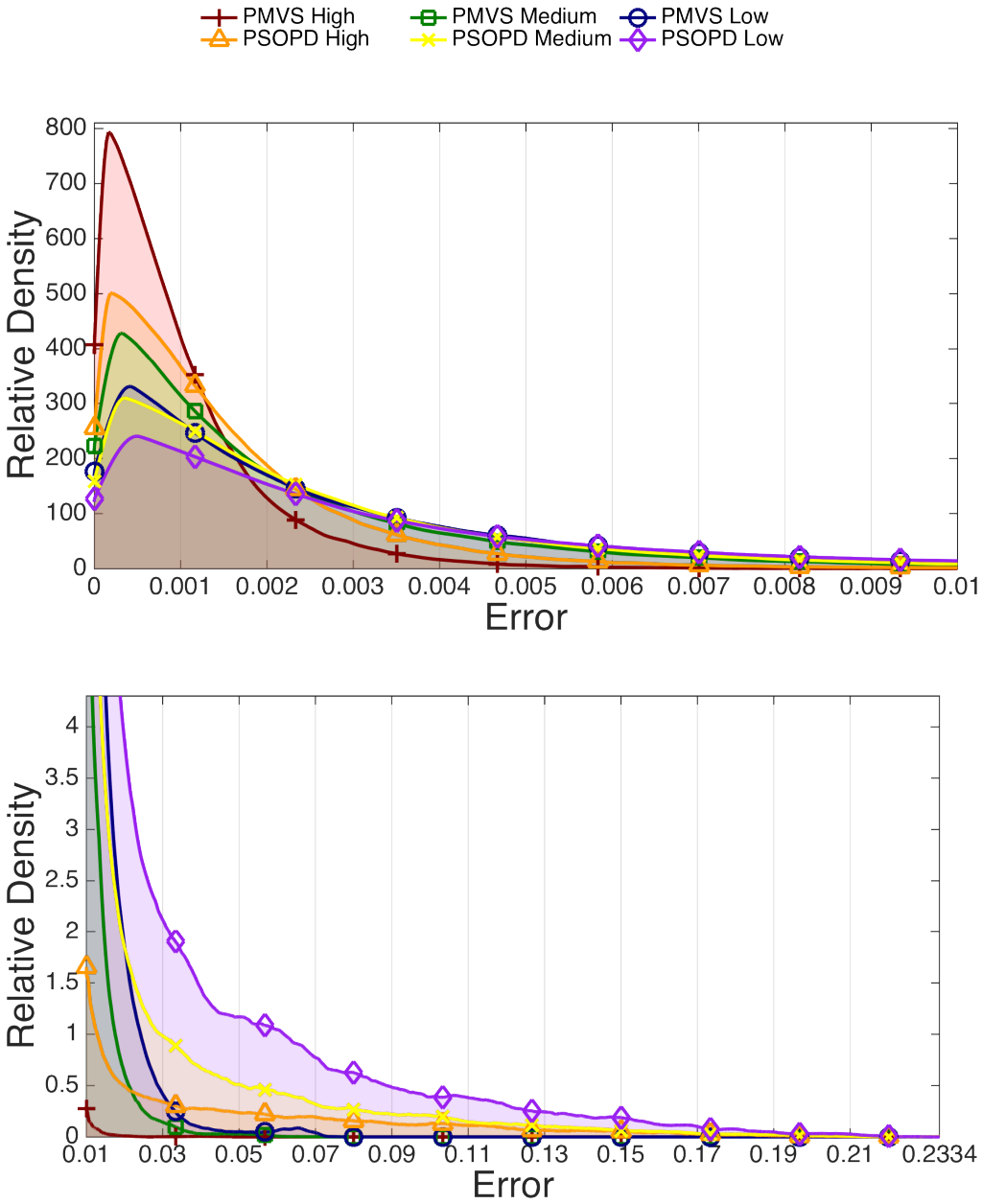


Figure 5.18: Probability distributions for the Buddha datasets
Top: Normalised probability distributions for the errors in patch refinement for running PMVS and PSOPD on each of the three different textured datasets.
Bottom: Zoomed in tail end of the distribution up to the largest error

		Mean	Median	Mode	Max
PMVS	High	0.0010	0.0007	0.0002	0.0449
	Med	0.0024	0.0014	0.0003	0.0678
	Low	0.0024	0.0019	0.0004	0.0737
PSO	High	0.0140	0.0023	0.0004	0.2327
	Medium	0.0156	0.0035	0.0007	0.2331
	Low	0.0204	0.0052	0.0010	0.2323
PSOPD	High	0.0033	0.0012	0.0002	0.2330
	Medium	0.0057	0.0020	0.0004	0.2334
	Low	0.0100	0.0027	0.0005	0.2300

		Residual Error							
		0.0	0.005	0.01	0.02	0.03	0.05	0.1	0.2
PMVS	High	100.00	0.82	0.06	0.01	0.00	0.00	0.00	0.00
	Med	100.00	12.08	2.61	0.41	0.13	0.02	0.00	0.00
	Low	100.00	20.01	6.27	1.31	0.39	0.13	0.00	0.00
PSO	High	100.00	28.54	20.18	16.57	14.19	10.42	3.86	0.08
	Medium	100.00	40.55	26.04	18.73	15.12	10.53	3.95	0.08
	Low	100.00	52.72	36.97	26.33	21.00	14.25	5.23	0.12
PSOPD	High	100.00	6.62	3.18	2.37	1.97	1.42	0.57	0.00
	Medium	100.00	20.39	8.85	5.08	3.80	2.38	0.79	0.02
	Low	100.00	32.51	18.35	11.32	8.61	5.59	1.91	0.05

Table 5.2: Statistics for the Buddha datasets
Top: Different averages of the reconstruction error on each dataset.
Bottom: The inverse cumulative frequency of errors as a percentage of entire patch cloud

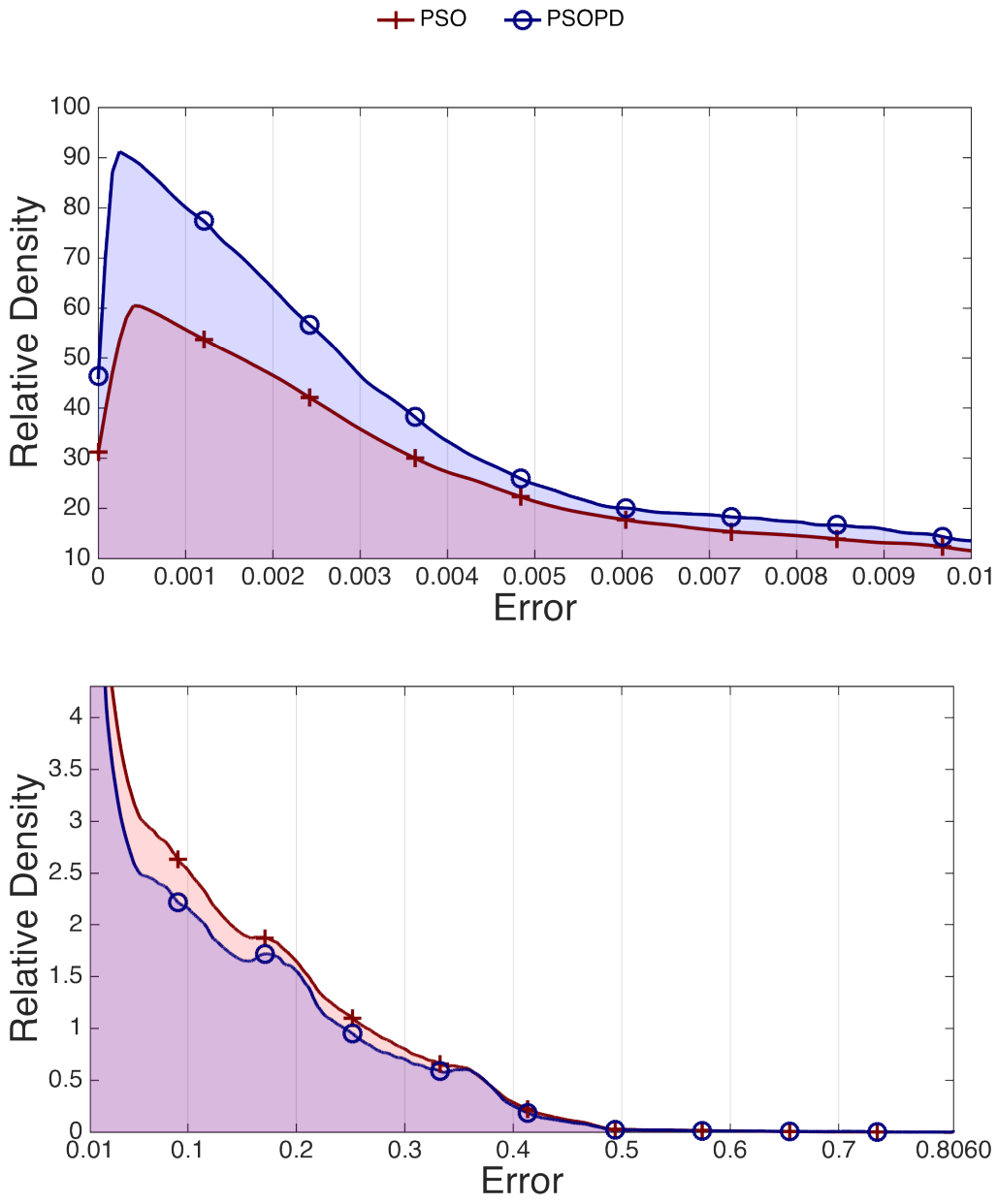


Figure 5.19: Probability distributions for the LynxCAD dataset
Top: Normalised probability distributions for the errors in patch refinement when running PSO and PSOPD on the dataset.
Bottom: Zoomed in tail end of the distribution up to the largest error

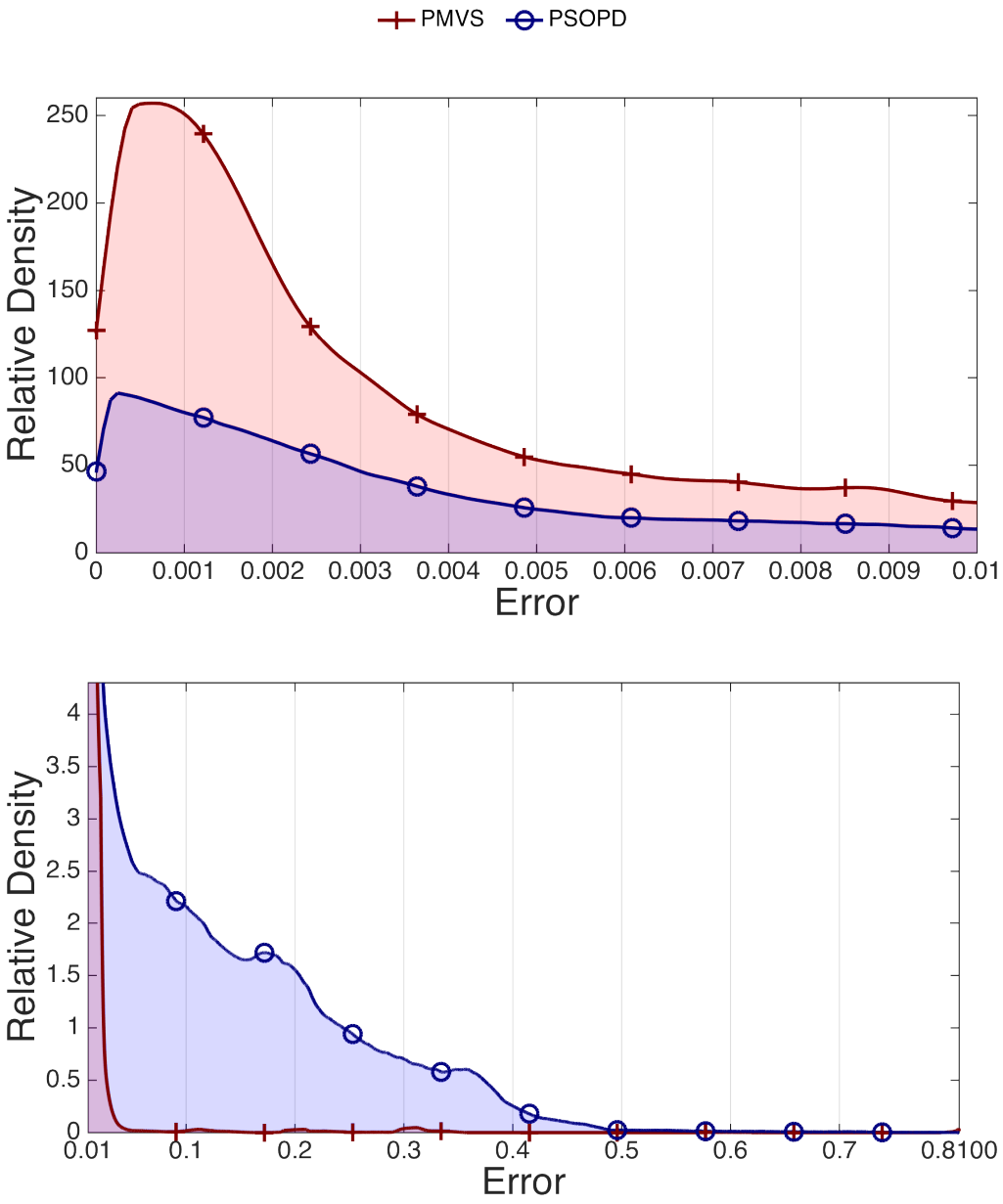


Figure 5.20: Probability distributions for the LynxCAD dataset
Top: Normalised probability distributions for the errors in patch refinement when running PMVS and PSOPD on the dataset.
Bottom: Zoomed in tail end of the distribution up to the largest error

	Mean	Median	Mode	Max
PMVS	0.0046	0.0023	0.0007	0.8100
PSO	0.1007	0.0592	0.0004	0.8060
PSOPD	0.0897	0.0365	0.0002	0.8060

	Residual Error							
	0.0	0.005	0.01	0.02	0.03	0.1	0.3	0.5
PMVS	100.00	27.72	7.80	0.96	0.53	0.38	0.16	0.01
PSO	100.00	79.58	71.90	64.90	60.24	38.95	7.42	0.30
PSOPD	100.00	71.82	62.84	56.21	52.31	34.60	6.81	0.30

Table 5.3: Statistics for the LynxCAD datasets
Top: Different averages of the reconstruction error on each dataset.
Bottom: The inverse cumulative frequency of errors as a percentage of entire patch cloud

5.2.3 Speed

		PMVS Run Time (sec)	Total Patches inside PMVS Cloud	PSO Run Time (sec)	Total Patches inside PSO Cloud	PSOPD Run Time (sec)	Total Patches inside PSOPD Cloud
Monkey	High	280	356120	883	461921	909	461921
	Med	298	328743	1008	461921	1107	461921
	Low	257	193085	1013	461921	1099	461921
Buddha	High	221	250264	732	399178	792	399178
	Medium	284	239817	879	399178	945	399178
	Low	356	217628	879	399178	941	399178
LynxCAD		1323	692776	4091	1805718	4508	1805718
Dino		731	338430	971	525354	1037	525354
Temple		390	239161	578	327818	615	327818
LynxReal		1670	494138	2550	1393262	2796	1393262

Table 5.4: Refinement times for PMVS, PSO and PSOPD

As the speed of reconstruction is a critical factor affecting the usefulness of the system, Table 5.4 presents the run times for computing the patch clouds with the PMVS, PSO and PSOPD frameworks. Our improved photo-detail-consistency metric requires more computation than the standard photo-consistency used by both PMVS and PSO. However, the table above shows that the speed of refinement is only slightly affected, and is negligible in terms of the overall run times.

As the photo-detail-consistency function approximates the level of texture over the patch, Equation 5.2, this requires the computation of many image derivatives. In order to achieve the run times documented in the table above, a simple optimisation was used. Before patch refinement begins, gradient responses are pre-computed by invoking a Sobel edge detection kernel over each of the input images, storing the results in a GPU buffer. Sobel edge detection is a well-studied area, with widely available documentation on its GPU extension, [122], therefore, we leave out the details of its implementation.

Once the gradient responses are computed, a second kernel is invoked which computes the texture responses, τ, in Equation 5.2. Constant memory is used to store the pre-computed weights, d, and local memory is used to read in the gradient responses to fast access memory, Section

3.3.3. The kernel follows a pixel per work-item mapping, where each work-item then sums over the gradient responses to compute each pixel's texture response, τ, and store it back to global memory. As in Equation 5.2, this is computed by summing the distance weighted gradient responses, returned by running the Sobel kernel, over the $\mu x \mu$ area centred over the pixel. Once complete the results are copied back to the CPU and stored in RAM, ready to be used in the following patch refinement.

In practice, this operation takes no longer than a second, of course depending on the number and resolution of the image set, so its run times were not seen worthwhile to document. Once the texture responses have been pre-computed the weight computation required when the optimisation is processing is greatly reduced, allowing for the fast patch refinement documented in the table above. The pseudo-code for this texture weight kernel is given in Figure 5.21 below, where μ is the patch size, src is the result of running the Sobel edge detection for each pixel in the image, and dst is the image buffer in which the texture weights will be written to. The local memory buffer, $locSobel$, is used to avoid multiple work-items reading the same Sobel gradient response from a pixel value in global memory multiple times.

Algorithm 12 Texture Weight Kernel

procedure TEXWEIGHT($image_2d\ src$, $image_2d\ dst$, $const\ float\ *d$, $local\ float\ *locSobel$)

$p = get_global_id(0)$ ▷ Get pixel's index for work-item

$CopyGlobalToLocalMem(p, src, locSobel)$
$Barrier()$ ▷ Wait for copy to finish

$texRes = 0$ ▷ Start with 0 texture response
for $i = -\mu/2\ to\ \mu/2$ **do** ▷ Sum distance weighted texture over patch
  for $j = -\mu/2\ to\ \mu/2$ **do**
    $texRes = texRes + d.at(i,j) * locSobel.at(p.x + i, p.y + j)$
  end for
end for

$CopyLocalToGlobal(p, texRes, dst)$ ▷ Write back texture response
end procedure

Figure 5.21: The flow of our GPU based PSO texture weight computation kernel

5.3 Discussion

5.3.1 Technique Synopsis

This chapter has described, in detail, our novel approach to adaptively weighting a newly defined photo-detail-consistency function. The purpose of which, is to improve the clarity of the function and the probability that the globally optimum parameters, along a patch's entire viewing ray, remain correctly relating to the underlying surface characteristics. By improving the similarity metric, the intentions were to reduce the number of incorrectly refined patches, in turn removing some of the noise from the final reconstructions.

To achieve this goal the standard photo-consistency metric was adapted to take into account two cases where the NCC can fail, these being when the texture over the surface is low, and when the patch is viewed and sampled from shallow angles. In order to measure the texture over the patch, the gradient of each pixel can be taken and summed over the extent of the patch window. Colour channels with low texture response are then weighted lower than channels with high texture response. In order to account for the fact that texture further from the centre of the patch is likely to be less reliable than that closer to the centre, when summing the response over the patch window, image gradient is weighted based on its radial distance. Using bilinear interpolation to sample a patch, when its angle to image plane is shallow, results in a textureless effect. As such a second weighting was defined per view, taking into account the angle between the patch normal and image plane, measuring the suitability of each view for sampling in.

5.3.2 Results and Limitations

Results have shown that our photo-detail-consistency function succeeds in significantly reducing the number of badly refined patches. A side effect to improving the similarity metric also allows patches to better adapt to the underlying surface properties, meaning that patch refinement based on our photo-detail-consistency function is in fact of notably highly accuracy than using the standard similarity metric alone. The level of detail in mod-

elling high frequency surface variation, obtained by our PSOPD approach, was far superior to reconstructions acquired by PSO and achieved results on par with PMVS whilst achieving a denser reconstruction.

It has been shown that as texture is reduced the number of incorrectly refined patches still increases; the reason for this is that the photo-detail-consistency metric only accounts for textureless colour channels. When the texture over all the image channels is low, the performance of our metric degenerates to the standard photo-consistency model. Patch refinement under these circumstances still succeeds in many cases, whilst the global optimum still relates to the correct surface position and orientation. However, occasionally the 'texturelessness' of the patch distorts the global optimum to no longer align with the surface, predicting when this will occur is unfortunately not possible.

Unfortunately, it is not yet feasible to design a similarity metric that guarantees that the global optimum always relates correctly to the underlying surface properties. As such the current framework still produces some badly refined patches in textureless regions, all be it far fewer than our preliminary PSO framework from the previous chapter. Therefore, to remove this noise, either the patch size must be increased to include some texture, or a certain level of filtering is required.

5.3.3 Improvements

Adding texture to these textureless patches is potentially the only way to allow patch refinement to succeed in such cases. Therefore, to further improve this method patches should adapt to the level of texture by increasing their extent, hopefully meaning that some texture will now be included inside the patch. However, this severely impacts the run time, with more patch samples increasing the level of computation as well as greatly smoothing the patch cloud, Section 4.1.1.

To avoid increasing the patch size, reducing the computational overhead, whilst further accounting for textureless regions and the addition of filtering steps, a multi-resolution technique could be utilised. Switching to a multi-resolution technique, is in some ways equivalent to increasing the patch size, without increasing the patch samples. Using a 5x5 patch, as the resolution is lowered the extent of the patch increases, incorporating more surface area and texture.

Whilst this technique would offer additional improvements in terms of patch refinement, it also allows for the inclusion of filtering steps, removing badly refined patches. Noisy patch refinement should not be consistent through the multi-resolution scales, whereas correct patch refinement should be. This offers a way to detect when noisy patch refinement is occurring and remove these erroneous patches from the final reconstruction.

5.4 Summary

This chapter has defined our enhanced photo-detail-consistency metric, allowing for the reduction in the amount of noise produced when refining patch clouds. It has outlined the following key approach and subsequent benefits:

- Improving the similarity metric can be achieved by accounting for two factors leading to the failure of the NCC. These being the texture over the patch, and the angle at which the patch is viewed and sampled over.
- Using our enhanced photo-detail-consistency metric greatly improves the number of correctly refined patches, but also more significantly, improves the fidelity of refinement.
- Our PSOPD approach outperforms PSO especially in well-textured regions and the acquisition of high frequency surface variations, which PMVS fails to reconstruct densely. However, incorrect patch refinement can still occur inside textureless regions, all be it to a lesser extent than PSO.

6

A Multi-Resolution Approach to 3D Reconstruction

Throughout this thesis, the intentions of this work have been to improve upon the density and accuracy at which 3D patch clouds can be extracted under textureless circumstances. Striving for a technique that increases the number of patches in low textured circumstances or across high frequency surface details, whilst also defining a method that can operate effectively when the number of input images is low. Particular attention has been paid to the top performing patch-based algorithm, PMVS.

In the previous chapters, we have described a new patch optimisation technique and improvements to its patch refinement stages that have substantially increased the number of patches produced with equivalent accuracy to PMVS. Whilst being able to densely refine patches over

textured surfaces, unpredictable and incorrect patch refinement can occur in textureless regions using our approach, this results in high levels of noise in the final patch cloud. The intentions of this final chapter are to further reduce and filter noise, whilst retaining the features producing a dense reconstruction. This is achieved through the incorporation of a multi-resolution scheme into our reconstruction framework, as will be described in this chapter, such an approach both verifies correctly refined patches and further enhances noisy patch refinement.

Analogously to increasing the patch size, we found that keeping the patch size constant whilst decreasing the resolution of input images, refining patches at this reduced resolution, has a similar effect but reduced computational cost. While improving the robustness to textureless surfaces, lower resolution images also introduced smoothing, therefore only creating patch clouds at lower resolutions reduces the ability to fully define high frequency surface details. The robustness of patch clouds produced at lower resolutions is therefore made use of to more tightly restrict the optimisation space of each patch, propagating these results up through the remaining resolutions. Once the highest resolution is reached a filtering of each patch cloud occurs. To avoid filtering by redundancy, used in standard depth map techniques, which is susceptible to removing correctly refined patches when the input image set is sparse, we instead propose a per-view filtering. For a patch whose refined location relative to its neighbours, in only its reference view, is used to define whether it should be removed.

The results have shown a surprising increase in the accuracy and density of refining patches within textureless regions, patch clouds are visibly more dense and of higher accuracy. Due to difficulties in defining noise to slightly inaccurate patch refinement, some error still persist inside textureless regions. However, comparing our results to PSOPD, defined in the previous chapter, shows a significant improvement in all aspects of the patch cloud. Comparing our method to PMVS has shown that we can be equally accurate in high textured regions, whilst producing much denser reconstructions inside textureless regions. While the accuracy of patch refinement inside these textureless regions, by their very nature, are not as high, the improvements in density far outweigh the drop in accuracy and run time. Small levels of noise or inaccuracy in the output can be removed in a post process, whereas gaps have to be filled via an inaccurate surface interpolation scheme, discussed in Section 6.1.

Beginning the chapter, Section 6.1 describes multi-resolution methods. Section 6.1.1 outlines the limitations in the expansion and filtering stages of PMVS, explaining why we have chosen not to use a similar approach to filter noise in our patch clouds. After this, Section 6.1.2 explores the feasibility of using a multi-resolution approach, providing some preliminary experimental results into the advantages of such a technique. Following this, Section 6.1.3 details our novel multi-resolution framework, including information on how it is implemented. Section 6.2 gives the results from our multi-resolution framework, analysing the noise reduction and accuracy improvements when compared to PSOPD and its performance against PMVS. A final discussion of these results, presenting a review on the achievements of this algorithm can be found in Section 6.3. This chapter is concluded with a short summary in Section 6.4.

6.1 Multi-Resolution Patch Refinement

In Chapter 4, our preliminary patch based reconstruction framework using PSO to optimise a patch's position and orientation was presented. Showing promising results for increasing the density of patch clouds, however, in situations where the global optimum no longer aligned correctly with the underlying surface characteristics, the position and normal, noisy patches occurred. To reduce the number of noisy patches, Chapter 5 proposed the use of a novel photo-detail-consistency metric, improving the likelihood that the globally optimal solution aligns correctly with the surface to be reconstructed. This both significantly reduced the amount of noise, whilst also improving the accuracy of patch refinement. However, due to the difficulties in predicting exactly when patch refinement will succeed or fail, noise still persists in the final reconstruction. The intentions of this final chapter are to further reduce and improve upon the effects of this noise.

Defining exactly what is meant by 'noise' is often application dependant. Obviously, patches with a large distance to the underlying surface could clearly be classified as erroneous and thus noisy. However, the classification between noisy patches and minor inaccuracies is unclear when the error on patch refinement is much smaller. Removing all but the highest accuracy patches will in most cases degenerate the density and overall quality of the entire patch cloud, whereas tolerating a certain level of error can allow for a more complete and dense coverage.

In cases where a surface mesh is required, noise in the patch cloud is removed as a post-process, involved in the generation of the best fitting triangulated surface. In situations where only the highest accuracy patches are kept, surface meshing routines tend to fail at accurately generating a mesh inside empty regions, relying on interpolation over large regions with no information. Noise in a patch cloud can be recovered from, whereas, large empty regions are not so easy to deal with as will be discussed in the following paragraphs.

Mesh based methods, such as [32, 81, 131], actually perform better in the presence of a small amount of noise, as the meshing operation does not have to rely solely on interpolation. To be more specific, Pound et al. [131] use PMVS to generate an initial patch cloud, whilst modifying its parameters to produce more noisy patches, better filling textureless regions. Consequently, these noisy patches provide better guidance for their meshing procedure, where patches are grouped and clustered onto best fitting planes, removing noise and polygonising the surface. However, it is a tricky task to get PMVS to generate any results in textureless regions, regardless of how its parameters are defined, thus large amounts of over smoothing and interpolation must occur. Poisson surface reconstruction, [81], is a highly efficient tool for meshing patch clouds, however, in the presence of no data, surfaces produced by this technique tend to bulge and distort from the desired model. The authors of PMVS, Furukawa and Ponce [53], suggest using a mesh based deformation algorithm to fill in gaps and further improve the accuracy of the patch-based reconstruction. However, their results show that this requires significant run time, which does not scale well to larger image sets or for industrial purpose, where run times can run into days in some cases.

Our aim is therefore to remove obvious noise from our patch clouds, whilst still producing patches in these textureless regions, even if their accuracy is not highly reliable. In most situations, it is desired that we accurately refine as many patches as possible, forming a dense coverage, then compromise on patch noise, as it is better to have some inaccurate points than none whatsoever to guide the meshing procedure.

Typically, patch based methods such as PMVS [53], Habbecke and Kobbelt [63] or Lhuillier and Quan [98], have achieved noise removal through the use of expansion and depth map filtering techniques. However, as was described in Chapter 2 and was seen in the results of PMVS

in Chapter 4, this can lead to degeneration in the final patch cloud, where expansion normally fails to move into and capture larger textureless regions. To obtain both high frequency details and fill in low textured data, expensive energy minimisation must be performed, rather than Poisson surface reconstruction, [53].

Such expansion and filtering steps have been shown to become more effective when the number of input images is large. PMVS, for example, achieves significantly better results for datasets with smaller baseline and a larger number of input views, [53]. In these cases there are more pixel expressing surface details, allowing expansion to more densely cover these regions. This fact makes PMVS ideal for use in internet scale MVS problems, such as [4, 55], where input image sets are large and detailed. However, when dealing with smaller input image sets with wider baselines, such as the ones we focus on in this thesis, the expansion and filtering steps are not effective at producing dense high accuracy patch clouds as was shown in Chapter 4.

Ideally we would like to adapt our current framework to allow for the removal of obvious erroneous patches, which do not sit close to the correctly defined surface; whilst at the same time retaining our dense coverage and further improving the performance in textureless regions. Patches in these textureless regions may not be the most highly accurate, due to the limitations in comparing textureless patches, however these are important to keep for dense reconstruction. To achieve this goal, a multi-resolution framework will be studied, allowing for the adaptation of patch sizes, enhancing the ability to refine textureless surfaces, as well as to filter out obviously incorrect patch refinement.

Results obtained in the previous two chapters have shown that patch refinement via PSOPD, over well-textured regions, is incredibly accurate and unlikely to be affected by noise. Whereas refinement in textureless regions, can in many cases succeed as well as fail, however, the success of refinement is hard to predict pre-optimisation. Therefore it seems reasonable, rather than simply not attempting reconstruction in these areas, to first try to refine a patch, then remove it if it's inconsistent. To avoid the expansion and filtering steps, a multi-resolution approach is taken, however, this should be carefully assessed as to its feasibility and limitations. Therefore, we will start this section by analysing the limitations of expansion and filtering used by PMVS, then look at the viability of a

multi-resolution scheme to solve these problems. Once we have verified the multi-resolution scheme, we will give the details of our complete multi-resolution PSO based framework, optimising the photo-detail-consistency metric defined in the previous chapter. To distinguish between PSOPD from the previous chapter and the multi-resolution version presented here, we will term this new framework MRPSO.

6.1.1 PMVS Expansion and Filtering Stages

As previously discussed, in Section 2.4.4, the expansion and filtering steps in PMVS are effective at producing accurate and dense patch clouds when the number of input images are high, or when there is only a small baseline in the input views. However, when these factors do not apply, removing all but the accurate patches, via expansion and filtering, causes holes and subtle inaccuracies to occur. Here, we discuss these issues, further analysing reasons why they are not applicable but could be helpful.

Furukawa and Ponce [53] describe how PMVS begins creating a patch cloud through a robust feature matching and refinement scheme, Figure 2.3, this initial patch cloud is sparse but of high accuracy. The sparsity of the initial patch cloud is controlled by the texture of the underlying surface, creating more or fewer feature points. PMVS then relies on gradient descent to refine a patch's parameters, as such requiring precise initialisation. In order to add more patches to the cloud, an expansion procedure is used to initialise patches close to those already defined, such that refinement can take place, Figure 2.4. The assumption being that this initialisation is close enough to the optimal solution, such that gradient descent will not become trapped in the incorrect local minima. When refinement is complete, if the NCC score for that patch is above a certain threshold or the patch has moved more than a given angle from the neighbour it expanded from, the patch is removed as it's likely to have been incorrectly refined. Once a patch has passed this test, it blocks patches being created for the pixels that it projects into throughout the other views.

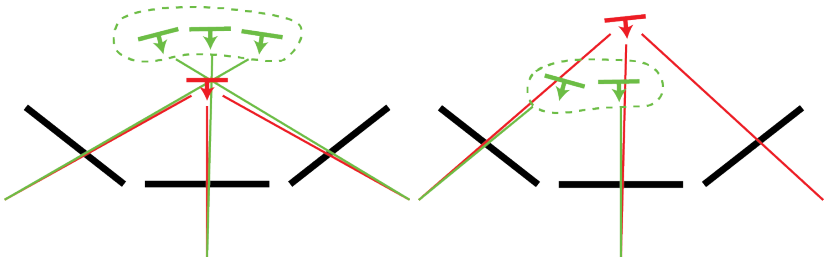
This expansion approach is fairly successful, but allows small errors to creep in where gradient descent did not precisely define the correct patch. When expansion is iterated these small errors are propagated through to surrounding patches, generally lowering the overall accuracy of the

patch cloud. This also forces patch expansion to stop inside textureless areas or across depth discontinuities, where the NCC is naturally higher. The larger the textureless regions, the smaller the amount of feature matches produced and the less likely patch expansion is to move into these regions. As well as this, the blocking nature of patch expansion stops fully dense reconstruction of high frequency surface details. Blocked pixels in corresponding views can no longer create patches close the current patch, thus cannot fully express high frequency surface details. This can be alleviated using a higher number of input views, such that there are more available 'non-blocked' pixels to express the surface detail. The issues with expansion are discussed and can clearly be seen in the results of Chapter 4.

As our patch-based approach does not rely on precise initialisation, this expansion procedure is largely unnecessary, hence we were able to create the feature free, expansion free, dense PSO algorithm, capable of expressing high frequency surface details. Chen et al. [26] replaced the gradient descent optimisation scheme in PMVS with GLN-PSO, using expansion to restrict the search space to nearby positions. This showed some provisional improvement in the overall quality of reconstruction. The use of expansion steps, allowing for the restriction of a patch's search space, improves the probability of the global optimum inside the restricted search space aligning correctly with the surface, so patch refinement should succeed more often. However, our PSO algorithm in Chapter 4, showed that relying on expansion and search space restriction is unnecessary in most well textured cases. In particular restricting the search space across depth discontinuities can mean that the space no longer contains the correct patch parameters, hence the need for strict filtering stopping expansion into textureless regions. In theory, restricting the search space in textureless regions based on the unrestricted results of textured patches, could provide improvements in noise. However, relying on expansion to inform this restriction gives no guarantee that the global optimum inside the restricted search space will correspond to the correct parameterisation. Therefore, we intend to use this theory to more correctly inform a search space restriction.

Through the expansion scheme above, PMVS can produce incorrect patches that need to be removed. In particular, gradient descent returns incorrect patch refinement when expansion moves over a depth

discontinuity, or alternatively when patches move into a textureless region. Therefore, Furukawa and Ponce [53] describe three filtering rules to remove erroneous patches. The first two are based on a patch's visibility. Firstly a patch must be visible, via a depth map test, in γ images. This is provisionally set to three, so that every patch in the patch cloud must be seen by at least three views, ensuring an accurate photo-consistency measure can be taken over the patch. In wide baseline cases, this must be reduced to two, accounting for the fact that it is unlikely for certain surfaces to be visible in more than two views, essentially making this filter redundant. Secondly, neighbourhood visibility is accounted for by collecting the set of patches that project into the same pixel neighbourhood as the current patch. Occlusion of these patches with the current patch is observed, Figure 6.1. If the current patch is in front or behind all of its neighbours, then it is removed as noise. Finally, the strictest filtering is the enforcement of regularisation. For each patch, a set of patches in all the pixels it projects into and adjacent pixels are found, if the proportion of patches that can validly be classified as neighbours to the current patch is lower than 0.25, the current patch is removed.



(a) Occluding (b) Occluded

Figure 6.1: Erroneous patches (Red) are removed when their occlusion is inconsistent with their neighbours (Green)

Although these filtering steps are valid for the expansion stage above, testing visibility via neighbouring patches is only well adapted to the errors induced in the expansion stage of PMVS. These do not account well for multiple patches expressing the higher frequency surface information which may be contained within a single pixel in certain arrangements of views, using these filters would result in the removal of valid patches. Secondly, these filters rely on a certain level of redundancy in the depth maps to allow for their effectiveness, however, this is clearly not possible when the input image set is sparse. When patch refinement fails in our PSOPD framework it is normally of significant random deviation from the correct position, thus should be simpler to detect without risking the removal of correct yet high frequency surface patches.

6.1.2 Multi-Resolution Study

The previous section has outlined the factors in PMVS that cause problems or that are not applicable to our patch based reconstruction technique. In this section, we look towards a multi-resolution approach to help overcome and outperform the expansion and filtering stages of PMVS, allowing the incorporation of better patch refinement in textureless areas and a filtering methodology. We address the question as to whether there is another way, instead of PMVS's expansion and filtering stages, to reduce noise. The intention being to obtain a patch cloud that is as dense and accurate as possible, albeit with some compromise in terms of noise.

6.1.2.1 Benefits of Multi-Resolution

Currently our PSOPD approach produces noise in textureless regions, however detecting whether severe erroneous patch refinement will occur is a difficult problem pre-optimisation. Results from Chapters 4 and 5 consistently show that patch refinement is significantly more likely to succeed and be of higher accuracy in the presence of texture. Habbecke and Kobbelt [63] also note the robustness of patch refinement in textured situations, therefore, they proposed adaptively changing the patch size based on the level of texture over its extent. To do this the size of the patch is increased such that the intensity variance, over the extent of the patch in its reference view, is above some predefined threshold, σ^2. Accordingly, pixels with a lower intensity variance refine larger patches than those with high variance. By ensuring that each patch contains some texture, increases the probability that patch refinement will succeed in the majority of cases.

The most obvious issue with increasing the size of the patch is the computational burden this inherits. To refine a patch via the NCC, a larger number of samples over its surface are required. As well as the cost of sampling, there is a rise in the number of calculations needed to compare two patches, further increasing the refinement time. Where scalability in terms of the run time is concerned, allowing more patch samples and deterioration in refinement speed is often not a feasible solution.

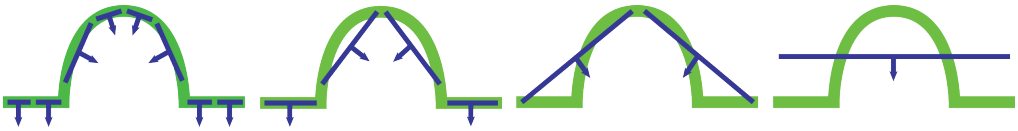


Figure 6.2: The effects of varying the patch size on the surface fidelity, assuming that patches are correctly refined

A less obvious disadvantage of increasing the patch size is the level of accuracy that can be obtained when refining its parameters, this was briefly described in Section 4.1.1. Figure 6.2, demonstrates the effect increasing the patch size has on its ability to accurately reflect the underlying surface. When patches (blue) are small, they represent a smaller area on the surface (green) of the model, hence, assuming patch refinement is successful, they can fully define the highest frequency surface variations. As the patch size is increased, the linear least squares solution corresponding to the NCC, results in the smoothing and a drop in accuracy of the patch cloud. Klowsky et al. [86] studied this effect rigorously, showing the importance of selecting the correct patch size on the ability to correctly reconstruct surface details. Therefore, smaller patch sizes are capable of reconstructing surfaces to a higher accuracy, however, as we have seen they are susceptible to incorrect refinement in textureless regions. Larger patch sizes are theoretically more robust in textureless regions, however result in a smoothing of patch refinement. Unfortunately, where patch refinement fails, there is no choice left but to increase the patch size, in order to include some texture over its extent. Therefore, increasing the patch size in textureless areas is likely to mean that the highest frequency surface details cannot be reconstructed, thus reducing the accuracy in these cases.

Alternatively, in order to retain computation efficiency whilst allowing the inclusion of texture over textureless regions, the resolution of the input imagery can be reduced whilst keeping the patch size constant (5x5). This is equivalent to increasing the patch size, in the fact that a single pixel at a lower resolution expresses a larger surface area. However, lower resolution input imagery can smooth out higher frequency surface variations, so we first analyse patch clouds created over multiple resolutions to validate this theory in terms of PSO. Our assumption being that reconstructed patches for a lower resolution image set are likely to be less noisy, as the reduction in resolution induces texture over textureless patches. However, patch refinement may also be less accurate, caused by the smoothing introduced by larger patch sizes in world space.

6.1.2.2 Creating Multi-Resolution Image Sets

In order to reduce an input image's resolution, careful thought should be placed on the technique used. Traditional image pyramid techniques blur the input image with a Gaussian kernel, then downsample it by a factor of two, reducing the resolution and smoothing out high frequency information. Although this is a valid approach, nowadays more beneficial and complex techniques are available. This work will take advantage of the second-generation wavelet transform (SGWT), [148], allowing for the creation of a full multi-resolution signal analysis of the input image. One of the many advantages of using the wavelet transform is the fact that it returns both a lower resolution image, in a set of approximation coefficients, as well as the corresponding texture frequencies, in a set of detail coefficients.

Though the mathematics of wavelets has been well studied along with the development of a multitude of different applications, this section aims to highlight crucial features in the decomposition pipeline that are utilised in this MVS technique. The interested reader should refer to [3, 12, 29, 33, 35, 78, 107, 108] for a complete mathematical breakdown of the wavelet transform.

The wavelet transform operates by convolving an input image with two functions, firstly a *wavelet* is applied which acts as a high pass filter, stripping away low frequency details and returning only high frequency components of the image. These values returned are known as the detail coefficients. Then a second function, the *scaling function*, is applied which acts as a low pass filter, removing the high frequency components and leaving only the low frequency information. The values returned are then sampled and stored in a lower resolution image, known as the approximation coefficients. Applying the wavelet transform to an image produces a lower resolution approximation image along side a set of detail coefficients describing which frequencies were removed and where they were removed. The same transform can then be repeatedly applied to the lower resolution approximation image to create a full multi-resolution decomposition.

One of the most significant advantages of using the SGWT is that it can be implemented as a lifted transform, rather than a traditional filter bank implementation, [148]. The lifted transform allows for much

faster and inplace (requires no extra memory) computation of the transform. Sweldens and Schröder [149] provide a full description of the lifted wavelet transform and [34, 147] describe the mathematical details of the approach. The main idea in a second-generation transform is to move away from designing Fourier based filters, removing high and low frequencies, instead to design wavelets spatially with certain polynomial degree and number of vanishing moments. This allows control over which parts of the signal are suppressed into to the approximation coefficients, and which parts fall into the detail coefficients, [147].

To perform the lifted SGWT, first the signal is split into even, λ_i, and odd parts, δ_i, in a process commonly refereed to as the *lazy wavelet transform*, [77]. This splitting is arbitrary, allowing the formation of any shaped sampling lattices. Then a prediction model, ρ, is formed for the signal, creating a *dual lifting* step, defined by Sweldens and Schröder [149]. Here the even parts of the signal are taken, and are used to predict to odd parts in Equation 6.1 below:

$$\delta_{i+1} = \delta_i - \rho(\lambda_i) \tag{6.1}$$

Just as the splitting was arbitrary, so is the prediction model, ρ. Here it can be understood spatially as an interpolation filter. If a linear model is chosen, having 1 vanishing moment, the prediction operator will take the two even samples either side of an odd sample and use this to linearly interpolate the value at the odd part, $\rho(\lambda_i)$. This prediction operation creates the set of detail coefficients for the odd samples, by replacing the odd sample with the difference between the interpolation and the actual value of the sample, via Equation 6.1. Therefore, the set of detail coefficients returned by such a transform can be understood as the *failure of the current coefficient to match its given prediction model*!

Once the odd coefficients are replaced by their failure values, the set of even coefficients, the approximations, need to be transformed before they are cascaded to the next level. Sweldens and Schröder [149] defined the method to compute the approximation coefficients in Equation 6.2 below:

$$\lambda_{i+1} = \lambda_i + \mu(\delta_{i+1}) \tag{6.2}$$

Normally for a multi-resolution representation, one would like the lower resolution version of the signal to retain certain properties of the input signal, such as its mean. Therefore, a *primal lifting* step is formed, as in Equation 6.2. This primal or update step makes maximal use of the work already done, by updating the even values based on the predicted odd values. Due to the fact that the update is based on the transformed odd coefficients, $\mu(\delta_{i+1})$, a double buffer of the input image is not required as the odd values are instantly replaced by their predicted value. Therefore, this transform requires no extra memory than its input size, and thus can be performed 'inplace'. The lifted SGWT is idea to implement on GPUs given their inplace nature and memory requirements, as is well documented in [151].

In Chapter 5 our photo-detail-consistency metric, Equation 5.1, was defined which requires not only a set of image pixels to calculate the measure, but also the texture channel weight, $\mathbf{w}_t$. The wavelet transform can offer these features at no extra computational cost. Previously, at the highest image resolution, the results of passing a Sobel edge detection kernel over the input image was used to compute the texture response, τ, in Equation 5.2. This is summed over the extent of the patch to compute its weight, $\mathbf{w}_t$.

With the availability of the detail coefficients at lower image resolutions, these can be used in place of the Sobel edge detection responses, Section 5.2.3, requiring less unnecessary calculation and the opportunity to better tune performance. In order to replace the texture responses of the Sobel kernel with the detail coefficients of the wavelet transform, a proper understanding of what these coefficients reflect is required.

Wavelets with a single vanishing moment can be thought of as edge detection kernels [3], where the detail coefficients respond when the signal is non-constant, and thus has some texture. For these reasons, we have chosen to utilize the separable integer Haar wavelet transform inside a second-generation lifted framework, [34]. At the cost of a small amount of accuracy resulting in a large memory saving, utilising the integer transform means that edge responses can be compressed into 8 bit integer values rather than 32 bit floats, as decimal storage is no longer required. The prediction and update steps of the integer Haar wavelet transform, [149], are given in Equation 6.3.

$$\begin{aligned} \delta_{i+1} &= \delta_i - \lambda_i \\ \lambda_{i+1} &= \lambda_i + \left\lfloor \frac{\delta_{i+1}}{2} \right\rfloor \end{aligned} \tag{6.3}$$

In order to further increase computational efficiency, the wavelet transform is performed inplace by running an optimised GPU kernel. Following a pixel to work-item mapping, image pixels at the higher resolution are first read into local memory, then each work-item calculates the approximation and detail coefficients for the lower resolution scale, when complete writing the results back to global memory. This is a well-studied area and the interested reader is refereed to [157] for implementation details. In order to pre-compute the texture responses, τ, before optimisation, the same kernel as previously defined in Section 5.2.3 is utilized. However, in order to approximate the image gradient the detail coefficients of the wavelet transform are used in the place of the Sobel kernel responses.

6.1.2.3 Multi-Resolution Experimentation

Now that the method for creating lower resolution datasets, and evaluating the photo-detail-consistency metric using the wavelet transforms detail coefficients have been defined, we can assess the validity of lower resolution reconstructions. To avoid redundant experimentation and for the sake of brevity, we chose to analyse the effects on patch error by running PSOPD over four differing image resolutions for the MonkeyLow dataset.

We chose the MonkeyLow dataset as the effects of noise were visually apparent in the highest resolution reconstruction. This dataset also provided the best example of large textureless regions, where ground truth is available to quantify the errors. As with previous chapters we give qualitative imagery, showing the patch clouds created over the multiple resolutions, as well as quantitative results of patch error, in the form of their probability distributions. Level 0 indicates the highest frequency image set, corresponding to the lowest pyramid level, where the higher the level the lower the image resolution.

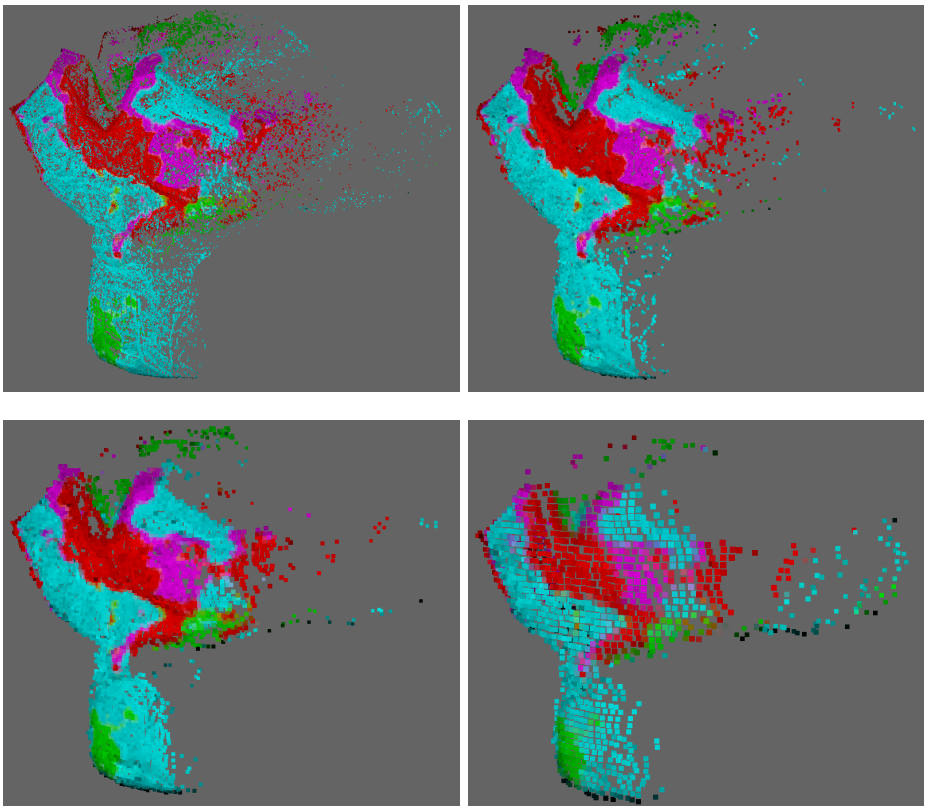


Figure 6.3: The patch clouds produced by running PSOPD on differing resolution images for the MonkeyLow dataset. The patch cloud created by the highest resolution image set, level 0, is displayed in the top left, through to the patch cloud created by the lowest resolution image set, level 3, in the bottom right

Figure 6.3 demonstrates the results of running PSOPD over the differing resolutions for only a single view in the dataset. For the best quality reconstruction, all patches should align correctly with the monkey's face and no holes or spurious patches should be present. This figure shows that, when refining patches at a lower resolution, there is a notable decrease in the amount of large erroneous noise, where more patches begin to align correctly with the face rather than sitting deeper inside the head. At the same time lowing the resolution reduces large errors in patch refinement, patches now aligning correctly with the surface begin to suffer an increase in surface smoothness. This is most noticeable on the monkey's mouth, eyes and eyebrows, where lower resolution reconstructions have more patches aligning correctly with the surface, but lack the sharp corners that were extracted in the high-resolution reconstructions. Interestingly for the MonkeyLow dataset, the lowest resolution tested showed very little 'noise' whatsoever, rather, only inaccuracies in terms of the level of surface smoothness.

By plotting the probability distributions, Figure 6.4 provides quantitative analysis of the effects of image resolution on patch error. These distributions show that as the image resolution is reduced, so are the effects of noise, however, more smoothing occurs. Erroneous noise in the reconstructions can be seen as errors higher than 0.1, whereas errors between 0 and 0.1 are seen as inaccuracies or smoothing in patch refinement. At lower resolutions this can be cased by the size of the patch, whereas at higher resolutions the level of texture over the patch causes smoothing over low textured regions.

The volume of patches in the tails of the distributions past errors of 0.1 are noticeably lower for levels 2 and 3 than that of levels 0 and 1. This shows that lower resolutions are much less likely to make large errors, i.e. noise. Whereas levels 2 and 3 have a lower peak and larger spread between errors from 0 to 0.1, showing that there are a large number of patches that lack the highest accuracy of levels 0 and 1, and thus represent a smoothing of the patch cloud.

One should note the effects of surface smoothness as the image resolution is reduced. Although the patch noise decreases, so to does the peak accuracy of the patch cloud. This reflects the fact that for lower resolution images, equivalent patch sizes in image space approximate a larger planar surface, similarly to increase the patch size whilst keeping the resolution constant. This results in smoother surface refinement, thus a drop in accuracy, widening the spread of the distribution.

From the results seen here we make the following judgements on the validity of a multi-resolution approach:

- Patches at lower resolutions are more robust to erroneous refinement, but lack high fidelity due to surface smoothing.
- Patches at the highest resolutions can achieve peak fidelity, however, are susceptible to errors as the textureless effect is increased.

To move forward with this approach we would like to combine the robustness of lower resolution refinement with the performance of higher resolutions. Naturally this leads to a propagation procedure where patches can be robustly refined at the lowest resolution and then the results can be used to infer the likely position of a patch at a higher resolution.

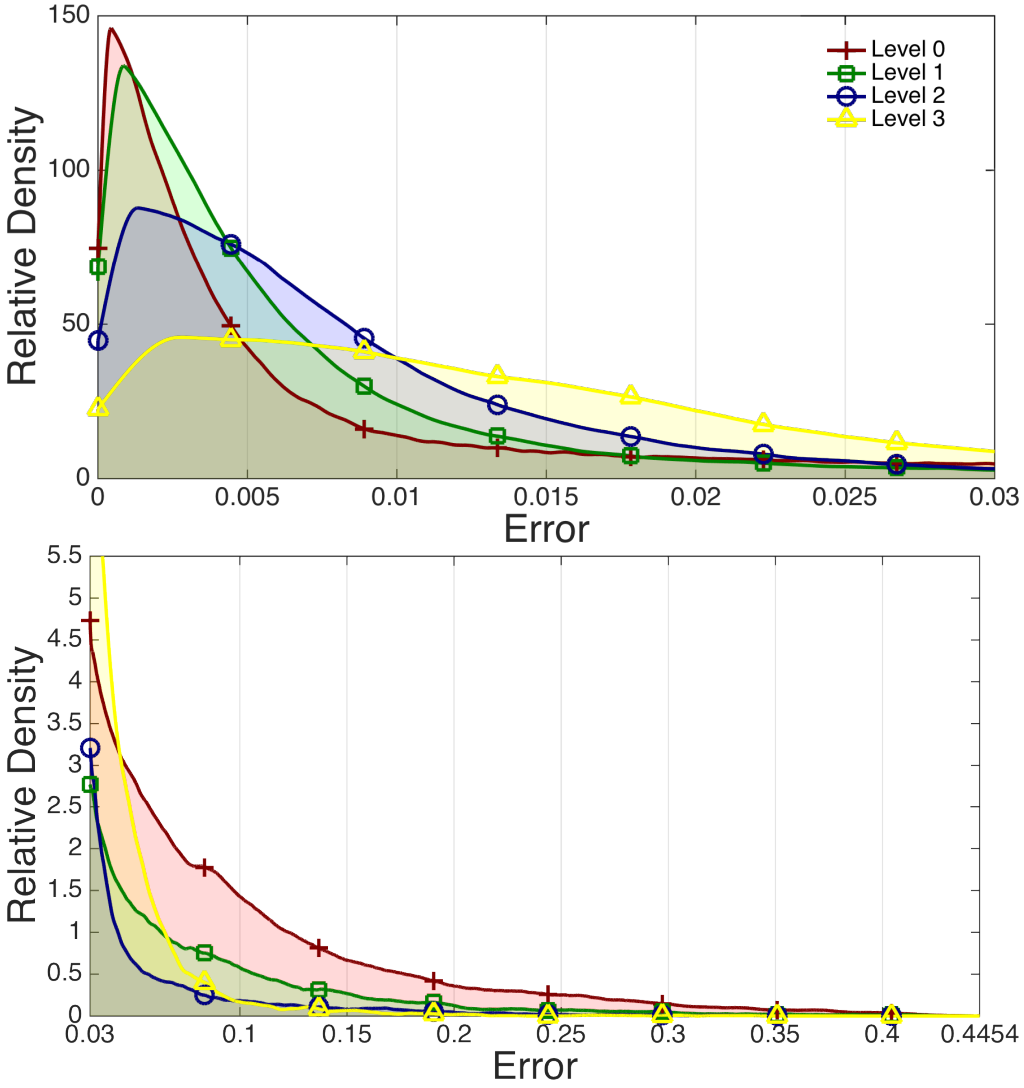


Figure 6.4: Probability distributions of the patch clouds created for the MonkeyLow dataset over each of the image resolutions

6.1.3 Multi-Resolution Framework

Now that we have tested the validity of a multi-resolution approach for PSO based refinement, we have seen that patches generated at lower resolutions have an equivalent effect to increasing the patch size, they are more robust yet increase smoothing. Keeping patch size constant whilst lowering the image resolution, removes the computational burden of extra sampling, this is an extremely desirable feature for algorithms where run time is an important factor. In this section our multi-resolution propagation technique, MRPSO, for 3D reconstruction will be defined as well as its relation to existing algorithms.

6.1.3.1 Previous Frameworks

The use of multi-resolution datasets and image pyramids within the stereo community has shown great advances in recent years, leading to some of these techniques being adopted into MVS. However, as the multi-view case is sufficiently more complicated than its binocular stereo equivalent, porting such techniques is not a straightforward operation. Fuhrmann and Goesele [49] discuss the issues of merging multiple depth maps of differing resolutions, showing that merging lower resolution depth maps containing global scene data, with local high-resolution depth maps can lead to significant surface smoothing unless dealt with properly.

Habbecke and Kobbelt [62] first employed a gradient based energy minimisation framework for fitting predefined planes to corresponding epipolar lines, they suggested using a Gaussian pyramid technique to define the plane parameters at a lower resolution and use these as initialisation for the same gradient based schemes at the highest resolution. This helped them to avoid local minima in the gradient based scheme, similarly to how feature matching and expansion aids PMVS. Liu et al. [103] then further demonstrate the use of multi-resolution approaches tailored to depth map schemes, by refining depth maps in a variational framework starting from differing image resolutions in what they termed a Multiple Starting Scales Framework. Results of their work showed the importance of initialisation, where the gradient based energy minimisation framework is extremely susceptible to local minima. They showed that starting from higher scales the variation framework was susceptible to local minima close to the visual hull used for initialisation. Whereas lower scales could

correctly recover deep concavities, however, inside textureless areas their approach led to a severe level of over smoothing. Whilst these frameworks have all shown improved results against starting at the highest resolution alone, they are specifically tailored to the limitations of a gradient based scheme. When using such schemes accurate initialisation of the input parameters is of the utmost importance. As we have shown, this is not necessary for PSO based techniques, where only the definition of the search space impacts on the overall refinement outcome. Therefore, such techniques do not extend fully to our framework or circumstance.

Yang and Pollefeys [167] suggest combining the robustness of large patch sizes with the accuracy of higher resolutions for stereo depth map creation. Their GPU based method operates by summing the resulting correlation curves along sampled epipolar rays over a multi-resolution scheme. Whilst they only use a Sum of Squared Difference similarity metric, their results show promise, but cannot guarantee that high frequency texture will contribute significantly to the total score. Since their method is only defined for the stereo case, summing correlation curves is not so easily extended to the multi-view case, where visibility would inevitably distort the summation procedure. Their use of discretised samples for computing the correlation curves also limits the accuracy when compared to a continuous approach. Although not strictly a multi-resolution approach, Goesele et al. [57] present an intriguing coarse-to-fine sampling strategy to deal with discretisation errors. First, the epipolar ray is sampled with a fixed sample rate, once the optimal position is found, a second search within a restricted interval is re-run to reduce the quantisation errors of sampling.

6.1.3.2 Our Framework

Our approach to defining a multi-resolution strategy, relevant to PSO, requires the fusion of the methods presented above. By actively restricting the search space for GLN-PSO, Chen et al. [26] improved the performance of their reconstruction. We wish to follow a similar procedure whilst avoiding the necessity for expansion and strict filtering, instead opting for a multi-resolution propagation approach.

By refining patches via PSOPD for the lowest resolution of each input image, we are able to robustly locate the approximate positions of the

patches. Following a coarse-to-fine strategy presented above, the results of patch refinement can be used to reduce the search space of the patches at the next resolution, propagating up our results with a certain level of confidence.

Our MRPSO algorithm thus proceeds as follows: First run PSOPD at the lowest resolution creating our base level patch cloud. For the next resolution, rather than search the entire viewing rays intersection with the visual hull, such as we did in standard PSOPD, we restrict the space to values around the previous levels depth only.

The amount of restriction is based on the confidence in the current patch's position, this is related to the current resolution we are reconstructing and the base resolution created. Results from the previous section showed how patches created at lower levels are robust to textureless refinement, however, the depth accuracy of these patches is less reliable. These results are modelled by our propagation technique by restricting the search space less at lower resolutions, where accuracy is lower but robustness is high, and more at high resolutions, where accuracy is higher but robustness lowers. Therefore; assuming that as the image resolutions are raised, the position of the patch becomes more reliable, the restriction factor is modelled in Equation 6.4 below:

$$k = \frac{\min(d_z, \min(d_x, d_y))}{(20 \cdot 2^{(s_n - 1) - s_c})} \tag{6.4}$$

Here, k is the restriction factor calculated based on the volume dimensions of the visual hull, d_x, d_y and d_z, base level, s_n, and current level, s_c. The volume dimensions are found by computing the bounding volume of the visual hull. The depth values defining the search space for PSOPD to search are restricted to $\lambda \pm k$, where λ is the depth value for the patch at the previously computed resolution. This search range is of course further clipped to ensure the it does not move out of the depth range defined by the intersection of the viewing ray to the visual hull. Whereas restricting the search space based on expansion from neighbours is not valid across depth discontinuities, our multi-resolution approach should adapt much better to these situations.

In our previous PSOPD algorithm a patch is only created for a pixel if its viewing angle to the current surface model is below fifty degrees.

In this improved MRPSO version, we adapt this threshold based on the level under reconstruction. Firstly at the base level this threshold is ignored, creating a patch for every pixel regardless of its viewing angle, this ensures adequate surface approximations are available when propagating to higher resolutions. For higher resolution reconstructions we gently loosen this threshold as we become surer of the patch's true position and orientation. This threshold is reduced using the Equation 6.5 below, where a is the angle threshold:

$$a = \begin{cases} \cos\left(\frac{(50+10s_c)\pi}{180}\right), & \text{if } 50 + 10s_c \leq 90 \\ 0, & \text{otherwise} \end{cases} \tag{6.5}$$

Since initialisation from the base level and propagation throughout the resolutions can still cause the restricted space not to contain the correct surface parameters, once all images build patch clouds to the highest resolution, a final filtering takes place. The idea of this filter is to remove patches that have been restricted such that they no longer align correctly with the surface.

As discussed, PMVS uses depth map filtering to enforce consistency, much like the depth map techniques presented in [140, 154]. However, to perform effectively, this relies on image redundancy created through large input image sets. Sparse image sets, which lack image redundancy, can cause these methods to filter out correctly refined patches, when a patch under construction is not seen and supported by more than two views, Section 6.1.1. To avoid these techniques, and account for sparser view sets retaining high density, we have chosen to experiment with a per-view filtration system.

Noise in our reconstruction framework, tends to be of significant deviation from its correct position. To avoid using redundancy to remove these outliers, we would like to use only the patch's reference view to determine its refinement success. Under a surface smoothness assumption, we observe the fact that if refinement was successful, the current patch and its neighbouring patches, within its reference view, should be at similar depths. This is of course violated across depth discontinuities, however, in such cases patch refinement is likely to be inaccurate, and the patch should be viewed better in a different image. Therefore, patches sitting directly over depth discontinuities should be removed too.

To validate the position of a patch, p, we first collect all patches within a given radius, r, to p. The larger this radius the more surface smoothness is enforced. As we would like to keep high frequency surface details, setting this radius to the 8-pixel neighbourhood of the current patch is required. If there are fewer than p_1 patches in the neighbourhood of the current patch, p is filtered as an outlier. In our experiments we set p_1 to 3, which was sufficient in all cases. Even though p may be correct when p_1 is low, since there is not enough support to infer the correctness of p, we choose to reject it. This is a feasible solution as patches that do not have many neighbours are most likely to sit on the boundary of the foreground and background, thus patch refinement is less likely to be correct in these locations.

Once a patch has collected its neighbours, P_N, the average depth difference, d, between its world space location, $\mathbf{p}$, and its neighbours locations, $\mathbf{p_n}$, is computed and weighted with the patch's normal, $\mathbf{n_p}$, via Equation 6.6 below:

$$d = |P_N|^{-1} \sum_{p_n \in P_N} (\mathbf{p} - \mathbf{p_n}) \cdot \mathbf{n_p} \tag{6.6}$$

To account for the fact that patches further from the camera centre are likely to have a larger depth difference than those nearer the viewing plane, the approximate depth volume, v, should be found. The depth volume, v, is computed on a per patch basis as it varies per pixel. This is computed based on the camera's world space centre, $\mathbf{c}$, and the camera's focal length, fl, in Equation 6.7 below:

$$v = \frac{|\mathbf{c} - \mathbf{p}|}{2^{s_c} fl} \tag{6.7}$$

A patch whose average depth value varies more than the scaled depth volume is then marked as an outlier in the following binary filter function, ϵ, in Equation 6.8 below:

$$\epsilon(p) = \begin{cases} 1, & d > v \cdot r \\ 0, & \text{otherwise} \end{cases} \tag{6.8}$$

6.2 Experimental Results and Analysis

In this evaluation section, the improvements made by our MRPSO technique are analysed with respect to its noise reduction compared to PSOPD, and improved density in relation to PMVS. The aims of our MRPSO framework are to enhance the reconstruction accuracy over textureless regions and remove spurious noise that falls far from the correct surface position. Thus defining a new patch based technique that is equivalent in accuracy to PMVS, whilst also creating patches inside large textureless regions, where PMVS fails.

For consistency with the previous chapters, reconstruction results are analysed over both the real and synthetic datasets. Results have shown the distinct advantage of our method for creating patches inside textureless regions, thus creating denser reconstruction when compared to PMVS. The propagation of restriction factors and filtering methods proposed clearly improve the accuracy as well as the noise in patch refinement over using PSOPD alone, however small amounts of noise still persist.

As with PSO and PSOPD presented in the previous two chapters, the parameters for running MRPSO were kept consistent with those used earlier. The exceptions being the viewing angle threshold and depth search range, which are both adapted throughout the levels of reconstructions by our MRPSO approach, Section 6.1.3. The lowest resolution used for the base level of the Monkey, Buddha, Dino and Temple datasets was level 3. Given the higher image resolution of the LynxCAD and LynxReal datasets, we were able to use level 4 as the base level for these reconstructions.

Qualitative results are presented, highlighting the overall improvement of our MRPSO approach when compared to PMVS. The improved density of reconstructions are clearly visible, with well-textured regions achieving equivalent accuracies, however, some noise still persists in textureless regions. Quantitative results compare MRPSO with both our PSOPD algorithm and PMVS. These show that our propagation and filtering techniques successfully reduce patch noise, whilst also providing improvements to the overall accuracy of extracted patch clouds. Numerically comparing our method to PMVS shows equivalent results, however,

our method still produces slightly more inaccurate patches, most predominantly caused by inaccuracies in textureless regions. Unsurprisingly, analysis of the speed of reconstruction shows a significant increase in run time, caused by the increased workload of refining patches over multiple resolutions. We argue that although the run times are higher, the GPU parallel nature of this approach makes it scalable to better more advanced hardware systems, which would undoubtedly vastly improve execution time.

6.2.1 Qualitative

Figures 6.5 to 6.10 show the resulting patch clouds for running PMVS against those created by our MRPSO framework. The aims of this final chapter where to further reduce noise and improve on the overall accuracy of reconstruction, whilst also introducing some filtering methodology to remove badly refined patches. Therefore, in this section we will analyse the achievements of this MRPSO technique in terms of the following factors: The reduction in patch error and noisy refinement, the effect on high frequency detail, accuracy and smoothness when texture is reduced and finally any persisting limitations of the system.

Noise and Error

The patch clouds created using our multi-resolution approach are clearly of higher quality in terms of showing a significant improvement in the density of reconstruction over the entire object's surface. Where patch expansion in PMVS fails to move into textureless regions, our MRPSO approach densely reconstructs such areas. Observing the MonkeyLow or Dino reconstructions, Figures 6.5 and 6.8, our dense coverage of the surface where no prevalent features exist, such as the chin of the Monkey and back of the Dino, is clear. It should be note that even in high textured situation, such as the MonkeyHigh or BuddhaHigh datasets, Figures 6.5 and 6.6, PMVS failed to densely cover all surface regions, whereas our approach had no issues in doing so.

Though our propagation procedure ensures an overall improvement in noisy refinement, particularly inside large textureless regions, this alone cannot remove all noise; therefore, our per-view filtering method

is required. Looking at the results for MRPSO, our filtering method has clearly succeeded in removing patches that lie at a large distances from the correct surface, such as the noise we experienced in the ears of the Monkey dataset, Figure 6.5. Whilst still successfully reconstructing almost the entire object, gaps in the object are caused by the fact no image in the input set observes these locations, thus no patches can be produced in these regions, such as the very top of the Monkey's head. Alternatively, there are a few situations where patch refinement can fail and the filtering method removes these patches, causing holes in the surface. This can be predominantly seen in the base and lid sections of the LynxCAD or LynxReal datasets, Figures 6.7 and 6.10; however, the effects of this are far less severe than PMVS.

Analogously to the previous chapters, Figures 6.11 to 6.13, compare the patch clouds created via MRPSO to PSOPD, for the synthetic datasets that have been filtered to remove all patches lying at a distance further than 0.005 units from the ground truth. Whereas, Figures 6.14 to 6.16, display the reduced effects of noise for the real life datasets, via a single view's patch cloud computed via MRPSO when compared to PSOPD.

Figures 6.11 to 6.13 clearly show the improvement in density of our MRPSO approach, there are now many more patches lying within 0.005 units of the surface than compared to using PSOPD alone. Especially of note is the fact that many of these patches now lie inside textureless regions, showing that through our MRPSO scheme, refinement of patches in textureless regions is now possible. In fact comparing these clouds with the results of running PMVS, in the previous figures, shows a significant improvement in terms of density. However, these results also highlight an issue with our filtration method. Previously, patches correctly refined in sparse areas were left unfiltered, however, with our filtering methodology patches must be supported by at least 3 others in a nearby area. When this is not the case, such as along the top section of the base for the Buddha model, Figure 6.12, these sparse but correctly refined patches are removed, leaving a gap in the model that was not present in PSOPD.

For all real life datasets, Figures 6.14 to 6.16, patch clouds generated by MRPSO are of consistently higher quality with a significant reduction in all patch noise, however some noise still persists. In particular the effectiveness of our propagation technique is clearly visible, patches that were previously noisy in our PSOPD reconstruction are now correctly

aligned with the surface, whilst noisy patches with large distances to the correct surface have been removed, see Figure 6.14. Some remaining noise in the Temple and LynxReal datasets, Figures 6.15 and 6.16, are still visible, caused by the inabilities of the per-view filtering process to remove these patches, nevertheless, the result show a drastic improvement in all aspects of the patch cloud.

High Frequency Details, Accuracy and Smoothness

Note in particular, when surface texture drops, patch refinement in PMVS becomes inaccurate, see the stomach or base section of the BuddhaLow dataset, Figure 6.6. Through the propagation of restriction factors within our MRPSO framework, refinement is of notably better quality over equivalent regions, defining a surface with more high frequency detail. Again the LynxCAD and Dino datasets, Figures 6.7 and 6.8, show the improvements propagation and restriction make to completely textureless regions. Restricting the depth values through a multi-resolution approach has clearly improved the quality and accuracy at which patches can be refined in these regions. However, when comparing patches inside textureless regions to those in highly textured circumstances, these patches are of slightly lower accuracy, brought about by small errors made inside the restricted search space. This can be seen in the Monkey reconstructions where sharp edges are captured in the high textured model, but are less sharp and smoothed in the low textured model, Figure 6.5.

Some noise still affects the accuracy and smoothness of the solution, seen particularly around the most high frequency surface details, such as the columns of the Temple reconstruction, Figure 6.9. There are large clusters of patches in similar positions but different orientations, our filtering method leaves these patches assuming they reflect high frequency surface detail. PMVS was more successful in removing this type of noise, using their pixel blocking approach. Whereas in our setting, multiple images can produce equivalent patches, such redundancies are not removed by our filter and can lead to some noise over high frequency reconstructions.

Limitations

By carefully observing patches produced over textureless regions, one can notice a limitation to our current framework. Although restriction of a patch's depth range clearly enables it to converge onto the correct depth position, we chose not to restrict the patch's normal parameters in a similar fashion. We found that restricting a patch's normal parameters over the differing scales was highly susceptible to errors as the resolution grows. This is cased by the fact that at high resolutions, patches can express significant deviation from their lower resolution counterparts, as they begin to incorporate high frequency surface details. Therefore, we found that restricting the normal search space was not appropriate. However, without restriction in textureless areas the normal parameters can experience some noise over these regions, this can be seen in the slight illumination variance over textureless surfaces such as the chin of the MonkeyLow dataset, Figure 6.5. Of course when there is sufficient texture, normal refinement is not an issue, such as along the equivalent MonkeyHigh reconstruction.

The effectiveness of our filtration system has shown to be linked to the abilities of the base level to refine patches to an acceptable standard. If patch refinement at the base level fails and restriction removes the correct parameters from the search space, this error is propagated to the top level, creating a large area of patches within the same incorrect vicinity. Our filtering method then fails to remove these patches through its outlier rejection scheme. The LynxCAD, Temple and LynxReal reconstructions, Figures 6.7, 6.9 and 6.10, have fallen victim to this limitation. Note the remaining noisy patches in the concavity of the Temple or the spurious points inside the lid and at the base of the LynxReal model, Figure 6.16. We hypothesise that in these cases, reducing the resolution actually created a textureless effect rather than inducing texture, blurring out all texture detail. When patch refinement was run at the lowest level, this failed and propagated bad results upwards.

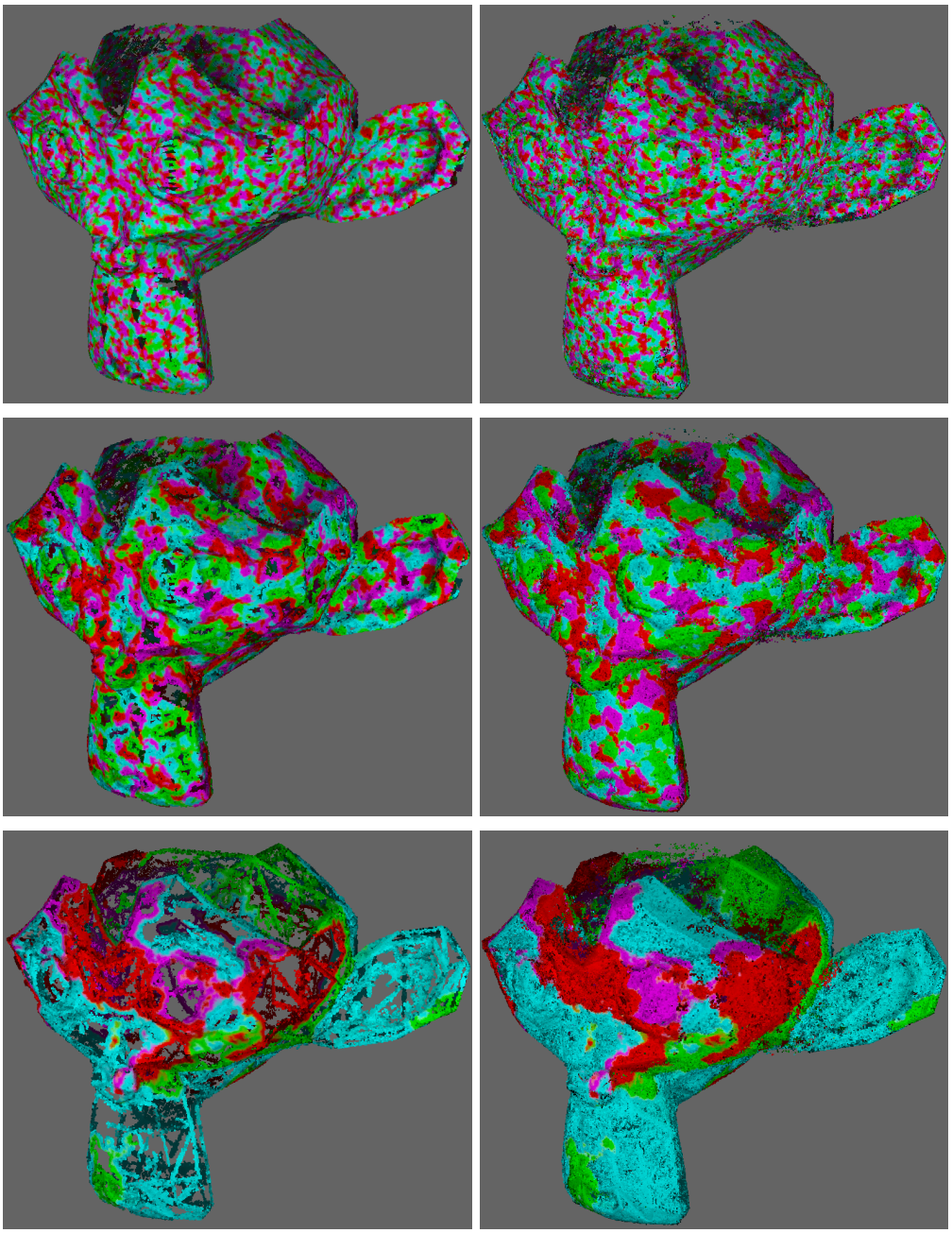


Figure 6.5: Reconstruction results for the Monkey dataset
with varying levels of texture.
Left Column: PMVS results. **Right Column**: MRPSO results

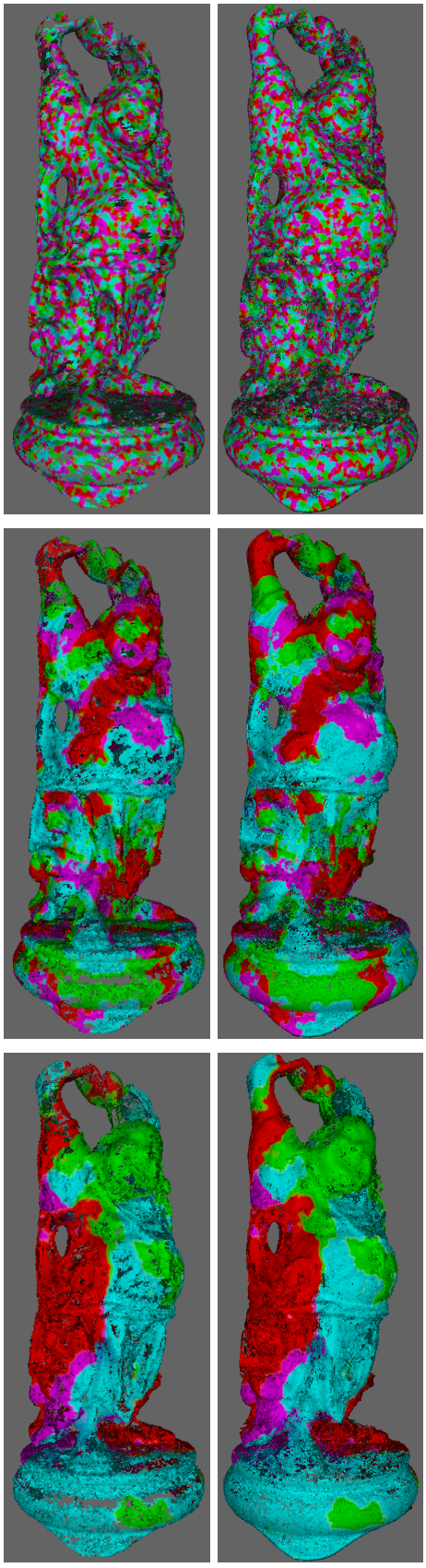


Figure 6.6: Reconstruction results for the Buddha dataset
with varying levels of texture.
Left Column: PMVS results. **Right Column**: MRPSO results



Figure 6.7: Reconstruction results for the LynxCAD dataset
Left Column: PMVS results. **Right Column**: MRPSO results

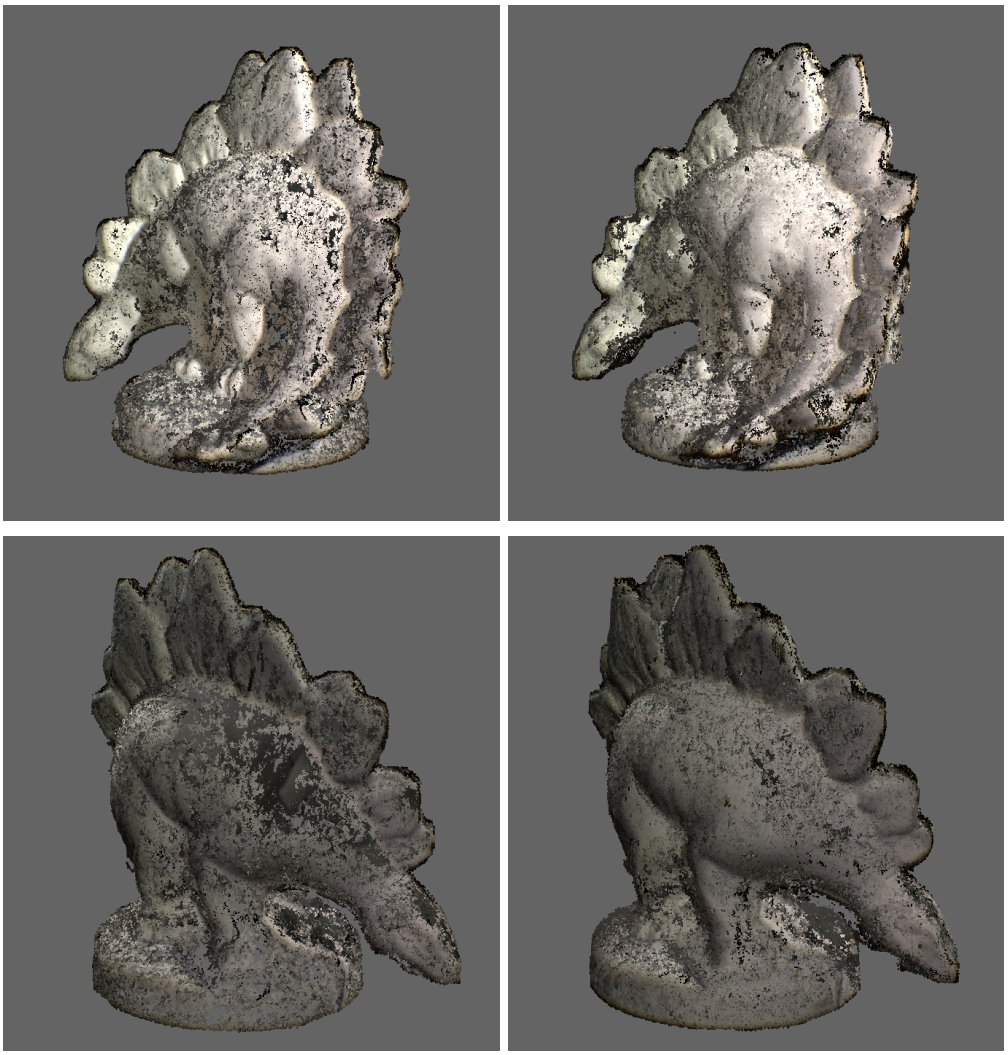


Figure 6.8: Two views of the reconstruction results for the Dino dataset
Left Column: PMVS results. **Right Column**: MRPSO results



Figure 6.9: Two views of the reconstruction results for the Temple dataset **Left Column**: PMVS results. **Right Column**: MRPSO results

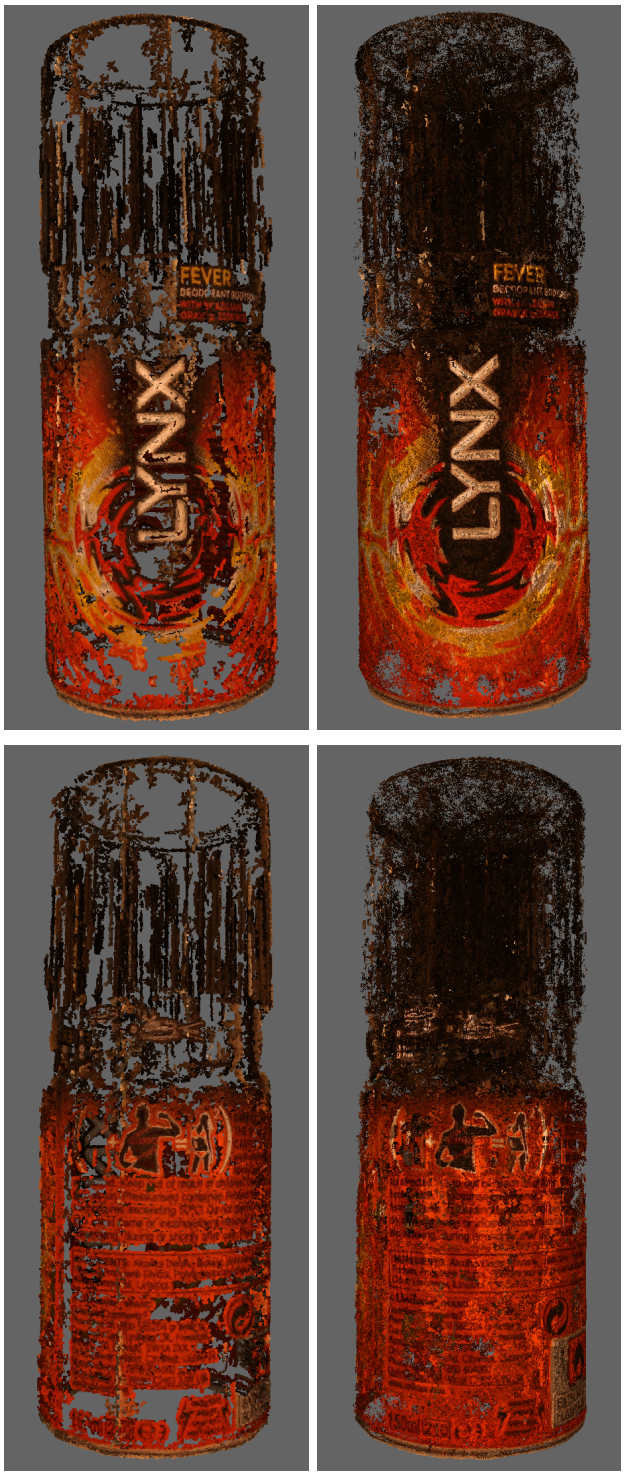


Figure 6.10: Two views of the reconstruction results for the LynxReal dataset
Left Column: PMVS results. **Right Column**: MRPSO results

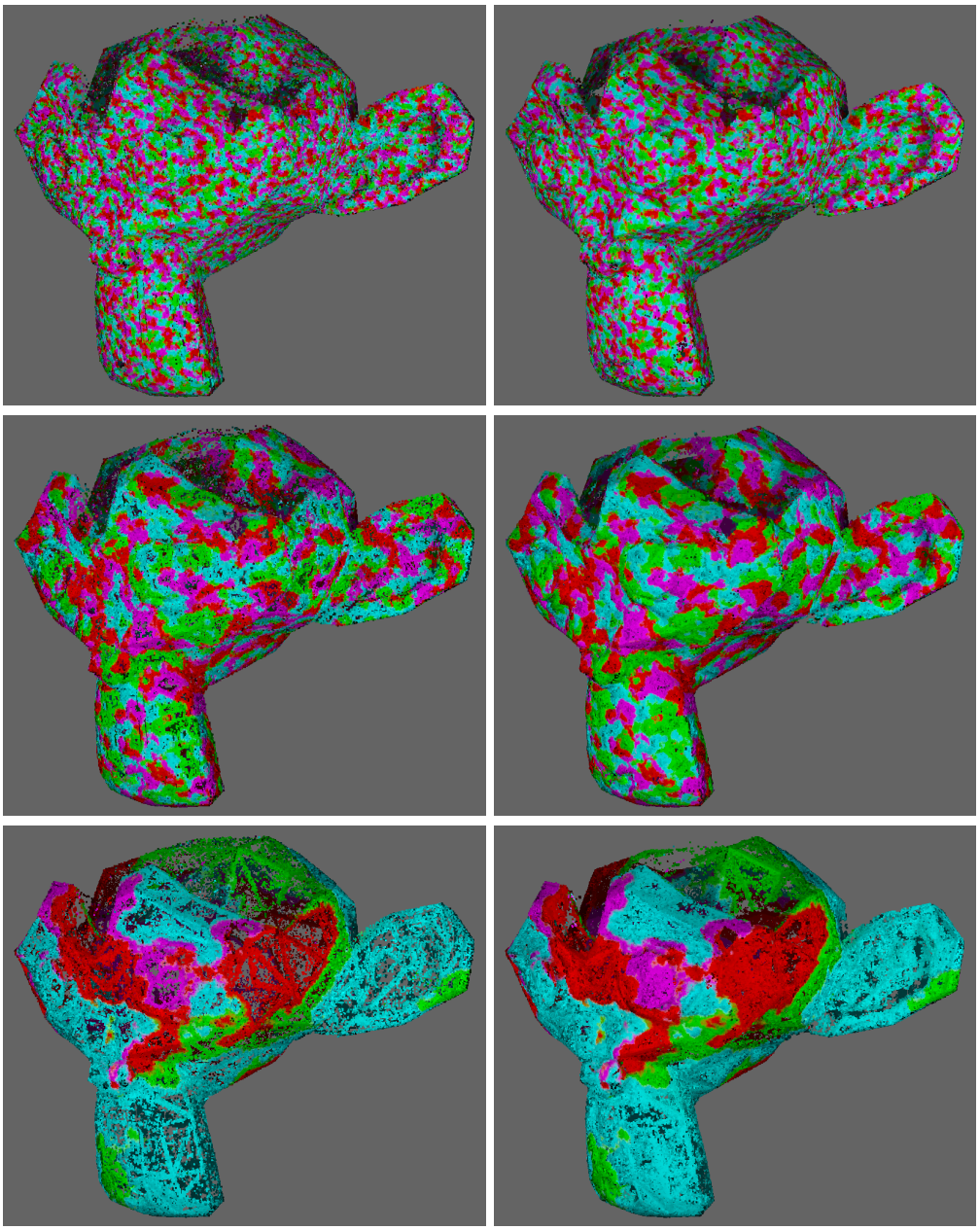


Figure 6.11: Noise reduced patch clouds created for the Monkey datasets by PSOPD (left) and MRPSO (right), these are filtered by removing all patches in the cloud whose distance is more then 0.005 units from the surface

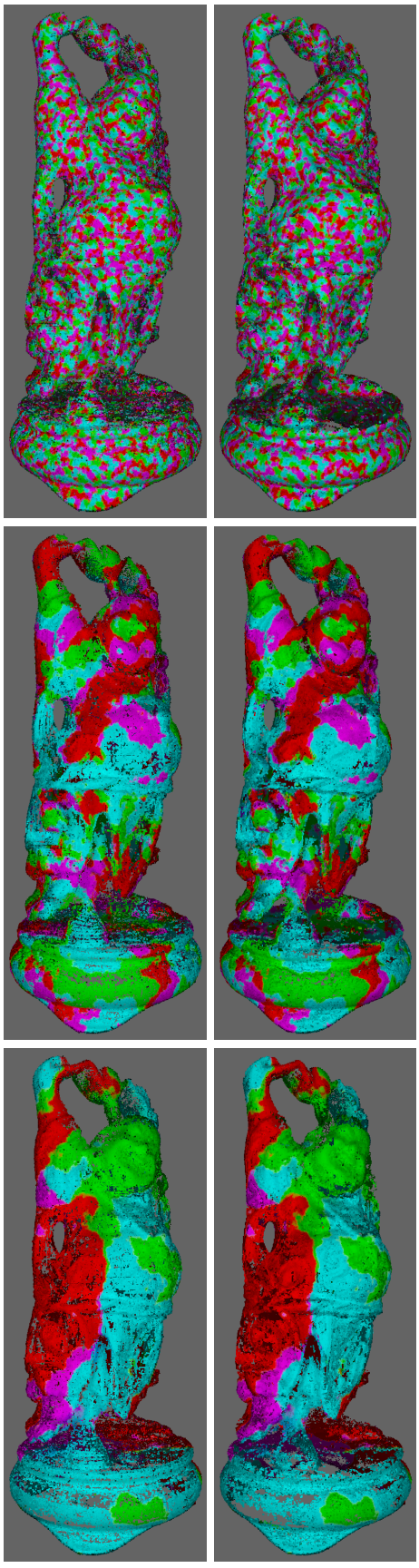


Figure 6.12: Noise reduced patch clouds created for the Buddha datasets by PSOPD (left) and MRPSO (right), these are filtered by removing all patches in the cloud whose distance is more then 0.005 units from the surface



Figure 6.13: Noise reduced patch clouds created for the LynxCAD dataset by PSOPD (left) and MRPSO (right), these are filtered by removing all patches in the cloud whose distance is more then 0.005 units from the surface

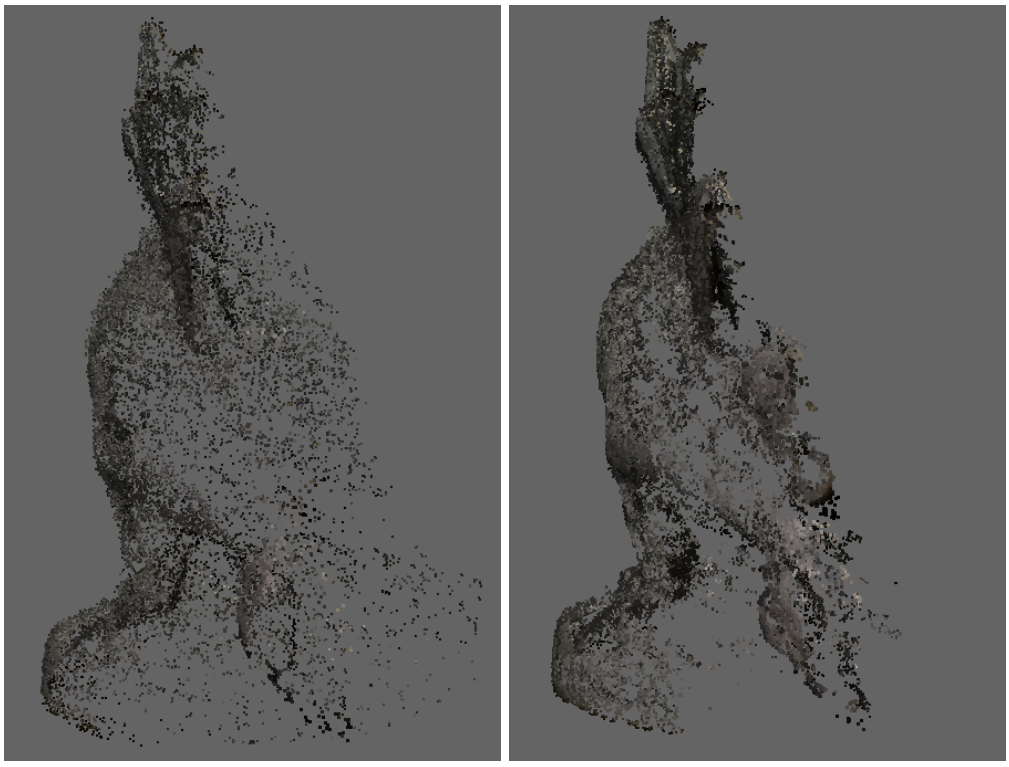


Figure 6.14: Patches created by PSOPD (left) and MRPSO (right) for a single view from the Dino dataset

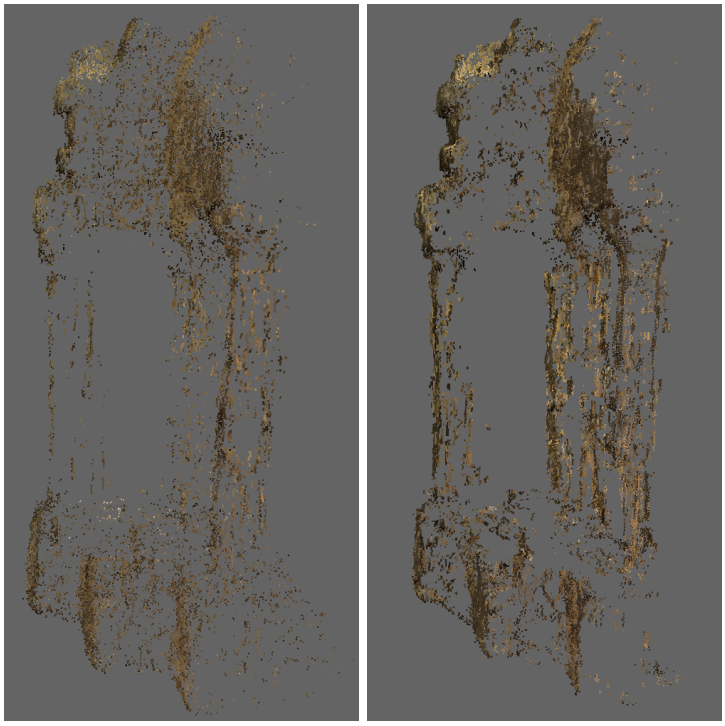


Figure 6.15: Patches created by PSOPD (left) and MRPSO (right) for a single view from the Temple dataset



Figure 6.16: Patches created by PSOPD (left) and MRPSO (right) for a single view from the LynxReal dataset

6.2.2 Quantitative

Throughout the qualitative results presented in the previous section, the improvements of our MRPSO approach towards creating denser more accurate patch clouds are visually apparent, this section provides numerical analysis of these substantial gains. Similarly to analysis in the previous chapters, probability distributions for the patch error, measured by the one-sided Hausdorff distance metric, are given in Figures 6.17 to 6.22. Initially, the probability distributions for the patch clouds created via MRPSO are plotted along side their corresponding PSOPD counterparts, allowing for the analysis of the accuracy improvements made by our propagation and filtering methods. Following this the distributions for MRPSO are then plotted along side their equivalent PMVS reconstructions, allowing for the assessment of our methods performance in comparison to the current top performing approach. For firm numerical results, Tables 6.1 to 6.3 provide statistical information after the error distributions are given for each reconstruction. To conform to the abbreviations used in the previous chapters, MRPSO High, MRPSO Med and MRPSO Low, will correspond to the reconstructions created for the high, medium or low textured datasets via our multi-resolution framework respectively. This section will discuss the improvements in our MRPSO technique via the following two factors: The improvement made when using MRPSO as opposed to PSOPD, and MRPSO's performance in comparison to PMVS.

MRPSO Performance Comparison with PSOPD

Section 6.2.1 gave visible evidence of the density improvements made by MRPSO. The overall improvement in the number of patches correctly and accurately refined via our MRSPO technique when compared to PSOPD is further reflected by observing the inverse cumulative frequency tables. The percentage of patches with an error greater than 0.005 has been dramatically reduced in all cases, showing that there is now much fewer badly refined patches. As well as this the mean values for all distributions are considerably lower, observe that the mean values for the Monkey reconstructions are an order of magnitude smaller for MRPSO than the equivalent PSOPD reconstruction, Table 6.1. This shows that the overall accuracy the entire patch cloud achieves has now improved. The spread of each distribution has also been reduced, showing that there are

now many more patches within the low error sections of the distribution. This reflects the effectiveness of our propagation technique in accurately refining patches regardless of their texture.

The benefits of restricting the search space, in our multi-resolution propagation technique, has lead to an increase in the quality and accuracy of patch refinement. Utilising PSO inside a smaller search space, means that for the same number of iterations, particles can more thoroughly explore the space. This results in the fact that particles have an improved ability to find the absolute global optimum to a higher accuracy. The advantage restriction has made is reflected in the increase in the peak performance and decrease in the mode for each of the datasets tested. Observe the fact that the mode for both the Monkey and Buddha reconstructions, Tables 6.1 and 6.2, are now lower than using PSOPD alone.

Interestingly, the residual errors for the Monkey and Buddha reconstructions, Tables 6.1 and 6.2, for errors higher than 0.02 units, are lower for the medium textured datasets than that of their highly textured counterparts. One potential explanation for this is that the patch size (5x5) is more suited to the medium texture frequency than the high textured frequency, producing better results with less large error than that of the high frequency version, however since the difference is negligible we do not dwell on this point. Akin to this, the maximum error for the Buddha medium reconstruction, Table 6.2, is notably lower. Again, we attribute this to the patch size better suiting the texture frequency when propagation occurs in the lower resolutions, thus resulting in a distinct performance improvement. This highlights the importance of choosing the correct patch size for the object under reconstruction.

The incorporation of a filtering step into our MRPSO framework is another aspect that has led to improved performance against PSOPD. The intentions of this filtering stage were to remove patches that sat at a large distance from the true surface. The success of this filtering stage is visible in the reduced tails of each distribution. Comparatively to PSOPD, the tails of each distribution are considerably shorter and of lower volume. Analysing the inverse cumulative frequency tables shows that the percentage of patches with an error higher than 0.02, and thus definitely classed as noise rather than simple inaccuracies in refinement, has dropped considerably across all reconstructions.

However, not all badly refined patches could be removed by our filtering stage. This is shown by the fact that the max error throughout all reconstructions has only seen a small decrease, this highlights a potential area of future work. A hypothesis on the cause of where this happens was discussed in Section 6.2.1, and we recap here to highlight the problem. A limitation to our framework is the assumption that patch refinement is correct at the lowest resolution; when dropping the resolution of a dataset, in certain areas, this can increase the textureless effect, resulting in incorrect patch refinement at the base level. Propagating this incorrect position upwards whilst restricting the search space, then removes the correct location for the higher resolution patches and results in a large cluster of patches within incorrect locations. Our filtering system then fails to remove these clusters as their neighbourhood remains at consistent but incorrect depths.

The accuracy at which patches can be refined inside textureless regions remains lower than that of a well-textured patch, even through our propagation technique. As described in Section 3.2, this is due to the nature of a textureless patch and the fact that the photo-consistency score is less well defined in these situations. Through our MRPSO technique, the inclusion of more patches in low textured regions, which have been correctly refined but to a slightly lower degree of accuracy, has an interesting effect on the results. This is perhaps best displayed when comparing MRPSO to PSOPD for the LynxCAD dataset, Figure 6.22. Previously patch refinement in PSOPD either worked very well over textured regions, or failed randomly over all the textureless regions, this resulted in a peak at low error but with a large elongated tail. Now that our MRPSO framework is inplace, reconstructions contain significantly more patches within these low textured areas whose patch refinement succeeded but was naturally lower accuracy. This resulted in a shift in the peak performance towards a higher error. This also lead to the volume of patches at higher errors (0.1 and higher) in PSOPD shifting into the tail of MRPSO between 0.003 and 0.1, improving the overall accuracy but still incurring error inside textureless regions.

MRPSO Performance Comparison with PMVS

To assess MRPSO's true performance, we compare our results to PMVS, which accurately extracts patches, but only in well-textured regions. Analysing MRPSO compared to PMVS, shows that the peak performance and mode for the Monkey and Buddha reconstructions, Tables 6.1 and 6.2, are now exactly equal. This shows that the majority of patches via our MRPSO approach can be refined to an equivalently accuracy as PMVS, with the small inaccuracies in both methods most likely a consequence of bilinear interpolation used to sample a patch and the fact that the patch size was set to 5x5. This claim is further supported by the matching alignment and shape of the distributions for the highly textured datasets.

An important factor to consider when assessing a reconstruction framework, is its scalability in terms of the 'texturedness' of the object. As the texture over the object is reduced, qualitative results in Section 6.2.1 showed that our method is noticeably denser than PMVS and clearly less noisy than PSOPD. The quantitative results presented here show the effects of this added density. Our framework produces and keeps patches in textureless regions, as opposed to PMVS, which either cannot create patches in these circumstances or they are removed by filtering. Due to the fact that refining patches within textureless regions is not as accurate as within highly texture regions, for MRPSO the variation in error increases as the level of texture reduces. This can be seen in the results of the Monkey datasets, Table 6.1, where the mean grows as the texture decreases, whereas the mean stay roughly consistent for PMVS. Whilst we can produce many high accuracy patches for well-textured datasets, thus resulting in low error and a reduced spread, as the texture is reduced our framework will naturally include more textureless patches having lower accuracy, accounting for the rise in mean value. This increases the spread and lowers the peak performance when compared to PMVS.

As our distributions are closely aligned for high textured datasets, with the alignment decreasing with the texture, we account for this effect in the fact that our reconstructions are denser and contain a larger number of textureless patches with lower accuracy. Therefore, we argue that in low textured cases our method actually outperforms PMVS due to the density it inherits, all be it with a compromise in terms of the accuracy of these patches.

On the other hand the mean and residual errors passed 0.005 are higher for MRPSO than that of PMVS, suggesting that PMVS is in fact performing better. In many cases this is true, the patches PMVS create are only of high quality with very little noise, however they remain sparse. Whereas our approach extracts the equivalent patches along with a set of lower accuracy patches inside low textured regions, as well as being effected by some persisting noise. In the case where only the highest accuracy patches are desired it would be better to use PMVS, however, when attempting to reconstruct low textured objects, it would be beneficial to use MRPSO to reconstruct patches that would otherwise be missed by PMVS.

Finally, we make a last remark on the effectiveness of our filtering method verse that of PMVS. PMVS is highly efficient at producing only high accuracy patches, where their filtering technique uses image redundancy and regularisation to enforce strict rules on patch filtering. For the reasons discussed in the previous sections, we chose to experiment with a per-view filtering technique to avoid unnecessary sparsity. Our filtering is not as strict as PMVS, which can be seen by tails of each distribution having larger volume. Therefore, density is achieved but a compromise must be taken on the overall accuracy. As previously mentioned, the max errors remained high for our MRPSO approach due to some limitations of our filtering approach and assumption that lower resolution patches are correct. Interestingly the max error for the LynxCAD reconstruction, Table 6.3, was lower for MRPSO than it was for PMVS, representing the fact that restricting the search space via a multi-resolution approach can be more effective than that of expansion from neighbours.

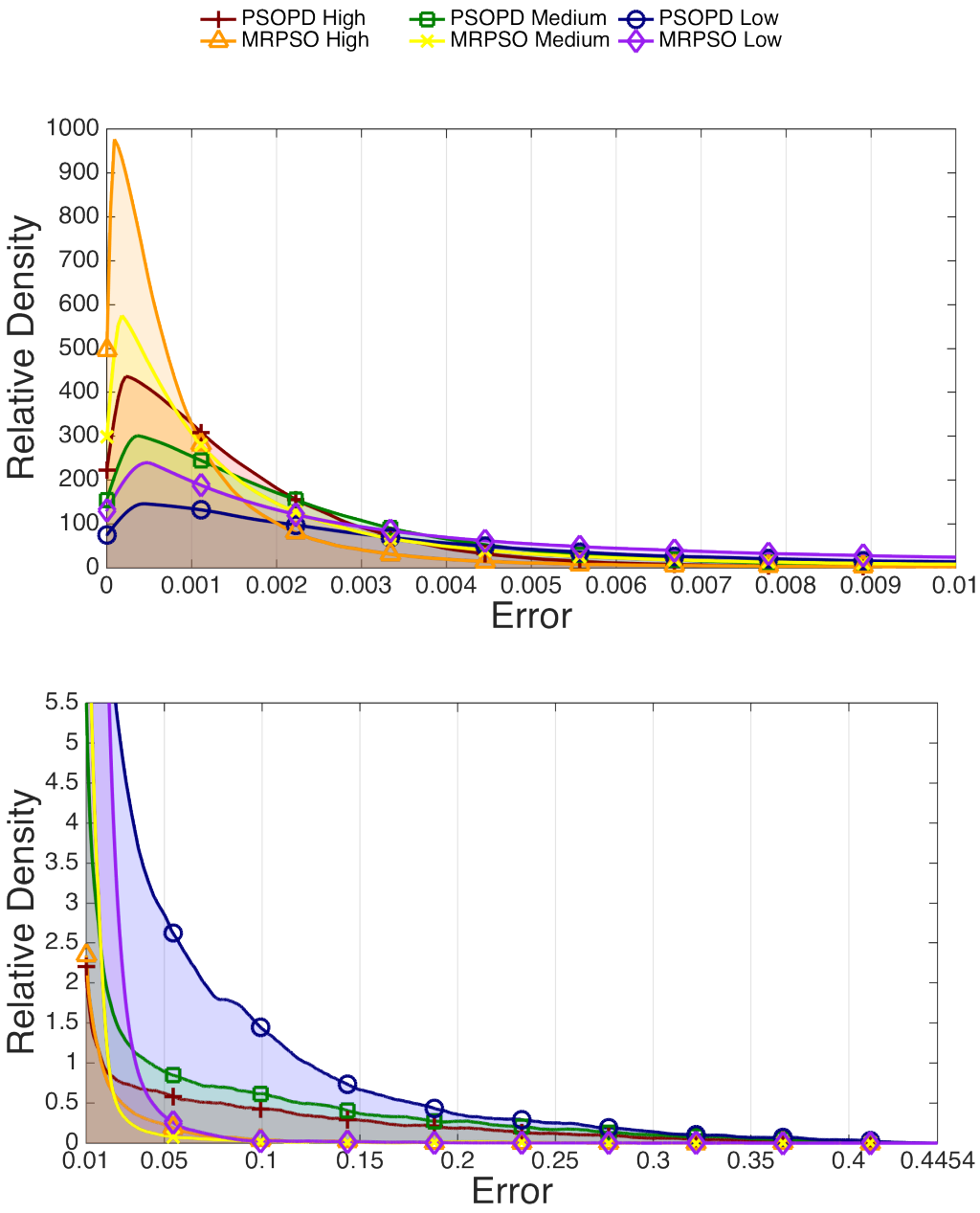


Figure 6.17: Probability distributions for the Monkey datasets
Top: Normalised probability distributions for the errors in patch refinement for running PSOPD and MRPSO on each of the three different textured datasets.
Bottom: Zoomed in tail end of the distribution up to the largest error

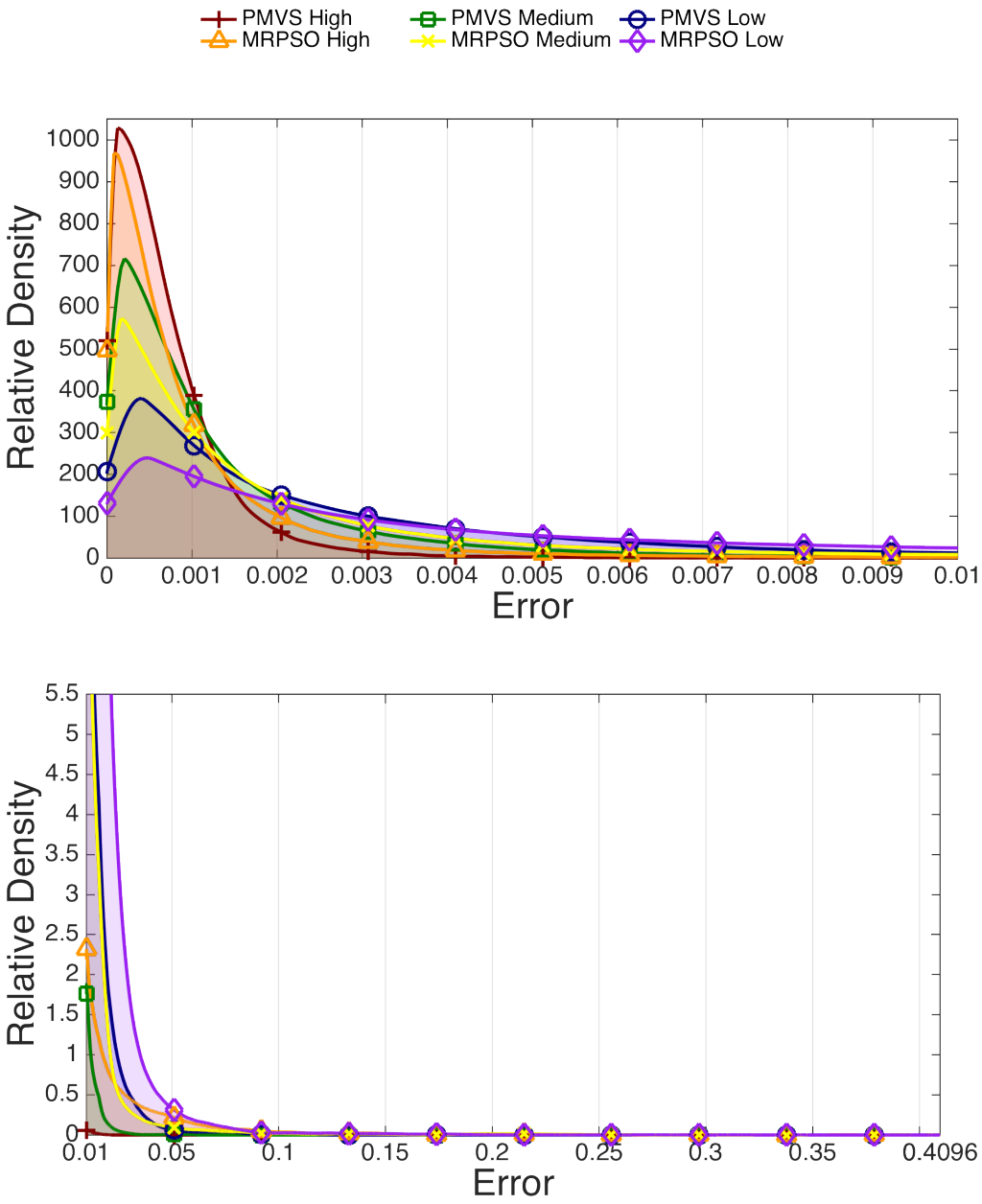


Figure 6.18: Probability distributions for the Monkey datasets
Top: Normalised probability distributions for the errors in patch refinement for running PMVS and MRPSO on each of the three different textured datasets.
Bottom: Zoomed in tail end of the distribution up to the largest error

		Mean	Median	Mode	Max
PMVS	High	0.0007	0.0005	0.0001	0.0180
	Med	0.0014	0.0008	0.0002	0.0580
	Low	0.0033	0.017	0.0004	0.1751
PSO	High	0.0377	0.0032	0.0006	0.4446
	Medium	0.0371	0.0039	0.0008	0.4439
	Low	0.0505	0.0148	0.0010	0.4481
PSOPD	High	0.0123	0.0014	0.0002	0.4437
	Medium	0.0179	0.0021	0.0004	0.4391
	Low	0.0335	0.0060	0.0005	0.4454
MRPSO	High	0.0022	0.0006	0.0001	0.4011
	Medium	0.0029	0.0012	0.0002	0.4096
	Low	0.0062	0.0030	0.0005	0.4025

		Residual Error							
		0.0	0.005	0.01	0.02	0.03	0.1	0.2	0.3
PMVS	High	100.00	0.34	0.02	0.00	0.00	0.00	0.00	0.00
	Med	100.00	4.43	0.61	0.06	0.02	0.00	0.00	0.00
	Low	100.00	19.49	6.07	1.48	0.59	0.04	0.00	0.00
PSO	High	100.00	41.94	34.91	31.23	28.61	14.98	5.09	1.21
	Medium	100.00	46.07	35.65	30.63	27.62	14.48	5.02	1.21
	Low	100.00	67.66	56.20	46.84	40.80	18.42	6.14	1.52
PSOPD	High	100.00	13.85	10.38	9.14	8.34	4.48	1.59	0.37
	Medium	100.00	24.51	16.68	13.65	12.13	6.43	2.42	0.62
	Low	100.00	54.24	42.38	33.31	27.82	10.89	3.36	0.87
MRPSO	High	100.00	5.63	3.19	1.91	1.31	0.19	0.03	0.00
	Medium	100.00	12.95	4.93	1.22	0.74	0.18	0.07	0.00
	Low	100.00	37.23	19.22	5.52	2.20	0.17	0.02	0.00

Table 6.1: Statistics for the Monkey datasets
Top: Different averages of the reconstruction error on each dataset.
Bottom: The inverse cumulative frequency of errors as a percentage of entire patch cloud

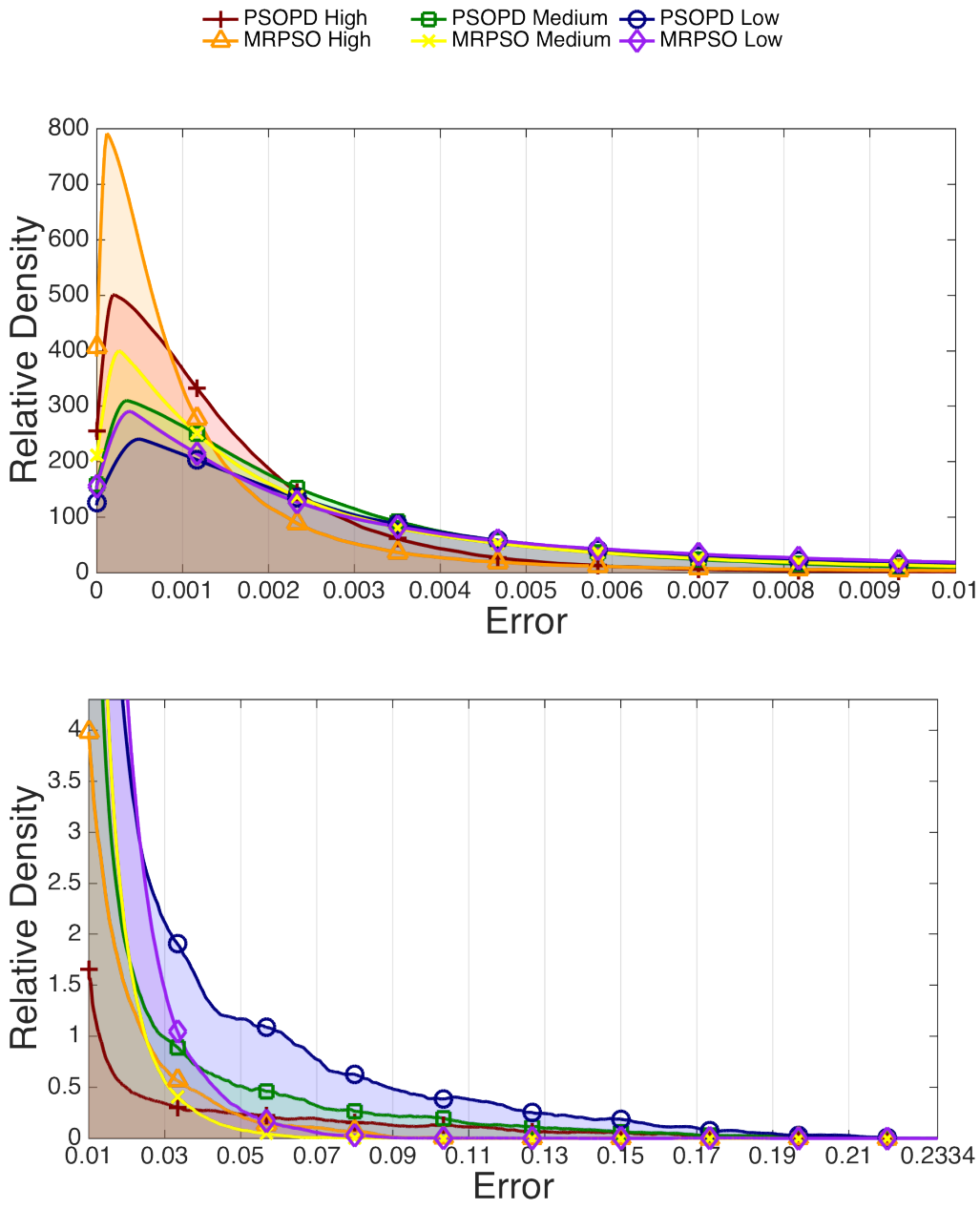


Figure 6.19: Probability distributions for the Buddha datasets
Top: Normalised probability distributions for the errors in patch refinement for running PSOPD and MRPSO on each of the three different textured datasets.
Bottom: Zoomed in tail end of the distribution up to the largest error

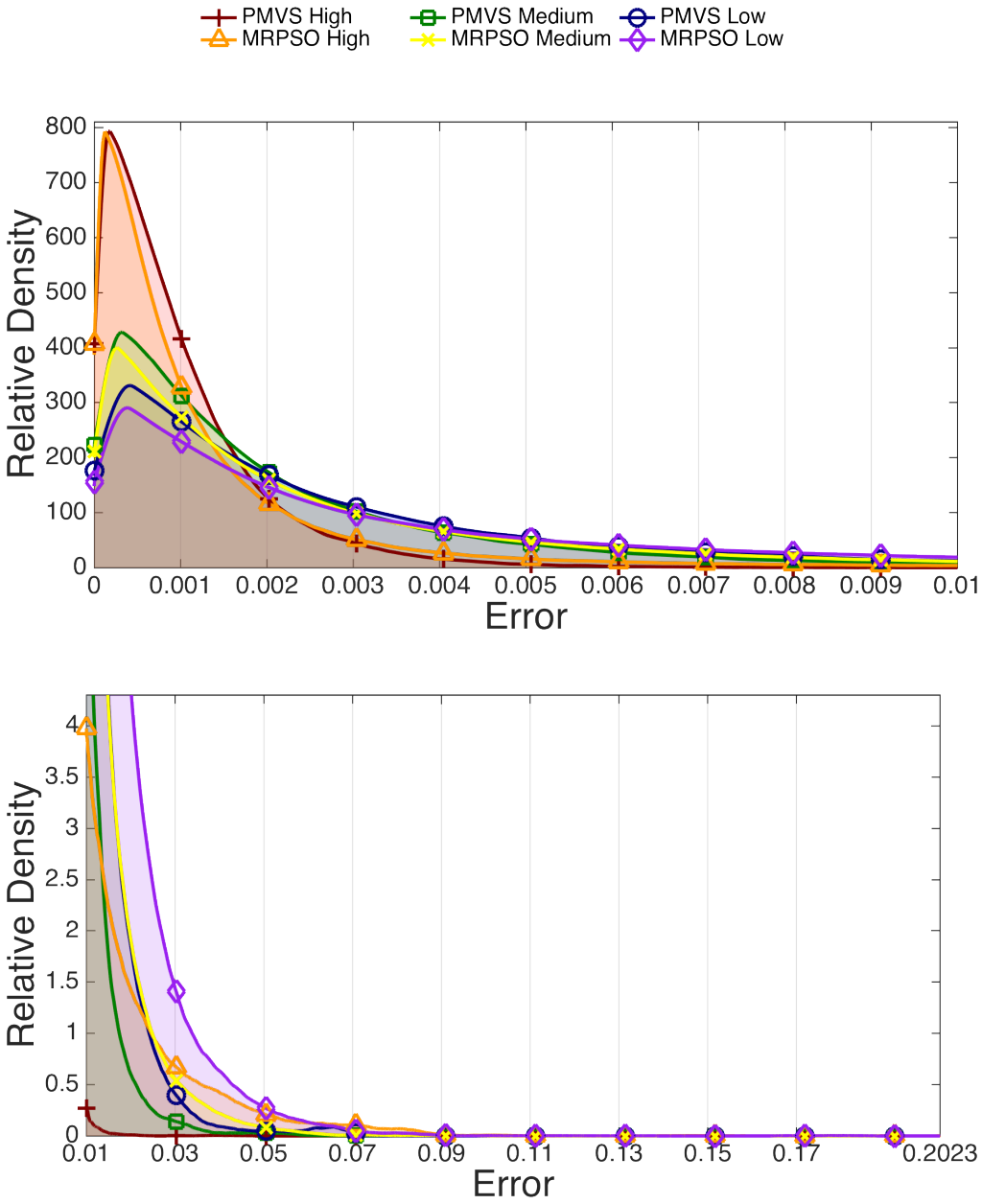


Figure 6.20: Probability distributions for the Buddha datasets
Top: Normalised probability distributions for the errors in patch refinement for running PMVS and MRPSO on each of the three different textured datasets.
Bottom: Zoomed in tail end of the distribution up to the largest error

		Mean	Median	Mode	Max
PMVS	High	0.0010	0.0007	0.0002	0.0449
	Med	0.0024	0.0014	0.0003	0.0678
	Low	0.0024	0.0019	0.0004	0.0737
PSO	High	0.0140	0.0023	0.0004	0.2327
	Medium	0.0156	0.0035	0.0007	0.2331
	Low	0.0204	0.0052	0.0010	0.2323
PSOPD	High	0.0033	0.0012	0.0002	0.2330
	Medium	0.0057	0.0020	0.0004	0.2334
	Low	0.0100	0.0027	0.0005	0.2300
MRPSO	High	0.0024	0.0008	0.0001	0.2024
	Medium	0.0032	0.0017	0.0003	0.0978
	Low	0.0049	0.0024	0.0004	0.1856

		Residual Error							
		0.0	0.005	0.01	0.02	0.03	0.05	0.1	0.2
PMVS	High	100.00	0.82	0.06	0.01	0.00	0.00	0.00	0.00
	Med	100.00	12.08	2.61	0.41	0.13	0.02	0.00	0.00
	Low	100.00	20.01	6.27	1.31	0.39	0.13	0.00	0.00
PSO	High	100.00	28.54	20.18	16.57	14.19	10.42	3.86	0.08
	Medium	100.00	40.55	26.04	18.73	15.12	10.53	3.95	0.08
	Low	100.00	52.72	36.97	26.33	21.00	14.25	5.23	0.12
PSOPD	High	100.00	6.62	3.18	2.37	1.97	1.42	0.57	0.00
	Medium	100.00	20.39	8.85	5.08	3.80	2.38	0.79	0.02
	Low	100.00	32.51	18.35	11.32	8.61	5.59	1.91	0.05
MRPSO	High	100.00	8.49	4.59	2.25	1.26	0.42	0.01	0.00
	Medium	100.00	18.61	6.38	1.63	0.58	0.08	0.00	0.00
	Low	100.00	29.64	13.62	4.25	1.72	0.36	0.01	0.00

Table 6.2: Statistics for the Buddha datasets
Top: Different averages of the reconstruction error on each dataset.
Bottom: The inverse cumulative frequency of errors as a percentage of entire patch cloud

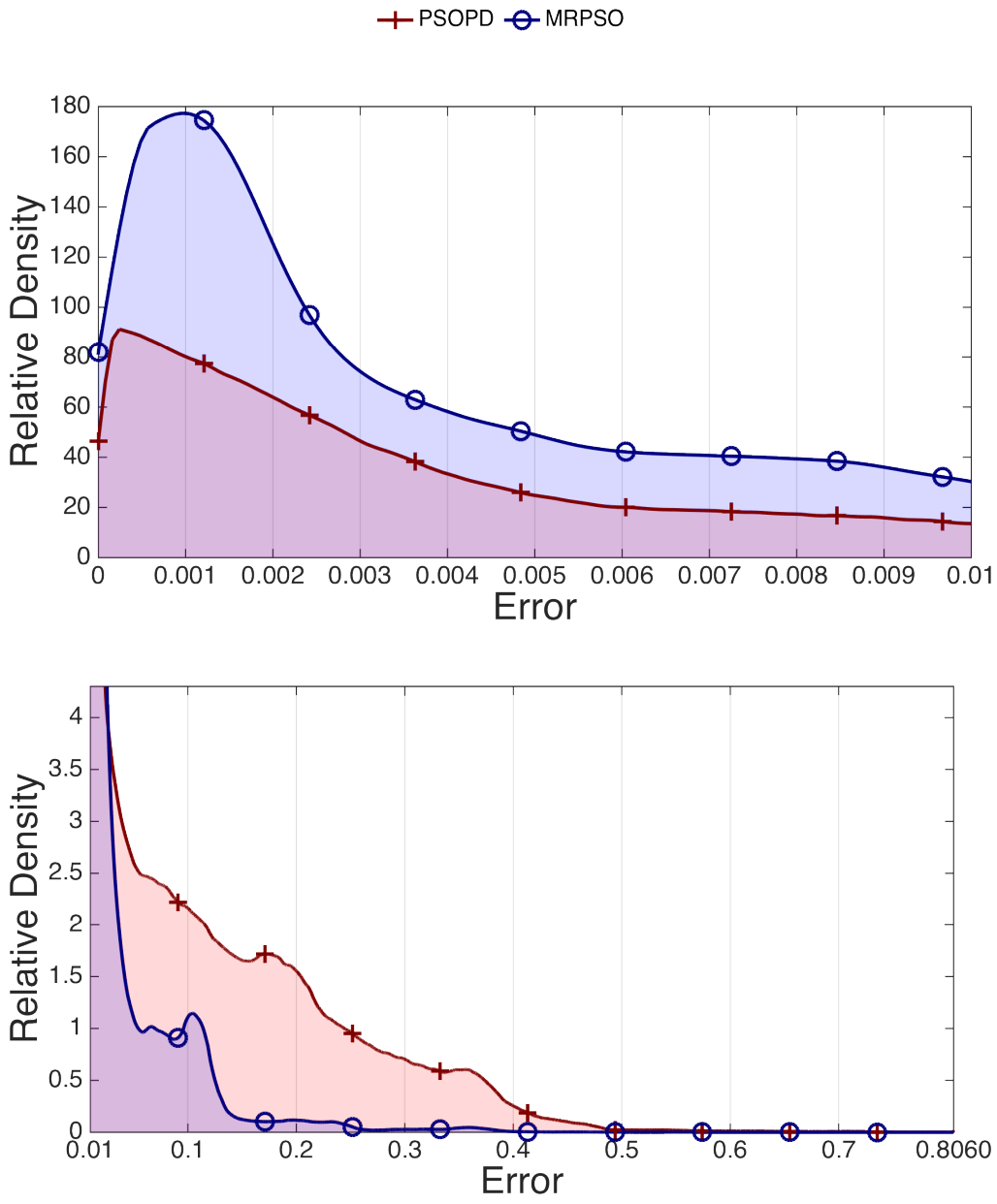


Figure 6.21: Probability distributions for the LynxCAD dataset
Top: Normalised probability distributions for the errors in patch refinement when running PSOPD and MRPSO on the dataset.
Bottom: Zoomed in tail end of the distribution up to the largest error

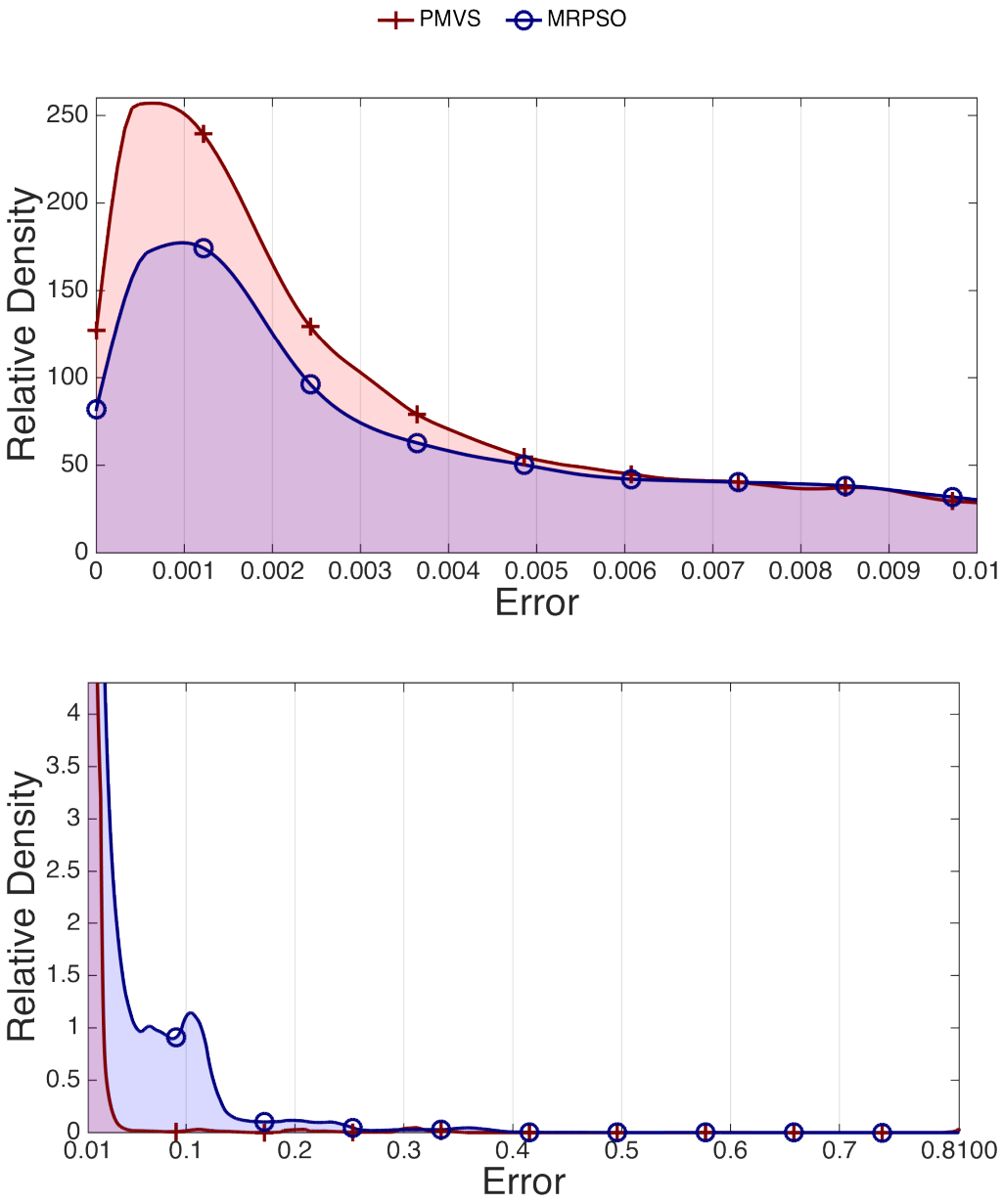


Figure 6.22: Probability distributions for the LynxCAD dataset
Top: Normalised probability distributions for the errors in patch refinement when running PMVS and MRPSO on the dataset.
Bottom: Zoomed in tail end of the distribution up to the largest error

	Mean	Median	Mode	Max
PMVS	0.0046	0.0023	0.0007	0.8100
PSO	0.1007	0.0592	0.0004	0.8060
PSOPD	0.0897	0.0365	0.0002	0.8060
MRPSO	0.0165	0.0044	0.0010	0.6535

	Residual Error							
	0.0	0.005	0.01	0.02	0.03	0.1	0.3	0.5
PMVS	100.00	27.72	7.80	0.96	0.53	0.38	0.16	0.01
PSO	100.00	79.58	71.90	64.90	60.24	38.95	7.42	0.30
PSOPD	100.00	71.82	62.84	56.21	52.31	34.60	6.81	0.30
MRPSO	100.00	47.19	27.57	16.18	12.49	4.52	0.32	0.00

Table 6.3: Statistics for the LynxCAD datasets
Top: Different averages of the reconstruction error on each dataset.
Bottom: The inverse cumulative frequency of errors as a percentage of entire patch cloud

6.2.3 Speed

		PMVS Run Time (sec)	PSO Run Time (sec)	PSOPD Run Time (sec)	MRPSO Run Time (sec)	Total Patches inside MRPSO Cloud
Monkey	High	280	883	909	3651	1191841
	Med	298	1008	1107	3787	1180802
	Low	257	1013	1099	3925	997981
Buddha	High	221	732	792	2723	910941
	Medium	284	879	945	2657	875061
	Low	356	879	941	2688	801188
LynxCAD		1323	4091	4508	20444	3473140
Dino		731	971	1037	3244	998697
Temple		390	578	615	1886	600668
LynxReal		1670	2550	2796	10017	3473140

Table 6.4: Refinement times for PMVS, PSO, PSOPD and MRPSO

With the introduction of a multi-resolution framework into the refinement procedure, the work required to produce the final patch cloud has been significantly increased. Although MRPSO is computationally more demanding than its PSOPD counterpart, its ability to properly refine patches inside textureless regions is a distinct advantage. Table 6.4 displays the run times for reconstructing an entire patch cloud.

Our MRPSO algorithm does not add any further computation to the refinement of a single patch, thus its run times are scaled by approximately 3 or 4 times compared to PSOPD. This represents the fact that our multi-resolution results are created by propagating the results up from the third resolution (fourth for the LynxCAD and LynxReal datasets). As well as this, when refining the highest resolution, the surface model from the level below is more accurate than the visual hull model used in PSOPD. This results in more patches at the highest resolution passing the viewing angle test, thus an increase in the amount of computation required. This is seen by comparing the number of patches created in our MRPSO patch clouds, Table 6.4, to that of PSOPD's patch clouds, Table 5.4.

Although the computation times in the table above are high, we do not see this as a limitation to our approach. As mentioned previously, Section 4.3.3, running this algorithm on more specialised hardware, whilst refining

multiple patches at once, will no doubt massively improve these speeds. Alternatively, a simple approach to reducing the speed is to skip every other resolution, resulting in a net reduction of the total number of patches that require refinement.

6.3 Discussion

6.3.1 Technique Synopsis

Chapters 4 and 5 developed a novel patch based 3D reconstruction framework that does not rely on feature based matching or expansion which is utilized in standard patch techniques; thus making it possible to recreate scenes for datasets with low texture or a small number of input images. Whilst demonstrating compelling results, the success of PSO in patch refinement was largely controlled by the texture of the underlying surface. Refinement in high textured regions was reliable and highly accurate, whereas, in low textured circumstances patch refinement was susceptible to failure.

The objectives of this chapter were to further reduce the amount of noise by improving the texture over a patch. Therefore, meaning refinement is more likely to succeed in correctly extracting the surface parameters. Traditionally, to improve the texture over the patch, its size is increased to include more surface over its extent. As we have studied, this inherently results in two issues: firstly an increase in the number of samples required over the patch's surface, and also the stronger the enforcement of local planarity. These factors both increase computation time and surface smoothness, removing high frequency surface details.

To avoid these shortcomings, this chapter proposed the use of multi-resolution analysis to achieve a similar effect to increasing the patch size. Although multi-resolution analysis also increases the computation time, our propagation technique should allow for better acquisition of high frequency surface details in textureless areas. This cannot be achieved whilst using an increased patch size alone. By refining patches at a lower resolution the robustness of patch refinement can be increased. The assumption being that large textureless regions should be compressed into smaller areas. The outcome of refining patches at the lowest resolution is then used to restrict the search space of the patches in the next level, propagating up results with increasing confidence.

In areas where patch refinement fails, we introduced a per-view filtering system to remove incorrect patches. We chose to use a per-view filtering system, as this avoids some of the issues with typical depth map techniques. Traditionally, filtering a point cloud relies on redundancy throughout the image set in order to isolate and remove outliers. When the input image set is sparse, the levels of redundancy are highly reduced, which can result in the removal of many correct patches. By observing the properties of the noise in patch refinement, we were able to define a per-view filtering technique whereby image redundancy was not necessary. Noise in refinement was characterised by a break in the surface smoothness assumption. Therefore, patches whose depth difference relative to its surrounding patches was large were removed as outliers.

6.3.2 Results and Limitations

Whereas refinement of a textureless patch along the entire viewing ray proved to be very noisy in textureless situations; by restricting it's search space to only a very small section along the ray, results were substantially improved. Our MRPSO framework showed large improvements in density when compared to PMVS, but also increased accuracy when compared to PSOPD. Patches that were previously noisy in PSOPD, are now correctly refined within their restricted search space for MRPSO. Therefore, our MRPSO framework has addressed the research question as to whether it is possible to create dense reconstruction of low textured datasets, even when the number of input views is low.

Our filtering methodology proved to be effective at removing noise that occurred in high resolutions refinement, but its effectiveness was limited in terms of removing low-resolution errors. When patch refinement fails at the base level and propagates up incorrect restriction values, this creates many incorrectly refined patches within a similar depth range, which our per-view filtering method cannot remove. Also high frequency noise induced over highly detailed surfaces when combining the patch clouds, could not be removed by any per-view filtering method.

Through experimentation we have found out that although our framework performs incredibly well on certain datasets, such as the Monkey, Buddha and Dino reconstructions, it is still limited in its application on others. We chose to reduce the image resolution with the assumption

that patch refinement at a lower resolution would be more robust, in the majority of cases this is true. However, in certain cases, such as the low textured lid of the Lynx Can or the Temple dataset, patch refinement at a lower resolution was not stable. On closer inspection this was caused by the blurring introduced by moving to too lower resolution. In the case of the Lynx Can, even when dropping the resolution, the large textureless regions would still contain no texture. Interestingly, for the Temple dataset, although at higher resolutions it appeared to be very well textured, when lowing the resolution this texture was quickly blurred out and the Temple would become textureless. Therefore, we have isolated the problems that can arise to situations where the low-resolution texture assumption is violated. This will occur for datasets that are incredibly low textured, or when the only textural information is held within high frequency variations.

6.3.3 Improvements

By incorporating a model of the surface texture into our multi-resolution scheme, our technique could be extended to cope with objects that contain only high frequency texture. Currently, our framework drops the resolution and creates a patch for every pixel regardless of the underlying texture. If this was accounted for in a texture model, a further adapted approach could be taken to only refine a patch once the resolution has reached a level such that there is sufficient texture to do so. This would stop incorrect patch refinement at the base level propagating up incorrect results thus allowing perfect reconstruction for datasets such as the Temple, and the top sections of the LynxCAD model.

Unfortunately, it seems unlikely that our approach can be improved in order to extend its application to highly textureless datasets. The reason for this being that when there is no texture it is not possible to match features allowing for triangulation. To handle these situations, different techniques would need to be used. For example, using Structured light rather than shape from stereo, where active light patterns can be projected onto the scene or a light source is actively moved whilst the object is kept constant. These methods essentially texture the object in a way that reflects its shape. Alternatively, laser range scanners, which capture depth automatically could be used. There is also potential to feed this algorithm with prior knowledge, allowing the use of model fitting to find the

optimal surface. Perhaps most intriguing is the potential to incorporate technological advances, in terms of Plenoptic cameras, which capture a Light Field rather than still image, [118]. By integrating a Light Field into the photo-detail-consistency metric, textureless patches could potentially be dealt with more robustly.

As well as improving the base level matching scheme through the addition of a texture model, another possibility is to adapt the filtering strategy used. We know that when base level matching fails, the properties of the noise are such that they should all be clustered. Therefore, there is scope to adapt the filtering function to detect and remove these clusters. However, detecting clusters of incorrect refinement is likely to require checking other views and thus require image redundancy.

6.4 Summary

This chapter has described our multi-resolution framework for creating high accuracy patch clouds, improving performance in low textured environments. It has both reduced noisy refinement and experimented with a per-view filtering system. The bullet points below highlight the key concepts of the following approach and the limitations identified through experimentation:

- Large improvements to the accuracy of the patch cloud, specifically in low textured circumstance, can be achieved using a multi-resolution matching approach, propagating a restriction factor that reduces the size of the search space. This dramatically reduces noise and improves accuracy and density.
- When the assumption that moving to lower resolutions can induce texture is satisfied, per view filtering is sufficient to remove all but the highest frequency noise, which we chose to leave due to difficulties in defining noise to high frequency surface variations.
- In situations where lower resolutions do not induce texture, some of the lower resolution patches can fail refinement and propagate up bad results, which per-view filtering cannot remove, hence leaving some noise in the final patch cloud.

7

Conclusions and Future Work

3D reconstruction is a massively beneficial tool for many areas of computer vision. Because of these plentiful benefits, MVS has received much academic and industrial interest. This thesis developed a multi-resolution patch based reconstruction system; highlighting key factors and limitations which can be improved within the MVS community.

Previous patch based systems, although achieving high speed and accuracy, have been limited in their ability to reconstruct textureless regions; this makes them prone to producing patch clouds with large holes and missing high and low frequency surface details. Reliance on gradient descent and the need for feature matching and expansion

caused patch based techniques difficulties in extending to low textured scenes or datasets with a small number of input views. Through the introduction of PSO and our multi-resolution framework, we were able to define an updated reconstruction system that was not limited by the need for such procedures. Our multi-resolution approach shows significant improvements in densely reconstructing all areas of the object that led to an increase in the amount of computation required. Therefore, a scalable GPU PSO implementation was adopted, to take advantage of today's modern hardware. This thesis took a GPU PSO algorithm and specifically adapted it into the MVS domain, specially designing the evaluation kernel to suit the sampling requirements of the photo-consistency function; the intentions being to move towards a general-purpose framework applicable in the majority of academic and industrial situations.

This chapter begins in Section 7.1, by formalising the issues faced and improved upon, summarising the findings presented throughout this thesis and its relation to the overall context. The significance and contributions of this work relating to the aims outlined in Section 1.3 are then given in Section 7.2. To form a complete and fair assessment of our approach its limitations are given in Section 7.3. Finally, this thesis ends with a proposal of future work in Section 7.4.

7.1 Problems in Multi-View Stereo

Recovering depth from 2D images has many potential advantages, however, recreating this third dimension is an ill-posed problem; where many possible depths can reflect the true underlying surface. Deciding between the many possible solutions for valid depth values is the problem of 3D reconstruction. Due to the ill-posed nature of this problem there must always be some compromise in terms of finding a suitable solution.

Though many MVS algorithms exist they are limited in their approach regarding run times, memory usage and accuracy. In particular their scalability in terms of the datasets in which it is possible to reconstruct also varies wildly depending on the particulars of each algorithm. Top performing approaches are capable of high accuracy over well-textured regions with large input datasets. However, as either the texture drops or the view set becomes sparse, such techniques performance degenerates

rapidly. Given the plethora of different techniques there still does not seem to be a solution that is robust and viable enough for general usage. One purpose of this thesis has been to investigate shortcomings in state-of-the-art reconstruction systems and the issues surrounding MVS problems, identifying their limitations and key areas requiring improvement.

As the literature review, in Chapter 2, highlighted, low texture or sparse input sets are inherently difficult for MVS to solve, attributed to the reduction in feature matching capabilities. In all MVS cases, recovering depth from 2D images relies on the ability to match and triangulate feature points or surface patches throughout the image set. As the texture over the surface of an object is reduced, or the size of the input set decreases, the number of these feature points also diminishes, increasing the challenge for the reconstruction algorithm.

High fidelity energy minimisation techniques are capable of achieving high accuracy at the cost of significantly longer run times and large memory requirements. Most notably, the accuracy of the reconstruction is limited by the underlying volume resolution over which energy minimisation takes place. In textureless regions large amounts of over smoothing occur, where the regularisation term has higher influence than the photo-consistency cost.

Patch based techniques showed their potential at overcoming the limitations of a volumetric representation, instead defining sparse patch clouds obtained robustly and efficiently through feature matching. These sparse patch clouds then undergo an expansion and filtering process to increase their density. Given their localised approach to representing the surface, they show much potential in terms of defining high frequency details in a parallel execution. However, their reliance on gradient descent requires precise initialisation. In the case of low textured objects or only a sparse input set, these methods fail to accurately define all areas of the object, creating holes in the final patch cloud.

Given the speed, accuracy and low memory requirements of a patch based approach, the intentions of this thesis were to remove the reliance on gradient descent. Providing comparison with a state-of-the-art method showed that in doing so, we improved performance and robustness to textureless regions, extending patch based techniques towards robustly and efficiently reconstructing more general scene types.

Traditionally, many research papers present only high quality results, neglecting to show cases of failure. Throughout this thesis, we have tried to define these situations and present methods to tackle such circumstances. The intentions being to reduce the considerable gap between academic reconstruction systems and those applicable to the public domain.

7.2 Contributions

The contributions of this thesis are:

- The identification and classification of limiting factors and open problems in the field of MVS, in particular the issues relating to performance in terms of low texture and sparse input sets. These issues make MVS algorithms unsuitable for general purpose or industrial use.
- The replacement of gradient descent, in patch based stereo methods, by a global PSO scheme to remove the need for feature matching and expansion. To inform future research, an extensive experimentation and full discussion of the limitations of this method is documented.
- Taking full advantage of today's modern hardware features, the PSO algorithm implemented on the GPU is adopted to form the global optimisation procedure. The evaluation stages inside the GPU PSO algorithm are specially designed, in this thesis, to complement MVS reconstruction, accounting for the sampling requirements of the photo-consistency function.
- The adaptation of the photo-consistency function to take into account aspects that cause such methods to fail, therefore enhancing its abilities, without severely degrading its computation time.
- A comprehensive assessment of multi-resolution techniques and their adaptation into a MVS setting where PSO is utilised. Improving the performance of optimisation, specifically tailored to low texture circumstances.

- The definition of a general-purpose reconstruction framework, accounting for factors such as density and robustness, which are key to its industrial application.

In Chapter 4, our preliminary patch based reconstruction framework was presented, this removed gradient descent and the need for feature matching and expansion, which is limited in low textured circumstances; instead, using PSO to optimise the position and orientation of a patch. We also defined an efficient method for visual hull extraction that did not rely on high quality silhouettes. A GPU approach to the construction of a sparse voxel octree is taken. Obtaining the octree representation of the visual hull allows for efficient initialisation of our PSO method, without requiring feature matching.

Here we experimented with this global refinement procedure to assess its viability in terms of a new feature free initialisation strategy. We compared our results to those produced by PMVS, and assessed the effectiveness of this technique. Performing PSO, through an adapted GPU based algorithm, showed that the cost of evaluating the photo-consistency function could be highly reduced, improving the run time.

We showed that using PSO with only a coarse search space initialisation was equally as effective as feature matching and expansion in well-textured areas. This showed promising results for both increasing the density of patch clouds and removing the reliance of feature matching and expansion. However, we found that in situations where the global optimum to a patch's photo-consistency function no longer aligned correctly with the surfaces position and normal, refinement produces noisy parameters.

Chapter 5, proposed the use of a novel photo-detail-consistency metric, improving the likelihood that the globally optimal solution aligns correctly with the surface to be reconstructed, thus reducing noise. This metric was created by observing cases where NCC fails, this occurs in low textured settings or when the patch has a shallow grazing angle to the image plane.

Accounting for these effects in a photo-detail-consistency metric both significantly reduced the amount of noise, whilst also improving the accuracy of patch refinement. We showed how improving the ability of a patch to adapt to its underlying surface texture could be the key to improving

the results in low textured areas; although care must be taken in order to retaining the speed and accuracy of such an approach. Due to the difficulties in predicting exactly when patch refinement would succeed or fail, noise still persists in the final reconstruction.

Chapter 6, further reduced and improve upon the effects of noise. Defining a full multi-resolution 3D reconstruction framework capable of accurately reconstructing both textured and low textured datasets within a reasonable runtime. By matching patches at a lower resolution, stability could be improved similarly to increasing the patch size, without sacrificing accuracy.

The innovative propagation of a restriction factor allowed the search space of the patch to be reduced, helping to guarantee patch refinement would succeed. Our findings showed that by correctly reducing the search space, through multi-resolution analysis, patch refinement could succeed in textureless areas. This showed significantly improved results in terms of obtaining denser more accurate patch clouds.

However, patch refinement was not successful in every case, as such a per-view filtering technique was used to remove erroneous patches. In most cases per-view filtering is sufficient. However, when lower resolution patch refinement fails, our filtering method could not remove this low frequency noise. We found that a small amount of higher frequency noise over detailed surfaces is also induced when combining patch clouds, this is a limitation to per-view filter methods, without using image redundancy they cannot account for this type of error.

Though succeeding in accurately reconstructing low textured objects, we also defined the limitations of patch-based approaches. In extremely low textured situations no feature or patch matching can succeed at either high or low resolutions, representing the fact that patch based MVS methods are not suitable for these cases.

7.3 Limitations

Given the wide-ranging application of MVS throughout many aspects of academia and industry, there will always be limits and unexpected circumstances in which MVS algorithms fail. The discussion section of

Chapter 6 more thoroughly covers some of the outstanding issues that need to be resolved inside our multi-resolution framework, as well as highlighting some of the limitations of such an approach.

Our algorithm makes the assumption that lowering the resolution, should at some scale induce more texture over the surface, as large textureless regions are suppressed into single pixels, removing the textureless effect. This allows for more accurate patch refinement to occur. In particular, our current algorithm does not account for moving to too low a resolution and blurring out texture information. However, as suggested, the extension of a texture model for adaptively deciding when patch refinement should occur throughout the resolutions, should improve our frameworks performance on datasets with high frequency texture that do not respond well to decomposition.

Although our algorithm performs very well at densely reconstructing both low and high textured surfaces, the per-view filtering system is limited in the amount of high frequency noise it removes around extremely detailed surfaces. Therefore, when a system requires the acquisition of surface mesh, we suggest passing the patch cloud through a further filtering process to remove the finer scales of noise, before Poisson surface reconstruction or other model fitting approaches are used. This reflects the fact that noise can be reduced and recovered from, whereas missing patches are a bigger problem. In striving for a generalised approach, incorporating further filtering routines only serves to limit the application of our framework. This framework seeks to provide a flexible base for use in general purpose environments, enabling further customisation and post-processing if required.

The final limitation of our framework is on very low textured datasets, the multi-resolution texture assumption is violated and our systems performance will drop back to that of PMVS. Such circumstances are incredibly difficult to perform accurate reconstruction for any MVS algorithm that relies only on input images. Therefore, in such situations our algorithm will fail and we must rely on prior knowledge. This entails the use of model fitting or shape from shading approaches. Alternatively, depth values returned from a laser scanner or Kinect sensor could be used, as well as the possible integration of light fields into the photo-consistency measure that are captured by Plenoptic camera.

7.4 Future Work

This work has provided a multi-resolution, multi-view reconstruction system for generalised object recovery, bridging the gap between academic systems and robust real world solutions. Current academic systems focus on improving the performance of MVS algorithms under specific circumstances, neglecting to adequately deal with failure cases. This thesis has attempted to take such works and improve upon the generality of these techniques, heading towards a robust real world solution. With the addition of a texture model into our multi-resolution framework the system should be general enough to succeed in all feasible multi-view cases. Albeit that we have identified extremely low textured datasets as being too challenging without prior knowledge or increased data. However, that is not to say we do not see potential for future development and continued research in many aspects of the framework. In this section we discuss areas of potential future work in terms of three different categories: implementation, similarity metrics and theoretical concepts.

7.4.1 GPU Implementation and Efficiency

Memory and speed requirements are always a key factor in the success of a reconstruction system. Having an algorithm that can produce incredibly accurate results, but requires tens of thousands of hours and a super computer to calculate is not necessarily very useful. One of the overwhelming features of a patch-based approach is its parallelisable nature, which therefore motivated our use of a GPU algorithm. Our PSO based algorithm defined in Chapter 3 used this fact extensively. However, throughout the development process of this thesis, more sophisticated and efficient approaches became apparent. There is scope in its implementation to allow for more efficient and faster PSO.

In particular, we note how when evaluating the photo-consistency metric we start one work-item per input view, even though most of these view are not valid. A simple optimisation would be to reduce this workload by only starting work-items for views that can validly compute a photo-consistency measure. However, the simplicity of this idea hides the complexity of its implementation. As each particle within the swarm will be visible by a different set of views, defining a fixed work-item to view

mapping would not be possible and a second kernel invocation would be needed to calculate such mappings.

As discussed, with the rapid development of hardware increasing the number of cores on a modern GPU, running multiple optimisations at once could extend our PSO technique. To achieve peak performance, all cores of the GPU should be utilised, which can be achieved either by increasing the swarm size, or running multiple swarms in tandem.

As is the nature of technology, the OpenCL API is ever evolving. Recently the newer OpenCL standard, version 2.1 [61], has been released which allows for nested parallelism. This essentially means that one kernel running on the GPU can enqueue another. Before this was not possible, and the kernel had to notify the host device to enqueue the next kernel, resulting in a slower execution time. With the availability of nested parallelism, our entire PSO routine and reconstruction algorithm could easily be ported to run entirely on the GPU without requiring the host device to schedule each section of the reconstruction. By removing the high latency CPU scheduling, requiring only that the CPU upload the images to the GPU and invoke the first kernel, our method could be run entirely on the GPU. As discussed in Section 3.3.3, removing unnecessary host to device calls would undoubtedly achieve significant computational gains.

7.4.2 Improving the Similarity Metric

Chapters 5 and 6, consistently showed the improvement in accuracy and noise that can be obtained by improving a patch's ability to adapt to the underlying surface properties. In Chapter 5, we imposed a texture and viewing angle weighting to create a photo-detail-consistency function that was better defined. Then in Chapter 6, we restricted the search space over this function such that optimisation was more likely to succeed. Both of these advances reflect the fact that by adapting the patch and its search space to better reflect the surface properties, improved results can be obtained.

It would be an interesting area of research to explore the effects of the wavelet coefficients on the photo-detail-consistency function. Using different wavelets, the detail coefficients can reflect different surface

characteristics. Currently we use only the detail coefficients produced by a Haar wavelet transform, which represent step edges. Therefore, there is potential to further improve upon the weighting inside the photo-detail-consistency metric by searching for higher order texture consistency in the detail coefficients rather than edge responses. This could be achieved by using wavelets with a higher number of vanishing moments, such as the Morlet wavelet [101], which can search for specific texture responses under differing orientations.

Alternatively, the use of the wavelet coefficients can also introduce a second measure of a patch's correlation, its local detail structure. This can be used along side the NCC to compare the back projection of the detail structure over the patch as well as the colour consistency. In theory this should improve the results, but care would need to be taken in order to account for the rotational variance of the wavelet coefficients as the patch projects into different views. This would also double the amount of sampling required, so thorough experimentation as to its benefits would be required.

7.4.3 Theoretical Concept

One of the benefits of a patch-based approach is the local nature of their refinement procedure. This allows patches to express arbitrary surface details at the same scale as the patch size. However, this also represents their achilles heel, by separating patches into discrete surface regions, it becomes harder to enforce regularisation and filter noise.

Therefore, theory can be borrowed from energy minimisation frameworks relying on a re-projection error functional, [38, 56, 99, 143]. These surfaces evolve by projecting the current surface model into the views and generating synthetic images. These are used to measure the error between the observed and expected images. Therefore, the surface evolves based on what's already been generated and how much it matches the original images. This theory could be extended to our patch based approach to better inform our filtering and propagation of restriction values.

By measuring the global re-projection error, isolating noisy patches should be obvious, and thus filtering and regularisation could be per-

formed at every resolution. This also defines a second way to measure the confidence in our patch refinement when propagating restriction values to the next level. Currently, all patches have an equal confidence and thus are equally restricted. Using the re-projection error a better weighting of our confidence could be performed, thus allowing adaptive restriction on a per-patch basis. This would be particularly useful in textureless areas, as it would provide a definitive method to measure confidence over larger image regions.

Appendices

A

Datasets

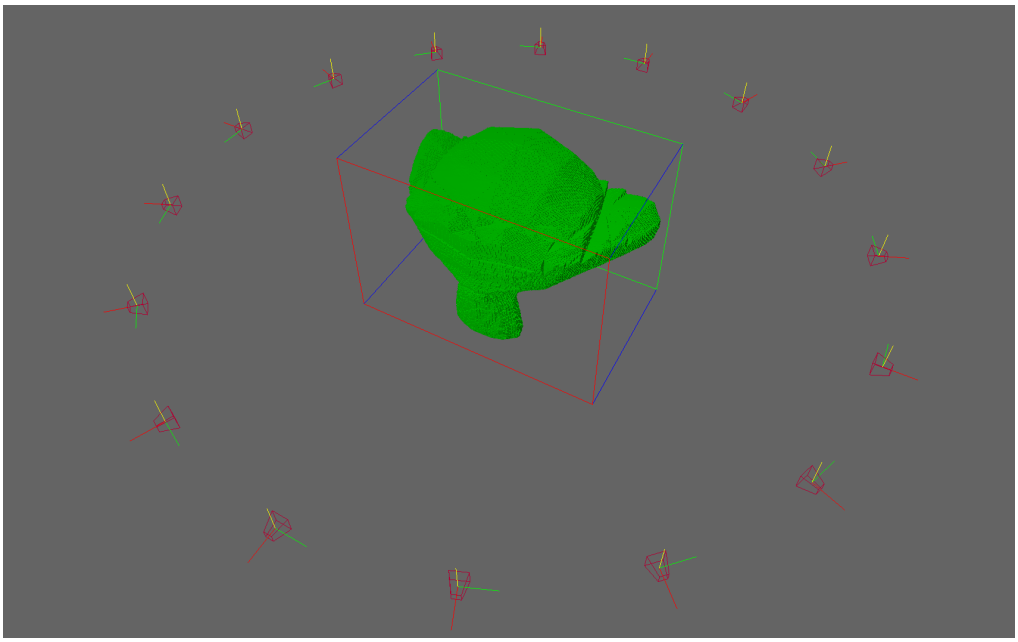


Figure A.1: Camera set up and visual hull for Monkey datasets, showing the positions at which the cameras were placed

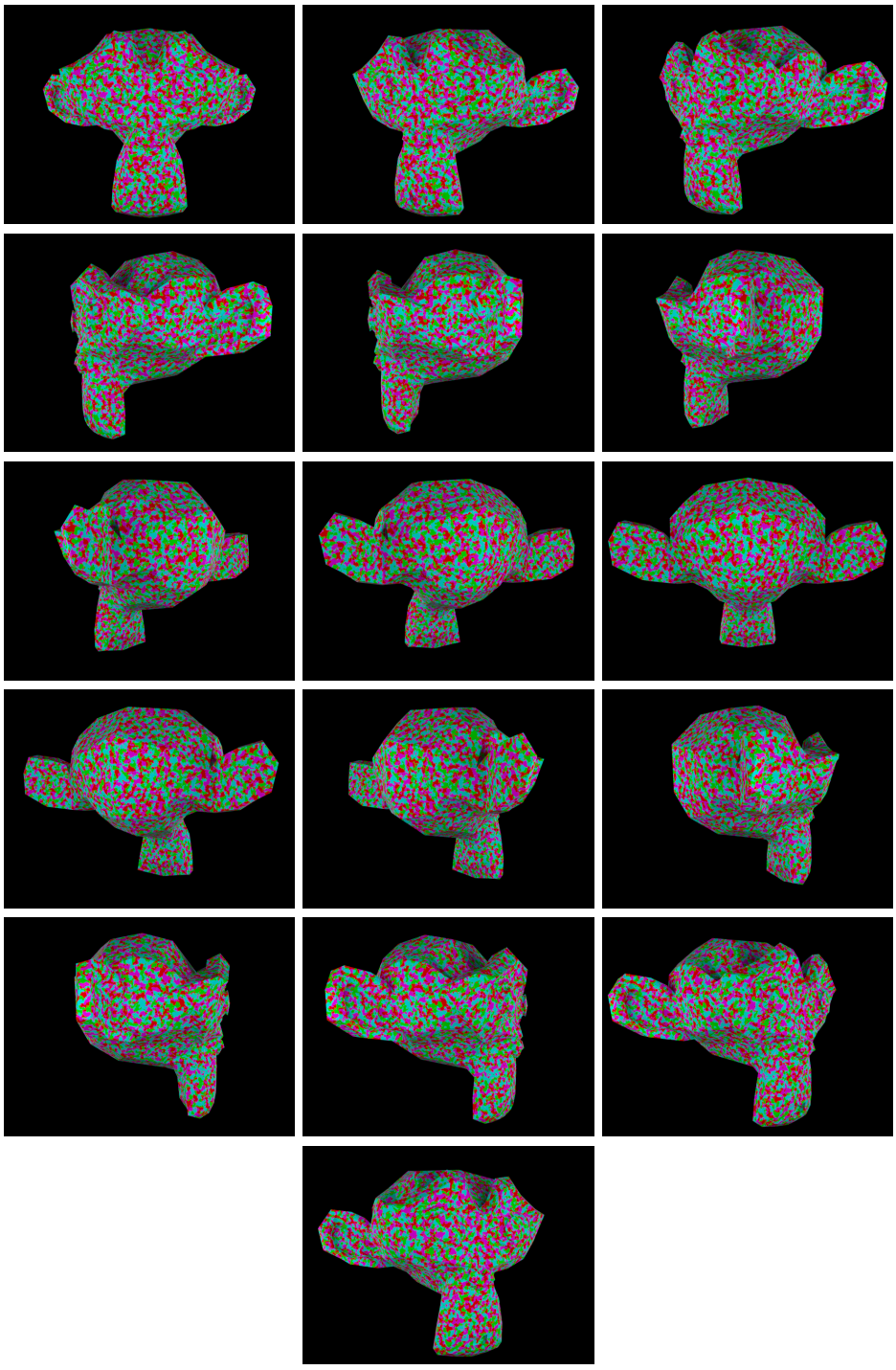


Figure A.2: The highly textured Monkey dataset (MonkeyHigh) consists of 16 images at a resolution of 640x480

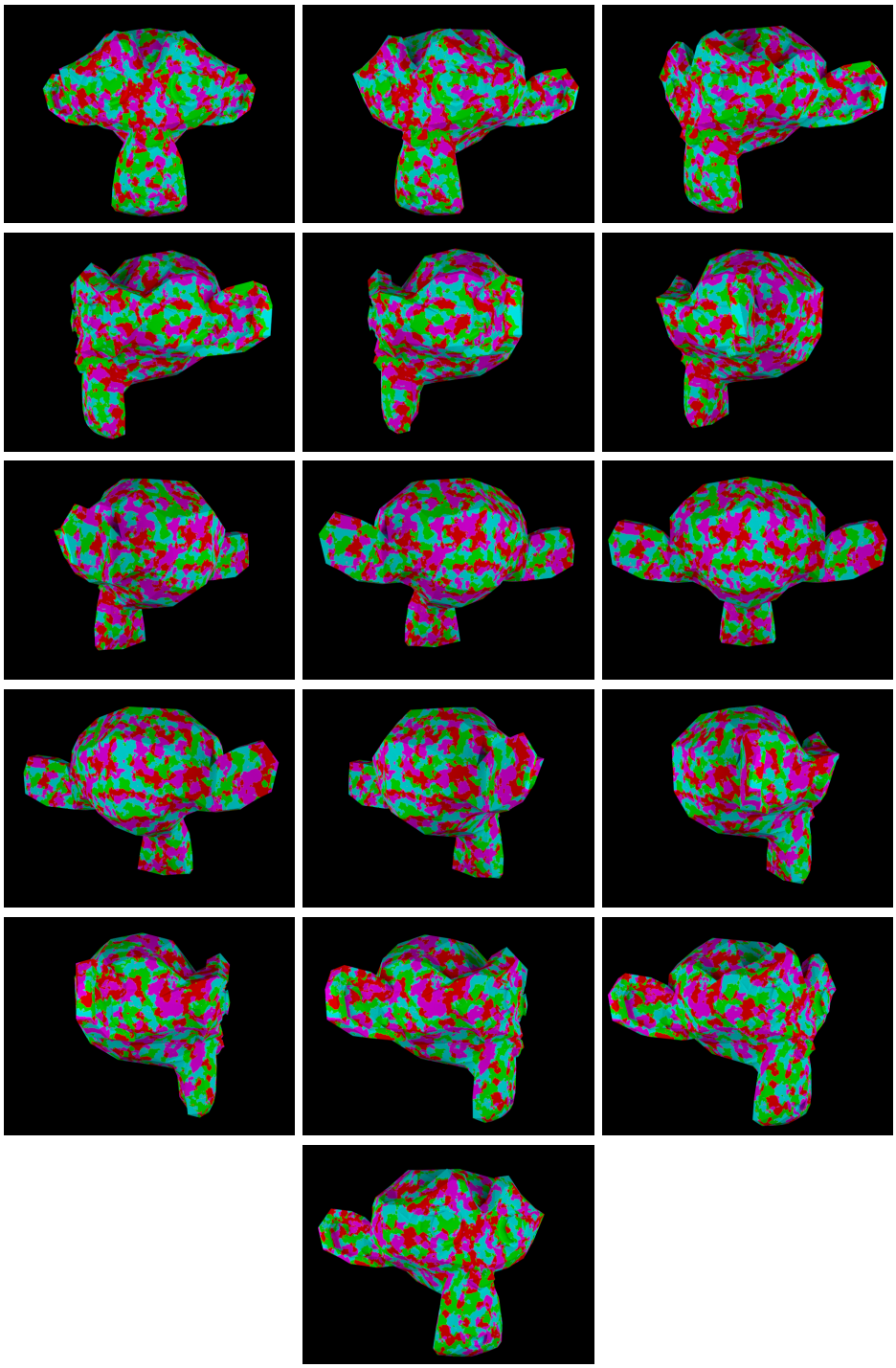


Figure A.3: The medium textured Monkey dataset (MonkeyMed) consists of 16 images at a resolution of 640x480

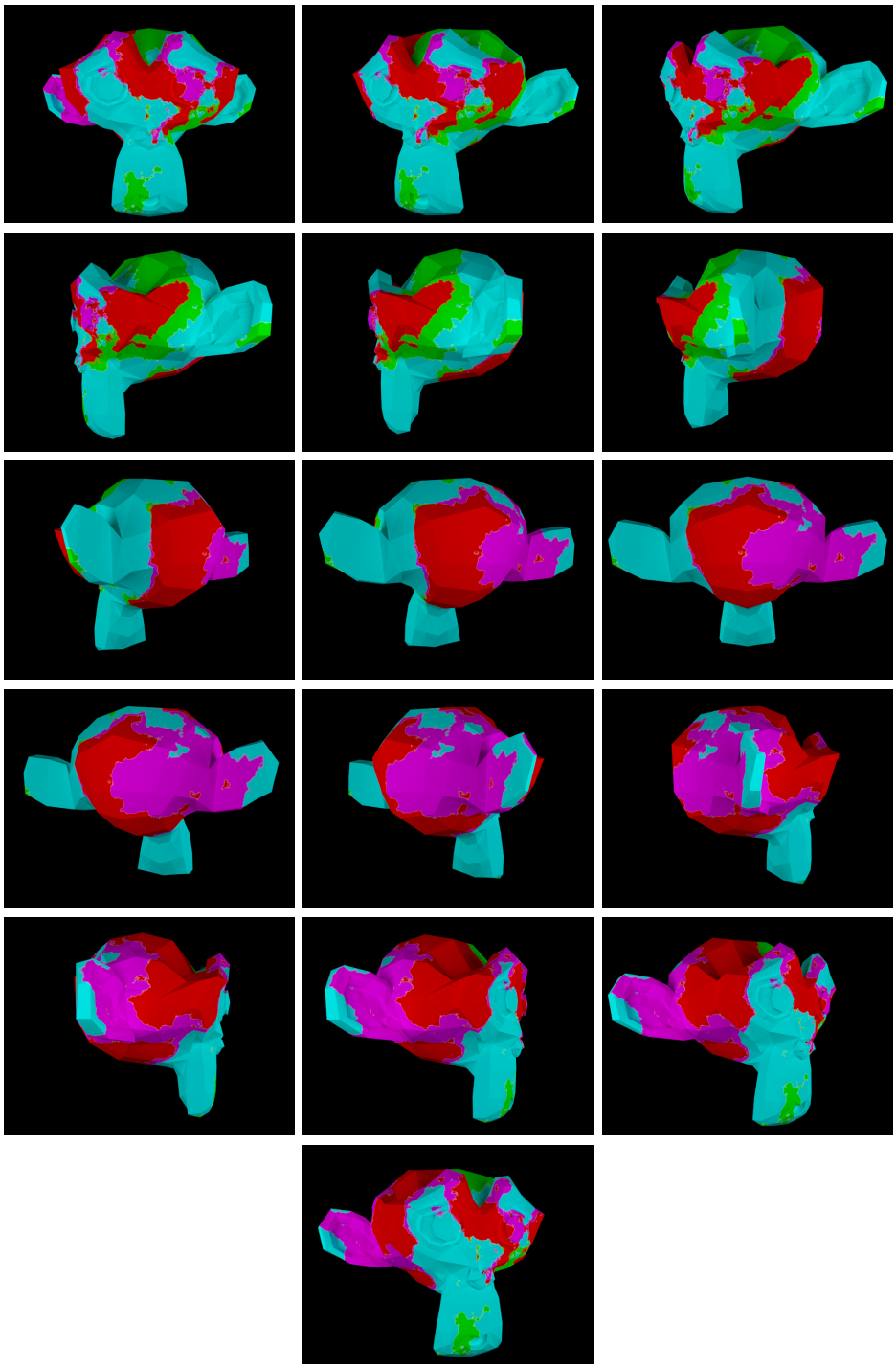


Figure A.4: The low textured Monkey dataset (MonkeyLow) consists of 16 images at a resolution of 640x480

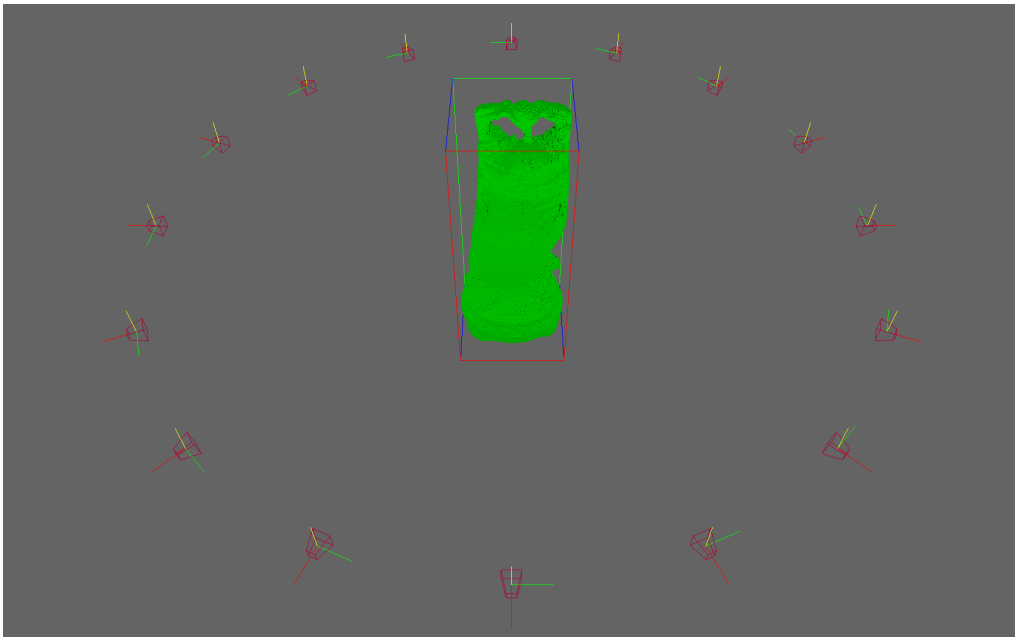


Figure A.5: Camera set up and visual hull for Buddha datasets, showing the positions at which the cameras were placed

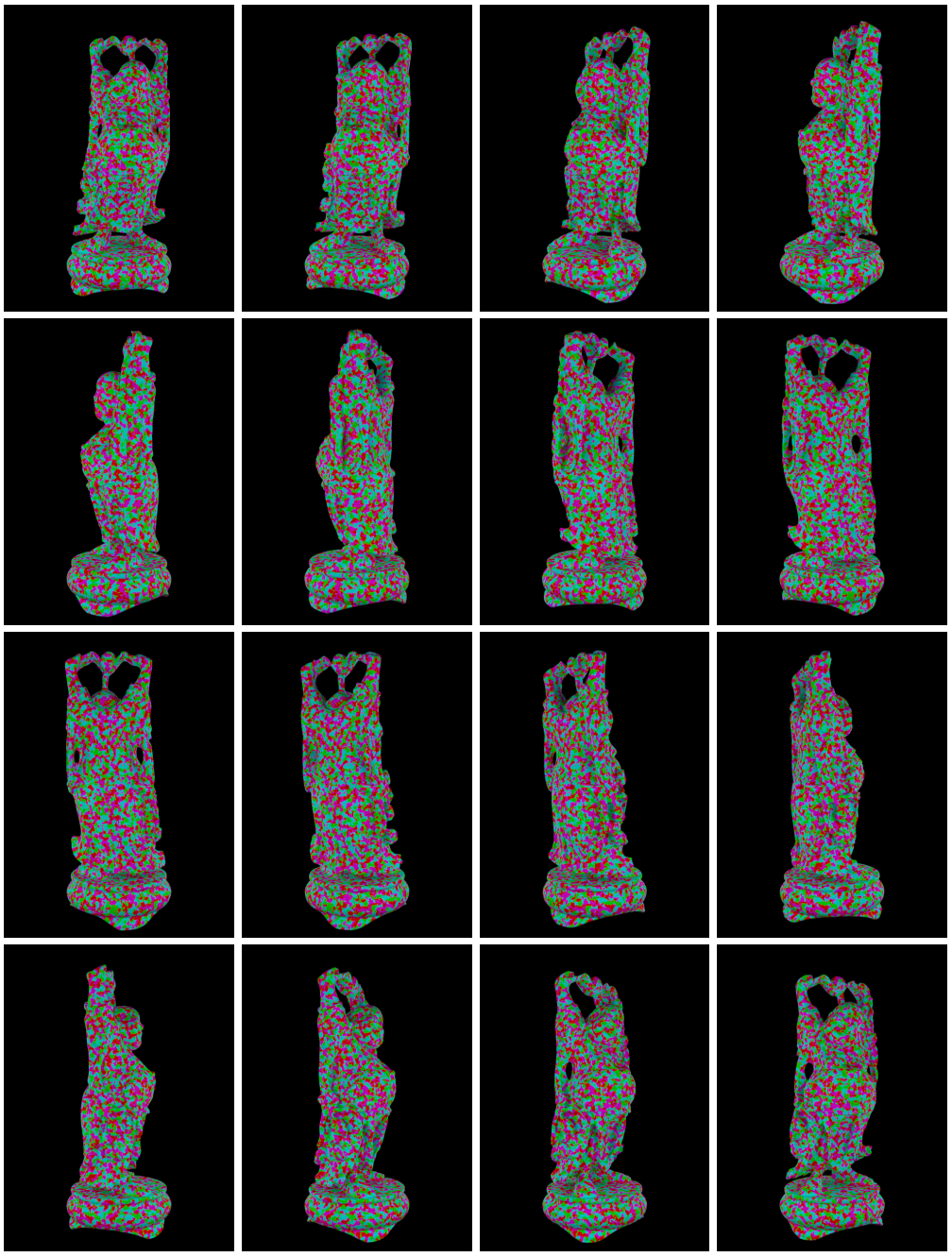


Figure A.6: The highly textured Buddha dataset (BuddhaHigh) consists of 16 images at a resolution of 480x640

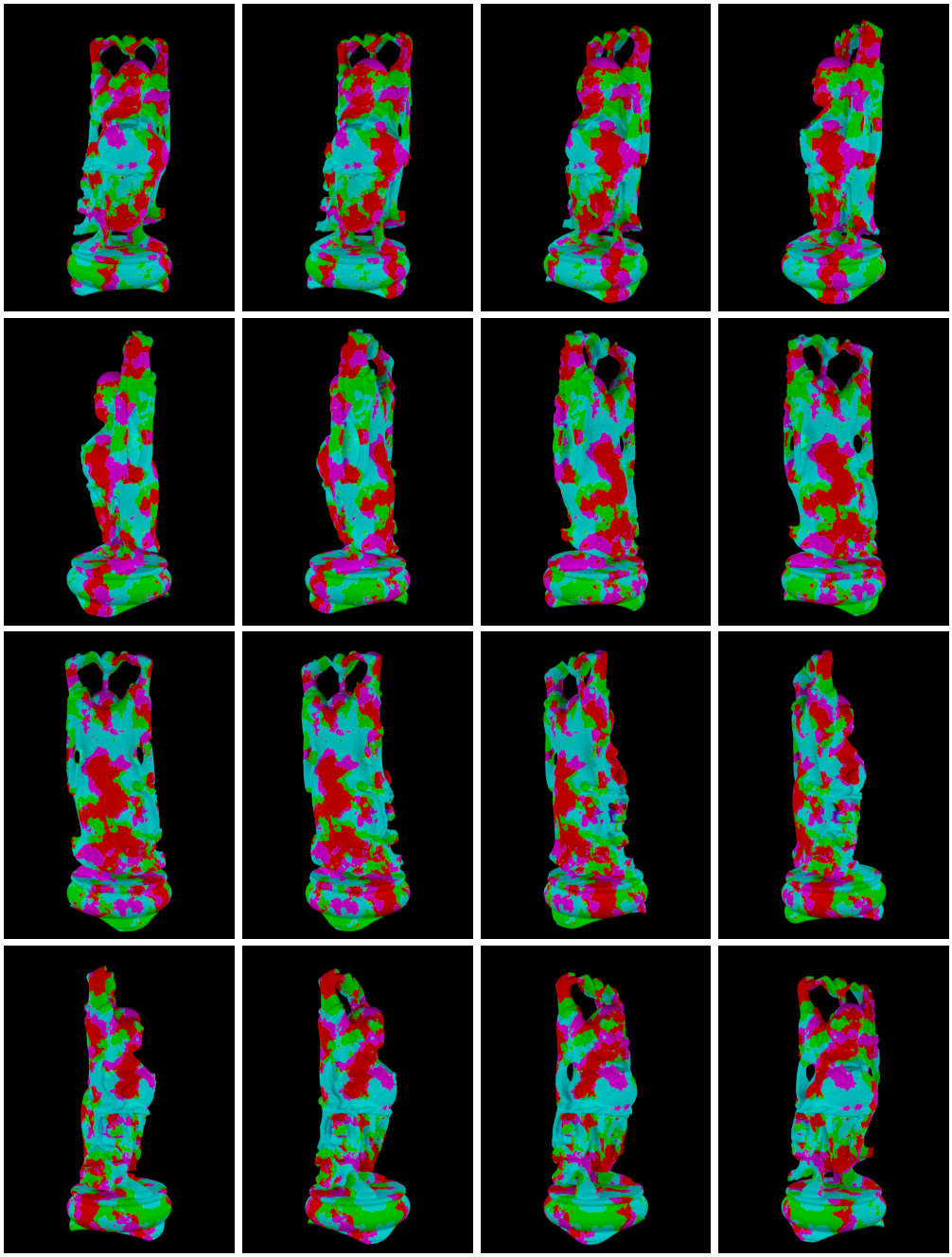


Figure A.7: The medium textured Buddha dataset (BuddhaMed) consists of 16 images at a resolution of 480x640

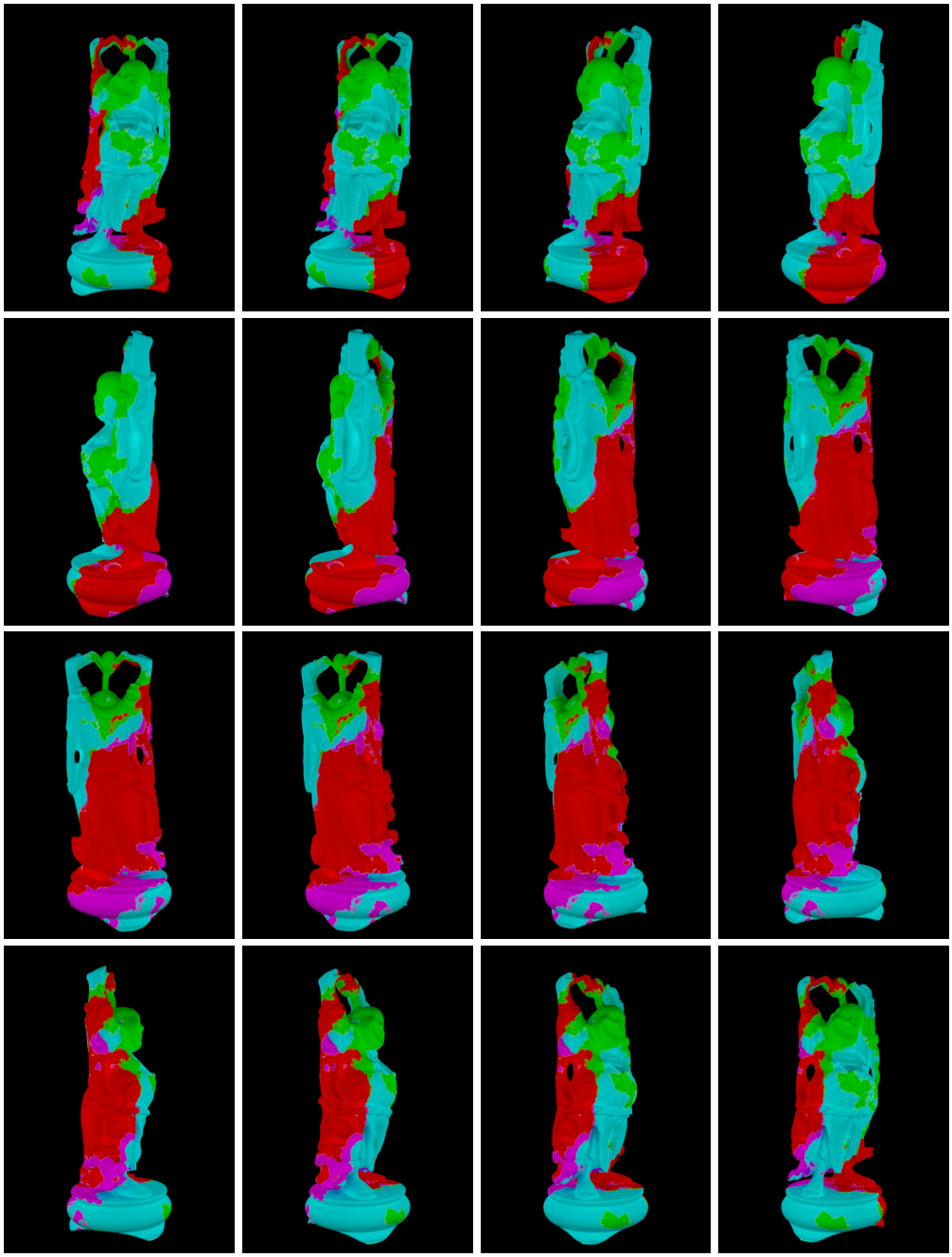


Figure A.8: The low textured Buddha dataset (BuddhaLow) consists of 16 images at a resolution of 480x640

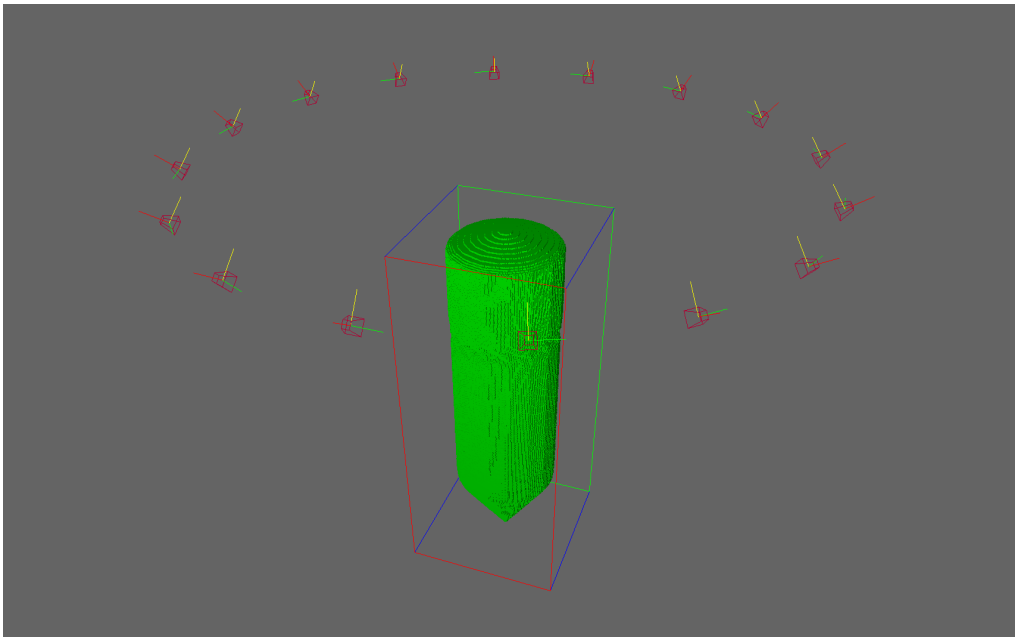


Figure A.9: Camera set up and visual hull for LynxCAD dataset, showing the positions at which the cameras were placed

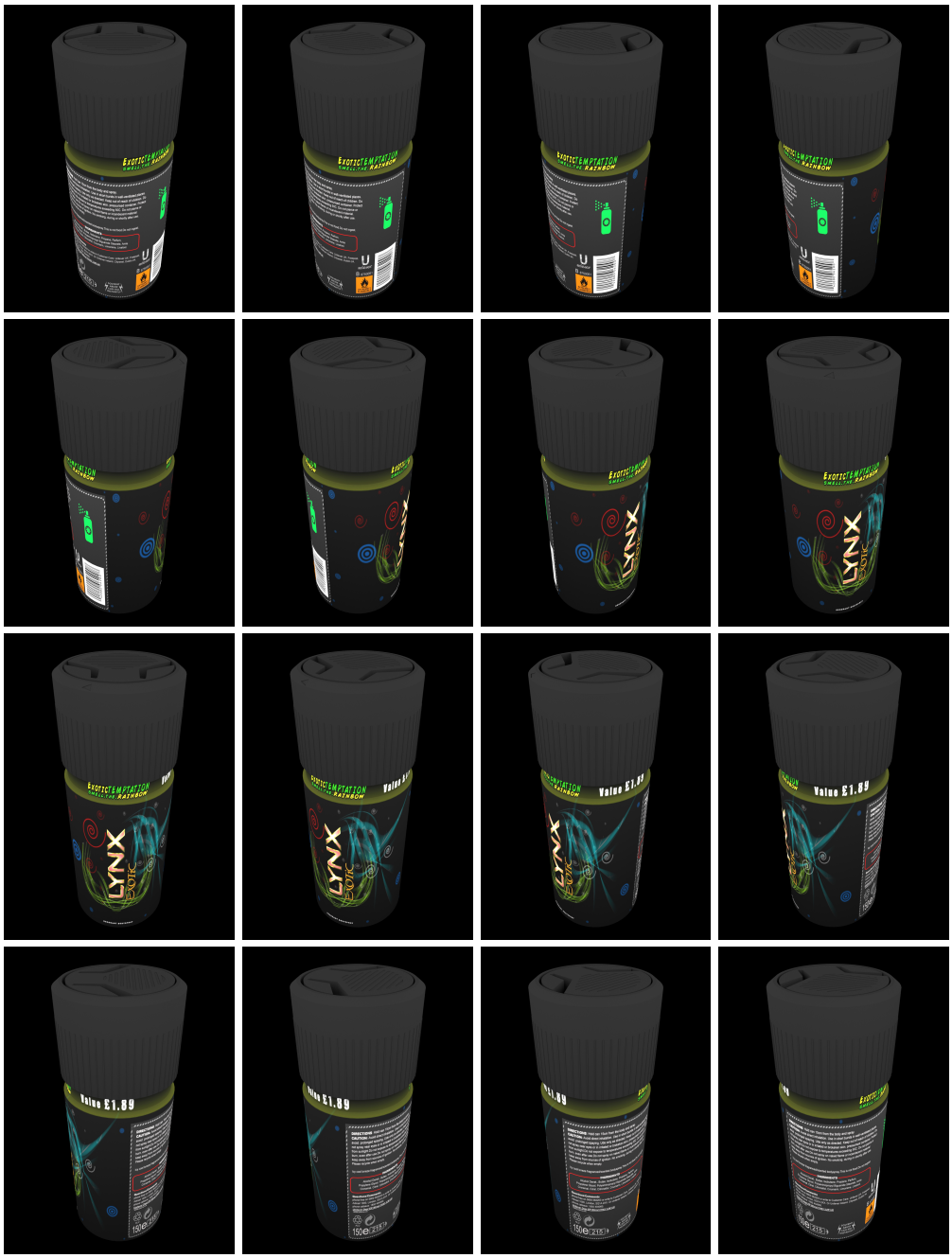


Figure A.10: The textured Lynx CAD dataset (LynxCAD) consists of 16 images at a resolution of 960x1280

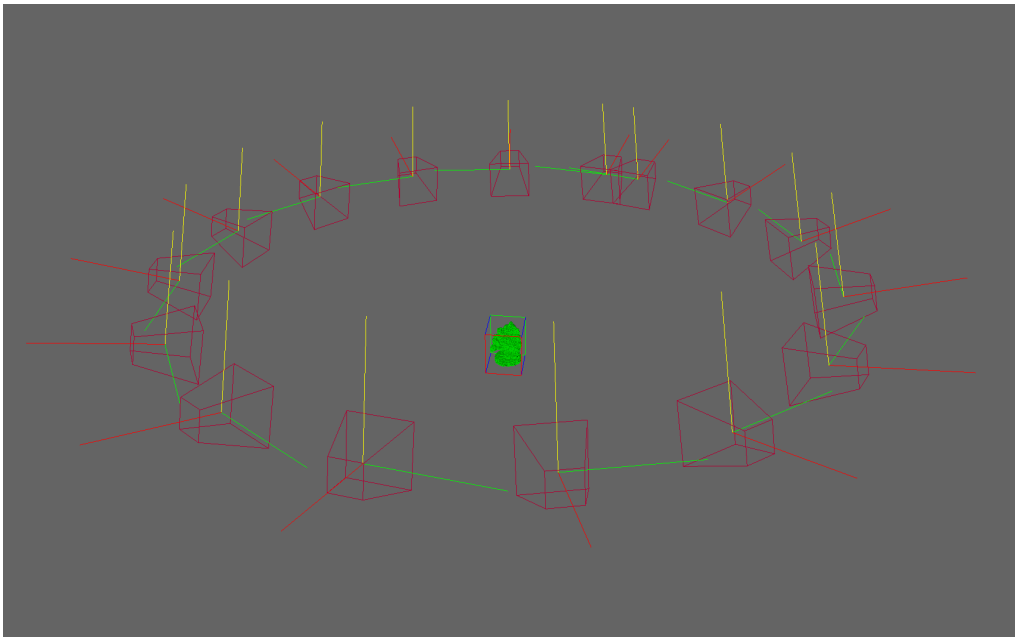


Figure A.11: Camera set up and visual hull for Dino dataset, showing the positions at which the cameras were placed



Figure A.12: The real life Middlebury Dino dataset (Dino) consists of 16 images at a resolution of 480x640

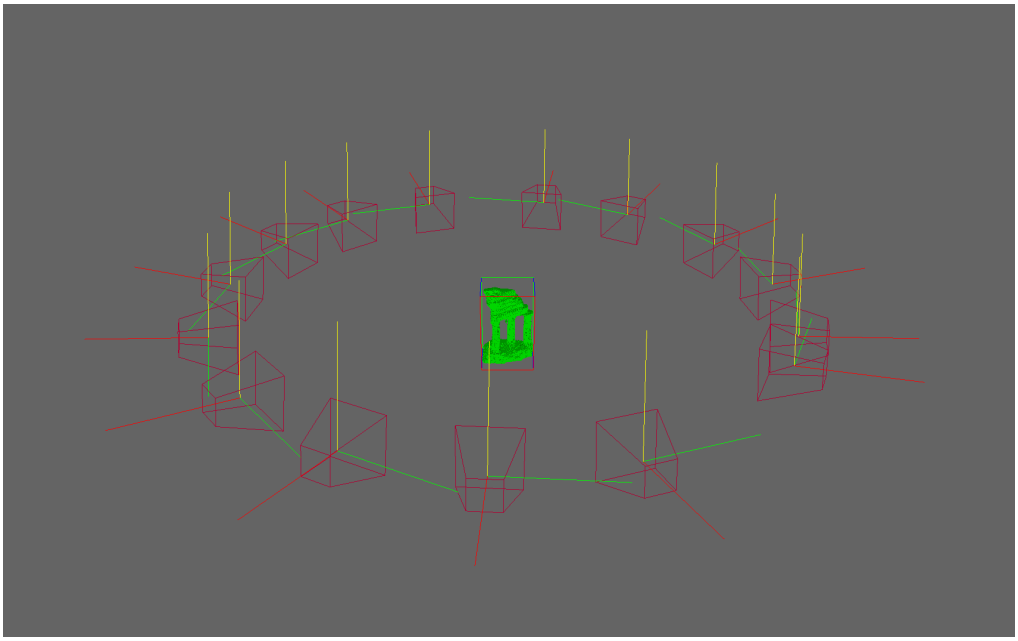


Figure A.13: Camera set up and visual hull for Temple dataset, showing the positions at which the cameras were placed



Figure A.14: The real life Middlebury Temple dataset (Temple) consists of 16 images at a resolution of 480x640

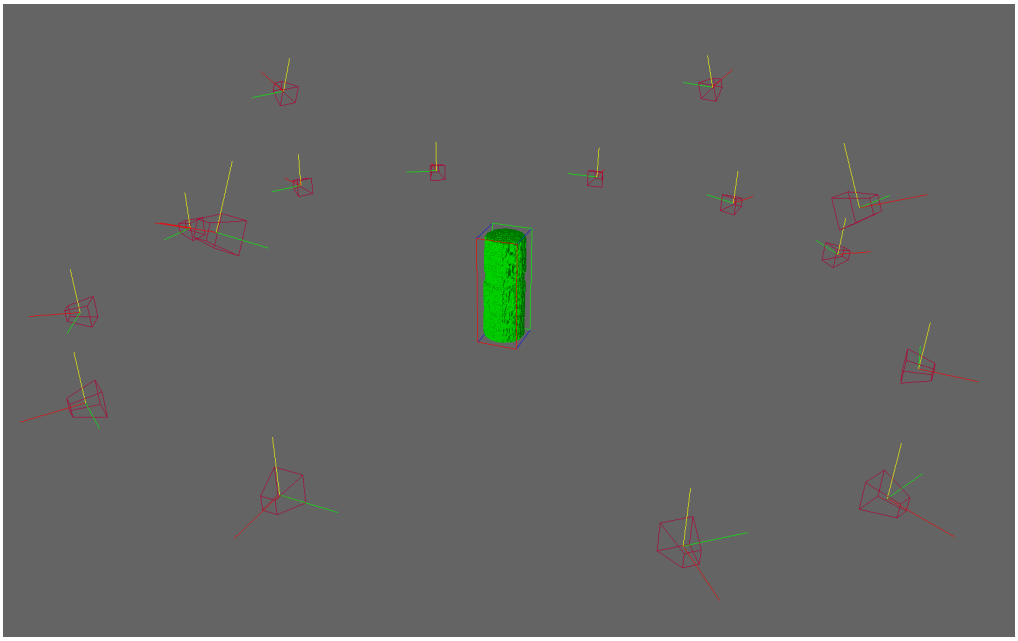


Figure A.15: Camera set up and visual hull for LynxReal dataset, showing the positions at which the cameras were placed

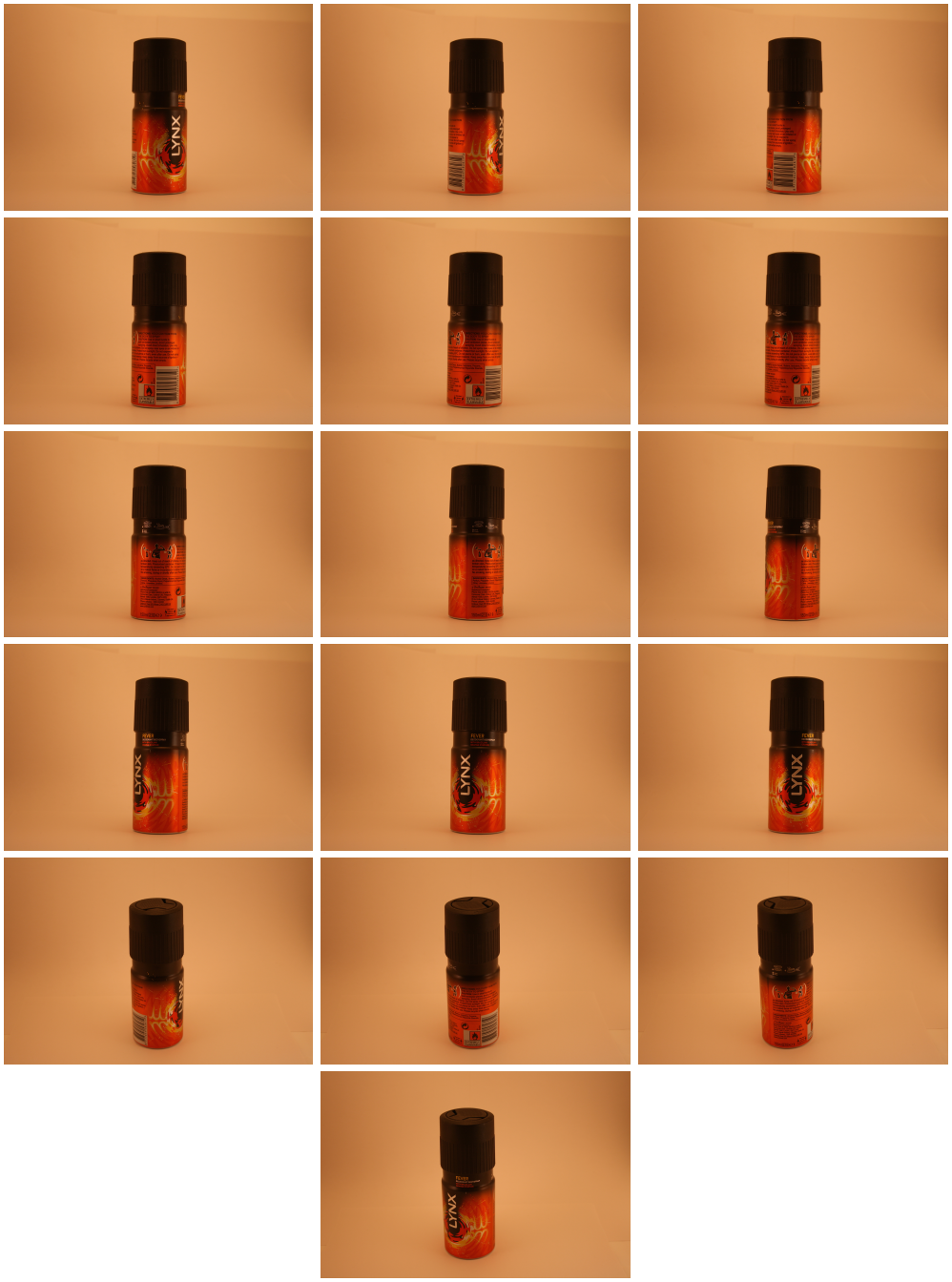


Figure A.16: The real life Lynx Can dataset (LynxReal) consists of 16 images at a resolution of 1944x1296

References

[1] 3dsom. 3dsom: 3d software object modeller. http://www.3dsom.com, 2015.

[2] Acute3D. Acute3d capturing reality. http://www.acute3d.com, 2014.

[3] P. S. Addison. *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance*. CRC Press, 2010.

[4] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):pages 105–112, 2011.

[5] Agisoft. Agisoft photoscan. http://www.agisoft.com, 2015.

[6] J. Allard, J.-S. Franco, C. Ménier, E. Boyer, and B. Raffin. The grimage platform: A mixed reality environment for interactions. In *Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on*, pages 46–46. IEEE, 2006.

[7] J. G. Allen, R. Y. Xu, and J. S. Jin. Object tracking using camshift algorithm and multiple quantized feature spaces. In *Proceedings of the Pan-Sydney area workshop on Visual information processing*, pages 3–7. Australian Computer Society, Inc., 2004.

[8] ARC3D. Arc 3d webservice. http://www.arc3d.be, 2015.

[9] Autodesk123D. Autodesk 123d catch. http://www.123dapp.com/catch, 2015.

[10] D. Avis and D. Bremner. How good are convex hull algorithms? In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 20–28. ACM, 1995.

[11] B. Baumgart. *Geometric modeling for computer vision.* PhD thesis, Stanford, CA, USA, 1974.

[12] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms i. *Communications on pure and applied mathematics*, 44(2):pages 141–183, 1991.

[13] Blender Online Community. Blender - a 3d modelling and rendering package. URL http://www.blender.org.

[14] J. Blinn. Models of light reflection for computer synthesized pictures. In *ACM SIGGRAPH Computer Graphics*, volume 11, pages 192–198. ACM, 1977.

[15] J.-Y. Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/, 2013.

[16] E. Boyer and M. Berger. 3d surface reconstruction using occluding contours. *International Journal of Computer Vision*, 22(3):pages 219–233, 1997.

[17] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 26–33. IEEE, 2003.

[18] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):pages 1222–1239, 2001.

[19] D. Bradley, T. Boubekeur, and W. Heidrich. Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[20] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 120–127. IEEE, 2007.

[21] S. Cagnoni, A. Bacchini, and L. Mussi. Opencl implementation of particle swarm optimization: a comparison between multi-core cpu

and gpu performances. In *Applications of Evolutionary Computation*, pages 406–415. Springer, 2012.

[22] N. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. *Computer Vision–ECCV 2008*, pages 766–779, 2008.

[23] N. D. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. Automatic 3d object segmentation in multiple views using volumetric graph-cuts. *Image and Vision Computing*, 28(1):pages 14–25, 2010.

[24] J. Y. Chang, K. M. Lee, and S. U. Lee. Multiview normal field integration using level set methods. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[25] V. Chari, A. Agrawal, Y. Taguchi, and S. Ramalingam. Convex bricks: A new primitive for visual hull modeling and reconstruction. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 770–777. IEEE, 2012.

[26] Z. Chen, W.-C. Chen, and P.-Y. Sung. A novel 3d dense reconstruction with high accuracy and completeness. In *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*, pages 1–4. IEEE, 2013.

[27] M. Clerc. Standard particle swarm optimisation. 2012.

[28] S. D. Cochran and G. Medioni. 3-d surface description from binocular stereo. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):pages 981–994, 1992.

[29] A. Cohen, I. Daubechies, and J.-C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 45(5):pages 485–560, 1992.

[30] R. Collins. A space-sweep approach to true multi-image matching. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 358–363. IEEE, 1996.

[31] D. Cremers and K. Kolev. Multiview stereo and silhouette consistency via convex functionals over convex domains. *Pattern Analysis*

and Machine Intelligence, IEEE Transactions on, 33(6):pages 1161–1174, 2011.

[32] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[33] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *Information Theory, IEEE Transactions on*, 36 (5):pages 961–1005, 1990.

[34] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *Journal of Fourier analysis and applications*, 4(3): pages 247–269, 1998.

[35] I. Daubechies et al. *Ten lectures on wavelets*, volume 61. SIAM, 1992.

[36] A. Delaunoy and M. Pollefeys. Photometric bundle adjustment for dense multi-view 3d modeling. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1486–1493. IEEE, 2014.

[37] A. Delaunoy and E. Prados. Gradient flows for optimizing triangular mesh-based surfaces: Applications to 3d reconstruction problems dealing with visibility. *International journal of computer vision*, 95 (2):pages 100–123, 2011.

[38] A. Delaunoy, E. Prados, P. G. I. Piracés, J.-P. Pons, and P. Sturm. Minimizing the multi-view stereo reprojection error for triangular surface meshes. In *BMVC 2008-British Machine Vision Conference*, pages 1–10. BMVA, 2008.

[39] L. Diosan and M. Oltean. What else is the evolution of pso telling us? *Journal of Artificial Evolution and Applications*, 2008:page 12, 2008.

[40] J. S. Documentation. Java random. http://docs.oracle.com/javase/7/docs/api/index.html, 2015.

[41] Y. Duan, L. Yang, H. Qin, and D. Samaras. Shape reconstruction from 3d and 2d data using pde-based deformable surfaces. *Computer Vision-ECCV 2004*, pages 238–251, 2004.

[42] C. Dyer. Volumetric scene reconstruction from multiple views. *Foundations of Image Understanding*, pages 469–489, 2001.

[43] R. C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 1, pages 84–88. IEEE, 2000.

[44] M. El-Melegy and N. Al-Ashwal. A multi-view 3d shape reconstruction system using level sets. *Int. J. Com. Dig. Sys*, 1(1):pages 31–40, 2012.

[45] A. A. Esmin, G. Lambert-Torres, and A. Z. de Souza. A hybrid particle swarm optimization applied to loss power minimization. *Power Systems, IEEE Transactions on*, 20(2):pages 859–866, 2005.

[46] O. Faugeras and R. Keriven. Variational principles, surface evolution, pde's, level set methods and the stereo problem. In *Biomedical Imaging, 2002. 5th IEEE EMBS International Summer School on*, pages 83–pp. IEEE, 2002.

[47] J. Franco and E. Boyer. Efficient polyhedral modeling from silhouettes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(3):pages 414–427, 2009.

[48] J. Franco, E. Boyer, et al. Exact polyhedral visual hulls. In *British Machine Vision Conference (BMVC'03)*, volume 1, pages 329–338, 2003.

[49] S. Fuhrmann and M. Goesele. Fusion of depth maps with multiple scales. In *ACM Transactions on Graphics (TOG)*, volume 30, page 148. ACM, 2011.

[50] S. Fukui, Y. Iwahori, H. Itoh, H. Kawanaka, and R. Woodham. Robust background subtraction for quick illumination changes. *Advances in Image and Video Technology*, pages 1244–1253, 2006.

[51] Y. Furukawa and J. Ponce. Carved visual hulls for image-based modeling. *Computer Vision–ECCV 2006*, pages 564–577, 2006.

[52] Y. Furukawa and J. Ponce. Accurate camera calibration from multi-view stereo and bundle adjustment. In *Computer Vision and Pattern*

Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008.

[53] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(8):pages 1362–1376, 2010.

[54] Y. Furukawa and J. Ponce. Patch based multi-view stereo. http://www.di.ens.fr/pmvs/, 2010.

[55] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1434–1441. IEEE, 2010.

[56] P. Gargallo, E. Prados, and P. Sturm. Minimizing the reprojection error in surface reconstruction from images. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[57] M. Goesele, B. Curless, and S. M. Seitz. Multi-view stereo revisited. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2402–2409. IEEE, 2006.

[58] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[59] B. Goldlucke, I. Ihrke, C. Linz, and M. Magnor. Weighted minimal hypersurface reconstruction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(7):pages 1194–1208, 2007.

[60] K. Group. The opencl specification 1.2, revision 19. 2012.

[61] K. Group. The opencl specification 2.1, revision 23. 2015.

[62] M. Habbecke and L. Kobbelt. Iterative multi-view plane fitting. In *Int. Fall Workshop of Vision, Modeling, and Visualization*, pages 73–80, 2006.

[63] M. Habbecke and L. Kobbelt. A surface-growing approach to multi-view stereo reconstruction. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[64] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*, volume 2. Cambridge Univ Press, 2000.

[65] R. Hassan, B. Cohanim, O. De Weck, and G. Venter. A comparison of particle swarm optimization and the genetic algorithm. In *Proceedings of the 1st AIAA multidisciplinary design optimization specialist conference*, pages 1–13, 2005.

[66] C. Hernández Esteban and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. *Computer Vision and Image Understanding*, 96(3):pages 367–392, 2004.

[67] V. H. Hiep, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1430–1437. IEEE, 2009.

[68] V. H. Hiep, P. Labatut, J.-P. Pons, and R. Keriven. High accuracy and visibility-consistent dense multiview stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(5):pages 889–901, 2012.

[69] A. Hornung and L. Kobbelt. Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 503–510. IEEE, 2006.

[70] A. Hornung and L. Kobbelt. Robust and efficient photo-consistency estimation for volumetric 3d reconstruction. *Computer Vision–ECCV 2006*, pages 179–190, 2006.

[71] T. Horprasert, D. Harwood, and L. S. Davis. A robust background subtraction and shadow detection. In *In Proceedings of the Asian Conference on Computer Vision*, 2000.

[72] A. Hosni, M. Bleyer, M. Gelautz, and C. Rhemann. Local stereo matching using geodesic support weights. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 2093–2096. IEEE, 2009.

[73] X. Hu and P. Mordohai. A quantitative evaluation of confidence measures for stereo vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):pages 2121–2133, 2012.

[74] D. P. Huttenlocher, G. Klanderman, W. J. Rucklidge, et al. Comparing images using the hausdorff distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(9):pages 850–863, 1993.

[75] Insight3D. Insignt3d: Open source image based 3d modelling software. http://insight3d.sourceforge.net, 2015.

[76] M. Jancosek, A. Shekhovtsov, and T. Pajdla. Scalable multi-view stereo. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1526–1533. IEEE, 2009.

[77] M. H. Jansen, P. Oonincx, and P. J. Oonincx. *Second generation wavelets and applications*. Springer, 2005.

[78] A. Jensen and A. la Cour-Harbo. *Ripples in mathematics: the discrete wavelet transform*. springer, 2001.

[79] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs. Large scale multi-view stereopsis evaluation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 406–413. IEEE, 2014.

[80] H. Jin, S. Soatto, and A. Yezzi. Multi-view stereo beyond lambert. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–171. IEEE, 2003.

[81] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[82] J. Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010.

[83] J. F. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm intelligence*. Morgan Kaufmann, 2001.

[84] H. Kim, Q. Hunter, M. A. Duchaineau, K. I. Joy, and N. L. Max. Gpu-friendly multi-view stereo for outdoor planar scene reconstruction. In *VISAPP (2)*, pages 255–264, 2012.

[85] M. Kimura, H. Saito, and T. Kanade. 3d voxel construction based on epipolar geometry. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, volume 3, pages 135 –139 vol.3, 1999.

[86] R. Klowsky, A. Kuijper, and M. Goesele. Modulation transfer function of patch-based stereo systems. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1386–1393. IEEE, 2012.

[87] J. Koenderink et al. What does the occluding contour tell us about solid shape. *Perception*, 13(3):pages 321–330, 1984.

[88] B.-I. Koh, A. D. George, R. T. Haftka, and B. J. Fregly. Parallel asynchronous particle swarm optimization. *International Journal for Numerical Methods in Engineering*, 67(4):pages 578–595, 2006.

[89] K. Kolev and D. Cremers. Integration of multiview stereo and silhouettes via convex functionals on convex domains. In *Computer Vision–ECCV 2008*, pages 752–765. Springer, 2008.

[90] K. Kolev, M. Klodt, T. Brox, and D. Cremers. Continuous global optimization in multiview 3d reconstruction. *International Journal of Computer Vision*, 84(1):pages 80–96, 2009.

[91] K. Kutulakos and S. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):pages 199–218, 2000.

[92] P. Labatut, R. Keriven, and J.-P. Pons. Fast level set multi-view stereo on graphics hardware. In *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pages 774–781. IEEE, 2006.

[93] A. Ladikos, S. Benhimane, and N. Navab. Efficient visual hull computation for real-time 3d reconstruction using cuda. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–8. IEEE, 2008.

[94] S. Laine and T. Karras. Efficient sparse voxel octrees. *Visualization and Computer Graphics, IEEE Transactions on*, 17(8):pages 1048–1059, 2011.

[95] A. Laurentini. The visual hull concept for silhouette-based image understanding. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(2):pages 150 –162, feb 1994.

[96] A. Laurentini. The visual hull concept for silhouette-based image understanding. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(2):pages 150 –162, feb 1994.

[97] S. Lazebnik, Y. Furukawa, and J. Ponce. Projective visual hulls. *International Journal of Computer Vision*, 74(2):pages 137–165, 2007.

[98] M. Lhuillier and L. Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(3):pages 418–433, 2005.

[99] Z. Li, K. Wang, W. Jia, H.-C. Chen, W. Zuo, D. Meng, and M. Sun. Multiview stereo and silhouette fusion via minimizing generalized reprojection error. *Image and vision computing*, 33:pages 1–14, 2015.

[100] C. Liang and K. Wong. Complex 3d shape recovery using a dual-space approach. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 878–884. IEEE, 2005.

[101] J. Lin and L. Qu. Feature extraction based on morlet wavelet and its application for mechanical fault diagnosis. *Journal of sound and vibration*, 234(1):pages 135–148, 2000.

[102] S.-H. Ling, H. H. Iu, K. Y. Chan, H.-K. Lam, B. C. Yeung, and F. H. Leung. Hybrid particle swarm optimization with wavelet mutation and its industrial applications. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 38(3):pages 743–763, 2008.

[103] Y. Liu, X. Cao, Q. Dai, and W. Xu. Continuous depth estimation for multi-view stereo. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2121–2128. IEEE, 2009.

[104] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Siggraph Computer Graphics*, volume 21, pages 163–169. ACM, 1987.

[105] N. B. M. Sainz and A. Susin. Hardware accelerated voxel carving. In *Proceedings of the 1st Iberoamerican Symposium in Computer Graphics*, pages 289–297, Portugal, 2002.

[106] R. Maini and H. Aggarwal. Study and comparison of various image edge detection techniques. *International journal of image processing (IJIP)*, 3(1):pages 1–11, 2009.

[107] S. Mallat. *A wavelet tour of signal processing*. Access Online via Elsevier, 1999.

[108] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(7):pages 674–693, 1989.

[109] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-based visual hulls. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 369–374. ACM Press/Addison-Wesley Publishing Co., 2000.

[110] W. McMillan. Polyhedral visual hulls for real-time rendering. In *Rendering Techniques 2001: Proceedings of the Eurographics Workshop in London, United Kingdom, June 25-27, 2001*, page 115. Springer Verlag Wien, 2001.

[111] Microsoft. Kinect for windows. https://www.microsoft.com/en-us/kinectforwindows/, 2015.

[112] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):pages 1615–1630, 2005.

[113] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International journal of computer vision*, 65(1-2): pages 43–72, 2005.

[114] A. Mulayim, U. Yilmaz, and V. Atalay. Silhouette-based 3-d model reconstruction from multiple images. *Systems, Man, and Cyber-*

netics, Part B: Cybernetics, IEEE Transactions on, 33(4):pages 582–591, 2003.

[115] A. Munshi, B. Gaster, T. G. Mattson, and D. Ginsburg. *OpenCL programming guide*. Pearson Education, 2011.

[116] L. Mussi, F. Daolio, and S. Cagnoni. Evaluation of parallel particle swarm optimization algorithms within the cuda™ architecture. *Information Sciences*, 181(20):pages 4642–4657, 2011.

[117] L. Mussi, Y. S. Nashed, and S. Cagnoni. Gpu-based asynchronous particle swarm optimization. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1555–1562. ACM, 2011.

[118] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR*, 2(11):pages 1–11, 2005.

[119] C. Nitschke, A. Nakazawa, and H. Takemura. Real-time space carving using graphics hardware. *IEICE TRANSACTIONS on Information and Systems*, 90(8):pages 1175–1184, 2007.

[120] H. Noborio, S. Fukuda, and S. Arimoto. Construction of the octree approximating a three-dimensional object by using multiple views. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(6):pages 769 –782, nov 1988.

[121] Nvidia. Nvidia cuda. http://www.nvidia.co.uk/object/cuda-parallel-computing-uk.html, 2015.

[122] K. Ogawa, Y. Ito, and K. Nakano. Efficient canny edge detection using a gpu. In *Networking and Computing (ICNC), 2010 First International Conference on*, pages 279–280. IEEE, 2010.

[123] OpenMVG. Openmvg: open multiple view geometry. http://imagine.enpc.fr/~moulonp/openMVG/, 2015.

[124] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. In *Computer graphics forum*, volume 26, pages 80–113. Wiley Online Library, 2007.

[125] PhotoModeler. Photomodeler: close-range photogrammetry and image-based modelling. http://www.photomodeler.com/index.html, 2015.

[126] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):pages 207–232, 2004.

[127] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, et al. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):pages 143–167, 2008.

[128] P. Pongchairerks and V. Kachitvichyanukul. Non-homogenous particle swarm optimization with multiple social structures. 2005.

[129] J. Pons, R. Keriven, O. Faugeras, and G. Hermosillo. Variational stereovision and 3d scene flow estimation with statistical similarity measures. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 597–602. IEEE, 2003.

[130] J.-P. Pons, R. Keriven, and O. Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 822–827. IEEE, 2005.

[131] M. P. Pound, A. P. French, E. H. Murchie, and T. P. Pridmore. Automated recovery of three-dimensional models of plant shoots from multiple color images. *Plant physiology*, 166(4):pages 1688–1698, 2014.

[132] R. Rodrigues and A. Fernandes. Accelerated epipolar geometry computation for 3d reconstruction using projective texturing. In *Proceedings of the 20th spring conference on Computer graphics*, pages 200–206. ACM, 2004.

[133] S. Roth. Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18(2):pages 109–144, 1982.

[134] H. Saito and T. Kanade. Shape reconstruction in projective grid space from large number of images. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 2 vol. (xxiii+637+663), 1999.

[135] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–195. IEEE, 2003.

[136] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):pages 151–173, 1999.

[137] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. Multi-view stereo evaluation web page. http://vision.middlebury.edu/mview/data/, 2006.

[138] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 519–528. IEEE, 2006.

[139] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):pages 1591–1595, 1996.

[140] S. Shen. Accurate multiple view 3d reconstruction using patch-based stereo for large-scale scenes. *Image Processing, IEEE Transactions on*, 22(5):pages 1901–1914, 2013.

[141] S. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 349–356. IEEE, 2005.

[142] N. Snavely. Bundler: Structure from motion (sfm) for unordered image collections. http://www.cs.cornell.edu/~snavely/bundler/, 2015.

[143] S. Soatto, A. Yezzi, and H. Jin. Tales of shape and radiance in multiview stereo. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 974 –981 vol.2, oct. 2003.

[144] Strata. Strata foto 3d cx. https://www.strata.com, 2015.

[145] C. Strecha, R. Fransens, and L. Van Gool. Combined depth and outlier estimation in multi-view stereo. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2394–2401. IEEE, 2006.

[146] C. Strecha, W. von Hansen, L. V. Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[147] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and computational harmonic analysis*, 3(2):pages 186–200, 1996.

[148] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal on Mathematical Analysis*, 29(2): pages 511–546, 1998.

[149] W. Sweldens and P. Schröder. Building your own wavelets at home. In *Wavelets in the Geosciences*, pages 72–107. Springer, 2000.

[150] R. Szeliski. Rapid octree construction from image sequences. *CVGIP Image Understanding*, 58:pages 23–23, 1993.

[151] C. Tenllado, J. Setoain, M. Prieto, L. Piñuel, and F. Tirado. Parallel implementation of the 2d discrete wavelet transform on graphics processing units: Filter bank versus lifting. *Parallel and Distributed Systems, IEEE Transactions on*, 19(3):pages 299–310, 2008.

[152] D. Tingdahl and L. Van Gool. A public system for image based 3d model generation. In *Computer Vision/Computer Graphics Collaboration Techniques*, pages 262–273. Springer, 2011.

[153] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(5):pages 815–830, 2010.

[154] E. Tola, C. Strecha, and P. Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications*, 23(5):pages 903–920, 2012.

[155] J. Tompson and K. Schlachter. An introduction to the opencl programming model. *Person Education*, 2012.

[156] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment—a modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 2000.

[157] W. J. Van Der Laan, A. C. Jalba, and J. B. Roerdink. Accelerating wavelet lifting on graphics hardware using cuda. *Parallel and Distributed Systems, IEEE Transactions on*, 22(1):pages 132–146, 2011.

[158] M. Vergauwen and L. Van Gool. Web-based 3d reconstruction service. *Machine vision and applications*, 17(6):pages 411–426, 2006.

[159] VisualSize. Visualsize. http://www.visualsize.com/index.php, 2015.

[160] G. Vogiatzis, P. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 391–398. IEEE, 2005.

[161] G. Vogiatzis, C. Hernandez, P. Torr, and R. Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(12):pages 2241 –2246, dec. 2007.

[162] H. Wang, Y. Liu, and S. Zeng. A hybrid particle swarm algorithm with cauchy mutation. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 356–360. IEEE, 2007.

[163] L. Wang, R. Chen, and D. Kong. An improved patch based multi-view stereo (pmvs) algorithm. In *3rd International Conference on Computer Science and Service System*. Atlantis Press, 2014.

[164] Y. Wexler and R. Chellappa. View synthesis using convex and visual hulls. In *Proc. BMVC*, 2001.

[165] C. Wu. Visualsfm : A visual structure from motion system. http://ccwu.me/vsfm/, 2015.

[166] Q. Yang, L. Wang, R. Yang, H. Stewénius, and D. Nistér. Stereo matching with color-weighted correlation, hierarchical belief

propagation, and occlusion handling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(3):pages 492–504, 2009.

[167] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–211. IEEE, 2003.

[168] R. Yang, M. Pollefeys, and G. Welch. Dealing with textureless regions and specular highlights-a progressive space carving scheme using a novel photo-consistency measure. In *null*, page 576. IEEE, 2003.

[169] A. Yezzi and S. Soatto. Stereoscopic segmentation. *International Journal of Computer Vision*, 53(1):pages 31–43, 2003.

[170] K.-J. Yoon and I. S. Kweon. Adaptive support-weight approach for correspondence search. 2006.

[171] K.-J. Yoon, E. Prados, and P. Sturm. Joint estimation of shape and reflectance using multiple images with known illumination conditions. *International Journal of Computer Vision*, 86(2-3):pages 192–210, 2010.

[172] C. Zach, K. Karner, B. Reitinger, and H. Bischof. Space carving on 3d graphics hardware. *algorithms*, 14:page 7, 2004.

[173] A. Zaharescu, E. Boyer, and R. Horaud. Transformesh: a topology-adaptive mesh-based approach to surface evolution. In *Computer Vision–ACCV 2007*, pages 166–175. Springer, 2007.

[174] G. Zeng, S. Paris, and L. Quan. Robust carving for non-lambertian objects. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 119–122. IEEE, 2004.

[175] G. Zeng, S. Paris, L. Quan, and F. Sillion. Accurate and scalable surface representation and reconstruction from images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(1): pages 141–158, 2007.

[176] Y. Zhou and Y. Tan. Gpu-based parallel particle swarm optimization. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 1493–1500. IEEE, 2009.

[177] Y. Zhou and Y. Tan. Particle swarm optimization with triggered mutation and its implementation based on gpu. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1–8. ACM, 2010.