VOLUMETRIC RECONSTRUCTION OF RIGID OBJECTS FROM IMAGE SEQUENCES

by

Naren Ramchunder

Submitted in fulfilment of the academic requirements for the degree of Master of Science in Engineering in the School of Electrical, Electronic and Computer Engineering at the University of Kwa-Zulu Natal, Durban, South Africa

Supervisor:
Mr. Bashan Naidoo

2012

To my parents, Roy and Maya

This document was created in LaTeX.

Acknowledgements

I would like to thank my supervisor, Mr Bashan Naidoo, for the consistent support and guidance he has provided me during my research and for always assisting me with great advice, patience and kindness. I am very grateful for the time and effort he has invested in reviewing my work and for the invaluable suggestions he has made.

Many thanks goes out to my sponsors, SAAB Grintek Communications and Armscor, for their financial support and interest in my project. I would specifically like to thank Mrs Franzette Vorster, Mr Danny Ignatov, Mr Preshon Jeebodh and Mr Francois van der Merwe for their involvement in this sponsorship. The Financial assistance of the National Research Foundation (NRF) towards this research is hereby gratefully acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

To my friend and office-mate Neil Hendry, my thanks for always being willing to 'lend an ear' to help solve problems. To the rest of my postgraduate colleagues and members of staff that I have been fortunate to get to know, thank you for the friendship and support you have given me.

Most importantly I would like to thank my parents, Roy and Maya, for their unwavering support and for the many sacrifices that they have made to allow me to pursue my studies. To my wonderful and understanding girlfriend Carminee, thank you for your encouragement and support. Lastly, to my close friends, thanks for the support and friendship that that you have shown me.

Abstract

Live video communications over bandwidth constrained ad-hoc radio networks necessitates high compression rates. To this end, a model based video communication system that incorporates flexible and accurate 3D modelling and reconstruction is proposed in part. Model-based video coding (MBVC) is known to provide the highest compression rates, but usually compromises photorealism and object detail. High compression ratios are achieved at the encoder by extracting and transmitting only the parameters which describe changes to object orientation and motion within the scene. The decoder uses the received parameters to animate reconstructed objects within the synthesised scene. This is scene understanding rather than video compression. 3D reconstruction of objects and scenes present at the encoder is the focus of this research.

3D Reconstruction is accomplished by utilizing the Patch-based Multi-view Stereo (PMVS) framework of Yasutaka Furukawa and Jean Ponce. Surface geometry is initially represented as a sparse set of orientated rectangular patches obtained from matching feature correspondences in the input images. To increase reconstruction density these patches are iteratively expanded, and filtered using visibility constraints to remove outliers. Depending on the availability of segmentation information, there are two methods for initialising a mesh model from the reconstructed patches. The first method initialises the mesh from the object's visual hull. The second technique initialises the mesh directly from the reconstructed patches. The resulting mesh is then refined by enforcing patch reconstruction consistency and regularization constraints for each vertex on the mesh.

To improve robustness to outliers, two enhancements to the above framework are proposed. The first uses photometric consistency during feature matching to increase the probability of selecting the correct matching point first. The second approach estimates the orientation of the patch such that its photometric discrepancy score for each of its visible images is minimised prior to optimisation.

The overall reconstruction algorithm is shown to be flexible and robust in that it can reconstruct 3D models for objects and scenes. It is able to automatically detect and discard outliers and may be initialised by simple visual hulls. The demonstrated ability to account for surface orientation of the patches during photometric consistency computations is a key performance criterion. Final results show that the algorithm is capable of accurately reconstructing objects containing fine surface details, deep concavities and regions without salient textures.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The task of achieving high quality video communications over bandwidth constrained systems is becoming an increasingly desirable capability. This is particularly true for hand held communications devices, where advancement in technology has made it possible for these devices to possess the necessary hardware to facilitate video communications. In military scenarios, low bandwidth communications systems are often preferred due to their robustness and flexibility of application. However, these communication links do not possess the required bandwidth capabilities for traditional live video communications. Model-Based Video Coding (MBVC) offers a technique of facilitating video communications over these low bandwidth radio links. A MBVC system uses three dimensional (3D) computer models to represent objects within the scene [2]. At the encoder, parameters which describe changes in object orientation and motion within the scene are determined and then updated for each frame of the video sequence. The only information exchanged between the encoder and decoder are these model update parameters. The decoder uses the received information to animate reconstructed objects within the synthesised scene. The work presented in this dissertation aims to address 3D model and scene reconstruction for the MBVC system.

Section 1.1 of this Chapter provides a conceptual overview of model-based video coding and its system components. Thereafter Section 1.2 discusses the main contributions of this dissertation together with the requirements for the 3D reconstruction component of a model-based coding system. The Chapter concludes with Section 1.3 where an outline of the dissertation is presented.

1.1 Model-Based Video Coding

Model-based video coding techniques [2–6] utilise 3D computer models to describe scene objects. They begin by analysing the input video sequence to determine information about the nature and location of objects within the scene. The information gained by this analysis is then used to synthesize 3D computer models of each object using computer vision techniques. The only information that is streamed across the network are the high level motion, deformation and lighting properties of the scene objects [3].

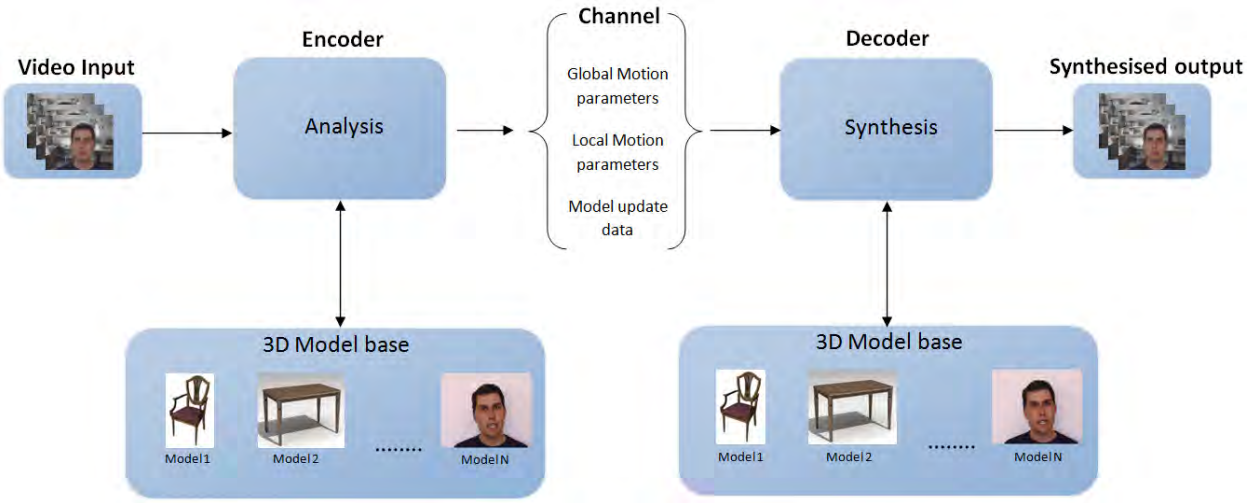


Figure 1.1: Generalised MBVC system - The system employs various models and model seeds to describe the scene [3; 7].

Figure 1.1 shows the basic structure of a generalised model-based video communications system. The scene at the encoder, which may be anything from a simple head and shoulder sequence to complex environments containing many objects, is captured using single or multiple cameras. During an initialisation phase the encoder uses the captured images to reconstruct 3D models for each object identified in the scene. This collection of objects forms the 3D model base of the system. Model information such as shape and texture are transmitted to the decoder only once and may be stored for future sessions. Thereafter the encoder analyses the video sequence and determines the 3D motion parameters of objects using the 3D model information. For a head and shoulder scene, the motion information would typically comprise of global head motion and local facial expression changes. Furthermore, the analysis may include model update data such as updated shape

and texture changes for objects as well as background. The transmitted motion parameters and model information is used at the decoder to modify the object models and synthesize a virtual view of the scene. An interesting point to note is that the rendered scene at the decoder may be viewed from a variety of viewing angles and is not restricted by the current viewpoint of the encoder. Since the amount of information transmitted for each frame of the video sequence is small, MBVC promises large reductions in bit rates compared to hybrid inter-frame coding schemes such as H.261/263/264 and MPEG-1/2. A study in [8] has shown that simple animated face models require only 0.5-1kbit/s, with more realistic models requiring 5-10kbit/s [2]. Achieving such low bit rates makes MBVC very desirable for applications such as video-conferencing.

1.1.1 MBVC System Components

The advancement of highly realistic computer graphics techniques and a set of standards for facial and body animation described by MPEG-4 [9] has essentially simplified syntheses at the decoder. Hence, much of the research effort in MBVC over the past few years has concentrated on the difficult analysis phase. Several areas that are of interest and require further development can be identified.

- ***Object detection*** is necessary to provide information such as the presence and location of objects of interest such as a person's face.
- ***Model initialisation*** is required to correctly reconstruct 3D models for identified objects or change the shape of existing models (generic models) to conform to the observed object's shape. An example of this in [10] where a morphable 3D face model, obtained from a database of facial scans, aids the reconstruction of high quality face models.
- ***Texture coding*** is needed to provide a realistic representation of the modelled objects at the decoder.
- ***Object tracking*** or global motion tracking ensures that the model mimics the motions of the observed objects. An extension to this is local tracking, where in the case of a facial model, would extract facial expressions represented by the motion and position of the mouth, eyes and so forth. This facial expression information is defined and can be encoded in MPEG-4 as facial animation parameters (FAPs) [9].

- ***Model refinement*** continuously updates the shape and texture of the fitted 3D models to match the observations at the encoder.

In a realistic model-based video coding system, each of the above components presents its own set of challenges that need to be addressed. This dissertation aims to address one of these problems, that of constructing 3D models of objects and scenes from images (model initialisation). 3D reconstruction from photographs has received much interest in recent literature and an overview of state-of-the-art 3D reconstruction techniques is provided in Chapter 2. For now Section 1.2 defines the requirements of the 3D reconstruction component of the MBVC system.

1.2 Problem Definition

This thesis proposes a solution to the model initialisation problem, where 3D models of objects and scenes are produced from image sequences that contain them.

PROBLEM DEFINITION: *From a single or multiple camera image sequence of a static rigid object, estimate as accurately as possible its 3D shape and texture.*

The inclusion of image sequences captured with single or multiple camera setups extends the flexibility of the system to include a variety of applications. In the case of video telephony over hand held devices, a single camera image sequence is preferred as the added expense of an additional camera is undesirable. However in situations such as dedicated video conference rooms the additional cameras are advantageous as they increase the field of view and depth-perception of the system.

This study is also limited to reconstruction of rigid objects. This constraint is necessary as it becomes difficult to accurately reconstruct the shape of the scene if it changes arbitrarily during the modelling phase [11]. Deformable objects such as the human face can be modelled as rigid objects during the initialisation phase. Thereafter, facial animation techniques such as [12; 13] can be used to animate the model and mimic the original video sequence.

The accuracy of 3D reconstruction is principally related to image resolution as well as other factors such as image quality and camera calibration accuracy. This research aims to produce reconstructions, of the best possible accuracy, given these factors. This is due to the fact that it is easy to

decrease the accuracy/resolution of a 3D mesh model using mesh simplification techniques such as Mesh Optimisation by H. Hoppe *et al* [14], whereas increasing the accuracy of the model once the modelling process is complete is impossible. In this dissertation a number of parameters that control the accuracy and reconstruction time of the algorithm are available.

1.3 Outline of Dissertation

Three dimensional reconstruction from photographs is a popular and well researched computer vision topic with a variety of application areas ranging from generating realistic 3D models for the entertainment industry to scientific metrology [15] and object analysis [16]. Figure 1.2 shows a schematic overview of the main processing modules within the 3D reconstruction pipeline employed in this dissertation.

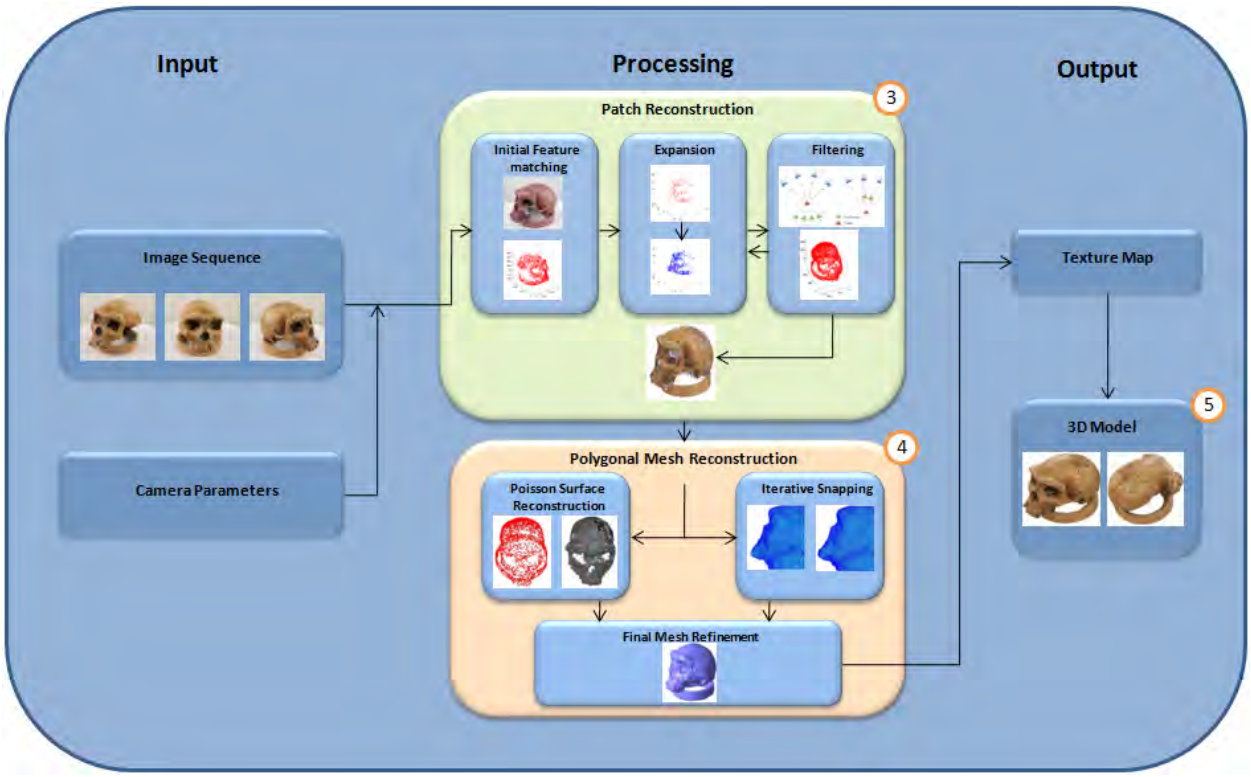


Figure 1.2: Schematic thesis overview - Numbered stages correspond to the associated chapter numbers

Raw data such as the input images and camera calibration parameters are transformed into a sparse 3D point model by the feature matching process. Subsequent expansion and filtering stages increase the density and quality of the point model. The point model is then converted into a mesh model and refined before being texture mapped, producing a realistic 3D model. A circled number attached to each significant process indicates the corresponding chapter number where the process is discussed.

Chapter 2 presents a review of state-of-the-art reconstruction algorithms together with a motivation for the proposed approach in relation to these methods. Chapter 3 discusses the Patch based reconstruction algorithm and begins by introducing fundamental concepts such as the patch model, photometric discrepancy function and image model. Thereafter the reconstruction processes consisting of Feature Matching, Feature Expansion and Patch Filtering are presented. Chapter 4 deals with converting the orientated point model, generated by the Patch Reconstruction algorithm, into a closed polygonal mesh model. The Mesh Reconstruction algorithm begins by firstly initializing a polygonal mesh model using either Poisson Surface Reconstruction or Iterative Snapping Mesh Reconstruction. Thereafter a Mesh Refinement process is undertaken to further improve the appearance of the mesh model. The final system implementation and results are discussed in Chapter 5. The chapter is divided into two sections. The first is Intermediate Results, where a discussion on the implementation of the algorithms is given together with preliminary test results. The second section presents the final results achieved by the system on a number of object and scene datasets. The chapter concludes with a quantitative evaluation of the reconstruction system against ground truth models.

Chapter 2

Literature Review

2.1 Introduction

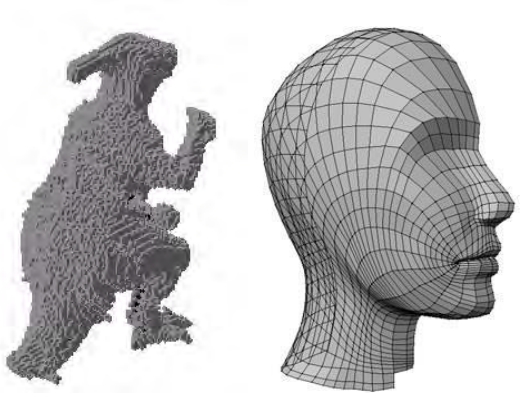
The task of acquiring three dimensional models from photographs, a process known as Image based modelling or Multi-view stereo (MVS) reconstruction, is a deeply researched domain. Three dimensional computer models have proved useful in many applications such as non-contact object inspection, digital archiving of cultural heritage artefacts, and generating realistic 3D graphics models for the entertainment industries. 3D models are traditionally captured using laser range scanning techniques [17–20] and produce a volumetric model by fusing together individual scans known as depth maps. These methods do however have some drawbacks. Apart from their relatively high cost $\approx$ \$100K, many of the older systems were slow due to the mechanical nature of scanning and lacked the ability to capture high resolution colour models [21; 22]. More recently, some of these issues have been addressed with systems such as the hand-held VIUscan colour laser scanner which can achieve a geometric resolution of 0.1 mm and capture full texture at a resolution of 250 DPI [23]. Nevertheless, these systems are still expensive and have limited depth of field. On the other hand, digital cameras available today are inexpensive and are able to acquire images at both high resolution and speed, thus sparking great interest in image based modelling. Over the years various methods with differing assumptions and experimental set-ups have been proposed, making classification of algorithms necessary. Multi-view reconstruction algorithms can be understood based on attributes such as *initialisation requirements*, *scene representation*, *photo-consistency measures* and *visibility model*. The following subsections provides a brief description of these attributes.

2.1.1 Initialization Requirements

Multi-view stereo reconstruction algorithms require input such as a set of calibrated images and some form of information about the geometry of the object or scene that is to be reconstructed [24]. For most algorithms a rough bounding box or volume serves well as an initial estimate [25–28]. Others extract an object silhouette from each image and reconstruct a visual hull that serves as an initial outer estimate of the object's geometry [24; 29–32]. In either case some form of constraint on the scene geometry is established. This can be used as an effective way to discard outliers.

2.1.2 Scene Representation

There are numerous ways of representing the geometry of an object or scene. Some common representations include voxels (Figure 2.1a), level-sets, polygon meshes (Figure 2.1b) and depth maps. Although it is common for algorithms to use a single representation throughout, a combination of different representations may be used for various stages in the reconstruction pipeline [24].



(a) **(b)**

Figure 2.1: Scene representation - (a) using voxels [26] and (b) using polygon meshes [33]

A common representation is the use of a regularly sampled 3D grid volume called a voxel cube. Space carving, Level-set and Graph cuts based algorithms refine this volume by removing voxels that do not satisfy either image discrepancy metrics (silhouettes) or photometric consistency cues [24]. The voxel representation is popular due to its simplicity and flexibility. However, storing each voxel in the cube consumes a large amount of memory and may pose a limitation for high resolution models.

Polygon meshes represent surfaces as a set of connected vertex structures. This topology gives them the ability to represent any type of structure. Their ability to be stored and rendered efficiently make polygon meshes a popular output format for MVS algorithms.

Depth map representations compute, for each pixel in the image, a discrete depth value resulting in a 2D depth map of the scene for each view. This technique avoids re-sampling the geometry on the 3D domain and is also efficient for smaller datasets [24].

2.1.3 Photo-consistency Measures

Photo-consistency measures are used as a means to assess the compatibility between the reconstructed model and the input images. MVS algorithms use photo-constancy measures to determine if the observed 3D point is part of the actual object surface. Photo-consistency measures can be categorized according to whether they operate in *image space* or *scene space* [24; 34].

Scene space measures operate by projecting either a point or a small region of geometry from the 3D scene into the input images. Thereafter the degree of correlation between those projections are evaluated using metrics such as computing the variance between projected pixels [26; 27] or performing window based matching using the sum of squared differences (SSD) or normalized cross correlation (NCC) [29; 35; 36].

Image space methods map an image region from one image to another by using an estimate of scene geometry [24]. A comparison between the predicted and measured image regions yields a photo-consistency measure termed prediction error [37]. Photo-consistency measures can be substituted from one method into another and are hence not intrinsic to any particular algorithm.

2.1.4 Visibility Model

Visibility models aid in identifying the images to be used for evaluating photo-consistency measures. This is particularly important as the visibility of scene objects change with viewpoint. Handling occlusions also becomes easier if one employs a technique for determining the visibility of a scene point. A number of approaches for evaluating visibility include geometric, quasi-geometric and outlier based techniques [24]. Geometric methods form a model of the image capture process together with the shape of the scene. This provides information such as which scene structures may be visible in specific images [24; 25; 30]. Quasi-geometric approaches use geometric constraints

to determine visibility. Occlusions can be minimised using this technique by constraining the photo-consistency assessments to groups of neighbouring cameras [29; 38]. Other approaches use a visual hull of the object to estimate visibility of neighbouring points [30; 31]. Outlier based approaches do not perform explicit geometric modelling and simply treat occlusions as outliers [29; 39]. This method performs well in situations where the majority of scene points are visible in each image with only a small amount of occlusions [24].

2.2 Multi-View Stereo Reconstruction Algorithms

Using the above properties of MVS algorithms we can divide them into roughly four classes namely Voxel-based methods, Polygonal surface-based methods, Multiple depth map-based methods and Patch-based methods. The following subsections provide a brief description of each class of reconstruction algorithm.

2.2.1 Voxel-Based Methods

Voxel based methods consists of Space carving, Level set and Graph cuts based techniques. These techniques use a voxel volume as a surface representation and perform a single pass through a 3D volume whilst computing a cost function. Voxels with costs below a certain threshold are then selected for reconstruction [26; 27]. These algorithms can handle both object and scene datasets however a lack of regularisation (smoothing) may cause noisy reconstructions, see Figure 2.2.

(a) **(b)** **(c)**

Figure 2.2: Voxel-based methods - (a) voxel colouring method by Seitz *et al.* [26]. A volumetric graph cuts method (b) by Vogiatzis *et al.* [40], and a Level-set based method (c) by Pons *et al.* [37]

Level set techniques represent surfaces as a time-varying volumetric density function. This function is then iteratively refined by maximizing photometric consistencies [35; 37; 41; 42]. Finally a

surface extraction algorithm such as Marching cubes [43] is used to obtain surface. Level set techniques can easily handle topological changes, however the use of voxels requires large amounts of memory for high-resolution input images. Graph cuts-based methods [30; 40; 44–47] express image discrepancies using weights of edges linking adjacent voxels in a volumetric model. The surface is then reconstructed by finding a minimum cut of the graph. Many of these voxel-based methods are able to handle both object and scene data sets however a volume bounding the object/scene needs to be known in advance.

2.2.2 Polygonal Surface-Based Methods

This class of algorithms employs polygonal surface meshes as a surface representation [29; 40; 48]. As part of an initialisation step, these algorithms extract silhouettes and construct a visual hull model onto which a surface mesh is initialized. The surface mesh is then refined by optimising the position of each vertex using photometric and geometric consistencies.

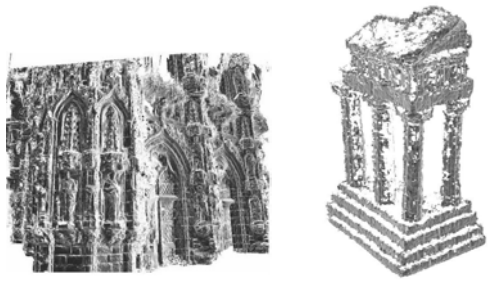
(a) (b) (c)

Figure 2.3: Polygonal surface based methods - (a) and (b) Carved visual hull method by Furukawa and Ponce [48]. An iterative deformation method (c) by Esteban *et al.* [29]

These techniques can produce very impressive results however any large errors in the visual hull cannot be recovered, hence there is a strong dependence on obtaining an accurate visual hull [29]. Furthermore topological changes cannot be handled easily due to their dependence on visual hulls for initialization [14]. Therefore these methods perform well on object datasets where silhouettes can be accurately extracted.

2.2.3 Multiple Depth Map-Based Methods

Multiple Depth Map-Based methods compute a depth map for each input image. One of two approaches is common, in [49] for each pixel a discrete depth value is estimated independently. Whereas [50; 51] use a Markov Random Field to model all the depth maps and the Expectation Maximization algorithm [52] is used to perform reconstruction.



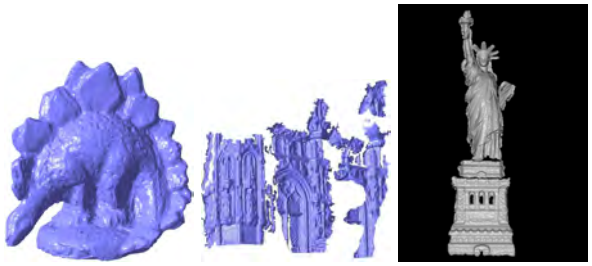
(a) **(b)**

Figure 2.4: Multiple depth map based methods - (a) Expectation-maximization is employed by Strecha *et al.* [51]. (b) Goesele *et al.* uses an algorithm that is simple yet robust [49].

In both cases the final reconstruction step requires that the individual depth maps be fused into a single 3D model [53; 54]. However, the fusion step may result in noisy reconstructions if the individual depth maps are inconsistent with each other. Multiple Depth Map-Based methods are able to handle object and scene data sets within the same framework.

2.2.4 Patch-Based Methods

The forth class of MVS algorithms are called Patch-Based methods [53; 55–58]. These algorithms use an unconnected set of surface patches as a surface representation. This flexible representation allows these algorithms to handle both object and scene datasets within the same framework. Reconstruction begins by first extracting and matching a set of feature points across the input images, resulting in a set of unconnected surface patches. A post processing stage where a polygonal mesh model is reconstructed from the surface patches is then undertaken to produce the final connected 3D model [59]. The work presented in this dissertation belongs to this category.



(a) (b) (c)

Figure 2.5: Patch based methods - (a) and (b) Patched based multi-view stereo method by Furukawa and Ponce [58]. (c) A method for reconstruction of community photo collections by Goesele *et al.* [53]

2.3 Data Sets

2.3.1 Object Reconstruction

The first category of datasets are referred to as *object datasets* and are classified as having a single object, completely visible, in an uncluttered set of images captured from all around it [60]. This set-up makes it relatively easy to segment the object from the background and compute it's visual hull [61–63]. Hence, this data type is the most commonly used in MVS. In some algorithms [26; 29; 40; 44; 45; 48] visual hull models are used as an initialization and thereafter a refinement process is used to improve the appearance of the model. Figure 2.6 contains examples of object datasets.

(a) (b)

Figure 2.6: Object data - A single object fully visible in an uncluttered set of images [24; 64].

Other approaches [46; 49; 50] simply use the segmentation information to improve reconstruction results. A quantitative comparison of multi-view algorithms for this type of dataset is presented in [24].

2.3.2 Scene Reconstruction

This dataset differs from object datasets in that the range of viewpoints may be limited resulting in partial occlusions of the target object(s). This also prevents the extraction of effective bounding volumes. Some examples of scene datasets, Figure 2.7, include: outdoor scenes, buildings and walls.

(a) **(b)**

Figure 2.7: Scene data - The range of viewpoints for the object of interest may be limited [65].

2.3.3 Crowded Scene Reconstruction

Crowded scene datasets differs from scene datasets in that moving obstacles may appear in front of the static object of interest in different places and in multiple images.

(a) **(b)**

Figure 2.8: Crowded scene data - Moving obstacles may appear in front of the static object of interest [65].

A typical example of this is when people pass in front of a building, Figure 2.8 while the images are captured, and thus occlude small portions of the building in those images. As the goal in MVS is to reconstruct any structure that is rigid with respect to the cameras, any moving obstacles should be detected and ignored by the reconstruction algorithm. This makes crowded scene datasets very challenging and only a few algorithms [50; 53; 58] are able to handle these data sets successfully.

2.3.4 Datasets for Testing

In this dissertation five real datasets are used for experimentation and evaluation. Three object datasets: skull, temple and dino, contain fine surface details, deep concavities and regions without salient textures. Dino and Temple form part of the Middlebury MVS evaluation project, and have been carefully calibrated by the authors of [24]. The skull dataset was acquired and calibrated by Jodi Blumenfeld and Steven R. Leigh, and can be seen to be particularly challenging because of the deep eye concavities. The two scene datasets used are: fountain and city hall, courtesy of Christoph Strecha. Viewpoints between images in these datasets vary considerably, making successful matching of image features very challenging. Appendix A contains illustrations of the datasets used in this dissertation.

2.4 Proposed System Choices

The aforementioned techniques for multi-view stereo reconstruction can be evaluated on several criteria that are crucial to the performance of any 3D reconstruction system. These include the type of datasets that can be handled, occlusion handling, robustness to changes in illumination within the scene, reconstruction accuracy and computational performance. However it is worth noting that algorithms generally seek to optimize performance in specific subsets of these criteria.

In terms of scene representation most algorithms simply employ a single representation throughout the reconstruction pipeline. However, it is more advantageous to use different representations for the initial modelling and final surface representations. In this dissertation an unconnected surface model (patch model) is used during the reconstruction process and a polygonal mesh surface representation is employed during refinement of the surface models. The use of an unconnected patch model allows modelling of arbitrary shapes and increases the flexibility of the algorithm. Once the shape of the scene has been estimated using the patch model it is converted into a polygonal mesh model for further refinement. This multi stage surface representation allows reconstruction of any scene topology and thus the algorithm is able to handle object and scene datasets within the same framework.

The extent to which a reconstruction technique is robust to occlusions and changes in illumination is related to the photometric consistency measure and visibility model employed. Scene space measures in combination with normalized cross correlation (NCC) are used in this dissertation to

effectively deal with these problems. During the patch reconstruction stage, occlusions and non-Lambertian effects are handled by simply ignoring images that have a bad photometric score against the reference image. Thus only images whose photometric scores against the reference image are above a certain threshold are used in the evaluation of the visibility model.

In this dissertation a reconstruction algorithm similar to that of Yasutaka Furukawa and Jean Ponce [58] (with some extensions) is proposed. Reconstruction begins by identifying , in each image, prominent image features such as corners, edges and blobs using the difference of Gaussian and Harris feature detectors. Thereafter each image feature is matched across multiple neighbouring images in an attempt to reconstruct a 3D point on the objects surface. An optimization procedure is undertaken to correctly estimate the point's position and orientation (surface normal) on the surface of the model. Prior to optimization we introduce a new estimate of surface orientation that allows the optimization procedure to converge faster and more accurately to the global minimum. Once all the image features have been matched an expansion procedure is undertaken to increase the reconstruction density. The expansion procedure begins by taking a matched point and generating neighbouring points that satisfy the expansion conditions set out in Chapter 3.3.1. The expansion stage is followed by three filtering stages that remove outlying points on the surface. The expansion and filtering stages may be iterated a number of times until the required density of the point model is achieved. To further improve the appearance of the model, the orientated point model is converted into a polygonal mesh model using one of two approaches based on the type of dataset. Thereafter the polygonal mesh model is refined based on reconstruction consistency and smoothness metrics.

The proposed system choices outline an algorithm that is accurate, robust to outliers and flexible in that it can handle different types of datasets within the same framework.

2.5 Summary

Traditional laser scanning techniques that fuse together individual depth maps to produce a 3D model require specialist equipment which can be expensive. Digital cameras available today are both inexpensive and can capture high resolution colour images. Thus 3D reconstruction using image based modelling techniques are increasing in popularity and performance.

A number of attributes such as *initialisation requirements*, *scene representation*, *photo-consistency measures* and *visibility models* are used to classify MVS algorithms. *Initialisation requirements* place constraints on the scene geometry using either bounding boxes or object silhouettes. *Scene representation* dictates the type of topology used to describe the scene. Several common scene representations such as voxels, level-sets, polygon meshes, and depth maps are used in MVS. *Photo-consistency measures* provide a means of assessing the reconstructed 3D model and are categorised as scene space or image space measures. In either case the amount of correlation between the image regions determines the validity of a particular 3D scene point. Finally, *visibility models* are used to determine the set of images in which a 3D scene point is visible. Determining the visibility of a scene point is important in evaluating photo-consistency measures as well as handling occlusions.

Multi-view stereo reconstruction algorithms are often classified according to the type of scene representation used. Hence the four primary classes of MVS algorithms are Voxel-based methods, Polygonal surface-based methods, Multiple depth map-based methods and Patch-based methods. All of the above methods, with the exception of Polygonal surface-based methods, are able to handle both object and scene datasets within the same framework. To increase the flexibility of the reconstruction algorithm. It is often more advantageous to use different scene representations at various stages in the reconstruction pipeline.

MVS datasets are classified as either an object, scene or crowded scene datasets. Object datasets are regularly used in MVS due to the relative ease of obtaining a visual hull model. Scene datasets are marginally more difficult as the lack of a tight bounding volume makes filtering outliers more difficult. Crowded scene datasets are by far the most difficult type of dataset. Moving obstacles in the scene should be detected and ignored by the reconstruction algorithm. Only a few point/patch based algorithms are able to handle these data sets successfully.

The proposed system choices are influenced by criteria such as the type of datasets that can be handled, occlusion handling, robustness to changes in illumination within the scene, reconstruction accuracy and computational performance. To address these criteria several key system choices are proposed. A two stage surface representation is employed to increase the flexibility and accuracy of the reconstruction algorithm. Robustness to occlusions and changes in illumination are addressed using a scene space photoconsistency measure in combination with normalized cross correlation.

Occlusions are also dealt with by not selecting images for the visibility model that have a photometric score that is below a certain threshold with the reference image. The reconstruction process begins by identifying image features such as edges, corners and blobs. Each feature is then matched across its visible images, resulting in an orientated point model. The density and accuracy of the model is improved by iterating between expansion and filtering stages. To simplify rendering and to further improve the accuracy of the model, the orientated patch model is converted to a polygonal mesh model and refined using reconstruction consistency and smoothness metrics.

Chapter 3 begins by introducing fundamental concepts used by the patch reconstruction algorithm and thereafter describes in detail the matching technique, expansion procedure and filtering algorithms. The conversion of the patch model into a polygonal mesh model and subsequent mesh refinement is presented in Chapter 4.

Chapter 3

Patch-Based Multi-View Stereo

3.1 Introduction

This chapter describes a Patch-Based Multi-View Stereo (PMVS) algorithm for 3D reconstruction from image sequences. Reconstruction is achieved using a three stage process consisting of initial feature matching, feature expansion and a patch filtering procedure to remove outliers. Image features such as corners, edges and blobs are identified in each image using difference of Gaussian (DoG) and Harris feature detectors. These feature points are then matched throughout the image sequence, resulting in a set of sparse orientated rectangular surface regions termed patches covering the surface of the object. These sparse patches are then iteratively expanded, increasing the density of the reconstruction. Outliers in the patch model are then filtered using visibility constraints. To provide a better understanding of the reconstruction algorithm, the following subsections introduce three fundamental concepts namely the patch model, photometric discrepancy function and image model used in the Patch-based algorithm.

3.1.1 Patch Model

The reconstruction algorithm employed in this dissertation uses a patch model for its surface representation. A patch p can be defined as a small rectangular plane tangent to the surface of the 3D model. The 3D properties of a patch are fully described by the coordinates of its centre $\mathbf{c}(p)$ and unit normal vector $\mathbf{n}(p)$, initially orientated towards its reference camera $\mathrm{R}(p)$ [58]. To simplify computations, the orientation of a patch is chosen such that one edge is parallel with its reference camera's $\mathrm{R}(p)$ x-axis. Finally, the extent of the rectangular patch is chosen such that the projection of p onto $\mathrm{R}(p)$ is $\mu \times \mu$ pixels (μ is typically 5 or 7 pixels) [58].

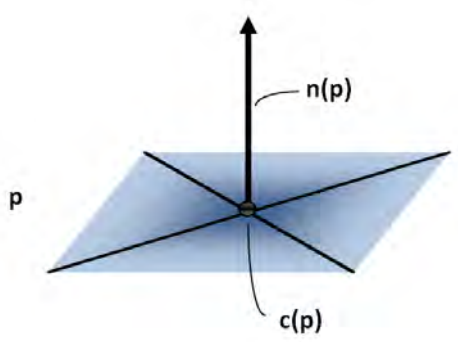


Figure 3.1: A 3D patch object - The figure shows a 3D rectangular patch with its centre and normal denoted as $\mathbf{c}(p)$ and $\mathbf{n}(p)$ respectively.

3.1.2 Photometric Discrepancy Function

The Photometric Discrepancy Function is used as a measure of inconsistency between two image regions. In this dissertation a scene space photo-consistency measure is used to determine the number of images that a candidate patch is visible in. Hence the Photometric Discrepancy Function provides a measure of confidence in the reconstructed patch. Initially the visibility of a patch is denoted by the set of images $\mathbf{V}(p)$ (Equation 3.7) and is determined using geometric constraints. Thereafter the true visibility of a patch is determined using the pairwise Photometric Discrepancy Function h(p, I_1, I_2), and is evaluated by firstly superimposing a $\mu \times \mu$ grid onto the patch p, projecting the grid points onto the images I_i and sampling pixel colours $\mathbf{q}(p, I_i)$ through bilinear interpolation. The pairwise photometric discrepancy function is then computed as one minus the Normalized Cross Correlation (NCC) score between $\mathbf{q}(p, I_1)$ and $\mathbf{q}(p, I_2)$ [58]. This metric is used because of the energy minimisation procedure employed in the optimisation stage.

The global Photometric Discrepancy Function g(p), proposed in [58], is a measure of the average discrepancy score for all the visible images of p, excluding R(p) and is defined as:

$$g(p) = \frac{1}{|V(p) \backslash R(p)|} \sum_{I \in V(p) \backslash R(p)} h(p, I, R(p)). \tag{3.1}$$

The patch optimisation stage minimises the global photometric discrepancy function such that the 3D position of the patch is optimised with respect to its visible images. Most multi-view reconstruction algorithms assume that the object or scene under reconstruction is perfectly Lambertian.

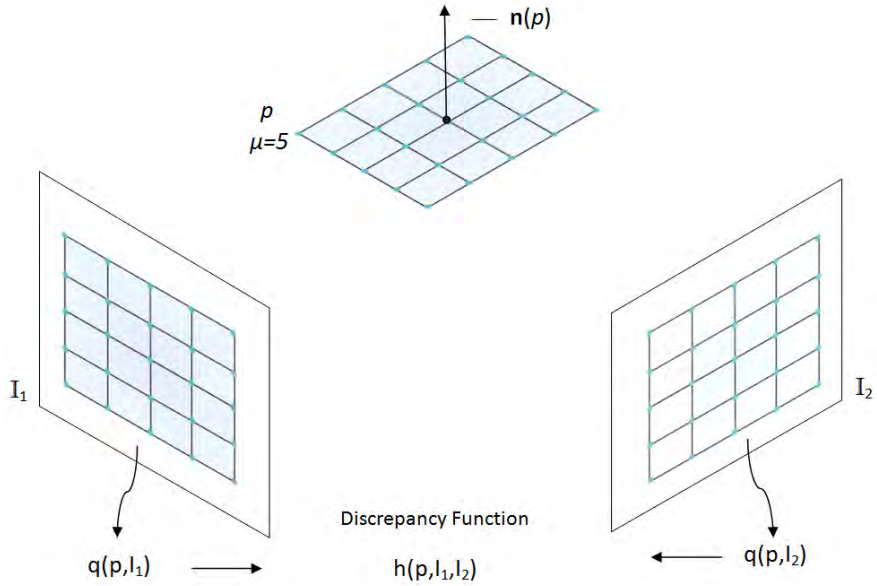


Figure 3.2: Photometric discrepancy - The photometric discrepancy h(p, I_1, I_2) of a patch p is given by one minus the normalized cross correlation score between sets of sampled pixel colours q(p, I_i) [58]

Here too in making this assumption, the discrepancy function g(p) 3.1 may not perform adequately for surface regions containing specular reflections or obstacles partially obstructing the static scene of interest. These effects are handled by simply ignoring images with bad photometric discrepancy scores. Hence, only images with discrepancy score's below a certain threshold α are used in the computations for visibility (α is typically 0.6 or 0.3). These images are denoted by the filtered visibility $V^*(p)$ [58].

$$V^*(p) = \{I \,|I \in V(p), h(p, I, R(p)) \leq \alpha\}\,. \tag{3.2}$$

$$g^*(p) = \frac{1}{|V^*(p) \backslash R(p)|} \sum_{I \in V^*(p) \backslash R(p)} h(p, I, R(p)). \tag{3.3}$$

The new filtered global photometric discrepancy function $g^*(p)$ 3.3 is given by exchanging V(p) in equation 3.1 with $V^*(p)$ given in equation 3.2. Note that equation 3.2 still contains the reference image R(p) by definition. Hence, even the new discrepancy function $g^*(p)$ still cannot handle non-Lambertian effects if present in the reference image. This failure mode is addressed successfully by the patch generation algorithm.

3.1.3 Image Model

One of the biggest advantages of the patch based surface representation is the flexibility with which it can represent arbitrary complex topologies. However accessing neighbouring patches or enforcing regularization cannot be accomplished easily due to the lack of connectivity information within the patch structure. To help perform these tasks, the reconstructed patches are projected onto their visible images and these locations are then recorded in image cells. This is accomplished by partitioning each image I_i into a uniform grid, $C_i(x, y)$, consisting of $\beta_1 \times \beta_1$ pixels cells, as shown in Figure 3.3.

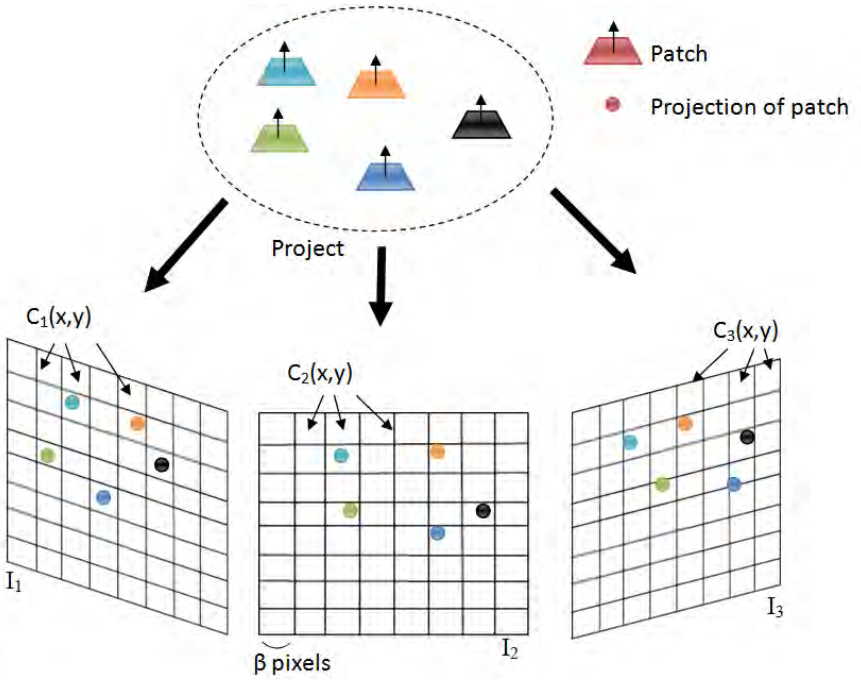


Figure 3.3: Image Model Cells - Each Image Cell stores the set of patches that project into it and are used to perform fundamental tasks such as accessing neighbouring patches and enforcing regularization.

A patch p is then projected onto each one of its visible images $\mathrm{V}(p)$, identifying the corresponding cell $C_i(x, y)$. Each image cell $C_i(x, y)$ then stores the patch set $Q_i(x, y)$ that project into it. Likewise, using the filtered visibility $V^*(p)$ instead of $\mathrm{V}(p)$, we have the set $Q_i^*(x, y)$ [58].

3.2 Initial Feature Matching

The Initial feature matching process aims to generate a sparse set of reconstructed patches evenly distributed throughout the surface of the object. Feature points identified using difference of Gaussian and Harris feature detectors are matched across multiple images in an attempt to generate a patch. The following subsections detail the processes involved in generating the initial feature matches.

3.2.1 Initial Feature Matching Algorithm

Initial Feature Matching is performed using a technique similar to that proposed by Furukawa and Ponce in [58]. The main difference is the approach taken to increase the probability of selecting the correct matching point first. This section details the steps involved in the Initial Feature Matching algorithm.

Initial feature matching begins by detecting features in each image using the DoG and Harris feature detectors. Thereafter for each feature f detected in image I_i, epipolar lines are drawn in the other images to narrow down the search for corresponding feature points. Features f' within two pixels from these epipolar lines are collected and form the set F[1]. These feature points are then sorted in increasing order of photometric discrepancy with the feature *f*. This heuristic increases the probability of selecting the correct matching point f' first, reducing the number of false matches generated and decreasing the computational time. Triangulation between the feature f and those in F are used to determine 3D points near the surface of the model. Each 3D point is then considered as a potential patch centre and an attempt is made to generate a patch using the following procedure [58]:

$$c(p) \leftarrow \{\text{Triangulation from } f \text{ and } f'\}. \tag{3.4}$$

$$n(p) \leftarrow \overline{c(p)O(I_i)}/\overline{|c(p)O(I_i)|}. \tag{3.5}$$

$$R(p) \leftarrow I_i. \tag{3.6}$$

[1]This should not be confused with **F** used to denote the fundamental matrix.

The patch centre is given by triangulation between the feature point f and f'. The patch normal is orientated towards the reference camera I_i. Initially the images in which a patch is deemed visible are computed by simply returning the set of images that satisfy the following geometric condition:

$$V(p) \leftarrow \left\{ I \left| n(p).\overline{c(p)O(I)} / \overline{|c(p)O(I)|} > \cos(\tau) \right. \right\}. \tag{3.7}$$

That is, a patch is deemed visible in image I_i when the angle between the patch normal and the ray from the optical centre, $\mathrm{O}(I_i)$, towards the patch is below a certain angle τ ($\tau = \pi/3$) [58]. The set of filtered images, $V^*(p)$, where the patch is truly visible can then be determined from $V(p)$ using equation 3.2. After initializing the patch candidate p, its geometric components $\mathbf{c}(p)$ and $\mathbf{n}(p)$ are refined using the patch optimization procedure described in section 3.2.3.6.

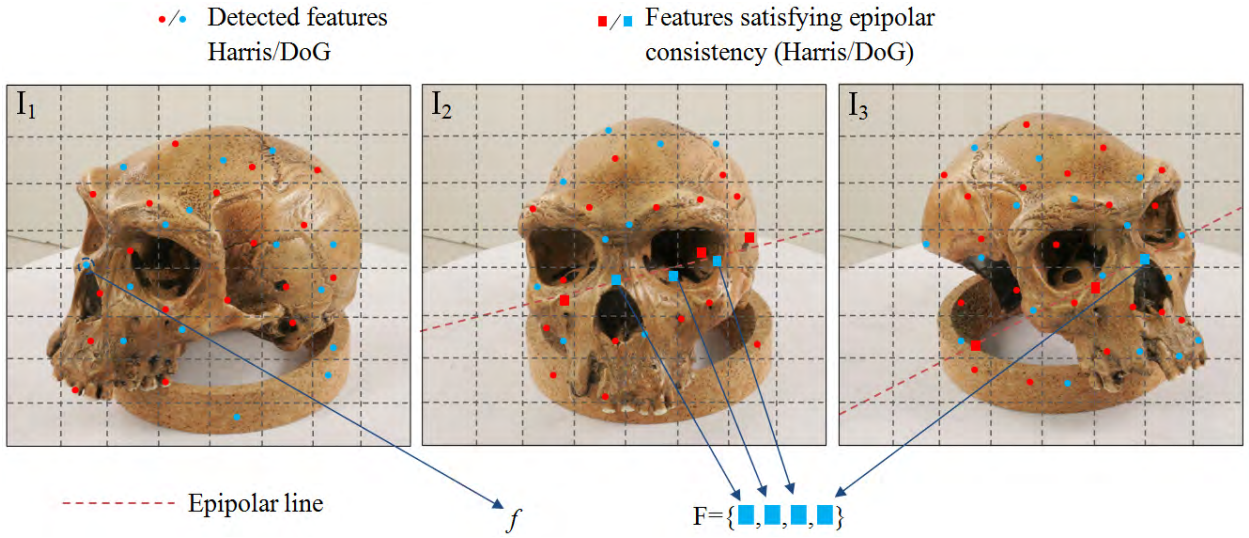


Figure 3.4: Initial Feature Matching algorithm - An illustration of the initial feature matching process. Feature f identified in I_1 is matched with features $f' \in \mathrm{F}$ that satisfy the epipolar constraint in images I_2 and I_3 [60].

Once the optimization is complete, the patch visibility information, V(p) and $V^*(p)$, is then updated using equations 3.7 and 3.2 respectively. The following heuristic is used by [58] to determine if the patch truly lies on the surface of the model. If $|V^*(p)| > \gamma$, that is, if patch p is truly visible by at least γ images, then the patch is said to lie on the surface of the model. The patch is then stored into the set of patches P and in image cells $Q_i(x, y)$ and $Q_i^*(x, y)$. To ensure that another patch is not generated in the same location, all features that are stored in the same image cell as patch p are removed. This also aids in speeding up computations. The Initial Feature Matching

algorithm is able to deal with surface regions containing specular reflections or obstacles partially obstructing the static scene of interest as follows. If a candidate patch is generated with feature f in R containing bad artefacts, the patch optimization will fail. However this does not stop the generation from succeeding in another image without such artefacts. The overall algorithm description is given in Algorithm 1.

Input: Features detected in each image I
Output: Initial sparse set of reconstructed patches P

1. For each image I_i with optical centre O(I_i)
2. For for each feature f detected in I_i
   F $\leftarrow$ {Features satisfying epipolar consistancy}
   Sort F in an increasing order of photometric discrepancy.
3. For each feature $f' \in F$
   Initialize $\mathbf{c}(p)$, $\mathbf{n}(p)$ and R(p)
   Initialize V($\mathbf{p}$) and $V^*(\mathbf{p})$
   Refine $\mathbf{c}(p)$, $\mathbf{n}(p)$
   Update V($\mathbf{p}$) and $V^*(\mathbf{p})$
4. If $|V^*(p)| > \gamma$
   Add p to the corresponding $Q_i(x, y)$ and $Q_i^*(x, y)$)
   Remove features from cells where p was stored
   Add p to P
   Exit loop 3. continue with loop 2.

Algorithm 1: Initial Feature Matching Algorithm [60]

3.2.2 Feature Detection

Feature detection forms the starting point for many computer vision algorithms and as a result many feature detection algorithms have been developed. Feature detection can be described as the process of identifying interest points in images. These interest points can be classified as **Edges**, **Corners** or **Blobs**.

Edges can be described as the boundary between two image regions i.e. regions where the intensity values change sharply. Edges can have any arbitrary shape and are usually defined as a set of image

points which have a strong gradient magnitude [66].

Corners are point like features in an image, which have local two or multi directional signal variations. Hence corner features are very stable across image sequences and are a popular choice for region based matching or object detection [1; 66].

Blobs / regions of interest detectors find image regions that differ in intensity from the surrounding points. Blob detectors can provide useful complementary information which may not be found by edge and corner detectors.

The performance of a feature detector can be judged by its invariance to factors such as scale, rotation, variations in illumination and image noise [67]. The following subsections deal with two popular interest point detectors, the Harris Corner/Edge Detector and the difference of Gaussian Edge/Blob Detector.

3.2.2.1 Harris Feature Detector

The Harris feature detector is a Corner/Edge detecting algorithm that is robust to small image rotations, scale, variations in lighting conditions and noise. It is a popular choice for feature detection and was first introduced in [68]. Feature detection is formulated by taking the local auto-correlation function of the image intensity signal after shifting patches by a small amount in different directions [67]. Given an image point (x,y) and a shift of (Δ x, Δ y), the auto-correlation function, as presented in [67], can be defined as,

$$c(x,y) = \sum_{W} [I(x_i, y_i) - I(x_i + \Delta x, y_i + \Delta y)]^2. \tag{3.8}$$

Where (x_i, y_i) denotes points within the Gaussian window [1],**W**, centred at (x, y) and $I(x_i+\Delta x, y_i+\Delta y)$ denotes the image function. Taking the first order Taylor expansion of the image function we have:

$$I(x_i + \Delta x, y_i + \Delta y) \approx I(x_i, y_i) + [I_x(x_i, y_i)\ I_y(x_i, y_i)] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}. \tag{3.9}$$

Where I_x and I_y are first order partial derivatives. Substituting this approximation into equation 3.8 we have,

[1]For clarity, the weighting factor $e^{-(x^2+y^2)/(2\sigma^2)}$ is omitted

$$
\begin{aligned}
c(x,y) = \quad & \sum_W [I(x_i,y_i) - I(x_i+\Delta x, y_i+\Delta y)]^2 \\
& \sum_W (I(x_i,y_i) - I(x_i,y_i) - [I_x(x_i,y_i) I_y(x_i,y_i)] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix})^2 \\
& \sum_W (-[I_x(x_i,y_i) I_y(x_i,y_i)] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix})^2 \\
& \sum_W ([I_x(x_i,y_i) I_y(x_i,y_i)] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix})^2 \\
& [\Delta x \Delta y] \begin{bmatrix} \sum_W (I_x(x_i,y_i))^2 & \sum_W I_x(x_i,y_i) I_y(x_i,y_i) \\ \sum_W I_x(x_i,y_i) I_y(x_i,y_i) & \sum_W (I_y(x_i,y_i))^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\
& [\Delta x \Delta y] \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\
& [\Delta x \Delta y] Q(x,y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}. \qquad (3.10)
\end{aligned}
$$

Matrix $Q(x,y)$ describes the image intensity shape of the neighbourhood near the origin. Let λ_1 and λ_2 represent the eigenvalues of matrix $Q(x,y)$. We can then characterize the "cornerness" H(x, y), by analysing the eigenvalues of $Q(x,y)$. We consider three cases:

1. When both eigenvalues are small, the pixels in the image region have approximately uniform intensity.

2. When one of the eigenvalues is larger than the other, we have a ridge shaped auto-correlation function. Shifts along this ridge cause little change in Q, whilst shifts in the orthogonal direction cause significant changes in Q, this situation indicates an edge [67; 68].

3. When both eigenvalues are large, $c(x,y)$ has a sharp peak. Hence shifts in any direction result in a substantial increase in Q, this situation signifies a corner [67; 68].

If Q(x,y) is symmetric and positive definite, we have:

$\lambda_1 \lambda_2 = detQ(x,y) = AC - B^2$, $\quad \lambda_1 + \lambda_2 =$ trace Q(x,y)= A + C.

Harris [68] suggested his measure of "cornerness" as:

$$H = \lambda_1\lambda_2 - 0.04(\lambda_1\lambda_2)^2. \tag{3.11}$$

In a typical implementation of this filter one would first smooth the input image using a Gaussian filter with $\sigma = 1$ (determined experimentally) to eliminate noise. Compute image derivatives A, B and C. Compute the measure of "cornerness" H and find corners as the local maxima in H(x, y).

3.2.2.2 Difference of Gaussian Edge/Blob Detector

The difference of Gaussian edge detector is derived from the stable Laplacian of Gaussian image filter. The Laplacian of Gaussian has been shown in [69] to outperform a range of image and gradient based functions such as the Harris corner function. Edges are identified by firstly convolving the original image with two Gaussian kernels of different standard deviations. These are then subtracted producing the difference of Gaussian (DoG).
The Laplacian of Gaussian is given by the following equation [1]:

$$\nabla^2 G(x, y, \sigma) = -\frac{1}{2\pi\sigma^4}(2 - \frac{x^2 + y^2}{\sigma^2})exp(-\frac{x^2 + y^2}{2\sigma^2}). \tag{3.12}$$

This can be normilized by σ^2 resulting in the ssLoG(scale space LoG)[70] which can be related to the DoG

$$\sigma^2\nabla^2 G(x, y, \sigma) = -\frac{1}{2\pi\sigma^2}(2 - \frac{x^2 + y^2}{\sigma^2})exp(-\frac{x^2 + y^2}{2\sigma^2}). \tag{3.13}$$

From [1; 71], taking the diffusion equation using the parametrization σ instead of the time parameter t, we have:

$$\frac{\partial G(x, y, \sigma)}{\partial\sigma} = \sigma\nabla^2 G(x, y, \sigma). \tag{3.14}$$

using the finite difference to expand the left hand side we can approximate equation 3.14 by,

$$\frac{\partial G(x, y, \sigma)}{\partial\sigma} = \lim_{\Delta\sigma\to 0} \frac{G(x, y, \sigma + \Delta\sigma) - G(x, y, \sigma)}{\Delta\sigma}. \tag{3.15}$$

Substituting $\Delta\sigma = \sigma(k - 1)$ we obtain,

$$\frac{\partial G(x, y, \sigma)}{\partial\sigma} = \lim_{k\to 1} \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{\sigma(k - 1)}. \tag{3.16}$$

Hence equation 3.14 can be estimated as,

$$(k-1)\sigma^2\Delta^2 G(x,y,\sigma) \approx G(x,y,k\sigma) - G(x,y,\sigma). \tag{3.17}$$

The difference of Gaussians is given by the right hand side of equation 3.17 and is defined as [1; 71],

$$D(x,y,\sigma) = G(x,y,k\sigma) - G(x,y,\sigma). \tag{3.18}$$

Note that $\sigma^2\Delta^2 G(x,y,\sigma) \approx D(x,y,\sigma)$ as k approaches 1. Lowe [71] found that this approximation has very little influence on the stability of detection, however $k = 2^{1/s}$, where $s > 1$ does provide good practical results. The DoG is less computationally expensive than the Laplacian of Gaussian (LoG) filter and one can easily change the width of the edge by adjusting the variances of each filter [1].

3.2.3 Feature Matching

Feature matching forms the second phase of the Initial Feature Matching algorithm. Interest points identified by feature detectors are matched across multiple images in an attempt to infer 3D information of particular points. Feature matching is carried out using the following procedure as described in [58].

Consider an image I_i with its optical centre denoted by $O(I_i)$. Each feature f detected in I_i is used to identify features f' of the same type (DoG or Harris) that are within two pixels[1] from the corresponding epipolar lines. These candidate matching points are then collected in the other images and form the set F. Triangulation between the feature f and those in F are used to determine 3D points near the surface of the model. These points are then considered as potential patch centres in order of increasing distance from $O(I_i)$. The following subsections cover the various aspects of feature matching.

3.2.3.1 Epipolar Geometry

The Epipolar geometry of a stereoscopic system expresses the geometric relationship between the 3D world and the 2D images of a rigid body taken from different viewpoints. This geometric relationship is determined by the cameras relative pose and internal parameters and is hence valid for

[1]This value was determined experimentally using the camera calibration parameters and resulting epipolar geometry.

any scene structure [72]. Essentially given two different perspective views of the same scene, the epipolar geometry can be determined from known camera parameters and relative position (calibrated cameras) or feature correspondences (uncalibrated cameras). Using the resulting epipolar geometry reduces the complexity of a search for a match from the entire 2D image plane to points along the epipolar line [73].

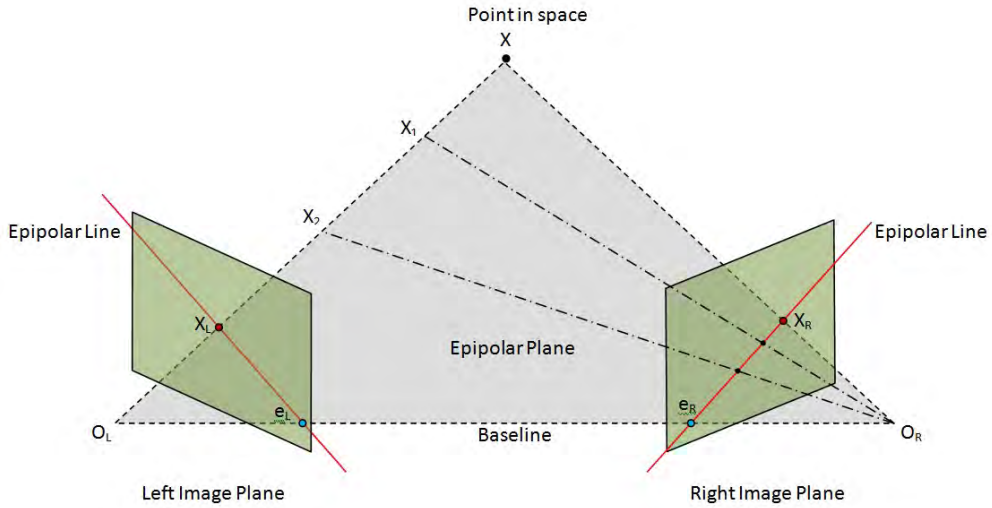


Figure 3.5: Epipolar Geometry - The epipolar geometry and epipolar constraint between two images

Figure 3.5 illustrates the concept of epipolar geometry. A 3D point $\mathbf{X}$ in space is viewed by two cameras with optical centres O_L and O_R behind their respective image planes. The epipolar plane $\prod_e$ is determined by point $\mathbf{X}$ and the optical centres O_L and O_R. The points X_L and X_R are image projections of the 3D point $\mathbf{X}$ on their respective image planes. The focal points O_L and O_R can be connected by a line called the baseline, the intersection of this line with the image planes defines points e_L and e_R known as the epipoles. For the special case when this line does not intersect with the image planes, the epipolar points lie at infinity [1]. An observation made from the left camera along the line $O_L - X$ will yield the point X_L on the image plane. When viewed from the right camera we observe the line $e_R - X_R$, known as the epipolar line. A similar observation is made when viewing the line $O_R - X$ from the right camera where we see the point X_R. However when viewing this line from the left camera we see the line $e_L - X_L$ [1]. This leads us to the Epipolar Constraint.

Epipolar Constraint: The image projection X_i of a 3D space point $\mathbf{X}$, must lie in the image plane along the corresponding epipolar line [1]. That is, given the point X_L, if the relative camera

parameters are known, the epipolar line along $e_R - X_R$ can be determined. Hence we can find X_R, the image point of $\mathbf{X}$, since it lies somewhere along the line $(e_R - X_R)$. This constraint restricts the 2D search for correspondences from the whole image plane to a line, which significantly reduces the complexity of the search. The epipolar geometry of a stereo system can be described using the Essential and Fundamental matrices.

3.2.3.2 The Essential and Fundamental Matrix

The epipolar geometry between two perspective views of a scene can be captured in an algebraic representation known as the fundamental matrix $\mathbf{F}$.

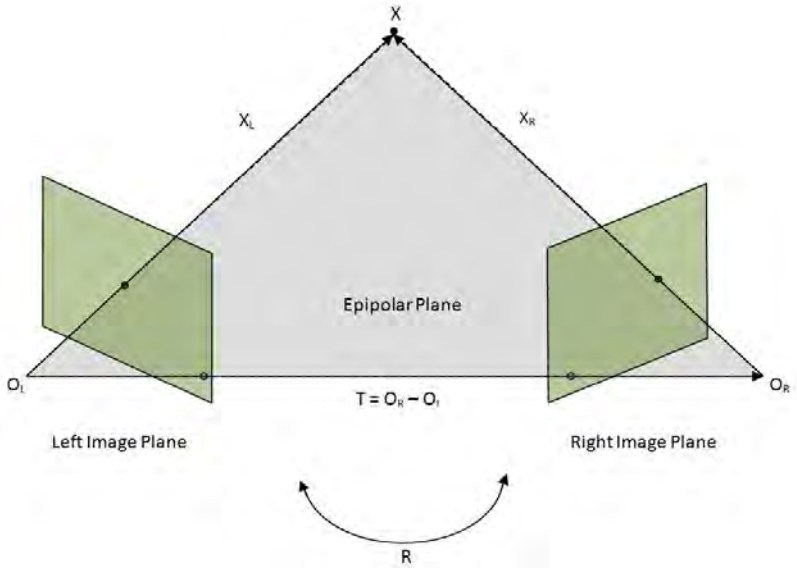


Figure 3.6: Epipolar Geometry - Epipolar plane vectors between two stereo images

Figure 3.6 shows a stereo camera system and resulting epipolar plane vectors X_L and X_R defined by a point $\mathbf{X}$, viewed by both cameras, and optical centres O_L and O_R of each camera. Each camera in a stereoscopic system has its own local coordinate system. The rotation matrix $\mathbf{R}$ and translation matrix $\mathbf{T} = O_R - O_L$ provide a means of exchanging between these two coordinate systems. Hence, for vectors X_L and X_R pointing at $\mathbf{X}$ in the 3D scene, the following relationship holds [1].

$$X_R = R(X_L - T). \tag{3.19}$$

In the left cameras coordinate system the epipolar plane $\prod_e$ is covered by the vectors X_L and T. Hence the vector $X_L - T$ also belongs to this plane. Hence we have the following coplanarity condition [1],

$$(X_L - T)^T.(T \times X_L) = 0 \tag{3.20}$$

Rearranging Equation 3.19 we have

$$R^T X_R = X_L - T \tag{3.21}$$

Substituting this into Equation 3.20 yields

$$(R^T X_R)^T.T \times X_L = 0 \tag{3.22}$$

Hence this crossproduct can be expressed as matrix multiplication, from [1]

$$T \times X_L = [T]_x X_L = S X_L \tag{3.23}$$

where

$$S \equiv [T]_x = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}. \tag{3.24}$$

From 3.22 we have

$$X_R^T R S X_L = 0 \tag{3.25}$$

and

$$X_R^T E X_L = 0 \tag{3.26}$$

where **E=SR** is known as the essential matrix, which gives the relationship between the extrinsic parameters of the stereo system and epipolar constraint [66].

Since the projection vectors X_L and X_R are generally unknown, E can be used to link them to the image projections, x_L and x_R, of point X onto the left and right image planes.

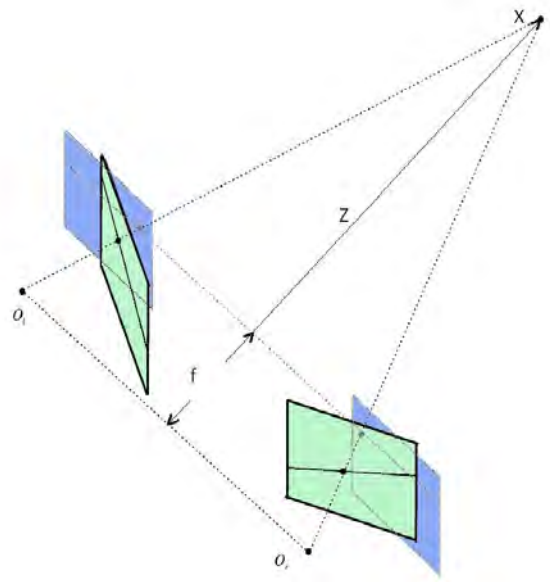


Figure 3.7: Perspective camera model - The perspective camera model for a pair of cameras.

The relationship between a 3D space point **X** and its image projection x is given by the equation of perspective projection, using similar triangles, we can show that:

$$X = \mathrm{x}\frac{Z}{f}. \tag{3.27}$$

Hence,

$$X_L = \mathrm{x}_L\frac{Z_L}{f_L} \qquad and \qquad X_R = \mathrm{x}_R\frac{Z_R}{f_R}. \tag{3.28}$$

Substituting these into Equation 3.26 and dividing through by $\frac{f_L f_R}{Z_L Z_R}$ gives.

$$\mathrm{x}_R^T E \mathrm{x}_L = 0 \tag{3.29}$$

which shows the relationship between x_L and x_R.
To relate the 3D world coordinates to the 2D image coordinates we require the cameras intrinsic parameters. These are the principal point, skew, focal length, and distortion coefficients of the camera. With M_L and M_R the left and right intrinsic parameter matrices, we can transform from world coordinates x_L and x_R to pixel coordinates $\hat{\mathrm{x}}_L$ and $\hat{\mathrm{x}}_R$ ie.

$$\hat{\mathrm{x}}_L = M_L \mathrm{x}_L \qquad and \qquad \hat{\mathrm{x}}_R = M_R \mathrm{x}_R. \tag{3.30}$$

Rearranging and substituting into Equation 3.29 yields

$$(M_R^{-1}\hat{\mathrm{x}}_R)^T E M_L^{-1}\hat{\mathrm{x}}_L = 0 \qquad or \qquad \hat{\mathrm{x}}_R^T F \hat{\mathrm{x}}_L = 0 \tag{3.31}$$

where $F = M_R^{-T} E M_L^{-1}$ is the fundamental matrix which relates pixel coordinates between the left and right images [1]. The epipolar geometry can be expressed in both the essential and fundamental matrices. However the fundamental matrix is defined in pixel coordinates and encodes the cameras intrinsic and extrinsic parameters, making it more practical to use. The essential matrix on the other hand only encodes information about the cameras rotation and translation and is only defined in terms of world coordinates [1].

3.2.3.3 Finding F From Point Correspondences

The 3 × 3 Essential and Fundamental matrices can be computed using equations 3.29 and 3.31 respectively. These equations are defined in the homogeneous coordinate system and therefore a solution can only be determined up to a certain scaling factor [1]. Cyganek *et al.* [1] show that only eight unique pairs of matched points are necessary to determine E or F. One of the most basic methods is known as the *eight-point algorithm* [1].
We can rewrite equation 3.31 as follows:

$$\sum_{i=1}^{3}\sum_{j=1}^{3} \hat{\mathrm{x}}_{Ri} F_{ij} \hat{\mathrm{x}}_{Lj} = 0 \tag{3.32}$$

Where $\hat{\mathrm{x}}_L$ and $\hat{\mathrm{x}}_R$ represent pixel coordinates in the left and right image planes. Equation 3.32 can be rewritten using a single summation as:

$$\mathbf{x}^T\mathbf{f} = r = \sum_{i=1}^{9} \mathrm{x}_i f_i = 0 \tag{3.33}$$

where x_i is the point coordinate component, r is the residual and f_i is a vector comprising of the nine elements of F, that is:

$$\mathbf{x} = [\hat{\mathrm{x}}_{L1}\hat{\mathrm{x}}_{R1}, \hat{\mathrm{x}}_{L2}\hat{\mathrm{x}}_{R1}, \hat{\mathrm{x}}_{R1}, \hat{\mathrm{x}}_{L1}\hat{\mathrm{x}}_{R2}, \hat{\mathrm{x}}_{L2}\hat{\mathrm{x}}_{R2}, \hat{\mathrm{x}}_{R2}, \hat{\mathrm{x}}_{L1}, \hat{\mathrm{x}}_{L2}, 1]^T. \tag{3.34}$$

and

$$\mathbf{f} = [F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33}]^T. \tag{3.35}$$

F can then be found using equation 3.33 and 8 unique sets of matched points. The eight point algorithm is simple, however it is particularly unstable to factors such as noise, numerical round-off errors and mismatched points. These can be overcome by normalizing the point correspondences prior to solving equation 3.33 and denormalizing afterwards [74].

In practice it is common to select more than eight matched points, $K \geq 8$, in this case a solution can be determined using the method of least-squares.

$$\min_{\|f\|=1} \|Q\mathbf{f}\|^2. \tag{3.36}$$

Where Q is a matrix containing a set of matched points in each row. The term $\|Q\mathbf{f}\|^2$ may be written as:

$$\|Q\mathbf{f}\|^2 = (Qf)^T(Qf) = f^T(Q^TQ)f. \tag{3.37}$$

The moment matrix $M = (Qf)^T(Qf)$ is a 9×9 matrix formed from Q. The authors of [75] show that a solution to equation 3.36 can be found using Lagrange-Euler multipliers and comprises of a minimal eigenvalue of matrix M. Another solution can be found using Single Value Decomposition (SVD) [75], where **F** is the column of matrix **S** corresponding to the lowest singular value in matrix **V** [1]. This method has a drawback in that factors such as noise, mismatches and discretization of matched points may cause the matrix F to not have a rank of 2.

Another approach can be taken which does not have this disadvantage. Here the matrix **F** is given as the eigenvector of the lowest eigenvalue of matrix **M**. This **F** minimizes the residuals:

$$R = \sum_{k=1}^{K} \hat{\mathrm{x}}_k. \tag{3.38}$$

Hence, **F** can be found by optimization [1]:

$$\min\{R\}. \tag{3.39}$$

Where R is,

$$R = \sum_{k=1}^{K} \hat{x_k} = \frac{f^T M f}{f^T J f}. \tag{3.40}$$

The normalization matrix J is given as J_1= diag[1, 1, . . . , 1]. To enforce the rank two constraint on F, Tsai and Huang [76] proposed that the lowest singular value should be set to 0 and then the fundamental matrix should be recalculated. This method proceeds as follows [1]:

$$F = SVD = S \begin{bmatrix} v_1 & 0 & 0 \\ 0 & v_2 & 0 \\ 0 & 0 & v_3 \end{bmatrix} D. \tag{3.41}$$

where $v_1 \geq v_2 \geq v_3 \geq 0$. Hence v_3 is set to 0 resulting in F_1 having a rank of two, as follows

$$F_1 = S \begin{bmatrix} v_1 & 0 & 0 \\ 0 & v_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} D. \tag{3.42}$$

Interestingly [1] indicates that distance between the smallest singular value of moment matrix M and zero provides method of assessing the quality of the matching points. If we set $J = J_1$ equation 3.36 and equation 3.39 become equivalent. Torr *et al.* [77] proposed selectively dividing f into f_1 =$[f_1, f_2, f_3, f_4, f_5]$ and f_2 =$[f_3, f_6, f_7, f_8, f_9]$. Thereafter f_1 is given as a solution of,

$$Df_1 = \lambda f_1. \tag{3.43}$$

Where

$$D = M_{11} - M_{12}M_{22}^{-1}M_{12}^T \qquad and \qquad M = \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^T & M_{22} \end{bmatrix}. \tag{3.44}$$

M is partitioned into M_{ij} giving,

$$f^T M f = f_1^T M_{11} f_1 + 2 f_1^T M_{12} f_2 + f_2^T M_{22} f_2. \tag{3.45}$$

Finally f_2 can be obtained from f_1, as follows [1]:

$$f_2 = -M_{22}^{-1} M_{12}^T f_1. \tag{3.46}$$

3.2.3.4 Finding F From Camera Projection Matrices

The eight point algorithm is simple and useful, however it is particularly unstable to factors such as noise, numerical round-off errors and mismatched points. These can be overcome, as described in Section 3.2.3.3, by normalizing the point correspondences prior to solving for **F** and denormalizing once a solution is found. For a more robust solution we prefer to use calibrated cameras and thus are able to estimate F using the corresponding [3×4] camera projection matrices P_L and P_R.

The equation of the ray back-projected from image point x to 3D space point **X**, is obtained by solving $\mathbf{PX} = \mathrm{x}$. From [72; 78] we have a set of parametric solutions of the form:

$$X(\lambda) = P_L^{-1}\mathrm{x} + \lambda C_L. \tag{3.47}$$

Where P^{-1} is the inverse ($PP^{-1} = I$) of P_L and C is the camera centre given by the null space of P. We consider two important points on the ray; $P^{-1}\mathrm{x}$ (at $\lambda = 0$) and the first camera centre C_L (at $\lambda = \infty$). In the right camera's image plane, these points are imaged at $P_R P_L^{-1}\mathrm{x}$ and $P_R C_R$. The epipolar line between them is then given by, $l_R = (P_R C_R) \times (P_R P_L^{-1}\mathrm{x})$.
The epipole in the second image, i.e. the projection C_L in the right image, is given by e_R. Consequently, $l_R = [e_R] \times (P_R P_L^{-1}\mathrm{x}) = F\mathrm{x}$. Where the fundamental matrix F is given by:

$$F = [e_R] \times P_R P_L^{-1}. \tag{3.48}$$

The skew symmetric matrix $[e_R]\times$ is given by:

$$[e']\times = \begin{bmatrix} 0 & -e'(3) & e'(2) \\ e'(3) & 0 & -e'(1) \\ -e'(2) & e'(1) & 0 \end{bmatrix}. \tag{3.49}$$

If the two camera centres are the same, the derivation breaks down and F is defined as the zero matrix [72]. Equation 3.48 shows how simply F is determined using the camera matrices, making this the preferred method of determining the Fundamental matrix.

3.2.3.5 3D Reconstruction

The 3D reconstruction process entails recovering 3D spatial information from corresponding image points observed from different views of the scene. The type of reconstruction possible depends primarily on the availability of camera parameters in the stereo camera set-up. Hence the availability

Table 3.1: Available calibration parameters and possible 3D reconstruction [1].

Available Calibration Data	Possible 3D Space Reconstruction
1.Intrinsic and extrinsic parameters. of the camera set-up	Result in 3D Euclidean coordinates (Precise reconstruction, known as triangulation).
2.Only intrinsic parameters.	Reconstruction up to a certain scaling factor.
3.Extrinsic and intrinsic data are not available.	Reconstruction up to a certain projective transformation.

of calibration data determines the degree of 3D reconstruction that is possible. Table 3.1 shows the type of calibration information available and the reconstruction that is possible.
Since we want an accurate reconstruction, we will focus on the technique of Triangulation. The Triangulation process aims to find the 3D location of a scene point $\mathbf{P}$ from corresponding image points p_r and p_l, in its visible images, and the camera's internal and external parameters. The scene point $\mathbf{P}$ is given as the point of intersection of the two rays emanating from the camera centres through the image points p_r and p_l (See figure 3.8).

The process seems fairly straight forward, however in practice, the two rays may not intersect due to imperfections in the camera calibration data, discrete image locations and the possibility of inaccurately estimating the positions of matched points p_r and p_l. A solution to this problem is found by finding an approximate crossing point $\mathbf{P_E}$ that lies a minimal distance from both rays simultaneously [1; 66].

$\mathbf{P_E}$ can be estimated from:

$$ap_l + c\mathbf{J} = \mathbf{T} + b\mathbf{R^T}p_r. \tag{3.50}$$

where a,b,c $\in \Re$ and $\mathbf{R}$ is the rotation and $\mathbf{T}$ the translation matrices of the stereoscopic system. The ray ap_l passes through the optical centre of the left camera $\mathbf{O_l}$ for a = 0, as well as image point p_1 for a = 1. Similarly, $\mathbf{T}$ + b$\mathbf{R}^T p_r$ is an equation of the ray through $\mathbf{O_r}$ and p_r. The vector $\mathbf{J}$ is constrained to be orthogonal to the two rays. Hence we have [1]:

$$\mathbf{J} = p_l \times \mathbf{R^T}p_r. \tag{3.51}$$

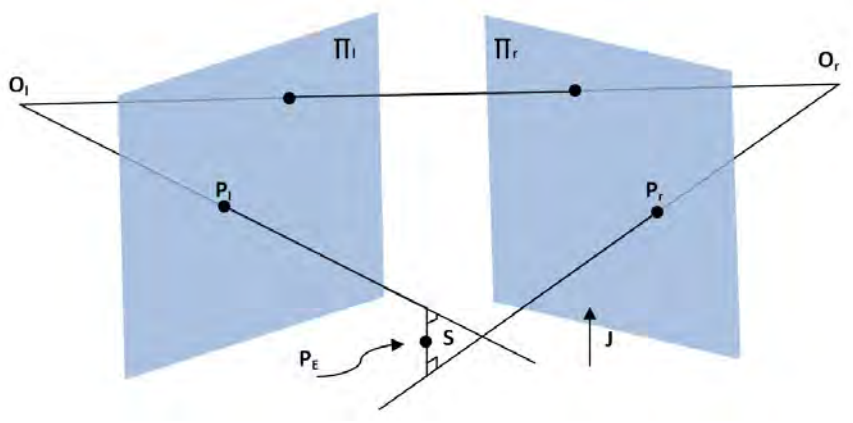


Figure 3.8: The triangulation process - Triangulation of 3D space point **P** from image points p_r and p_l. The 3D space point **P** is given as the approximate ray crossing point $\mathbf{P_E}$

From these two equations we have [66]:

$$ap_l + c(p_l \times \mathbf{R^T} p_r) = \mathbf{T} + b\mathbf{R^T} p_r. \tag{3.52}$$

The endpoints of S, the line joining the two rays, can be determined by solving equation 3.52. These endpoints are ap_l and $\mathbf{T}$ + b$\mathbf{R}^T p_r$ respectively. Once the endpoints are found, the triangulated point $\mathbf{P}_E$ is given as the midpoint of line segment S [1; 66]. The $\mathbf{R}$ and $\mathbf{T}$ matrices of the stereoscopic system can be determined from the left and right camera extrinsic parameters ($\mathbf{R_l}$, $\mathbf{R_r}$, $\mathbf{T_l}$ and $\mathbf{T_r}$) using the following relationships:

$$\mathbf{R} = \mathbf{R_r}\mathbf{R_l^T}. \tag{3.53}$$

$$\mathbf{T} = \mathbf{T_l} - \mathbf{R^T}\mathbf{T_r}. \tag{3.54}$$

3.2.3.6 Patch Optimisation

The patch optimisation algorithm aims to optimise the position and orientation of a patch p with respect to its visible images. This is achieved by minimizing the global photometric discrepancy function g*(p).

Optimization is performed using a modified approach of the algorithm first described in [58]. After initializing the patch parameters (See Section 3.2.1), a new estimate of surface orientation is made,

and the patch centre and normal vector are optimized such that g*(p) is minimized. To ease the complexity of the computations, the patch centre with 3 degrees of freedom is constrained to move only along the line joining the $\mathbf{c}(p)$ and the optical centre of R(p). This reduces the degree of freedom to one (depth). The patch normal is parametrized by Euler angles yaw and pitch. The rotation matrices for the yaw and pitch are given below [79].

$$R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.55}$$

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}. \tag{3.56}$$

This yields an optimization problem with three degrees of freedom (depth, yaw and pitch) which can be solved using a variety of gradient based methods. To enhance the performance of the Optimization algorithm an improved estimate of the orientation of the patch is made. Initially the patch is oriented such that it is parallel to the reference image. During optimization, this initial staring position may cause the algorithm to incorrectly converge to a local minimum. To avoid such situations the initial orientation of the patch is chosen such that its photometric discrepancy with respect to its visible images is minimized (see Section 5.2.2.6 for details).

In practice we use a variant of the conjugate gradient method to optimize g*(p) with respect to the three free variables. The conjugate gradient method has the advantage of faster convergence than the method of steepest decent whilst avoiding the expensive evaluation, storage and inversion of the Hessian. A description of the algorithm is given below [80–82].

1. Given any arbitrary starting point $x_0 \in dom\ f$ set d_0 = -g_0 ,where g is the gradient vector of the function at f(x)
2. Test a criterion for stopping the iterations $\|g_k\| \leq \epsilon$
3. Using the search direction d_k, update the variables $x_{k+1} = x_k + a_k d_k$, where the step-length a_k is found by minimizing f($x_k + a_k d_k$) by the strong Wolfe conditions:

$f(x_k + a_k d_k) \leq f(x_k) + c_1 a_k d_k^T \nabla\ f(x_k)$

$d_k^T \nabla\ f(x_k + a_k d_k) \geq c_2 d_k^T \nabla \mathrm{f}(\mathrm{x}_k)$

4. Calculate the new search direction d_{k+1}, using: $d_{k+1} = -g_{k+1} + b_k d_k$. The term b_k depends on the update formula chosen. We use Fletcher-Reeves :
   $\mathbf{b_k} = \frac{(g_{k+1})^T (g_{k+1})}{(g_k)^T (g_k)}$
5. Finally Compute $a_k = a_{k-1} \|d_{k-1}\| / \|d_k\|$ set k=k+1 and return to step two.

3.3 Feature Expansion

The set of patches generated by the initial feature matching stage may be sparsely distributed over the object's surface. The patch expansion stage is used to increase the density of the reconstruction by taking patches generated from the initial feature matching stage and reconstructing new patches in nearby vacant locations.

Expansion is carried out using the following procedure introduced in [58]. For each patch p a set of neighbouring empty cells is identified. Thereafter the patch expansion procedure is carried out for each one.

3.3.1 Identifying Cells For Expansion

For each patch p that is to be expanded, we first initialize the set of neighbouring image cells $\mathbf{C}(\mathrm{p})$ by collecting image cells surrounding the projection of p in each visible image $\mathbf{V}(\mathrm{p})$.

$$C(p) = \left\{ C_i(x', y') \,|p \in Q_i(x, y)\ , \left|x - x'\right| + \left|y - y'\right| = 1 \right\}. \tag{3.57}$$

Cells where the expansion process is deemed unnecessary are removed by the following two conditions.

1. Expansion is unnecessary if a patch has been reconstructed for one of the cells in $\mathbf{C}(\mathrm{p})$. That is, if $C_i(x', y') \in \mathbf{C}(\mathrm{p})$ includes a neighbouring patch of p, $C_i(x', y')$ is then removed from the set $\mathbf{C}(\mathrm{p})$. Patches are said to be neighbours if the following condition holds:

$$\left|(c(p) - c(p')) \cdot n(p)\right| + \left|(c(p) - c(p')) \cdot n(p')\right| < 2\rho_1. \tag{3.58}$$

Where the distance ρ_1 is determined as the image displacement of beta pixels, in reference image $\mathbf{R}(\mathrm{p})$, projected to the depth of the centres $\mathbf{c}(\mathrm{p})$ and $\mathbf{c}(\mathrm{p}')$ [58].

2. A patch should not be expanded if we encounter a depth discontinuity when viewing the patches from a certain camera view. In practice it is difficult to judge the presence of depth discontinuities without reconstructing the surface. This situation is handled by the following condition.

   If the filtered image cell $Q_i^*(x', y')$ contains a patch, then the expansion process is unnecessary and $C_i(x', y')$ is removed from the set $\mathbf{C}(\mathrm{p})$

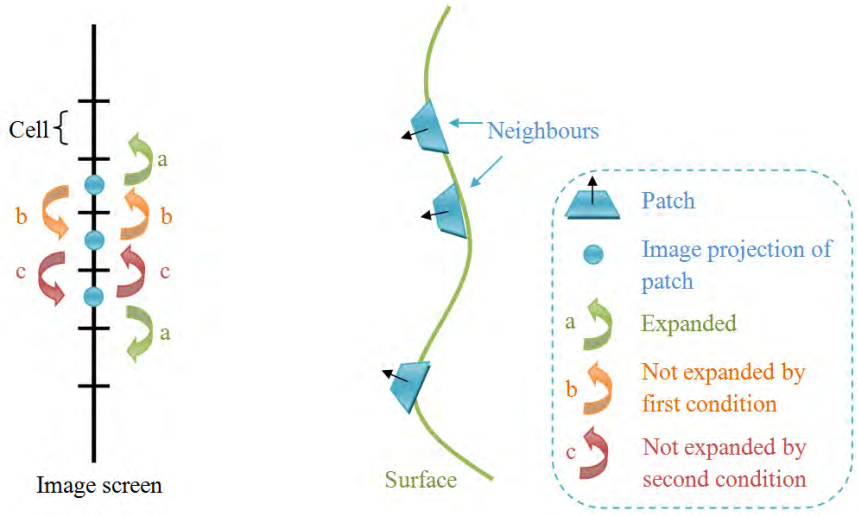


Figure 3.9: Patch Expansion technique - The patch expansion procedure generates patches in nearby empty cells. A patch is not expanded (b) if there is a neighbouring patch reconstructed there, or (c) if there is a depth discontinuity between the patches.

3.3.2 Expansion Procedure

Once the set $\mathbf{C}(\mathrm{p})$ is populated with the appropriate cells for expansion, the expansion process begins and generates a new patch p' using the following procedure.

We first initialize the patch candidate p' by setting $n(\mathrm{p}') = n(\mathrm{p})$, $R(\mathrm{p}') = R(\mathrm{p})$ and $V(\mathrm{p}') = V(\mathrm{p})$. The centre $c(\mathrm{p}')$ is then determined as the point where the ray from the optical centre through $C_i(x', y')$ intersects the plane formed by patch p. This step is crucial to insuring that p' and p are

neighbours in image I_i.

After computing $V^*(\mathrm{p}')$ from $\mathbf{V}(\mathrm{p})$, equation 3.2, We refine the patch parameters $c(\mathrm{p}')$ and $n(\mathrm{p}')$ using the patch optimization algorithm described in section 3.2.3.6. The optimization procedure ensures that $c(\mathrm{p}')$ projects into $C_i(x', y')$ by constraining $c(\mathrm{p}')$ to motions only along the ray from the optical centre to p′.

After the optimization, we add to $V(\mathrm{p}')$ visibility information determined from a cell depth-map test. The depth of each cell is computed by taking the average depth for all the patches that project into it. A patch is then visible in image I_i if the depth of the patch is less than the average depth for the cell plus ρ_1. This metric ensures that only the truly visible images of p are selected. $V^*(\mathrm{p}')$ is then updated according to Equation 3.2. Adding the depth-map visibility information to $V(\mathrm{p}')$ plays an important role in the subsequent filtering stages. The set of reconstructed patches may contain outliers, hence making the associated visibility information inconsistent amongst these patches. This fact is exploited in the patch filtering stage in order to reject outliers. Finally a patch is said to be successfully generated if $|V^*(\mathrm{p}')| > \gamma$. The patch p′ is then stored into the set of patches P and in image cells $Q_i(x, y)$ and $Q_i^*(x, y)$. The overall algorithm description is given in Algorithm 2

To significantly increase the density of the reconstruction, the Expansion algorithm is iterated three times. Within the Initial Feature Matching stage, the parameter α is set to 0.6 before and 0.3 after optimization. In the Patch Expansion stage α relaxed by 0.2 after each Expansion and Filtering iteration [58]. This allows reconstruction of challenging regions in the later iterations.

3.4 Patch Filtering Algorithm

The set of patches generated by the Initial Feature Matching and Feature Expansion stages may contain outliers. These outliers are detected and removed using a Patch Filtering technique presented in [58]. The following three filters are used to discard false matches near the surface of the model.

The first filter uses visibility consistency to filter away outlying patches. Patches p' that are inconsistent with the current visibility information for p are given by the set $U(p)$. That is, p and p' are not neighbours (equation 3.58), however they were incorrectly stored in the same cell for

Input: Set of initial feature matches P.
Output: Expanded set of reconstructed patches.

1. For each patch p in P
2. For each Image cell $C_i(x, y)$ containing p
   Collect the set $\mathbf{C}$(p) of candidate image cells for expansion.
3. For each cell $C_i(x', y')$ in $\mathbf{C}$(p)
   Set $n(\mathrm{p}') = n(\mathrm{p})$, $R(\mathrm{p}') = R(p)$ and $V(\mathrm{p}') = V(p)$
   Compute $c(\mathrm{p}')$
   Update $V^*(\mathrm{p}')$
   Refine $c(\mathrm{p}')$, $n(\mathrm{p}')$
   Add visible images from depth map test to $V(\mathrm{p}')$
   Update $V^*(\mathrm{p}')$
4. If $|V^*(\mathrm{p}')| > \gamma$
   Add p′ to the corresponding $Q_i(x, y)$ and $Q_i^*(x, y))$
   Add p′ to P

Algorithm 2: Patch Expansion Algorithm [58]

one of the images where p is visible. The patch p is then filtered out if following inequality holds [58]:

$$|V^*(p)|\,(1 - g^*(p)) < \sum_{p_i \in U(p)} 1 - g^*(p_i). \tag{3.59}$$

When p is an outlier, the number of true visible images $|V^*(p)|$ as well as the photometric discrepancy term $(1 - g^*(p))$ are likely to be small, and hence p is expected to be removed.

The second filter enforces a stricter form of visibility consistency [58]. The visibility of each patch p is determined using the cell depth-map test (See section 3.3.2). p is removed as an outlier if the number of visible images is smaller than γ.

The final filtering stage employs a subtle form of regularization using the following procedure [58]. For all the visible images $\mathbf{V}$(p) of patch p we collect the patches stored in the same cell as well as and neighbouring cells. Patch p is then filtered away if the proportion of neighbours (equation

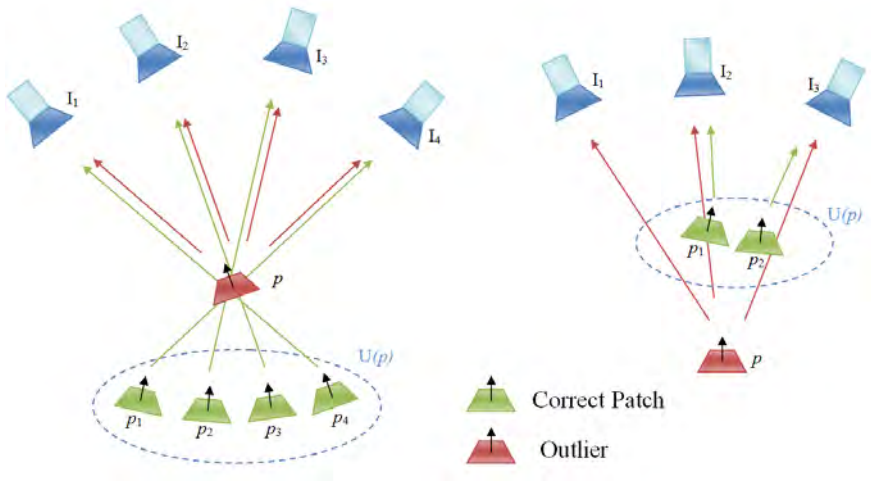


Figure 3.10: Patch Filtering technique - The first two filters employ visibility constraints to remove outliers. The third filter uses a surface regularization measure. For the first filter, the relationship between p_i and I_i i.e. $V(p_i)$ is represented by an arrow connecting them. $U(p)$ represents the patch set that is inconsistent with the current visibility information [58].

3.58) of p is below 0.25.

3.5 Summary

The Patch-Based Multi-View stereo reconstruction algorithm employed in this dissertation is both flexible and robust. Reconstruction begins by identifying strong feature points in the input images using the difference of Gaussian and Harris feature detectors. Image features such as corners edges and blobs are matched across the input images resulting in a sparse surface patch model. An expansion and subsequent filtering stage is then used to increase the density and remove outliers in the model. The resulting patch model is a dense collection of patches covering the surface of the 3D model.

A patch is defined as a small rectangular region, typically 5×5 pixels, tangent to the surface of the 3D model and has 3D properties described by its centre c(p) and normal vector n(p). To determine if a reconstructed patch lies on the surface of the model a photometric discrepancy function is employed. This function is evaluated by projecting the patch centre into each of its visible images (initially determined using geometric constraints) and computing the normalised cross correlation score between the windowed region in R(p) and each visible image. Images with scores below a

certain threshold are said to be truly visible images of p. A measure of the average discrepancy score, known as the global photometric discrepancy function, is used in the optimisation stage to refine the 3D properties of p such that its photometric score with its visible images is minimised. Inherently, the patch model does not have any connectivity information; to help with tasks such as identifying neighbouring patches and performing regularization an image model is employed. Each image has an associated image model formed by partitioning the image space into a uniform grid of cells. Each cell is then responsible for storing all the patches that project into it.

The patch reconstruction algorithm takes in features identified by the feature detectors and outputs a sparse 3D patch model. For each image feature identified in image I_i a set of features that lie within two pixels of the corresponding epipolar lines is collected in all other images. To ensure that the best possible match is selected first, the collected features are sorted in increasing order of photometric discrepancy with image I_i. Then for each feature a candidate patch is generated using the following procedure. The patch centre c(p) is determined by triangulation between the matching features. The patch orientation n(p) is initially set to be pointing towards the reference image R(p) = I_i. Initial visibility of the patch is determined using geometric constraints, and is thereafter improved using the photometric discrepancy function. Then, both geometric properties of the patch are optimised using the conjugate gradient method. The visibility information is re-computed using the new patch parameters and the patch is said to lie on the surface if the number of visible images is above a certain threshold. Lastly any features that lie in the same cell as p are removed. This avoids generating another patch in the same location.

To increase the density of the reconstruction, the sparse set of patches from the initial feature matching stage are expanded by generating new patches in nearby empty locations. The first step involves identifying vacant neighbouring cells that are fit for expansion. Cells are not selected if they contain a patch that is a neighbour of p or if there is a depth discontinuity when evaluated from a certain view. Once the set of cells have been identified the expansion procedure attempts to generate a patch by initialising the patch properties, performing optimisation and evaluating the true visibility of the patch. The patch is then constructed if the visibility is above a certain threshold.

The final step in the patch reconstruction algorithm involves filtering the patch model to remove outliers. Three filters are used to remove outliers that lie near the surface of the model. The first

two filters rely on visibility consistency information to remove outliers whilst the third filter uses a weak form of regularization and removes a patch if its proportion of neighbours is below 0.25. To increase the density of the reconstruction and effectively remove outliers, the expansion and filtering stages are iterated three times. After each iteration of the expansion stage, the parameter α (correlation matching threshold) is relaxed (increased) by 0.2 to allow better reconstruction of challenging regions.

Chapter 4

Polygonal Mesh Reconstruction

4.1 Introduction

The output produced by the Patch Reconstruction stage is a dense set of unconnected patch structures. Oriented point models are growing in popularity amongst the computer vision and computer graphics communities due to the ease with which it can represent arbitrary complex topologies [83]. However, its unconnected point representation makes performing image based modelling tasks such as smoothing or producing a closed surface, difficult. Consequently, it is beneficial to turn the reconstructed patches into a surface mesh model with which these tasks can be easily accomplished.

The Polygonal Mesh Reconstruction stage is designed to achieve that result and is split into two processes. The first is a mesh initialization stage, where one of two mesh initialization methods are preformed depending on the availability of segmentation information. The first approach uses Poisson Surface Reconstruction (PSR) to initialize a mesh from the reconstructed patches. This technique is used for datasets where segmentation information is unavailable. The second initialization method is called Iterative Snapping and is used for object datasets where a Visual Hull can be reconstructed from the segmentation information. The mesh model is then initialized at the boundary of the visual hull and iteratively refined using an energy minimization approach.

The second process is a Mesh Refinement Stage, where each vertex in the mesh is optimised using two vertex energy functions. The first is a geometric smoothness term and the second is a patch reconstruction consistency term. The following subsections detail the mesh reconstruction algorithm.

4.2 Poisson Surface Reconstruction

The Poisson Surface Reconstruction technique is used when its difficult or impractical to extract segmentation information from the object or scene. This situation often occurs for scene datasets where there may not be enough views of the scene to create an approximate bounding volume. The PSR technique takes the oriented point model from the Patch Reconstruction stage and reconstructs a water tight (closed) surface mesh model. In practice the implementation developed by Kazhdan *et al.* is used [59; 84].

The software outputs a closed mesh model, where the resolution of the mesh and hence the size of the triangles is determined by the density of the reconstructed points. Hence triangles in the mesh are large for regions with a small number of reconstructed patches. To remove these incorrect parts of the mesh, we simply reject triangles that have an average edge length larger than six times the average for the entire mesh.

4.3 Iterative Snapping

The PSR software can produce high quality meshes, however it cannot make use of segmentation information often available for object datasets. This second approach to mesh initialization uses segmentation information to compute a visual hull of the object. A mesh model is then initialized at the boundary of the visual hull and is subsequently refined by enforcing consistency with the reconstructed patches. The mesh deformation algorithm is based on the work presented in [48; 58].

In essence, the gradient descent method is used to optimize the position of all the vertices in the mesh model. Optimization is performed by minimizing the sum of two energy functions [58]. The first is a geometric smoothness energy term, $E_s(v_i)$, between a vertex v_i and its neighbours, and is given by [58]:

$$E_s(v_i) = \left|-\zeta_1 \Delta v_i + \zeta_2 \Delta^2 v_i\right|^2 / \tau^2 \tag{4.1}$$

Where v_i is the 3D position of vertex i. Δ is the discrete Laplacian operator [85–87] relative to the local parameterization of the tangent plane in terms of v_i and its neighbours. τ represents the average edge length of the mesh model. The weights ζ_1=0.6 and ζ_2=0.4 control the relative depth

and orientation variations in the neighbourhood of v_i [58].

The second function $E_p(v_i)$ is the photometric discrepancy term that imposes consistency with the reconstructed patches and is given by [58]:

$$E_p(v_i) = max\left(-\beta_1, min\left(\beta_1, \frac{d(v_i) \cdot n(v_i)}{\tau}\right)\right)^2 \tag{4.2}$$

Where $n(v_i)$ is the outward unit normal of the surface mesh at vertex v_i. The term $d(v_i)$ is evaluated by taking the signed distance between the vertex v_i and the reconstructed patches in the direction of $n(v_i)$. See figure 4.1. More concretely, $d(v_i)$ is computed as follows. For vertex v_i, we extend a line from v_i, along the direction of $-n(v_i)$, towards the reconstructed patches. Then for each patch whose normal is compatible ($n(p) \cdot n(v_i) > 0$) with that of v_i we collect the set Π (at most 10) closest patches to this line. The distance $d(v_i)$ is then computed as the sum of the weighted distance from v_i to the centres of the patches in $\Pi(v_i)$ along $n(v_i)$, from [58] we have:

$$d(v_i) = \sum_{p \in \Pi(v_i)} w(p)\left[n(v_i) \cdot (c(p) - v_i)\right] \tag{4.3}$$

Where the normalized weights $w(p)$, are Gaussian functions of the distance between $c(p)$ and the end of the line [60]. The standard deviation ρ is defined as the distance corresponding to an image displacement of $\beta = 2$ pixels at the depth of v_i.

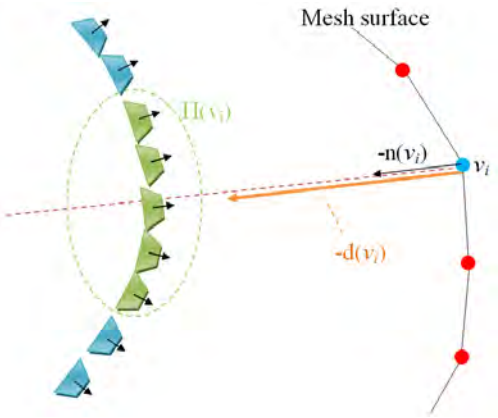


Figure 4.1: Iterative snapping algorithm - The reconstruction consistency term $E_p(v_i)$ is evaluated by determining the signed distance $d(v_i)$. The term $d(v_i)$ is in turn evaluated by computing the distance between v_i and its $\Pi(v_i)$ closest patches [58].

Note that the maximum strength of the consistency term $E_p(v_i)$ is bounded by $|\beta_1|$, which is set to 0.1 times the average edge length of the mesh model. This helps to ensure the stability of the deformation process [48]. In practice we use the following update rule for the gradient decent step $\mathbf{v} \leftarrow \mathbf{v} + \delta\mathbf{v}$, where

$$\delta\mathbf{v} = max\left(-\beta_1, min\left(\beta_1, d(v_i)\right)\right) n(v_i) + \left(-\zeta_1 \Delta v_i + \zeta_2 \Delta^2 v_i\right) \tag{4.4}$$

$\delta\mathbf{v}$ is obtained by differentiating $E_s(v_i)$ and $E_p(v_i)$. The vertex optimization process is iterated until convergence, whilst applying a re-meshing procedure (edge split, contract and swap) [14] once every five gradient decent steps. This has the effect of avoiding self-intersections [1] within the mesh and ensuring that the edge lengths become approximately uniform. Thereafter the resolution of the mesh is increased and the process is repeated until the required resolution is attained. In practice we iterate until the projection of the edges (τ) become approximately $\beta = 2$ pixels in length [58].

4.4 Mesh Refinement

The Mesh Refinement algorithm aims to optimize the position of each vertex in the mesh according to two vertex energy functions. The first is a geometric smoothness energy $E_s(v_i)$ defined in Equation 4.1. The second is a photometric consistency energy derived from the reconstructed patches. Mesh Refinement is then performed via energy minimization using a conjugate gradient method.

The photometric discrepancy function $E_p(v_i)$ is computed using the following two steps. Firstly the depth and orientation of the surface at each vertex is estimated using the patch optimization procedure. Optimization is applied independently using each pair of visible images, resulting in a set of reconstructed patches near the vertex. Secondly the estimated depth and orientation information is combined to compute the new photometric discrepancy energy function [58].

More specifically, let the visibility of vertex v_i be given by the set of images $V(v_i)$. In practice $V(v_i)$ is computed using a depth map test with the current mesh model. Thereafter, for each pair of images $(I_j, I_k) \in V(v_i)$, we set $c(p) \leftarrow c(v_i)$ and $n(p) \leftarrow n(v_i)$ and use the patch optimization procedure to minimize the photometric discrepancy function $h(p, I_j, I_k)$. This has the effect of

[1] self-intersections may arise as a result of large displacements in vertex positions without re-evaluating vertex connectivity.

generating a set of patches $P(v_i)$ on the tangent plane at vertex v_i.

The photometric discrepancy energy is computed as the sum of one minus (scaled) Gaussian function of the distance between each patch in $P(v_i)$ and the vertex v_i as follows:

$$E'_p(v_i) = \zeta_3 \sum_{p \in P(v_i)} 1 - exp\left(-\left(\frac{d'(v_i,p)}{\tau/4}\right)^2\right) \tag{4.5}$$

$$d'(v_i,p) = n(p) \cdot (c(p) - v_i) \tag{4.6}$$

Where $d'(v_i,p)$ is given as the (signed) distance between the patch p and the vertex v_i in the direction of patch normal $n(p)$ [58]. The term ζ_3 is the linear combination weight. Figure 4.2 illustrates this concept.

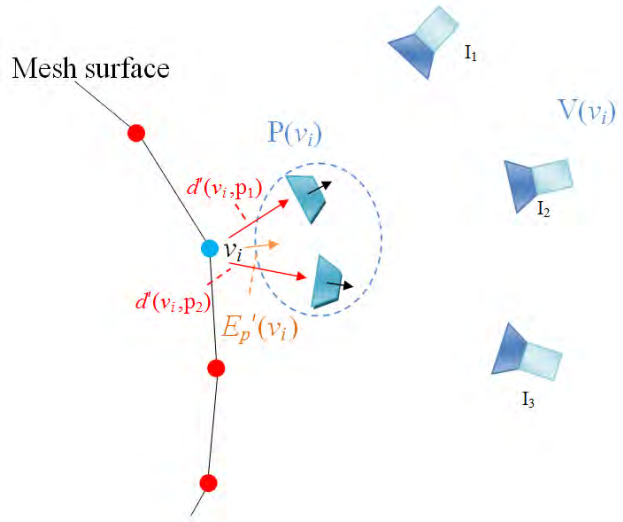


Figure 4.2: Mesh refinement process - In evaluating the patch reconstruction consistency term $E'_p(v_i)$, the distance term $d'(v_i,p)$ is evaluated by computing the signed distance between each patch p $\in P(\mathrm{v}_i)$ and vertex v_i [58].

Note that the set of patches $P(v_i)$ are computed only once during the pre-processing stage. Whereas the photometric discrepancy energy (Equation 4.5) is evaluated many times during the energy minimization procedure. The photometric discrepancy energy is formulated using the idea of occlusion robust photo-consistency proposed in [40]. Where first multiple estimates of depth and orientation from pairs of visible images are obtained, instead of using all the visible images at once to obtain

a single estimate. Thereafter, these multiple estimates are combined with Gaussian functions that are robust to outliers [58].

4.5 Summary

The polygonal mesh reconstruction stage is required to convert the orientated patch model into a closed polygonal mesh surface. Orientated point models can easily represent arbitrary complex topologies and is beneficial for the initial modelling of an object or scene. However for image based modelling applications, it is more advantageous to use a polygonal mesh surface representation due to the simplicity of storage and rendering. The mesh reconstruction stage is split into two main processes. The first is a mesh initialisation stage and the second is a mesh refinement stage.

Mesh initialisation is preformed using one of two methods depending on the availability of segmentation information. If segmentation information is unavailable, typically for scene datasets, the Poisson surface reconstruction technique is used. The PSR algorithm is able to directly convert an orientated point model into a surface mesh model; where the size of the mesh triangles depends on the density of the reconstructed points. If segmentation information can be accurately extracted from the input images, the iterative snapping algorithm is used. A visual hull model is reconstructed using the segmentation information and the surface mesh is initialised at the boundary of the visual hull. To further improve the initialisation, the position of each vertex in the mesh is optimised using geometric smoothness and patch reconstruction consistency functions.

The mesh refinement stage further refines the initialised mesh model. For each vertex in the mesh a set of visible images is computed using a depth map test. Thereafter the surface orientation at the vertex is estimated by generating a patch using the patch optimisation algorithm for each pair of visible images. This results in a set of patches on the tangent plane at the vertex. The photometric discrepancy energy function used in the optimisation is then computed as one minus the scaled Gaussian function of the distance between each reconstructed patch and the vertex.

Chapter 5

Implementation and Results

5.1 Introduction

This chapter details the implementation and evaluation of the 3D reconstruction system. Reconstruction is split up into two phases. The first is a Patch Reconstruction process, where an initial unconnected surface is extracted from pixel correspondences in multiple images. The second is a mesh based refinement stage, where a mesh is iteratively refined towards the reconstructed patches using smoothness and photometric consistency constraints. An overview of the reconstruction pipeline is presented in Section 5.1.1, and Section 5.1.2 contains a brief description of the resources used. Thereafter Section 5.2 provides implementation details and intermediate results from the individual stages within the reconstruction pipeline. Finally Section 5.3 presents the final results of the reconstruction system together with a quantitative evaluation of system performance in terms of reconstruction accuracy and completeness.

5.1.1 Reconstruction Pipeline

The reconstruction process can be viewed in terms of a pipeline, where the results of each stage feed into the subsequent stages. Figure 5.1 illustrates the reconstruction pipeline. The input images and camera calibration parameters can be seen as raw data. The data is then transformed into a sparse 3D point model by the feature matching process. The expansion and filtering stages increase the density and quality of the point model. The point model is then converted into a mesh model and refined before being texture mapped, producing a realistic 3D model.

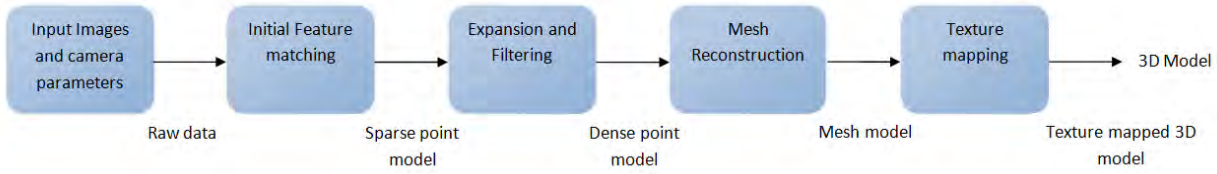


Figure 5.1: Reconstruction Pipeline - The figure shows how data flows from one process to the next in the pipeline.

5.1.2 Resources

The MATLAB development environment was chosen for testing and implementation of the algorithms. This aids in visualisation and rapid development of the abstract concepts presented in this dissertation.

5.2 Intermediate Results

The 3D Reconstruction process is divided into two main stages. The first is a patch reconstruction stage consisting of: feature detection, initial feature matching, feature expansion, and patch filtering. The second stage is the polygonal mesh refinement stage, consisting of a mesh initialization and final mesh refinement stage. The following subsections provide implementation details and intermediate results for each stage in the reconstruction system.

5.2.1 Feature Detection Results

Feature detection is the process of identifying interest points such as edges, corners, and blobs in images. These image features are often robust to changes in rotation, illumination and image noise. Hence, Feature Detection forms the start of many Multi-view Stereo Reconstruction algorithms that employ a region based matching technique. In our approach we use two types of feature detectors. The first is the Harris Feature Detector, which is a robust corner and edge detecting algorithm introduced in Section 3.2.2.1. The second is the difference of Gaussian edge and blob detector, introduced in Section 3.2.2.2, which is capable of identifying both stable edge features as well as complementary blob features. The feature matching stage uses the combination of features identified by both feature detectors.

5.2.1.1 Harris Feature Detector

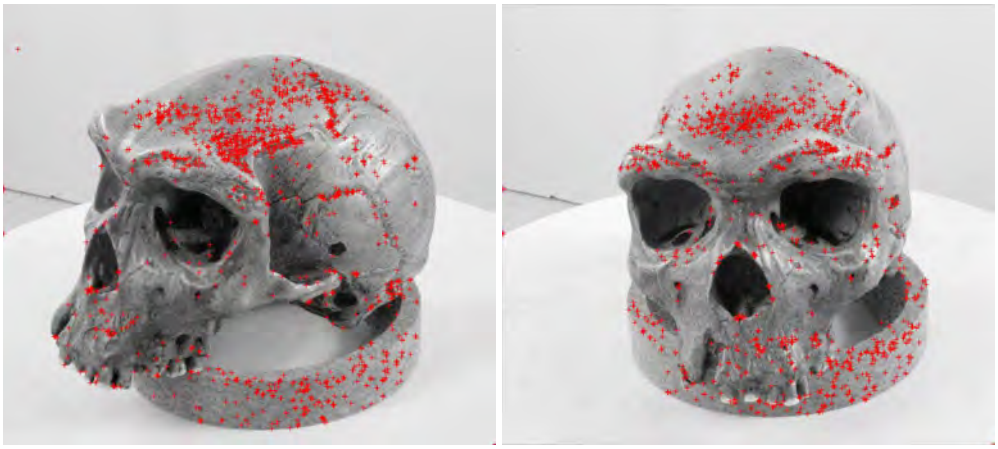
The Harris Feature Detector was implemented according to the procedure described in Section 3.2.2.1. Briefly, feature detection is achieved by first smoothing the input image I using a Gaussian filter with $\sigma = 1$ (determined experimentally) to eliminate noise, thereafter computing image derivatives A, B and C, using the convolution operator. From these, the measure of "cornerness" $\mathbf{H}$ is given by equation 3.11, and the actual corner points in the image are determined as the local maxima in $\mathbf{H}(x, y)$ that exceed a predetermined threshold.

Initial testing of the algorithm was performed on the "skull" dataset [88] which contains 24 images with a resolution of 3.2 to 3.6 Mega pixels. This image dataset is particularly good for testing the algorithm as it contains regions with strong highlights as well as large rotations between the views. Figure 5.2 shows a sample input image form the "skull" dataset.

Figure 5.2: Input image - An image from the skull dataset

Figure 5.3 shows the detected corner features for two different viewing angles of the skull, approximately 45 ° apart. For illustration purposes, the threshold has been set relatively high. Inspection of the detected features in Figure 5.3a and 5.3b indicate that similar features are found between the two views. From these results we can see that the implementation of the Harris corner detector is particularly robust to rotation and variances in illumination.

The Harris detector returns all features points above a certain threshold. Hence many points are returned for regions where the features are prominent. In order to achieve uniform coverage of the projected surface, a uniform grid of $\beta_2 \times \beta_3$ pixels is overlaid onto each image, after feature



(a) **(b)**

Figure 5.3: Harris Features - Comparison between features detected from different observation points. For illustration purposes, the threshold has been set relatively high.

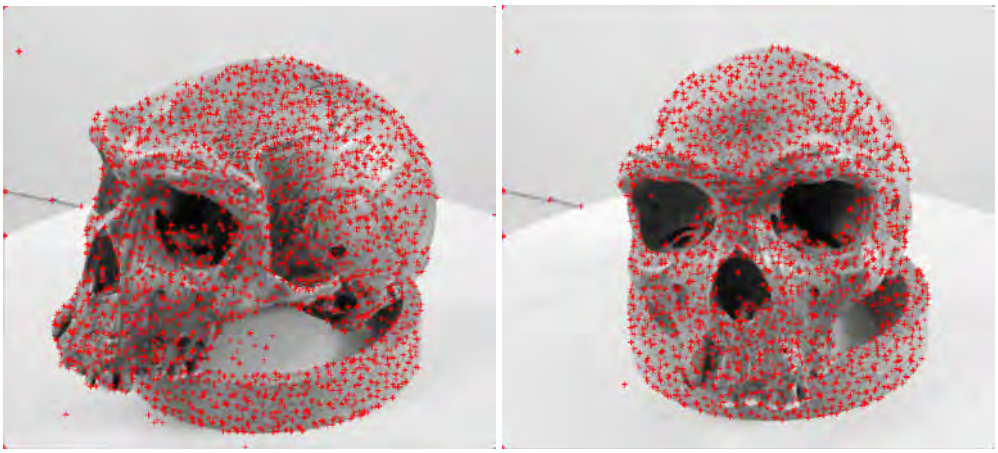
detection, and the η strongest responses above a certain threshold for each block are preserved. Figure 5.4 illustrates the result of this operation. The variables β_2, β_3, threshold and η control the number and density of the detected features. These variables can be tuned to achieve a suitable number of detected features.

5.2.1.2 Difference of Gaussian Feature Detector

The difference of Gaussian Feature Detector detects edge and blob features. Feature detection is achieved by convolving the original image with two Gaussian kernels of different standard deviations. These are then subtracted producing the difference of Gaussian.

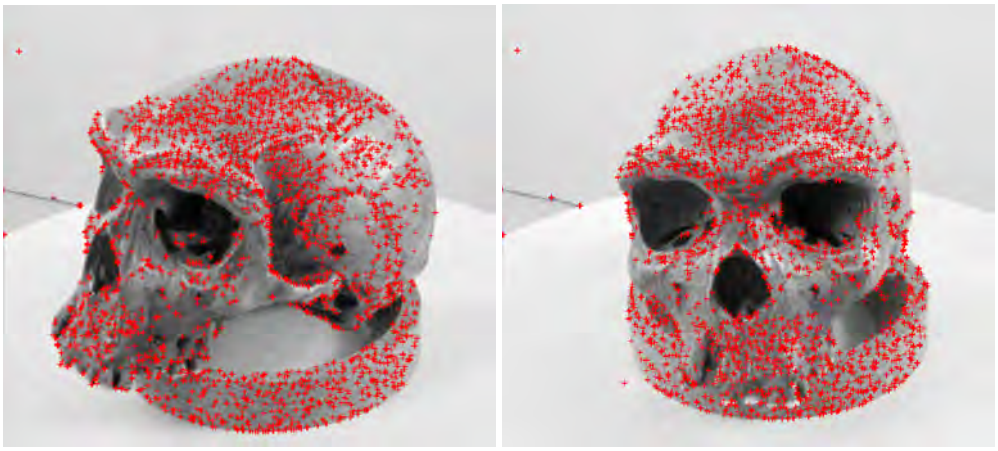
The Algorithm was implemented in Matlab using built-in functions from the Image Processing Toolbox. The function "fspecial" is used to produce the two Gaussian kernels with different standard deviations. The built-in function "imfilter" is then used to perform the convolution.

From Figure 5.5a and 5.5b we can see that the detector performs well and is relatively robust to rotations and variances in illumination. As with the Harris Feature Detector, we overlay a uniform grid of $\beta_1 \times \beta_2$ pixels onto each image after feature detection and return the η strongest responses in each block.



(a) **(b)**

Figure 5.4: Uniformly selected Harris feature points - A uniform distribution of feature points is obtained by overlaying a $\beta_1 \times \beta_2$ pixel grid over the detected features in image I and selecting the η strongest responses above a certain threshold in each block. Note that the threshold was reduced in order to allow detection in less prominent regions.



(a) **(b)**

Figure 5.5: Uniformly selected DoG feature points - A uniform distribution of feature points is obtained by overlaying a $\beta_1 \times \beta_2$ pixel grid over the detected features in image I and selecting the η strongest responses in each block. Note that the threshold was reduced in order to allow detection in less prominent regions.

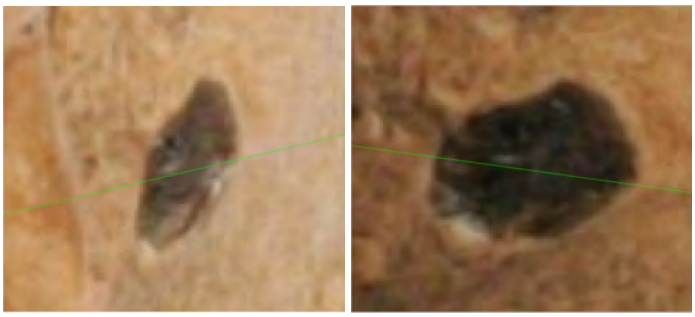
5.2.2 Initial Feature Matching

5.2.2.1 Epipolar lines

A set of Epipolar lines plotted onto the input images can be used to visually assess the accuracy of the camera parameters as well as epipolar geometry between the different views. Following from the epipolar constraint, Section 3.2.3.1, corresponding epipolar lines between the left and right views should pass through the same feature points. To assess the epipolar geometry of the "skull" dataset, a set of epipolar lines for two views are shown in Figure 5.6.

(a) (b)

Figure 5.6: Epipolar lines between two images of skull dataset - Epipolar lines of the same colour should pass through the same image feature points.



(a) (b)

Figure 5.7: Magnified views - The camera parameters can be visually assessed by examining the accuracy of the epipolar geometry.

Lines through the epipole in Figure 5.6a and points every step(100) pixels on the vertical axis are drawn in Figure 5.6a. The corresponding lines in Figure 5.6b are then found by estimating the Fundamental matrix **F** from the projection matrices using the algorithm introduced in Section 3.2.3.4. Close inspection of Figures 5.6a and 5.6b indicate that corresponding lines of the same colour do indeed pass through the corresponding image features in both images. A magnified view shown in Figure 5.7 shows the accuracy of the estimated epipolar geometry.

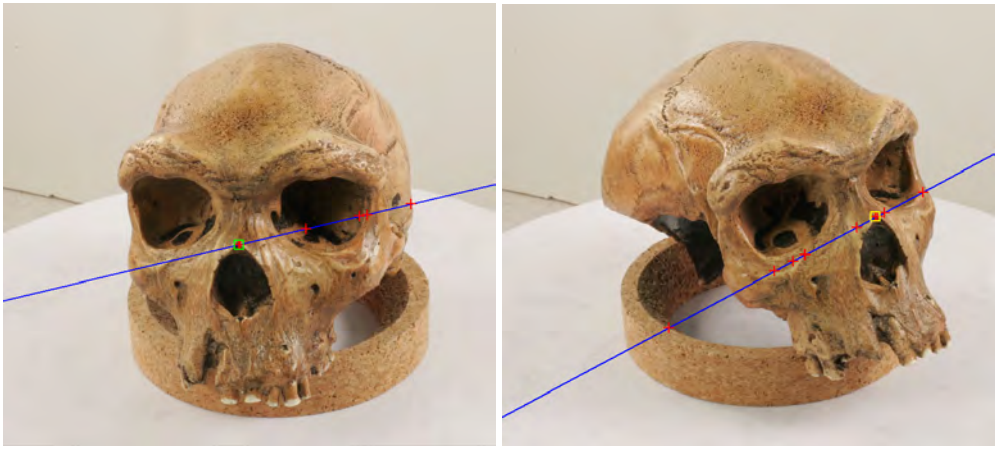
5.2.2.2 Feature Point Selection

In the Initial Feature Matching process introduced in Section 3.2.1, features identified using the Harris and DoG feature detectors are selected and then matched across multiple images. This section discusses the results obtained using epipolar geometry and a heuristic to select possible feature correspondences in multiple images. The selection process is optimized by rearranging the features in F such that the probability of selecting the correct matching point f' is maximized.

For each reference image R, a feature f is selected and the corresponding epipolar lines in the subsequent images are drawn. Features of the same type, either Harris or DoG, within two pixels of the corresponding epipolar lines are collected and form the set F. The candidate features $f' \in$ F are sorted in increasing order of photometric discrepancy with respect to a small image region around f. This has the effect of increasing the probability of selecting the correct matching point first, thus reducing the number of false matches and increasing the performance of the feature matching algorithm. Figure 5.8 shows the reference image R together with the feature point f (red +).

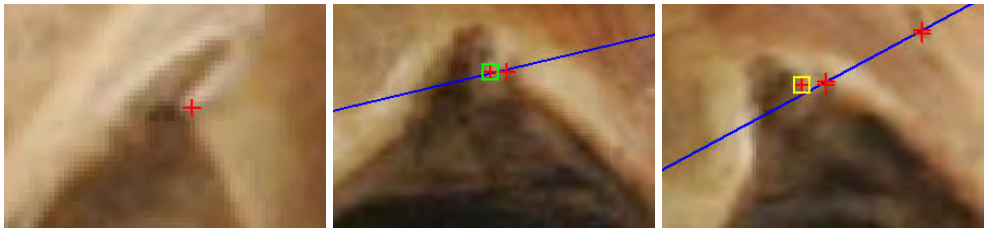
Figure 5.8: Reference image R and feature point f - A feature point f (red +) is selected in reference image R.

Figure 5.9 illustrates the selection of candidate matching points f' within two pixels of the corresponding epipolar lines for two subsequent images. The strongest matching point (smallest photometric discrepancy) in F is denoted by a green box around the feature f'. In this case the best match occurs for image 5.9a. The second strongest match is shown as a yellow box, intuitively this occurs for the other image, 5.9b.



(a) (b)

Figure 5.9: Candidate matching points - Feature points f' within two pixels of the corresponding epipolar lines are shown as red +. These features are then sorted in increasing order of photometric discrepancy with f. The strongest matching point is shown with a green box, the second is shown in yellow.



(a) (b) (c)

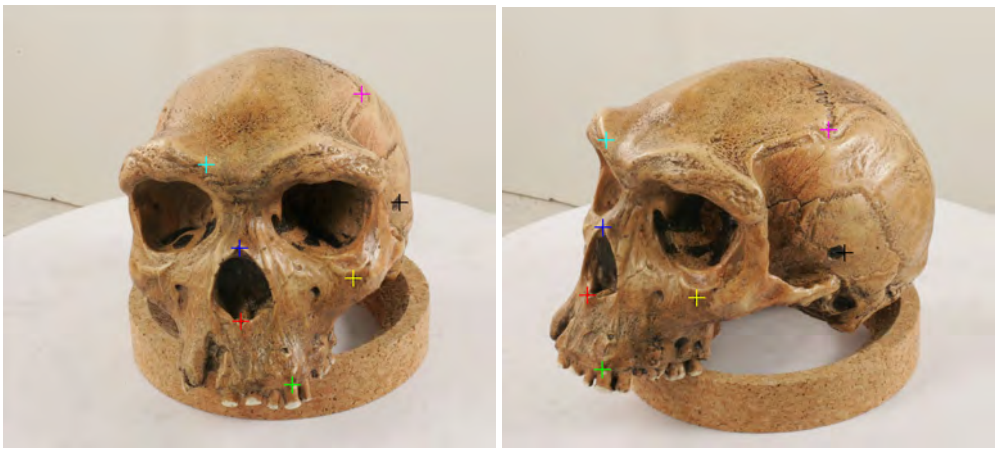
Figure 5.10: Magnified view of feature point f in reference image R and candidate matching points in subsequent images. - A feature point f (red +) in Figure 5.10a is selected in reference image R. Feature points f' within two pixels of the corresponding epipolar lines in Figures 5.10b and 5.10c are shown as red +. These features are then sorted in increasing order of photometric discrepancy with f. The strongest matching point is shown with a green box, the second is shown in yellow.

By rearranging the candidate feature points in order of increasing photometric discrepancy, the feature matching algorithm is able to reduce the number of false matches and improve performance in terms of reconstruction time as the reconstruction of false matches will be avoided.

5.2.2.3 Triangulation

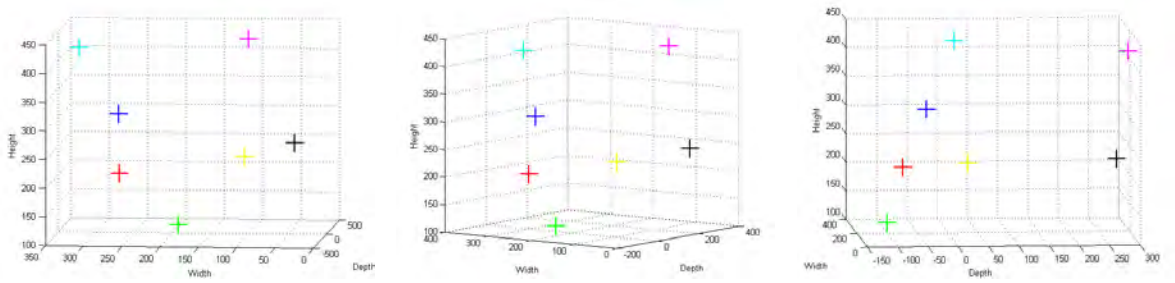
The triangulation process aims to recover the 3D position of a space point X from two image points x_l and x_r observed in the left and right images respectively. 3D reconstruction through the process of triangulation was presented in Section 3.2.3.5. This Section presents some preliminary test results for the triangulation algorithm.

To verify the correctness of the triangulation process, a set of corresponding feature points for two images in the "skull" dataset were manually selected. Figure 5.11 shows the two images, with corresponding feature points indicated by the same colour. The coordinates of the corresponding 3D space points, shown in Figure 5.12, were determined using the algorithm presented in 3.2.3.5. Figure 5.12a shows the points from the front view, Figure 5.12b at 45° to the front view and Figure 5.12b shows the right side view of the skull. By examining how the relative position of the points vary between the images, one can conclude that the triangulation preserves the 3D rigid structure of the object.



(a) **(b)**

Figure 5.11: Matched feature points - A set of manually selected feature points are used to test the triangulation algorithm. Matching points between the left and right images are shown in the same colour



(a) (b) (c)

Figure 5.12: Triangulation results - Three views of the 3D plot generated by triangulating the points f and f'.

5.2.2.4 Patch Visibility

During the Patch Reconstruction stage an initial estimate of the visibility of a patch is required. That is, a set of images in which a patch may be visible needs to be determined. A simple heuristic is used to identify potential visible images. A patch is said to be visible in an image if the angle between the patch normal and the ray joining the patch centre to the optical centre of the camera in question is below a certain threshold τ ($\tau = \pi/3$) [58] see Equation 3.7.

Figure 5.13 illustrates the concept of determining the visibility of a patch using the geometric constraints. The patch centre, shown as a red +, is generated using the feature point illustrated in Figure 5.8 and Figure 5.9a. Camera 0 is the reference camera and the patch normal is shown as the blue line from the patch centre towards the optical centre of the reference image. A few camera views are shown and the ray from the patch centre towards the optical centre of each camera is shown in the respective camera colour. The patch is visible in image I, when the angle between the patch normal (blue line) and the ray from the patch centre towards the camera (magenta, green or black line) is below $\tau = \pi/3$. In this example the visibility of the patch V(p) is determined as the set of images $\{1, 16\}$.

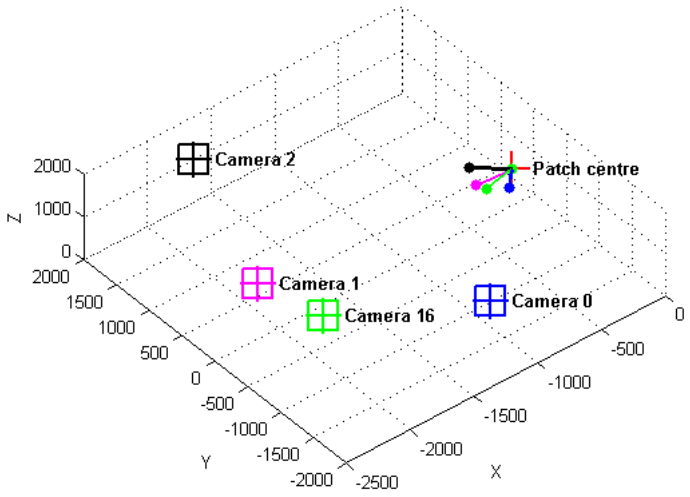


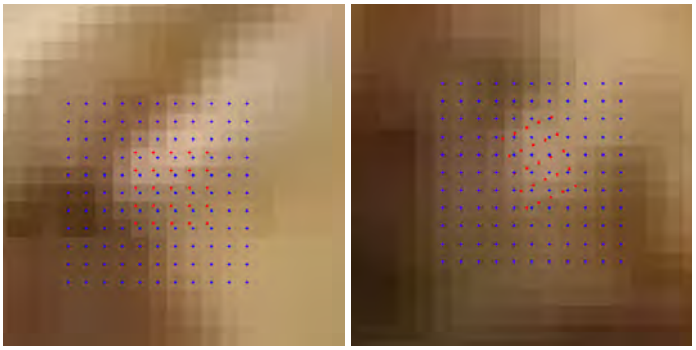
Figure 5.13: Visibility of a patch - Camera centres are given by the blue, green, magenta, and black boxes. The patch centre is shown as a red + and camera 0 is the reference camera. The ray from the patch centre towards the optical centre of each camera is shown in the respective camera colour. A patch is visible in image I when the angle between the normals is less than $\tau = \pi/3$

5.2.2.5 Pairwise Photometric Discrepancy Function

The Pairwise Photometric Discrepancy Function h(p, I, R), as defined in Section 3.1.2, provides a measure of correlation between two image regions. It is also used to define the set of images in which a patch is truly visible V*(p) (Equation 3.2).

As an example, the 3D patch associated with the feature points shown in Figure 5.8 (the reference image) and Figure 5.9a (candidate feature in image I^1) are used to demonstrate the computation of h(p, I, R) . The first step in evaluating h(p, I, R) is to project the patch centre $\mathbf{c}(p)$ onto the reference image R(p) (Figure 5.14a) and create a $\mu \times \mu$ pixel grid (red grid in figure 5.14a) centred at the image projection of $\mathbf{c}(p)$. The grid is oriented such that one of its edges is parallel to the image plane x-axis in R(p). The 3D grid at $\mathbf{c}(p)$ (Figure 5.15) is then determined by back projecting the grid coordinates in image R(p) to the depth of $\mathbf{c}(p)$ and orienting the 3D grid such that the points lie in the tangent plane at $\mathbf{c}(p)$. The 3D patch points (Figure 5.15) are then projected onto the reference image R(p) and image I, shown as the red grid in Figure 5.14a and Figure 5.14b respectively.

[1]Image I is any selected alternate camera view of the feature detected in R



(a) **(b)**

Figure 5.14: Photometric discrepancy computation - Image projections of patch p in R (p) (5.14a) and I (5.14b) are shown in red. The image region used for bilinear interpolation is shown in blue. The photometric discrepancy is given as 1 minus the NCC score between the interpolated image projections of p in R and image I.

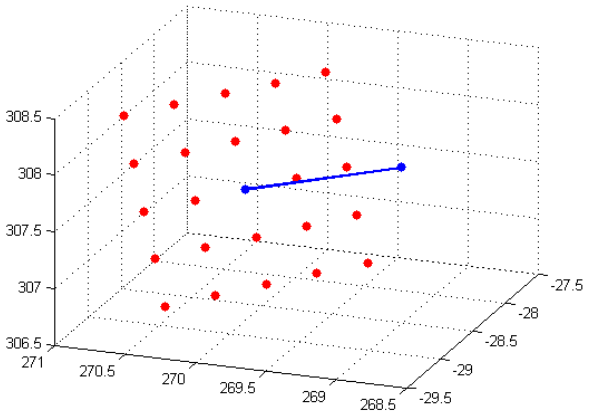


Figure 5.15: 3D Patch - The 3D patch grid is shown in red. The patch normal is shown in blue

The next step is to determine pixel colours of the projected patch points. Since we want an accurate representation of the grid colours we cannot simply snap the grid points to the nearest pixel and evaluate the colour. Interpolating the pixel colours results in a accurate representation of the patch colours and hence improves the correlation matching process. An 11×11 pixel grid is overlaid at the projection of the patch centre in R(p) and I, (blue grid). The extent of the grid is chosen such that the projected patch points always lie within the grid. Bilinear interpolation between the pixel grid points and the patch projections (red) is used to accurately determine the colours of the projected patch points. To further improve the matching process, interpolation is done independently for the Red, Green and Blue colour channels.

The final step involves computing the normalized cross correlation (NCC) between the interpolated patch projections in R(p) and I. The overall NCC score is given by the average of the NCC scores for the RGB colour space. The photometric discrepancy between the patches is then given by one minus the average NCC score.

The true visibility of a patch, V*(p), is determined from the set of visible images V(p) that have a photometric discrepancy score below a certain threshold α. This threshold α is set to 0.6 before optimization and 0.3 after. This is done because the photometric discrepancy score may be high due to imprecise geometry before optimization.

The algorithm was tested by evaluating the photometric discrepancy score at different patch locations such as those obtained by triangulation of the points shown in figure 5.11. Table 5.1 contains the results. The photometric discrepancy score for the patch associated with figure 5.14 is 0.2288. This is due to the precise triangulation of the patch centre c(p). Some of the patches have a photometric discrepancy score > 0.3 prior to optimization. The optimization stage refines $\mathbf{c}(p)$ and $\mathbf{n}(p)$ such that the global photometric discrepancy score g*(p) is minimized. Hence once optimized these patches have a better probability of being selected. Two incorrectly triangulated patch centres are selected for evaluating the algorithm. These patches will satisfy the first threshold but will fail to satisfy the post optimization threshold and are thus rejected, indicating that the method is effective.

Table 5.1: Pairwise Photometric discrepancy scores h(p, I, R) between the patches in Figure 5.11. A score below 0.3 indicates a good match. Incorrectly triangulated points are added to test the algorithm

3D Point (colour matches between R and I)	Photometric Discrepancy score
Blue	0.2288
Red	0.3010
Green	0.2906
Cyan	0.3846
Yellow	0.2262
Black	0.1845
Magenta	0.3983
Incorrectly triangulated	0.5295
Incorrectly triangulated	0.4645

5.2.2.6 Patch Optimization

The patch optimization algorithm is used to refine the position $\mathbf{c}(p)$, and orientation $\mathbf{n}(p)$, of a patch such that the global photometric discrepancy g*(p) is minimized. Optimization is achieved using the technique presented in Section 3.2.3.6. The optimization begins by initializing the patch parameters $\mathbf{c}(p)$ and $\mathbf{n}(p)$ (See Section 3.2.1), and thereafter iteratively refining the parameters using the conjugate gradient method.

To enhance the performance of the optimization algorithm an improved estimate of the orientation of the patch is made. Initially the patch is oriented such that it is parallel to the reference image (see Section 3.1.1 and Section 3.2.1). During optimization, this initial starting position may cause the algorithm to incorrectly converge to a local minimum, especially if the actual surface is far from perpendicular to the reference camera. To avoid such situations the following technique is used to provide an better estimate of patch orientation before optimization. For each visible image of p, the pairwise photometric discrepancy, h(p, I_i, R) where $I_i \in \mathrm{V}(p)$, and the normal in the direction of I_i are estimated. Thereafter a weighting factor for each normal is determined by combining the photometric discrepancy scores with a Gaussian kernel. The final surface orientation is determined as the average of the product between the normals and weights.

To illustrate this concept, the “temple” dataset is used since all the cameras are at the same height.

The process is nevertheless the same for all datasets. Figure 5.16 shows two images and the feature points that are used.

(a) (b)

Figure 5.16: Temple dataset - Two images from the temple dataset are used to illustrate the concept of improving the patch orientation using photometric discrepancy.

A 3D patch generated using the feature points shown in figure 5.16 is shown in Figure 5.17 as a red $\times$. The visual hull of the temple dataset (blue) as well as the camera centres for the visible images (green) are shown. The reference image for the patch is image 1 (Figure 5.16a) and the original surface normal is orientated towards this camera (cyan line in Figure 5.17). For each visible image of p $\{2-5, 23-29\}$ the pairwise photometric discrepancy with respect to R is computed and combined with Gaussian functions of the distance between R and I_i . Thereafter the normals are computed for each camera as the line from $\mathbf{c}(p)$ towards the respective camera. The normals are then combined with the Gaussian weights and the average is taken. In this example the strongest matches occur in the region of images 27, 28 and 29. Hence, the new estimated orientation of the patch is in this direction shown as the magenta line in Figure 5.17. Note that, for illustration purposes, the camera centres have been moved closer to the visual hull.
Practically this new estimate of surface orientation has greatly improved the optimization process, allowing it to converge to the correct depth and orientation in a fewer number of iterations whilst avoiding local minima.

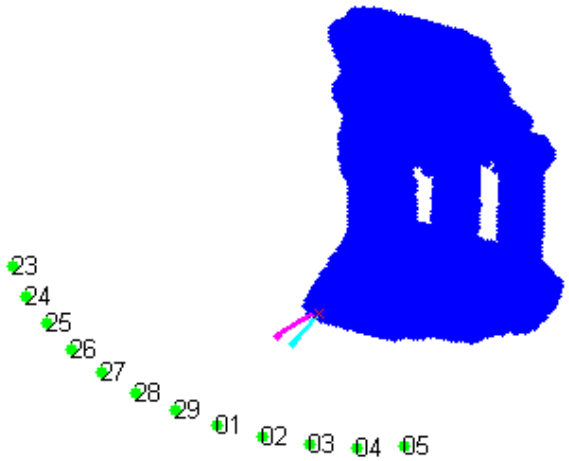


Figure 5.17: Improved surface orientation - The patch centre is shown in red and the objects visual hull is shown in blue. The new surface normal is estimated to be in the direction of the cameras that have low photometric discrepancy with R (magenta line, see text for discussion).

Table 5.2: Global photometric discrepancy g*(p) scores for points in Figure 5.11. The patch optimization algorithm is able to optimize c(p) and n(p) such that g*(p) is minimized.

3D Point (colour matches between R and I)	g*(p) before optimization	g*(p) after optimization	g*(p) after optimization using new orientation
Blue	0.3596	0.0893	0.0248
Red	0.3431	0.1630	0.1271
Green	0.2780	0.1651	0.1651
Cyan	0.4292	0.1125	0.0357
Yellow	0.1834	0.1496	0.0531
Black	0.1952	0.0743	0.0743
Magenta	0.4168	0.3557	0.2833
Incorrectly triangulated	0.5420	0.1238	0.1238
Incorrectly triangulated	0.4499	0.3960	0.3960

Table 5.2 shows results of patch optimization on the patches generated from the points shown in Figure 5.11. Note that scores for the global photometric discrepancy g*(p) with and without the new estimate of surface orientation are provided. A quick analysis of the results shows that the optimization is able to refine the parameters $\mathbf{c}(p)$ and $\mathbf{n}(p)$ sufficiently such that the orientation with respect to the visible images V*(p) is maximized, hence minimising the global photometric discrepancy g*(p) (column 3 Table 5.2). However, for a few patches the optimization gets stuck in a local minimum. This is avoided by using the new estimate of surface orientation before optimization.

Table 5.2 column 4, shows the final scores obtained by the optimization algorithm. Results for all the correctly triangulated 3D points show a significant decrease in g*(p) values. This will result in better scores for the Pairwise Photometric Discrepancy between image R and I and hence a higher probability of the patch being successfully generated with $\alpha = 0.3$ in Equation 3.2. Results for the incorrectly triangulated points are as expected. Due to the almost uniform colour and texture of the skull dataset, one would expect a large number of false positive's. However, it is interesting to note that optimization using the new surface orientation estimate, is unable to minimize g*(p) any further. This is due to high pairwise photometric discrepancies during the computation of the new surface orientation.

The optimization algorithm is able to improve both of the incorrectly triangulated points. In this example the first incorrectly triangulated patch might be falsely classified as a correct match ($|V^*(p)| > \tau$). As previously stated this is to be expected on this type of image dataset. It is up to the subsequent expansion and filtering stages to remove this false positive match from the set of patches $\mathbf{P}$.

5.2.2.7 Initial Feature Matching Process

The Initial Feature Matching Process is finally brought together by integrating the various components introduced in Section 3.2. The only input to the IFM algorithm is the image dataset and associated camera parameters. Feature matching begins by detecting prominent features in each image using the techniques presented in Section 3.2.2. Thereafter each feature is matched across multiple images using the feature matching algorithm presented in Section 3.2.1.
Initial test results of the algorithm on three images in the skull dataset are shown in Figure 5.18. Feature points in the reference image, Figure 5.18a, are matched across two subsequent images,

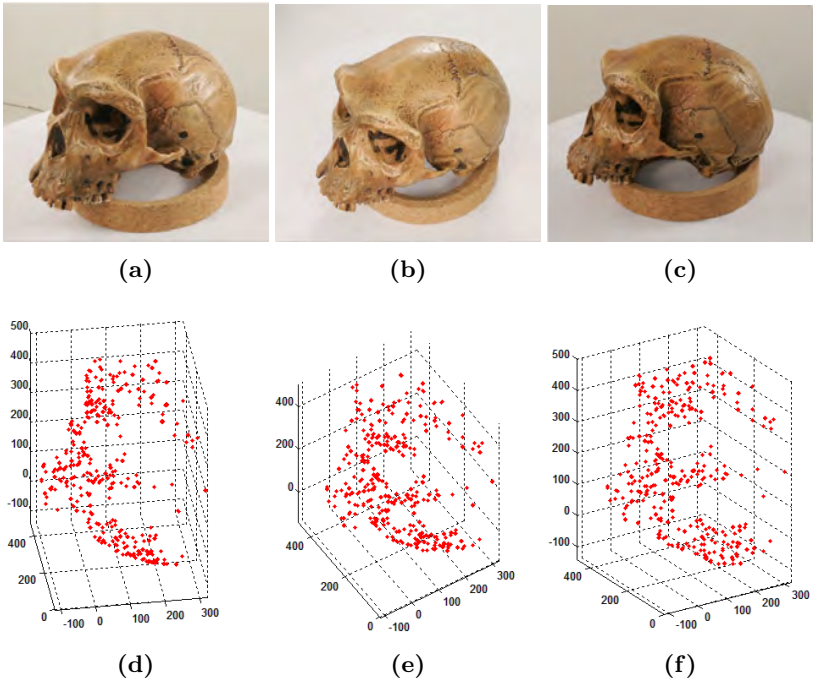


Figure 5.18: Initial feature matching using three images - Input images (a), (b), (c) and initial feature matches (d), (e), (f)

Figure 5.18b and Figure 5.18c. Different views of the resulting 3D point model are shown in Figure 5.18d, 5.18e and 5.18f. These results show that the algorithm was able to successfully: (1) select a feature in the reference image and find the best possible matching points in two successive images; (2) correctly triangulate these matches and determine the true visibility of the patch; (3) perform patch optimization where the position and orientation of the patch are refined; (4) reject patches as outliers if the number of visible images is $\leq\tau$ (τ=3 for skull dataset); (5) a successfully generated patch is then stored in the image model cell and feature points that lie in the same cell as p in all the truly visible images V*(p) are removed. This step is taken to ensure that the same patch is not generated by another feature in other images. Removing these features is a key step in increasing performance of the entire feature matching algorithm.

Further testing was done by performing matching on more views. The following image sets were used, where the first number in the set denotes the reference image: $\{1, 15, 16\}$; $\{2, 8, 17\}$; $\{3, 9, 18\}$ and $\{4, 10, 19\}$.

Figure 5.19 shows the results of adding more views to the model. The final model using reference

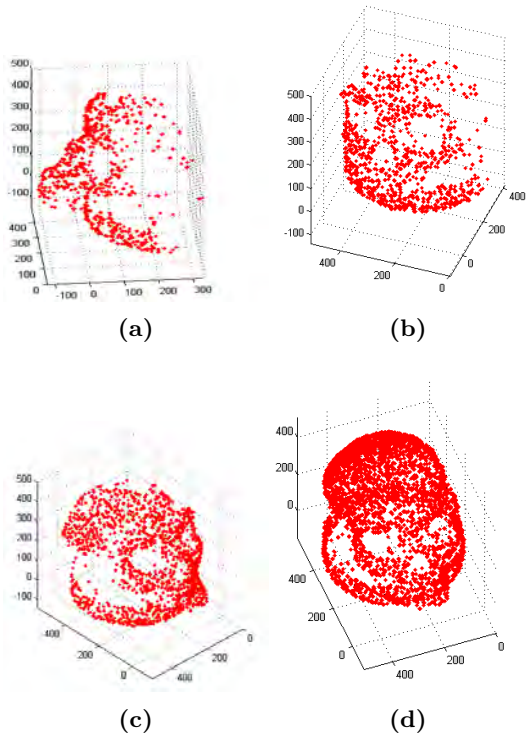


Figure 5.19: Initial feature matching on more views - Initial feature matches obtained by successively adding more views are shown in (a), (b), (c), and (d) contains matches from all views.

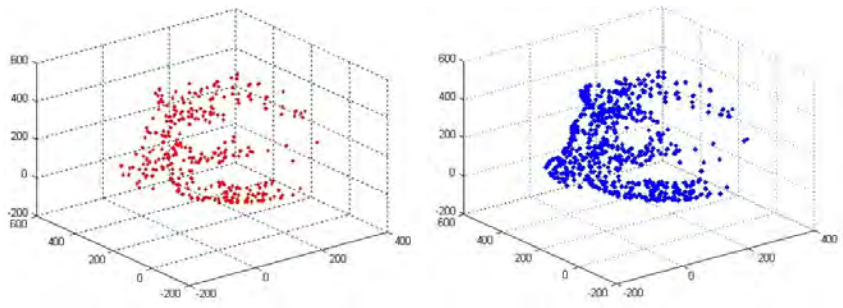
images $\{0, 1, 2, 3, 4\}$ is shown in Figure 5.19d.

These preliminary feature matching results look very promising. A visual assessment of the results using all camera views, Figure 5.19d, shows that the model contains very few outliers if any. Subsequently, Feature Matching was performed on the entire dataset and the results for different views are provided in Section 5.3.1.

5.2.3 Feature Expansion

The feature expansion process is used to increase the density of reconstructed patches. Feature expansion is carried out using the technique presented in Section 3.3. Patches from the Initial Feature Matching stage are expanded by firstly identifying suitable regions for expansion. Two conditions are used to identify regions where patch expansion is unnecessary. The first is a neighbour condition and the second is a depth discontinuity condition. Intuitively, a patch should not be

expanded for a surface region if a patch has already been reconstructed there. After the appropriate image cells for expansion have been identified, the expansion procedure (Section 3.3.2) is carried out for each patch with the goal of generating a patch in each image Cell $C_i(x, y)$.



(a) (b)

Figure 5.20: Patch Expansion - A subset of the initial Feature matches from the skull dataset, (a), is expanded using the Patch Expansion technique presented in Section 3.3. (b) shows the expanded patches.

Initial testing of the Expansion algorithm was done on a subset of the initial feature matches obtained for the skull dataset. Figure 5.20b shows the results of expansion on the patches shown in Figure 5.20a.

Figure 5.21 shows a magnified 3D view of the initial patches (red) and expanded patches (blue). The density of the expanded patches depends on the properties of the initial patch chosen for expansion. Hence, more patches are reconstructed for regions that are sparse.

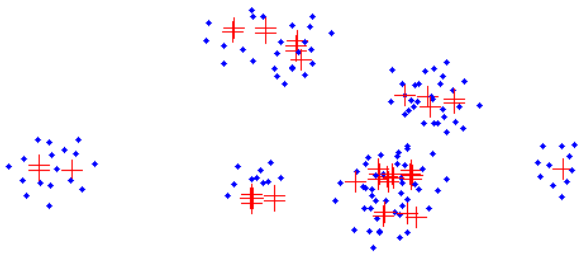


Figure 5.21: Close up view of expansion - Initial matches shown as red +, Expanded patches as blue dots.

5.2.4 Patch Filtering

The Patch Filtering stage is an important part of the reconstruction pipeline. The set of patches generated during the Initial Feature Matching and Expansion stages may contain false positives. The filtering stage uses three filters, presented in Section 3.4, to remove these outliers. The first two filters use visibility consistency to remove patches that do not lie on the surface of the model. Thereafter, isolated/lone patches are removed by the third filter using a regularization technique.

Initial testing of the Patch Filtering algorithm was done quite crudely using synthetic outlying patches to mimic situations where an outlier is present in the data. This was done because for object datasets, where a visual hull is used, outlying patches may lie within the visual hull, making it difficult to distinguish between an outlier and a correct patch.

The following technique was used to test the algorithm. The first filter (Equation 3.59) was tested by generating a patch that projects into the same cell as p, but is not a neighbour of p. This is accomplished by simply initializing the outlier to lie along the viewing ray between the patches and the optical centre of the corresponding camera (O_i). Thereafter the outlier is moved sufficiently away from O_i such that the outlier and the patches are not neighbours (Equation 3.58). This outlier is illustrated in Figure 5.22 as the red +.

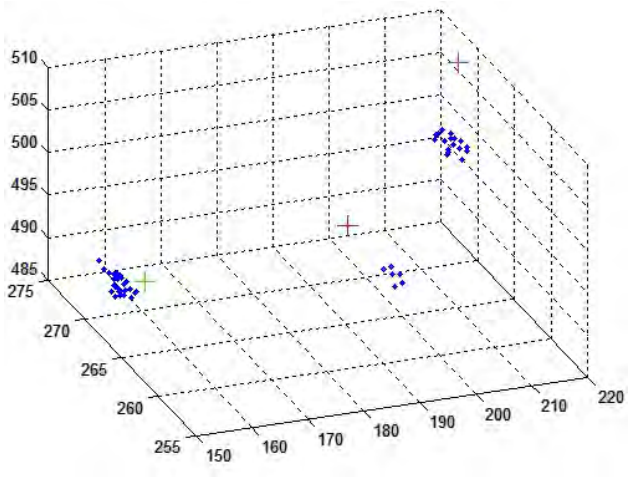


Figure 5.22: Initial testing of Patch Filtering algorithm - A subset of patch centres are shown as blue dots. Synthetic outliers are shown using coloured + symbols.

Under this situation, when Equation 3.59 is evaluated for the outlying patch, both the visible im-

ages $V^*(p)$ and its global photometric term $(1 - g^*(p))$ will be small. This is because the outlier will not project onto the true matching regions for its visible images. Hence the outlier is filtered out using Equation 3.59.

The second filter was tested by initializing the centre of the outlier to lie outside the object (Figure 5.22 green+). Hence the outlying patch will project into the same image cell as the patches in blue, however the depth for the patches and the outlier are not the same. The second filter removes this outlying patch by computing the number of images in which the patch is visible using a depth map test [58]. The patch is then removed if the number is less than γ. Patch visibility using the cell depth map test is determined as follows: (1) for each image I_i in the dataset; (2) project the patch into image I_i, identifying the corresponding cell; (3) compute the average depth for all the patches in that cell; (4) the patch is then visible in image I_i if its depth is less than the average depth for that cell. This filter can be aggressive and will possibly remove some valid patches together with the outliers. The iteration of the expansion and filtering stages many times resolves this as the average depth of the cell will improve with each iteration.

The third filter was tested by initializing a patch away from surrounding patches (Figure 5.22 magenta +). This patch is then projected into its visible images to determine the proportion of neighbours. If the number is less than 0.25(determined experimentally) the patch is said to be an outlier and is removed.

Initial testing of the Patch Filtering algorithm under these ideal conditions provided good results. The algorithm was able to effectively remove all three outlying patches. Section 5.3.3 provides results for real datasets.

5.2.5 Polygonal Mesh Reconstruction

This section presents intermediate test results for the Polygonal Mesh Reconstruction stage. Patches obtained from the Feature Matching stage are used to both initialize and refine surface mesh models. The mesh reconstruction process begins by initializing a surface mesh using Poisson Surface Reconstruction for scene datasets, and Iterative Snapping for object datasets. Thereafter, the initialized mesh is refined using an energy minimization technique.

5.2.6 Poisson Surface Reconstruction

The Poisson Surface Reconstruction algorithm of Kazhdan *et al.* [59; 84] is used for situations in which a visual hull cannot be determined from the images. These situations often arise for scene datasets. In practice we use a publically available implementation of the PSR algorithm available at [89].

5.2.7 Visual hulls

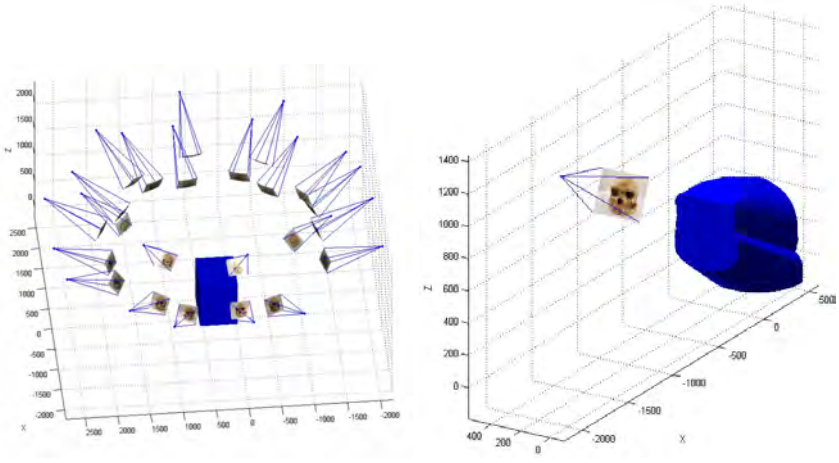
Visual Hulls provide a means of enforcing silhouette consistency on the reconstructed patches. Thus for object datasets, during the Feature Matching stage, patches that lie outside the visual hull of the object are rejected. The Visual Hull is again used in the Iterative Snapping algorithm as an initial estimate of the objects surface. Hence, for object datasets, determining good Visual Hulls is a vital part of the reconstruction process. This section discusses the technique used and results obtained for the reconstruction of Visual Hull models.

Segmentation is the first step in the reconstruction of a Visual Hull. For the "skull" dataset, segmentation information was provided in the form of edge contours. Silhouettes where produced by labelling pixels enclosed by the contour as foreground (white) and all other pixels are labelled as background pixels (black). For the "Temple" and "Dino" datasets such information was not provided. Silhouettes for these datasets were extracted using the following procedure: (1) Threshold the image such that majority of the foreground pixels retained; (2) close holes (contours) present within the silhouette; (3) fill holes; (4) dilate the image by u pixels; (5) erode the image by v pixels. The variables u and v are chosen such that the silhouette is conservative i.e. the silhouette lies a maximum of κ pixels outside the edge of the object ($\kappa = 3$ pixels for our experiments).

(a) **(b)**

Figure 5.23: Image Silhouette - Input image (a) and associated silhouette (b) for the skull dataset

The visual hull of the object is then determined using a simple Voxel Carving technique. Essentially, image silhouettes from different camera views are used to carve a voxel cube placed at the centre of the scene. The space carving process begins by determining the size and location of the voxel cube using the space spanned by all the camera views. This space is then filled with η voxels. Hence the number of voxels η, in the cube, determines the resolution of the final model.



(a) **(b)**

Figure 5.24: Voxel Carving process - (a) Cube of voxels and orientation of cameras observing the scene. (b) Result after 1 carving.

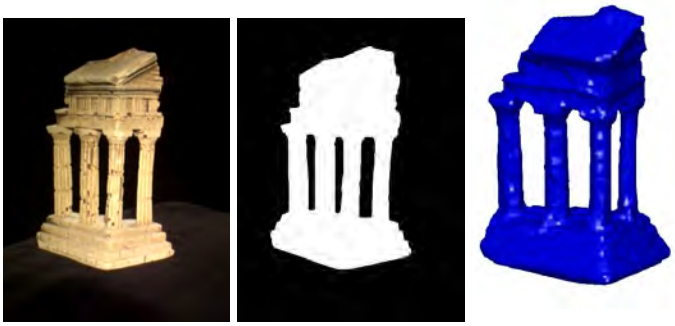
Figure 5.24a shows the initial voxel volume and the orientation of the cameras as well as the image associated with the corresponding camera. Figure 5.24b shows the result obtained after a single

(a) (b) (c)

Figure 5.25: Visual Hull model for the skull dataset - Three views of the Visual hull generated using the Voxel carving technique.

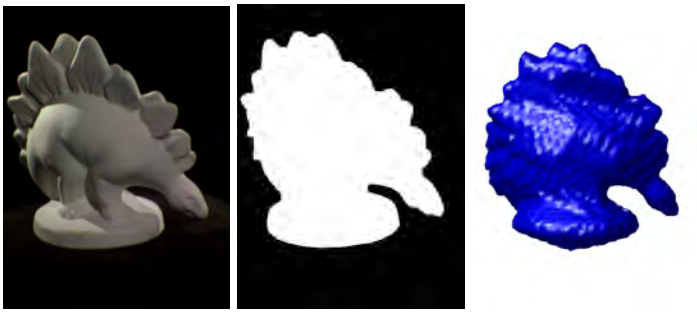
carving and also illustrates the basic idea behind the process. After carving the voxel space using all the camera views, real values are assigned to the voxels by moving all voxels a third of a square in each direction then seeing if they get carved off [90].

Figure 5.25 shows three views of the Skull Visual Hull model. Visual Hulls for the "Temple" and "Dino" datasets are shown in Figure 5.26c and Figure 5.27c respectively.



(a) (b) (c)

Figure 5.26: Visual Hull model for the Temple dataset - (a) an input image from the Temple dataset and its associated silhouette (b). The Visual Hull generated is shown in (c).

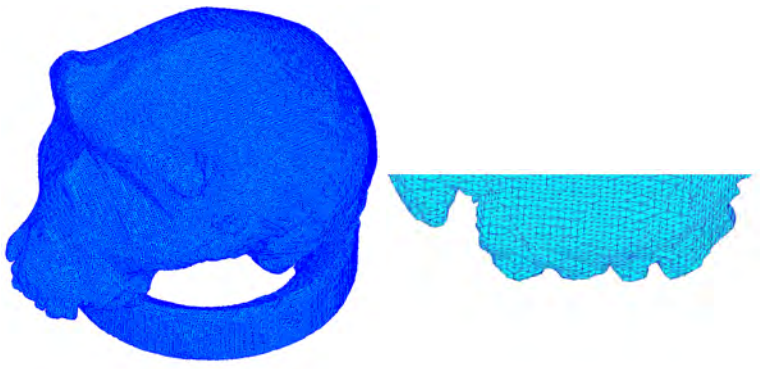


(a) (b) (c)

Figure 5.27: Visual Hull model for the Dino dataset - (a) an input image from the Dino dataset and its associated silhouette (b). The Visual Hull generated is shown in (c).

5.2.8 Iterative Snapping

This section presents preliminary test results for the Iterative Snapping mesh initialization algorithm presented in Section 4.3. The Iterative Snapping algorithm is used for object datasets where segmentation information can easily be extracted from the input images. This segmentation information is then used to generate a Visual Hull of the object. A surface mesh model is then initialized at the boundary of the visual hull using the Delaunay triangulation algorithm [91] and is refined using the Iterative Snapping technique.



(a) (b)

Figure 5.28: Mesh initialization - (a) A triangular mesh model initialized at the boundary of the Visual Hull. (b) Magnified view of the mesh model showing the triangles and their connectivity.

Figure 5.28a shows the result of mesh initialization using the Visual Hull of the Skull dataset. To

ensure an almost uniform triangle size, a uniform distribution of points on the voxel surface is used to initialize the mesh model. An important step in the mesh refinement process is the computation of the vertex distance metric $d(v_i)$. This metric forms part of the patch reconstruction consistency term $E_p(v_i)$ (Equation 4.2) and is essentially a measure of strength of the reconstructed patches in the region of the vertex v_i. Figure 5.29 illustrates test results obtained for the computation of $d(v_i)$ during the evaluation of the reconstruction consistency term $E_p(v_i)$. The magnitude of $d(v_i)$ (Equation 4.3) is a function of the distance between the reconstructed patches and the vertex v_i. Hence the effect of $E_p(v_i)$ is to move the vertex v_i towards the centroid formed by nearby reconstructed patches.

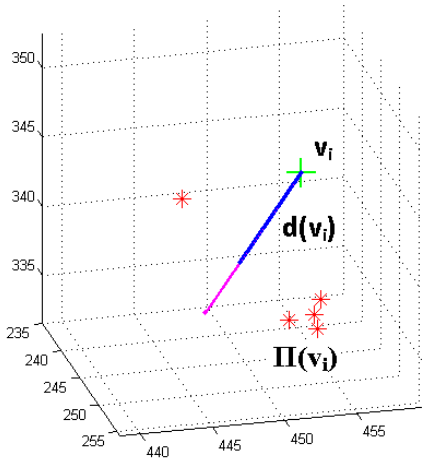
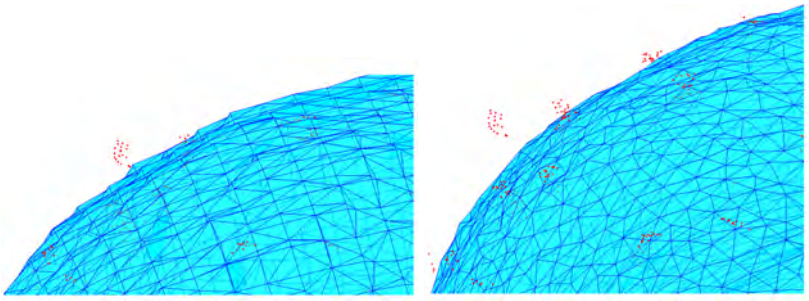


Figure 5.29: Computation of the signed distance $d(v_i)$ - Reconstructed patches compatible with vertex v_i (green) are denoted as the set $\Pi(v_i)$, and are shown in red. The magenta line denotes the negative of the vertex normal and the blue line denotes the distance and direction of $d(v_i)$.

Figure 5.30 shows the results after running the algorithm for 25 iterations. The patches are shown as red dots and lie near the surface of the mesh. Most noticeably the resulting mesh, Figure 5.30b, appears smoother; and the effect of outliers are reduced by the use of Gaussian weights. The change in triangle connectivity and size is also noticeable with the average edge length reducing every iteration due to the remeshing operations. The Iterative Snapping process is iterated until the average edge length becomes approximately 2 pixels in length at a depth of the vertex.



(a) (b)

Figure 5.30: Iterative snapping results - (a) Original mesh model with overlaid reconstructed patches(red dots). (b) Output of Iterative Snapping algorithm for 25 iterations showing the consistency with the reconstructed patches. Both surfaces are made transparent such that patches that lie beneath the surface can be viewed.

5.2.9 Mesh Refinement

The aim of the mesh refinement algorithm presented in Section 4.4 is to further improve the appearance of the model by attempting to recover fine surface details. This section discusses the intermediate results of the mesh refinement algorithm. The first step requires the computation of a set of visible images for each vertex using a depth map test. This is achieved by computing the distance between the vertex and each camera. The set of visible images is then given by the η [1] closest cameras. Figure 5.31 illustrates the set of vertices that are visible from camera 1 for the Skull dataset.

The next stage involves the computation of the photometric consistency term $E_p(v_i)$. Following from Section 4.4, for each pair of visible images a patch is generated using the optimisation technique presented in Section 3.2.3.6. Figure 5.32 illustrates the mesh surface and results obtained using this approach. The vertex centre, denoted by the red +, and vertex normal(yellow line) is optimised independently for each pair of its visible images. This results in two patches (magenta +) near the vertex. The patch normals are indicated by the red lines. The strength of the photometric consistency term, shown as a green line, is then given by evaluating Equations 4.6 and 4.5 on the resulting patches.

The patch generation procedure (vertex optimisation) is carried out only once for each vertex on the mesh resulting in a set of patches $\mathrm{P}(v_i)$. The mesh refinement process is then iterated until

[1] The number of cameras are chosen experimentally for each dataset. In essence the angle formed between the outer two cameras and the vertex should not exceed $\pi/3$.

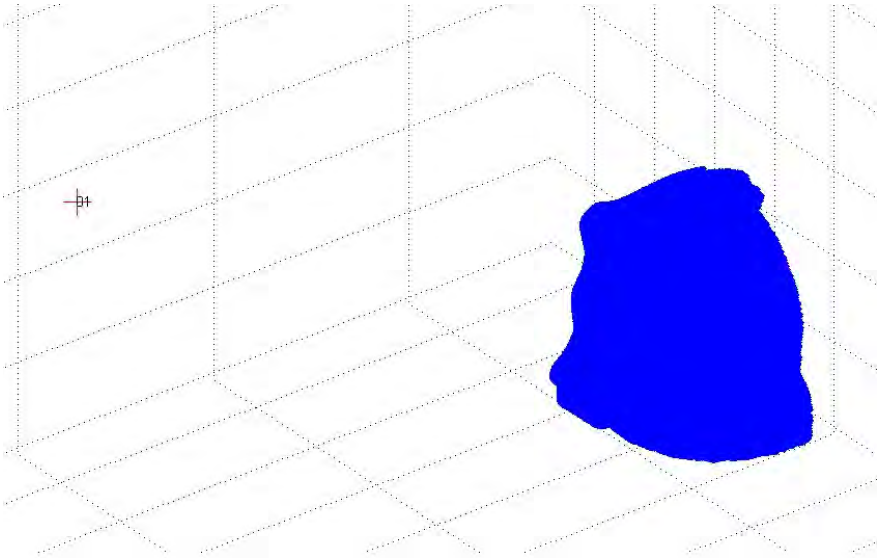


Figure 5.31: Vertices visible from a single camera view - To determine the visibility for a vertex, the distance between each camera and the vertex is computed. The visible images is then given by the η closest cameras. The red + indicates the camera centre for image 1. All vertices visible from this camera are shown in blue.

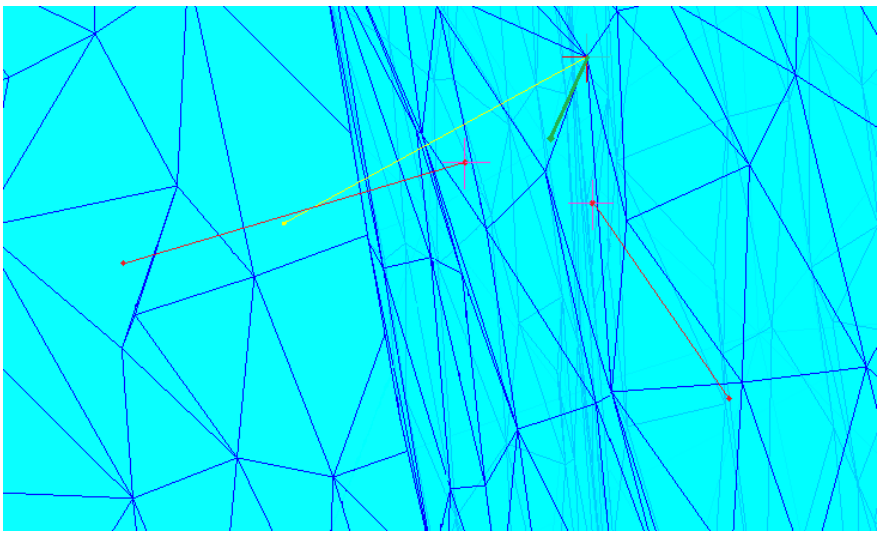


Figure 5.32: Computation of the photometric discrepancy term $E_p(v_i)$ -

the desired mesh resolution is achieved. Mesh simplification techniques such as edge split, contract and swap are performed once in every 5 iterations to avoid self intersections on the mesh. The final results of mesh refinement on each of the datasets are provided in Section 5.3.7.

5.3 Final results

In this section the final results of the 3D reconstruction system is presented. The section begins by discussing individual results from each stage within the reconstruction algorithm. Thereafter the system evaluation results obtained from the Middlebury evaluation is discussed. Results are presented for each stage as it is important in the context of MBVC for military applications. The extent to which the realism of the models appear convincing is what matters to a soldier in the field. Statistics and measurements are only important from a scientific point of view. To this end, the Middlebury evaluation [24] provides a platform for quantitative comparison of MVS algorithms. A collection of high-quality calibrated multi-view stereo datasets registered with ground truth 3D models are available together with an evaluation methodology for comparing multi-view algorithms.

5.3.1 Initial Feature Matching Results

This section discusses the Initial Feature Matching results obtained for object and scene datasets and shows the progress made by the algorithm as it builds towards the final result. Figure 5.33 shows the reconstructed patch model for the skull dataset.
Each patch is texture mapped by projecting the patch points ($\mu \times \mu$ grid) onto its reference image and sampling pixel colours. As illustrated by the figures, the reconstructed patches are sparse and possibly contain outliers. For object datasets, the corresponding visual hull models help reject outliers that lie outside the objects surface.
Figures 5.34 and 5.35 show similar results for the Dino and Temple models respectively. Here again the patches sparsely cover the surface and the presence of outliers is noticeable. The black colour around the edges of the models is caused when a portion of the patch extends beyond the edge of the model. These patches should be filtered out in the subsequent stages.

Figures 5.36 and 5.37 illustrate the results from the two scene datasets. With these types of datasets it is often difficult to obtain tight bounding volumes. For the Fountain dataset, a bounding volume was determined using markers to denote the extremities of the scene[1]. These end points

[1]These markers where determined by the authors of the dataset.

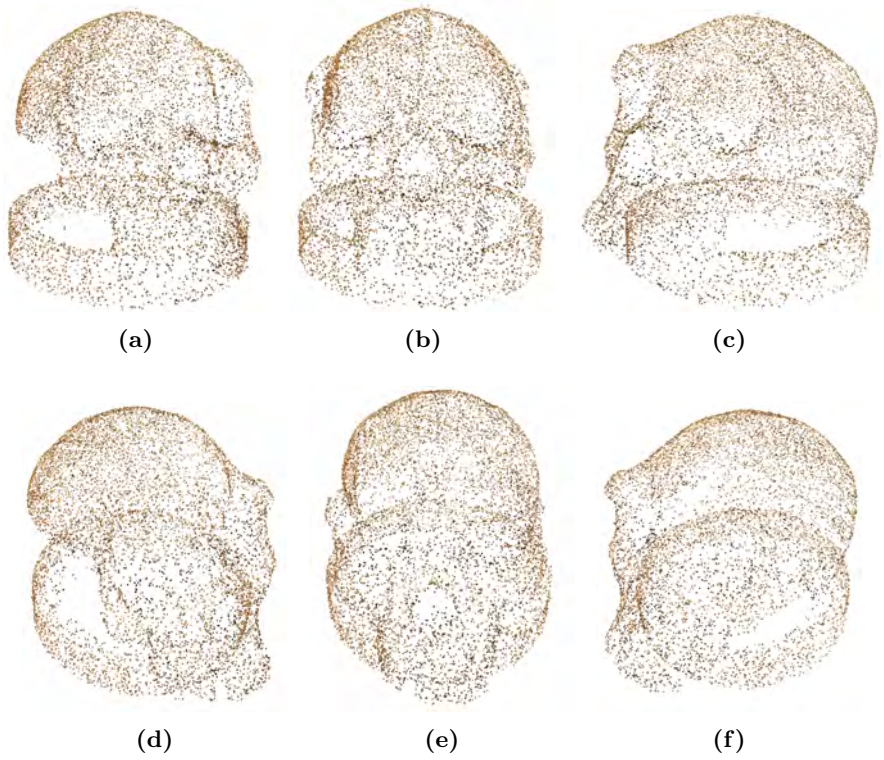


Figure 5.33: Initial feature matching results: Skull - Different views of the reconstructed patches after performing initial feature matching on the Skull dataset.

were then triangulated to give the 3D coordinates of the bounding box. A similar procedure was used for the City Hall dataset. Figure 5.36a - 5.36c illustrate the robustness of the IFM algorithm to outliers. Outliers, easily seen in Figures 5.36d and 5.36e are present in the patch model at this stage of the reconstruction.

Figure 5.37 contains the Initial feature matching results for the City Hall dataset. This dataset is very challenging, as the viewpoints change significantly from one image to the next. Nevertheless the IFM algorithm is able to recover most of the objects surface structure with a small number of outliers shown in Figure 5.37d. However, as this is a large scene, outliers closer to the actual surface are more difficult to identify.

The results of the Initial Feature Matching stage show that the algorithm is able to reconstruct point models for both object and scene datasets with a minimum number of visible outliers[1]. Table

[1]At this stage in the reconstruction it is very difficult to quantify outliers. The only method available is visual

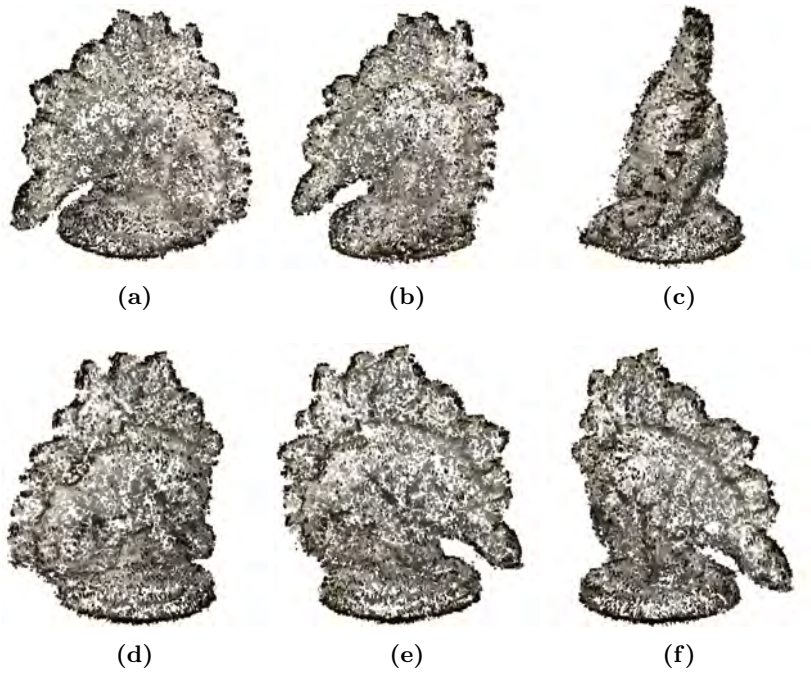


Figure 5.34: Initial feature matching results: Dino - Different views of the reconstructed patches after performing initial feature matching on the Dino dataset.

5.3 shows, for each dataset, the number of feature points detected and the resulting number of patches generated by the Initial Feature Matching algorithm.

assessment.

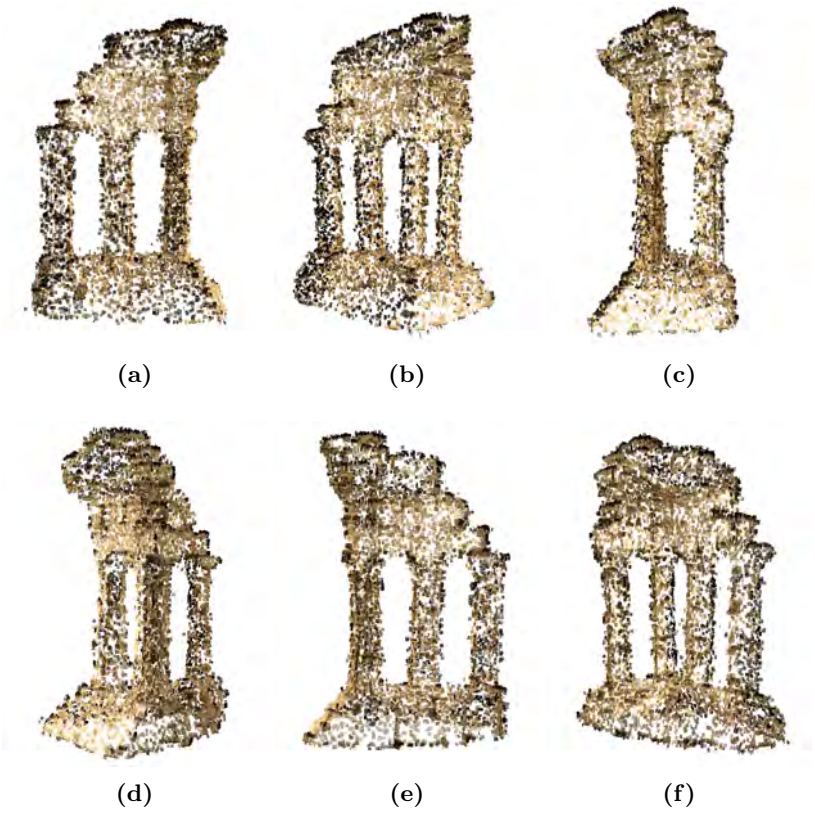
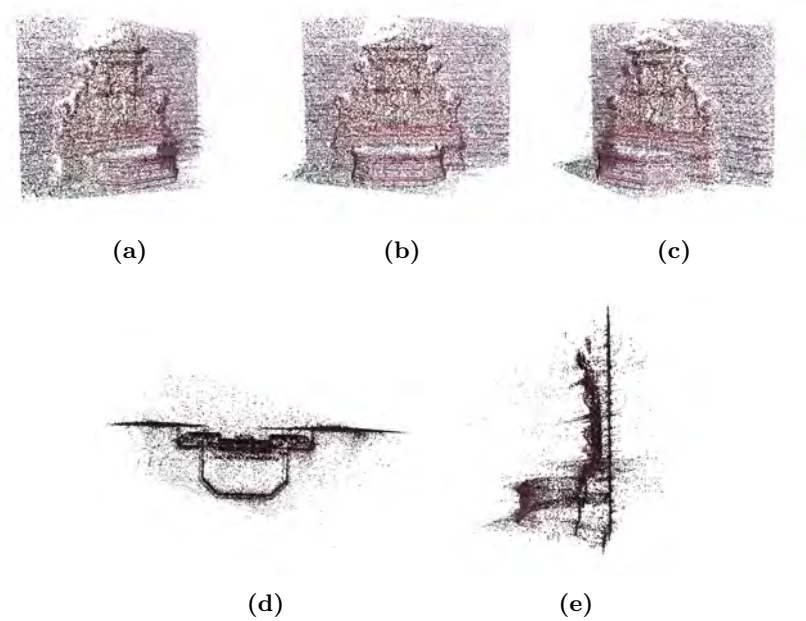


Figure 5.35: Initial feature matching results: Temple - Different views of the reconstructed patches after performing initial feature matching on the Temple dataset.

Table 5.3: Initial feature matching results - Features detected and the resulting number of IFM patches generated

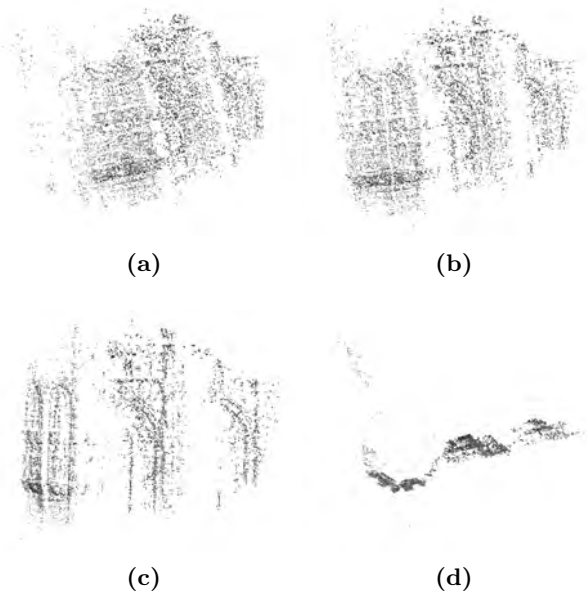
Dataset	DoG Features	Harris Features	IFM Patches Generated	Reconstruction Time (H)
Skull	66561	70576	20725	36
Dino	29933	24194	60862	34
Temple	21052	20492	24501	39
Fountain	98129	92351	90581	109
City Hall	72336	74480	18727	73



(a) (b) (c)

(d) (e)

Figure 5.36: Initial feature matching results: Fountain - Different views of the reconstructed patches after performing initial feature matching on the Fountain dataset.



(a) (b)

(c) (d)

Figure 5.37: Initial feature matching results: City Hall - Different views of the reconstructed patches after performing initial feature matching on the City Hall dataset.

5.3.2 Patch Expansion Results

This section presents results of the Patch Expansion algorithm. The expansion procedure is iterated 3 times in conjunction with the filtering stage to increase the density of the model as well as to remove any outliers that may be present. To simplify matters, only the output of the first expansion procedure is presented. The results after 3 expansion iterations in conjunction with patch filtering is presented in Section 5.3.4. Table 5.4 shows how the density of the patches increases with each iteration.

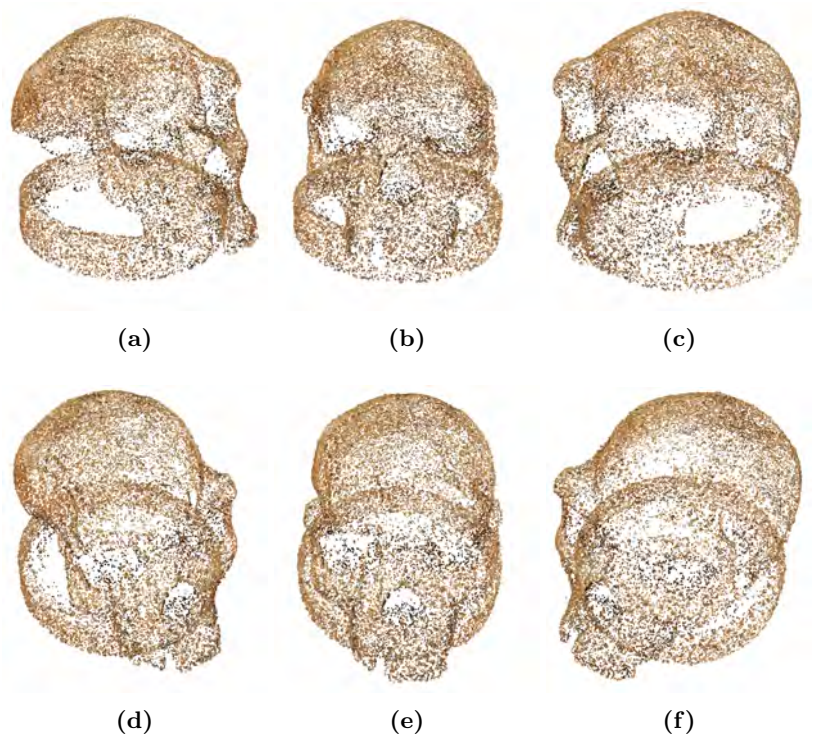


Figure 5.38: Patch expansion results: Skull - Different views of the reconstructed patches after performing Patch Expansion on the Skull dataset.

The expansion procedure is unaware of outliers and will expand any patch that satisfies the conditions set out in Section 3.3. Hence the density of outliers may be increased up to a point where the filters regard them as part of the actual surface. Figure 5.38 shows different views of the expanded patches for the Skull dataset. The reconstructed patches have increased in density making the surface structure more apparent.

Similar results are shown in figures 5.39 and 5.40 for the Dino and Temple models respectively.

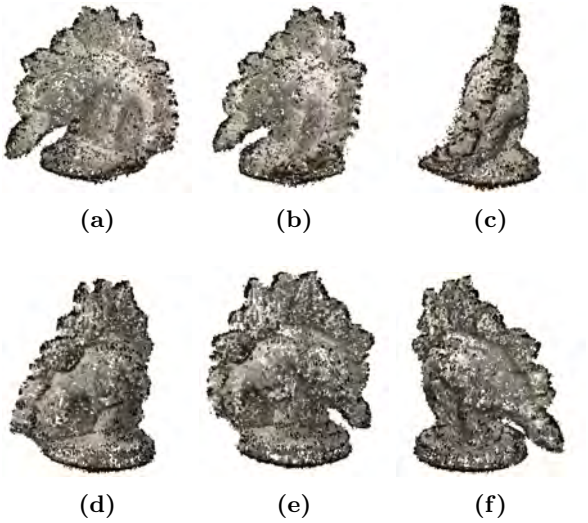


Figure 5.39: Patch expansion results: Dino - Different views of the reconstructed patches after performing patch expansion on the Dino dataset.

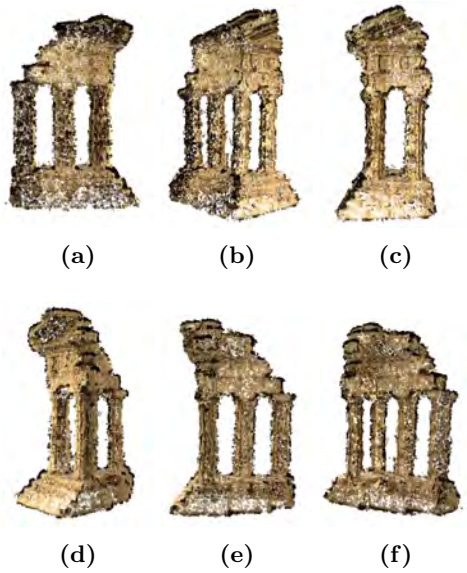
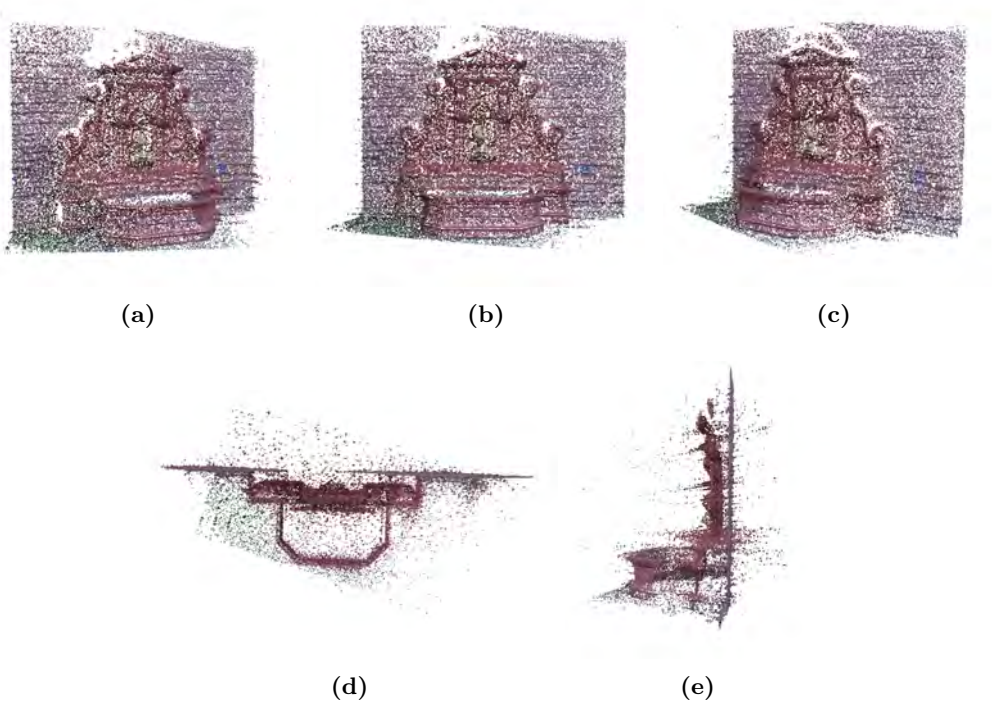


Figure 5.40: Patch expansion results: Temple - Different views of the reconstructed patches after performing patch expansion on the Temple dataset.



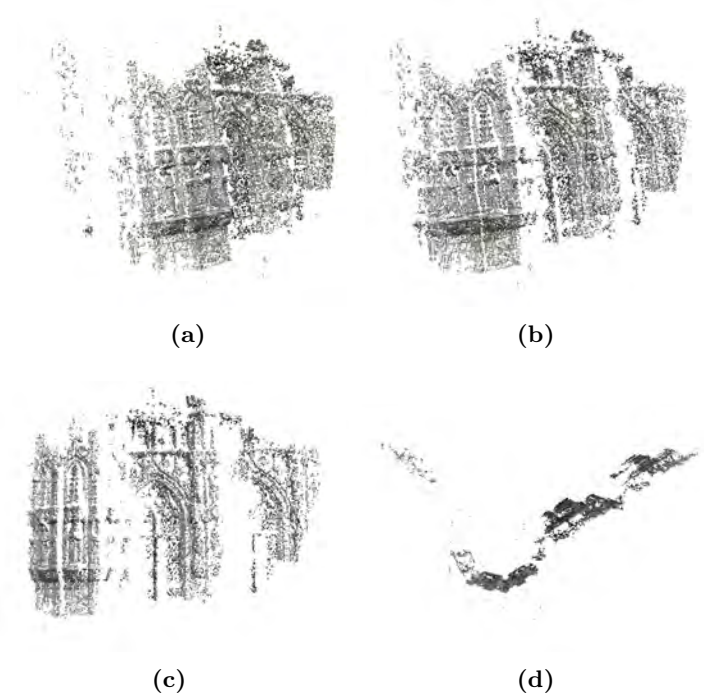
(a) **(b)** **(c)**

(d) **(e)**

Figure 5.41: Patch expansion results: Fountain - Different views of the reconstructed patches after performing patch expansion on the Fountain dataset.

Figures 5.41 and 5.42 show expansion results for the Fountain and City Hall models respectively. The increased density of the models are immediately noticeable. However, Figures 5.41d, 5.41e and 5.42d show that many of the outliers are also expanded. This makes the task of the filtering stage more difficult.

Table 5.4 also indicates the total reconstruction time for the patch reconstruction stage. The primary reason for the poor performance (speed) is that the patch reconstruction stage is very sequential and iterative and thus cannot be vectorized. Matlab is not suited to iterative processes and performs poorly. The sheer number of points (millions for each image) evaluated by the algorithm at each stage is ultimately what makes the process slow. However, for the mesh reconstruction stage, which can be vectorized, significant speed up is achieved (see Table 5.5).



(a) **(b)** **(c)** **(d)**

Figure 5.42: Patch expansion results: City Hall - Different views of the reconstructed patches after performing patch expansion on the City Hall dataset.

Table 5.4: Resulting number of patches after IFM, Expansion and Filtering - IFM patches are iteratively expanded and filtered 3 times to produce the final patch model.

Dataset	IFM	Expand 1	Filter 1	Expand 2	Filter 2	Expand 3	Filter 3	Time (H)
Skull	20725	66523	20564	81889	43237	367020	296582	515
Dino	60862	95740	45972	76997	68268	97366	85783	287
Temple	24501	169236	87545	489992	157470	219762	103307	328
Fountain	90581	376758	40930	79351	68764	141890	129732	624
City Hall	18727	98032	66543	80106	67366	83360	77275	301

5.3.3 Patch Filtering Results

Here the results from the Patch Filtering stage are presented. Again only the results from the first iteration are shown. Patches that are identified as outliers by the Patch filtering algorithm, Section 3.4, are removed from the point model. Figure 5.43 shows the filtering results for the Skull model. The filters are aggressive and many patches are removed, some possibly true positives. However, subsequent iterations of the expansion stage will ensure that some of these true positives are reconstructed. Table 5.4 shows the number of patches removed by the filtering stage at each iteration.

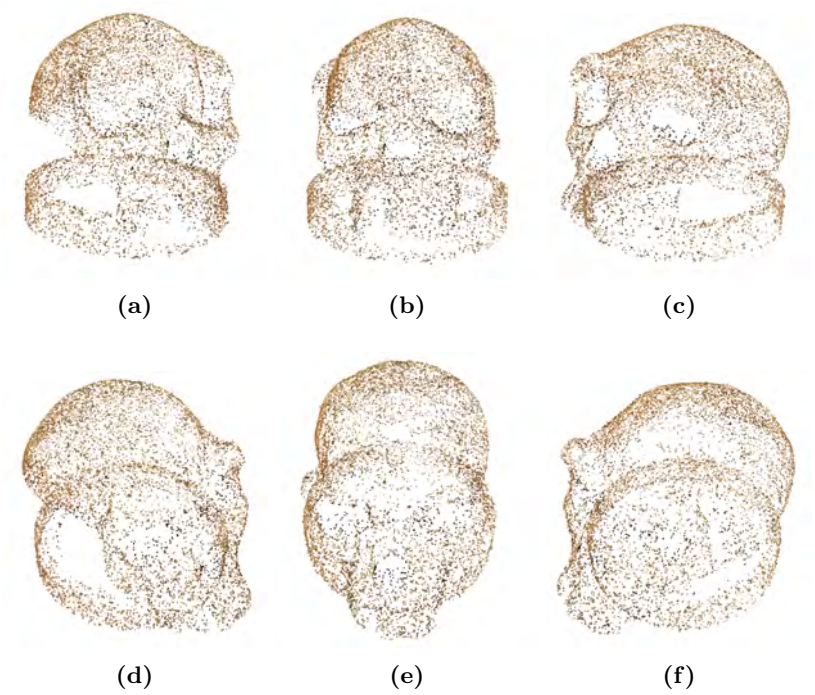


Figure 5.43: Patch Filtering results: Skull - Different views of the reconstructed patches after performing patch filtering on the Skull dataset.

Figure 5.44 and 5.45 present filtering results for the Dino and Temple models. Most noticeable with these two models is the reduction of patches that project outside the object surface (black patches). However as stated before the effect of the expansion may have classified some of these outliers as part of the surface. This effect is reduced by iterating the expansion and filtering stages a few times.

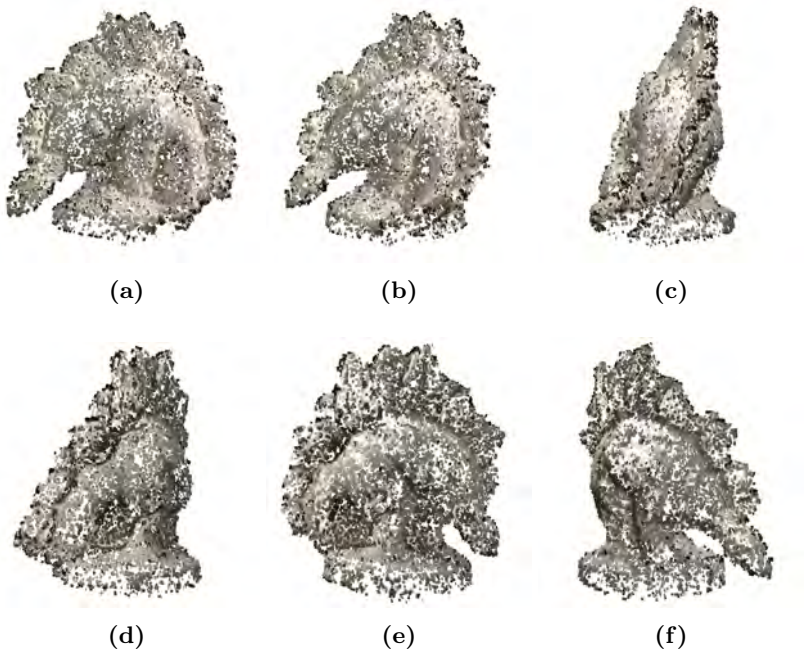


Figure 5.44: Patch Filtering results: Dino - Different views of the reconstructed patches after performing patch filtering on the Dino dataset.

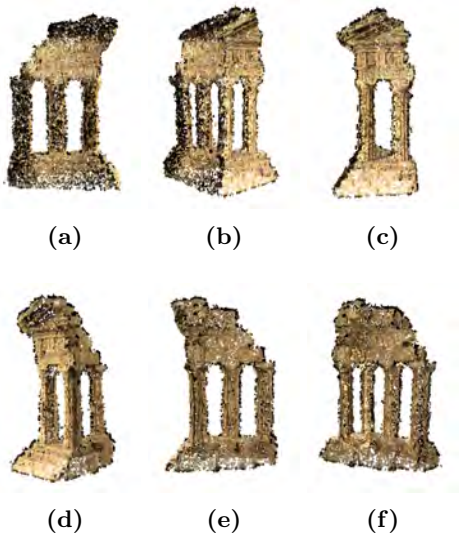


Figure 5.45: Patch filtering results: Temple - Different views of the reconstructed patches after performing patch filtering on the Temple dataset.

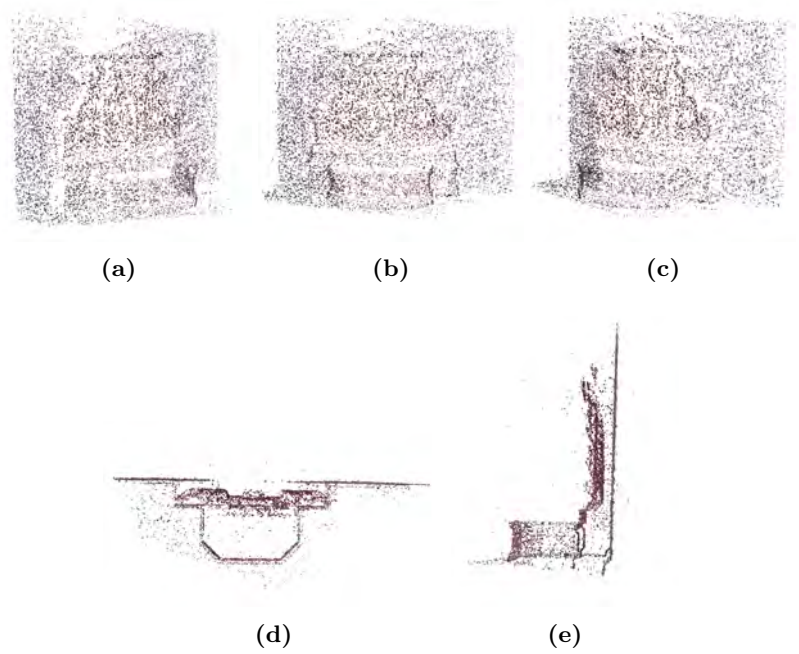
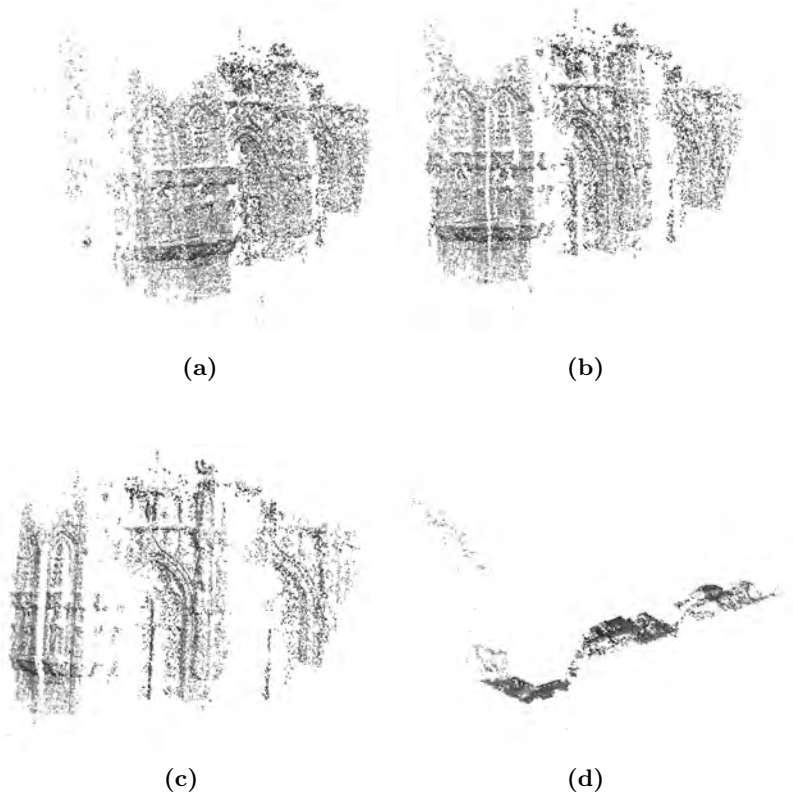


Figure 5.46: Patch Filtering results: Fountain - Different views of the reconstructed patches after performing patch filtering on the Fountain dataset.

For the scene models Figures 5.46 and 5.47, it is more difficult to judge the removal of outliers as the distance between adjacent groups of patches can be large. Nevertheless the filters do perform well with many of the outlying patches being removed. Figures 5.46d, 5.46e and 5.47d illustrate this.



(a) (b)

(c) (d)

Figure 5.47: Patch Filtering results: City Hall - Different views of the reconstructed patches after performing patch filtering on the City Hall dataset.

5.3.4 Patch Reconstruction Results

This section presents the final reconstructed patches after being formed by the Initial Feature Matching stage and then repeatedly expanded and cleaned by the Patch Expansion and Filtering stages. The final patch models should be dense enough and contain as few outliers as possible to enable accurate reconstruction of the final mesh models.

Figure 5.48 illustrates the results of the patch reconstruction algorithm for the Skull dataset. The model is dense and accurate enough to enable visualisation of the surface using only the reconstructed point model. Regions where the surface contains bad texture or is not covered by enough cameras are not present in the final models. These regions are seen as holes in the point model.

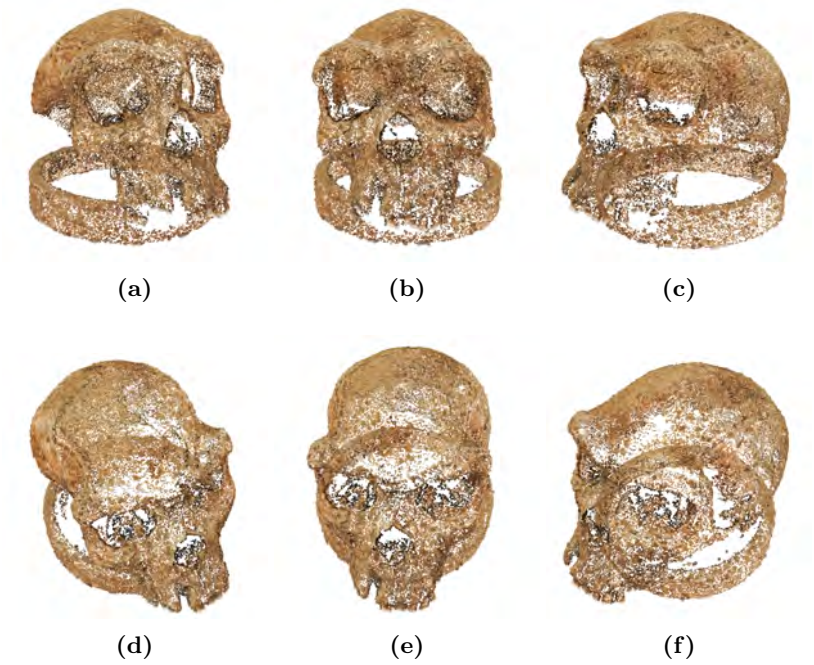


Figure 5.48: Patch Reconstruction results: Skull - Different views of the final reconstructed patches after performing feature matching, expansion and filtering on the Skull dataset. Note that the object appears to be translucent. This is because the surface is formed by unconnected groups of patches. Hence patches from surfaces to the 'rear' are visible through the gaps in the 'front' surface.

Results for the Dino, Figure 5.49, and Temple, Figure 5.50, show that the surfaces have been covered well, however there are still a few regions where the extent of the patch projects outside the objects surface. Optimisation of the patches independently of each other is the main cause of this type of error. The Final Mesh Refinement algorithm overcomes this problem by optimising the vertices taking into account the vertex neighbours.

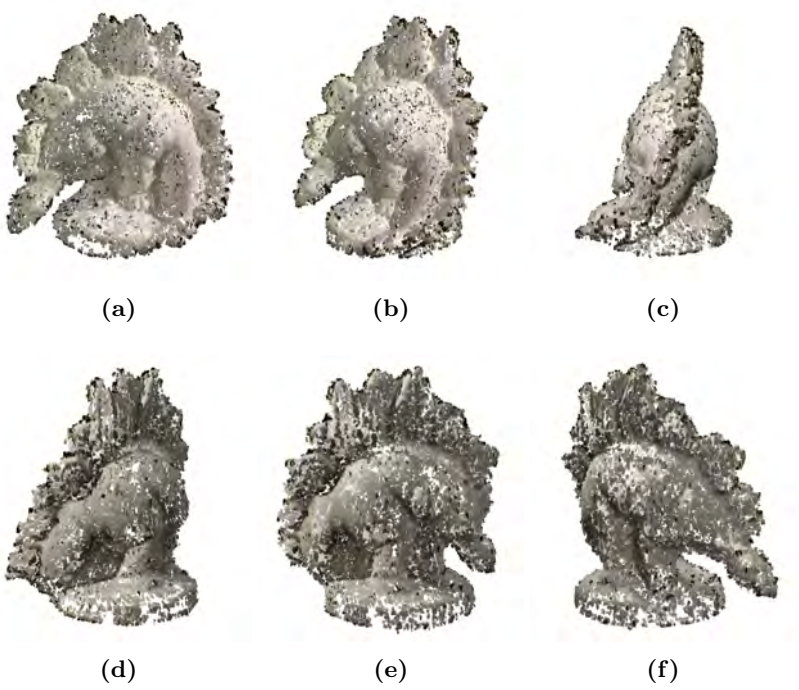
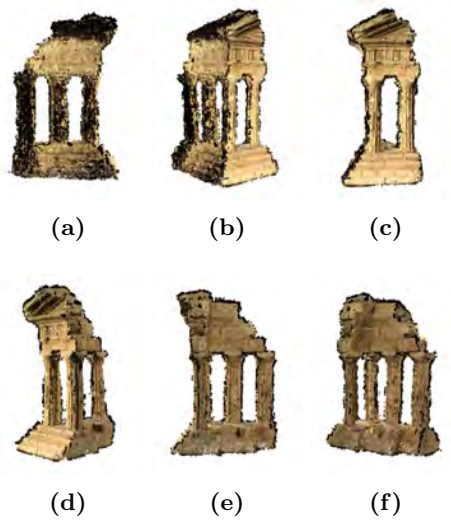


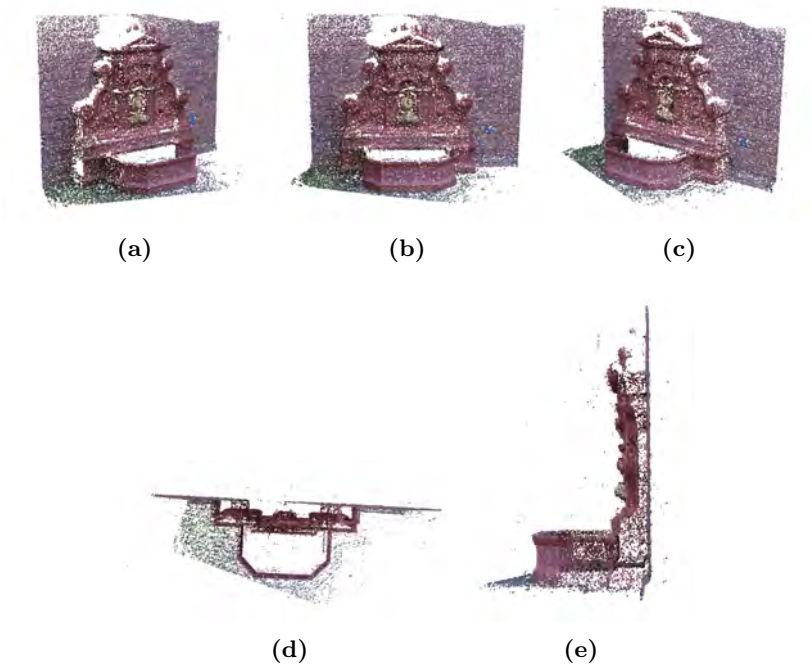
Figure 5.49: Patch Reconstruction results: Dino - Different views of the final reconstructed patches after performing feature matching, expansion and filtering on the Dino dataset.



(a) (b) (c)

(d) (e) (f)

Figure 5.50: Patch Reconstruction results: Temple - Different views of the final reconstructed patches after performing feature matching, expansion and filtering on the Temple dataset.

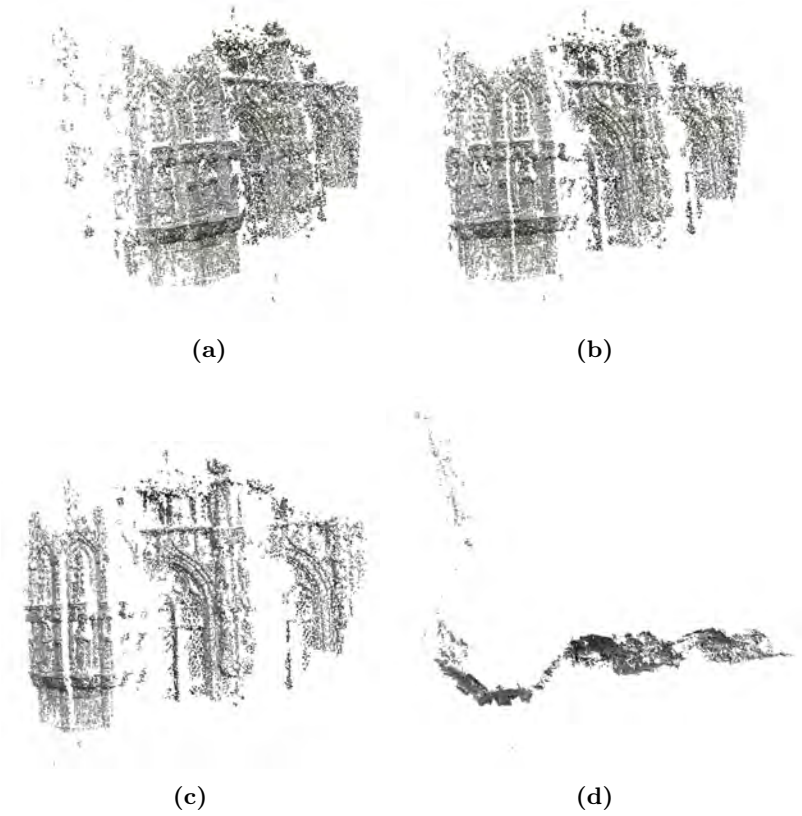


(a) (b) (c)

(d) (e)

Figure 5.51: Patch Reconstruction results: Fountain - Different views of the final reconstructed patches after performing feature matching, expansion and filtering on the Fountain dataset.

Figures 5.51 and 5.52 show the final reconstructed patch model for the Fountain and City Hall datasets. For both these models the scene surface has been recovered well with only a few outliers present. These outliers are negligible and are ignored by the Poisson Surface Reconstruction stage. Table 5.4 shows the number of patches generated by the patch reconstruction algorithm.



(a) **(b)**

(c) **(d)**

Figure 5.52: Patch Reconstruction results: City Hall - Different views of the final reconstructed patches after performing feature matching, expansion and filtering on the City Hall dataset.

5.3.5 Mesh Reconstruction Results

This section presents results for the mesh reconstruction algorithms. Mesh reconstruction is performed in two stages. The first is a mesh initialisation process where a triangular mesh model is initialised using either the object's visual hull or by utilizing the Poisson Surface reconstruction technique. Thereafter the initialised mesh is refined to improve the overall accuracy of the model. Figures 5.53, 5.54 and 5.55 show different views of the mesh models initialised using the corresponding visual hulls for the Skull , Dino and Temple models respectively. The visual hulls were obtained using the Voxel Carving technique presented in Section 5.2.7. The resulting mesh models approximate the surface as close as possible and work well for convex regions. However, concave regions such as the eyes of the Skull model, stairs of the Temple and feet of the Dino model cannot be recovered using this technique. Nevertheless the resulting mesh models form a good starting point for the refinement algorithms. Table 5.5 shows the number of vertices initialised for each dataset.

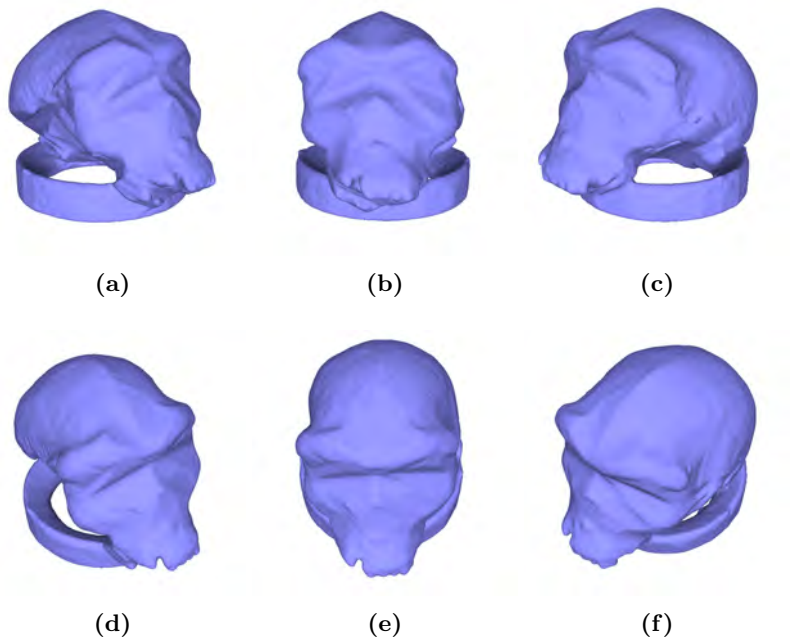


Figure 5.53: Mesh Reconstruction results: Skull - Different views of the visual hull model for the Skull dataset. Note that because this is a closed surface it no longer appears translucent.

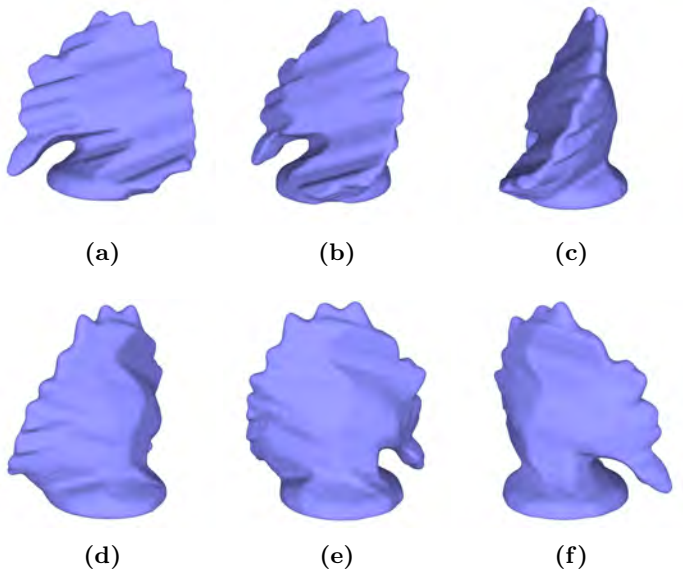


Figure 5.54: Mesh Reconstruction results: Dino - Different views of the visual hull model for the Dino dataset.

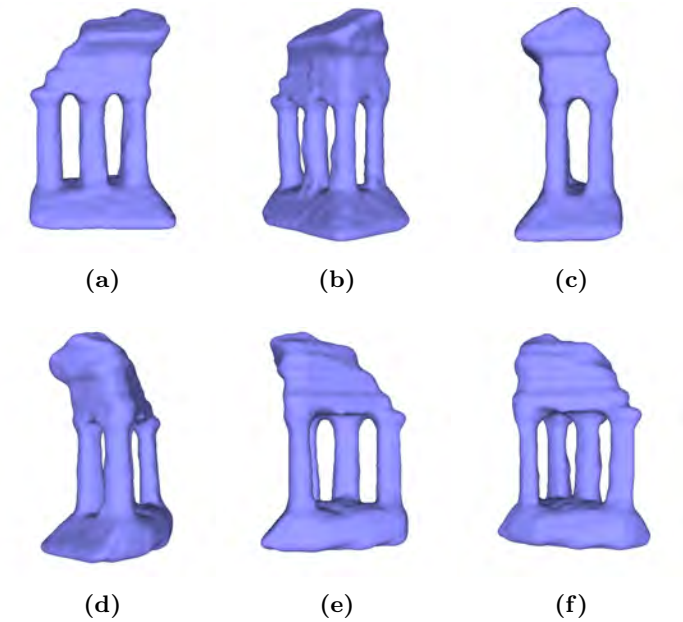


Figure 5.55: Mesh Reconstruction results: Temple - Different views of the visual hull model for the Temple dataset.

Mesh models for scene datasets are initialised using the Poisson Surface reconstruction algorithm developed by Kazhdan *et al.* [59; 84]. Figures 5.56 and 5.57 show the cleaned output of the PSR algorithm for the Fountain and City Hall Models. The resulting meshes were cleaned by removing triangles with an edge length larger than six times the average edge length of the entire mesh. The output of the PSR software produces high quality mesh models with only a small amount of noise present, seen as bumps on the surface. These models form a good starting platform for the Mesh refinement stage. Table 5.5 shows the number of vertices initialised using the PSR software for these two datasets.

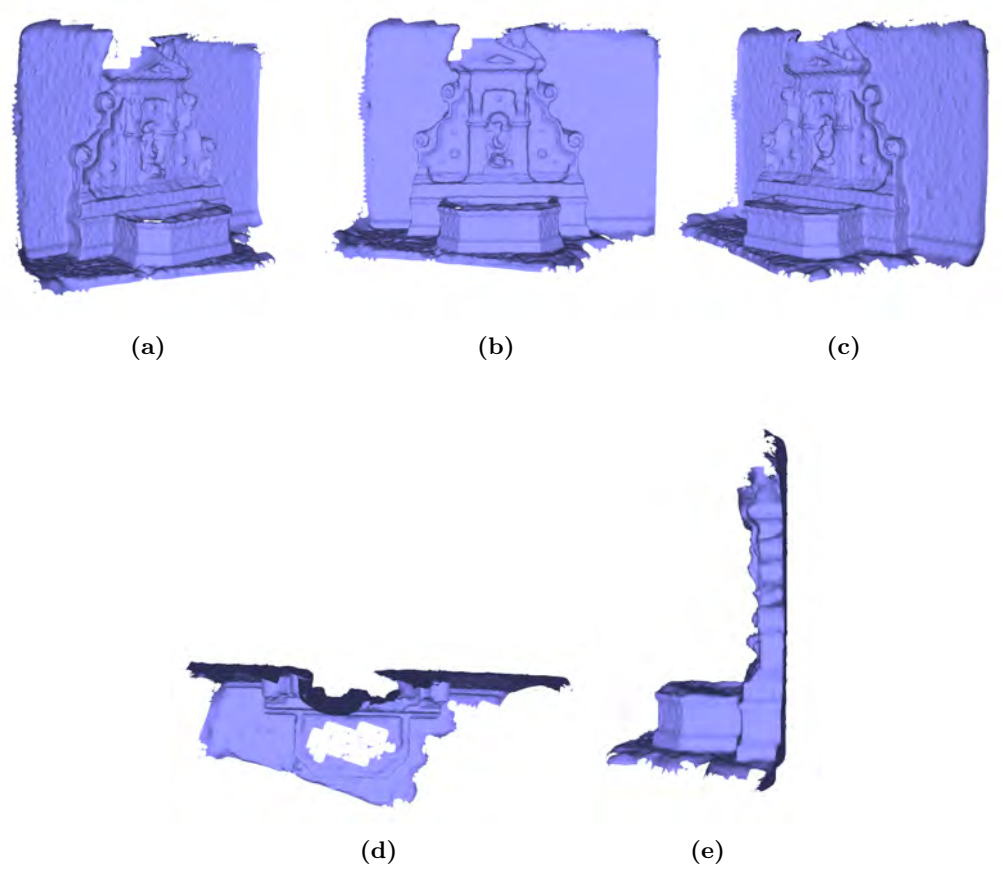
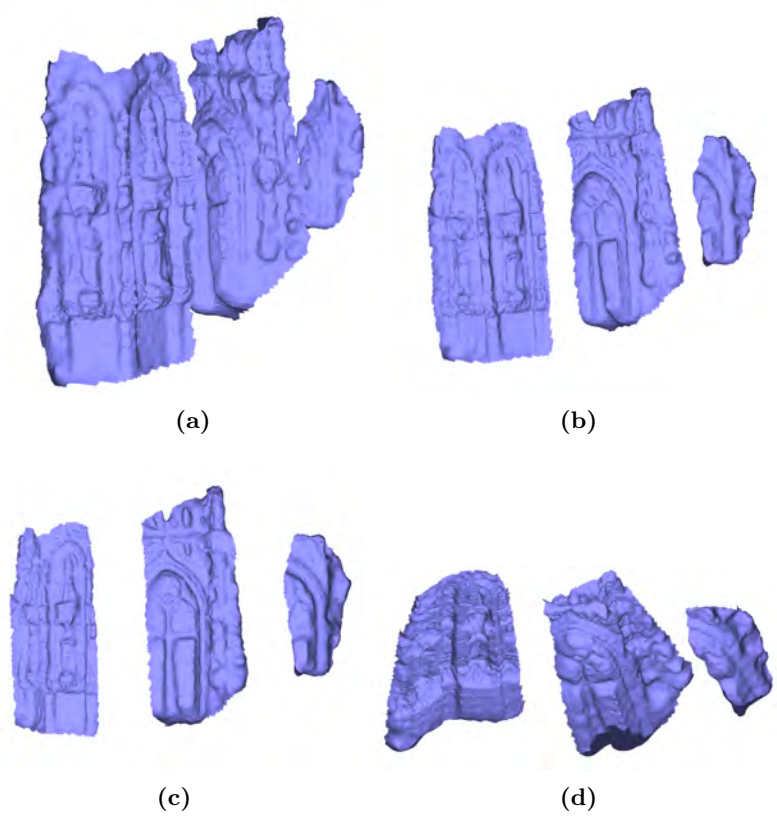


Figure 5.56: Mesh Reconstruction results: Fountain - Different views of the surface generated by the Poisson Surface Reconstruction algorithm for the Fountain dataset.

Table 5.5: Mesh reconstruction results - vertices generated after each stage in the mesh reconstruction algorithm.

Dataset	Visual Hull/Poission	Iterative Snapping	Mesh Refinement	Time (H)
Skull	171098	167020	241520	12
Dino	150901	143158	143752	10
Temple	106828	88004	89470	6
Fountain	204110	-	262734	13
City Hall	136649	-	345614	15



(a) **(b)** **(c)** **(d)**

Figure 5.57: Mesh Reconstruction results: City Hall - Different views of the surface generated by the Poisson Surface Reconstruction algorithm for the City Hall dataset.

5.3.6 Iterative Snapping Results

In this section the results of the Iterative Snapping algorithm for object datasets are presented. Figure 5.58 illustrates different views of the output of the algorithm for the Skull model. The resulting model is noticeably more accurate than the visual hull model with the deep concavities of the eyes clearly visible. The iterative snapping algorithm is even able to recover the fine surface details such as the teeth and the sutures [92] visible in Figures 5.58c, 5.58d, 5.58e and 5.58f. The notch at the bottom of the nose is also recovered well. However the top of the nose is not, this is because there are no patches reconstructed for that region, see Figure 5.48. The mesh also contains a few noisy regions, mainly on the top of the skull seen in Figures 5.58d, 5.58e and 5.58f. These bumps on the surface are caused by errors in the reconstructed patch model.

Similar results (assessed visually) were obtained for the Dino, Figure 5.59, and Temple, Figure 5.60, models. Most notably the algorithm is able to recover the deep concavities near the feet of the Dino, the structure of the stairs and inner wall of the Temple model.

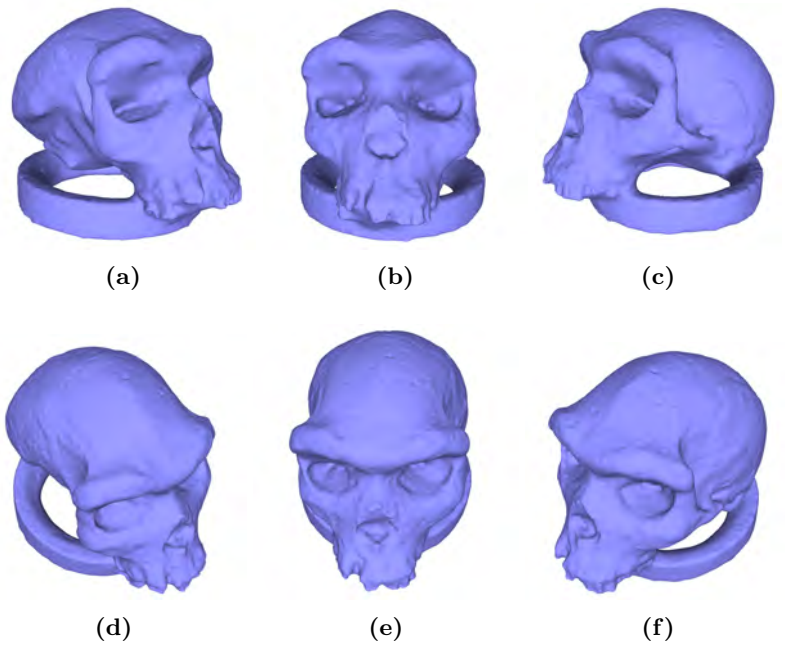


Figure 5.58: Iterative Snapping results: Skull - Different views of the mesh model for the Skull dataset after Iterative Snapping.

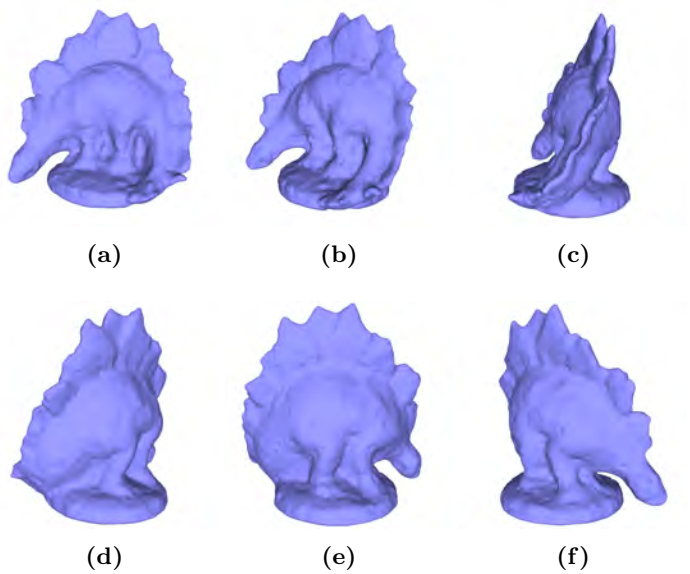


Figure 5.59: Iterative Snapping results: Dino - Different views of the mesh model for the Dino dataset after Iterative Snapping.

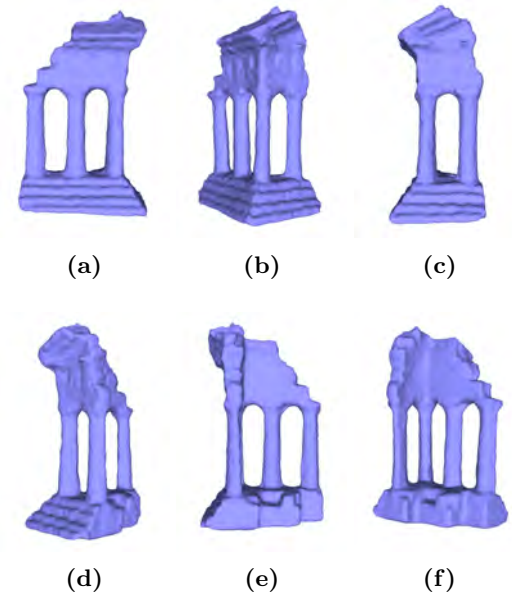


Figure 5.60: Iterative Snapping results: Temple - Different views of the mesh model for the Temple dataset after Iterative Snapping.

Table 5.5 shows the number of vertices after refinement by the iterative snapping algorithm. Note that the total number of vertices decrease because of the re-meshing operations.

5.3.7 Mesh Refinement Results

This section presents the Mesh refinement results for all the datasets. The mesh refinement algorithm, Section 4.4 aims to recover fine surface details that may have been overlooked by the Iterative snapping or Poisson surface reconstruction algorithms.

Figure 5.61 illustrates the results of mesh refinement on the Skull model. The final model appears smoother and more refined when compared to the Iterative Snapping results of Figure 5.58. Problematic regions such as the bumps on the top of the head (which can be seen upon close inspection of the image dataset), have been successfully recovered as shown in Figures 5.58d, 5.58e and 5.58f.

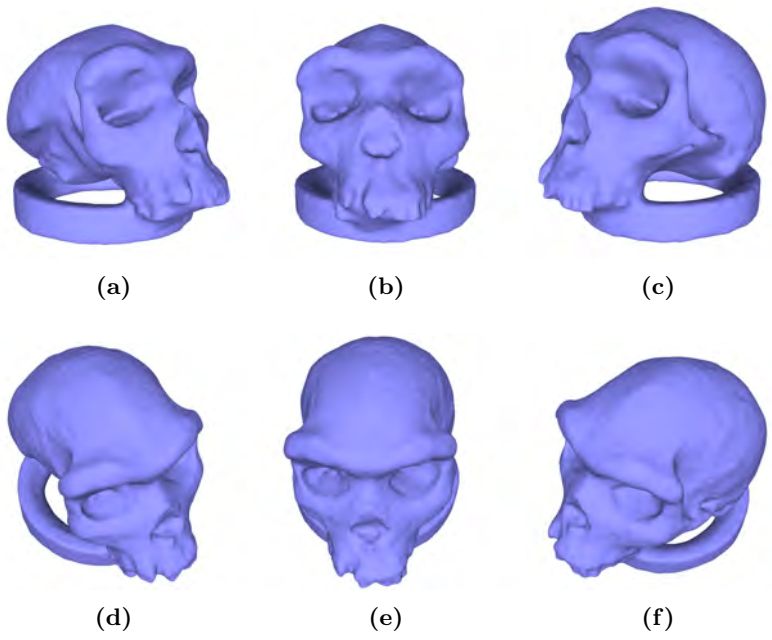


Figure 5.61: Mesh Refinement results: Skull - Different views of the mesh model for the Skull dataset after Mesh Refinement.

The results obtained for the Dino and Temple models, Figures 5.62 and 5.63 respectively, show similar improvements. However these models still contain some noisy regions. These may be improved by iterating the algorithm for a longer period.

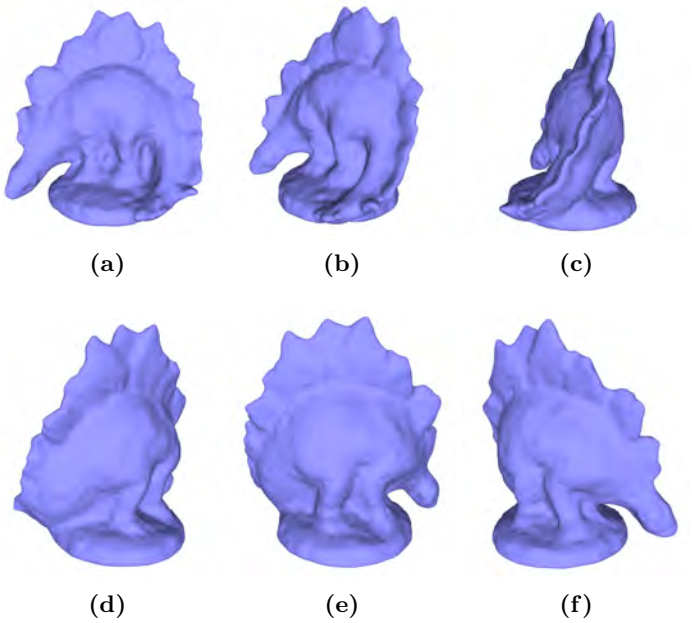


Figure 5.62: Mesh Refinement results: Dino - Different views of the mesh model for the Dino dataset after Mesh Refinement.

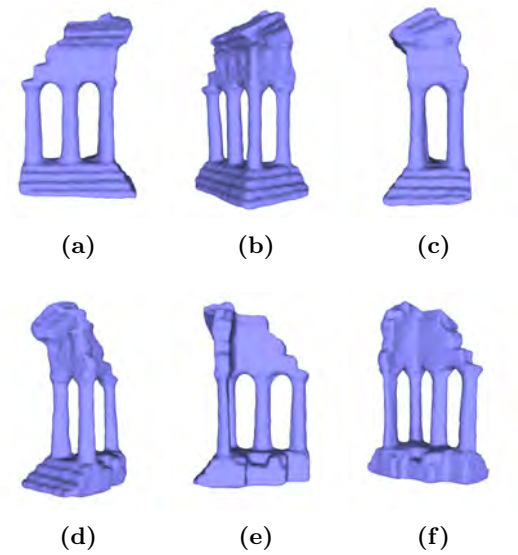


Figure 5.63: Mesh Refinement results: Temple - Different views of the mesh model for the Temple dataset after Mesh Refinement.

The Mesh Refinement results for the Fountain, Figure 5.64 and City Hall, Figure 5.65, models show a marked improvement (assessed visually) in terms of reduction of noise on the surfaces.

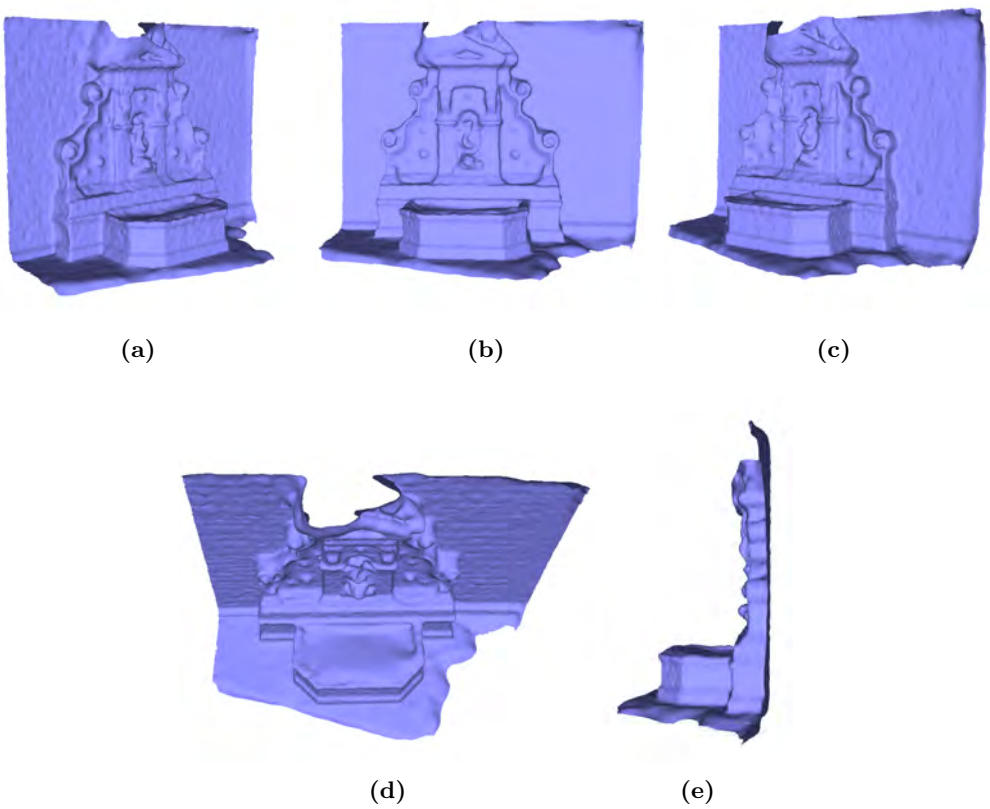
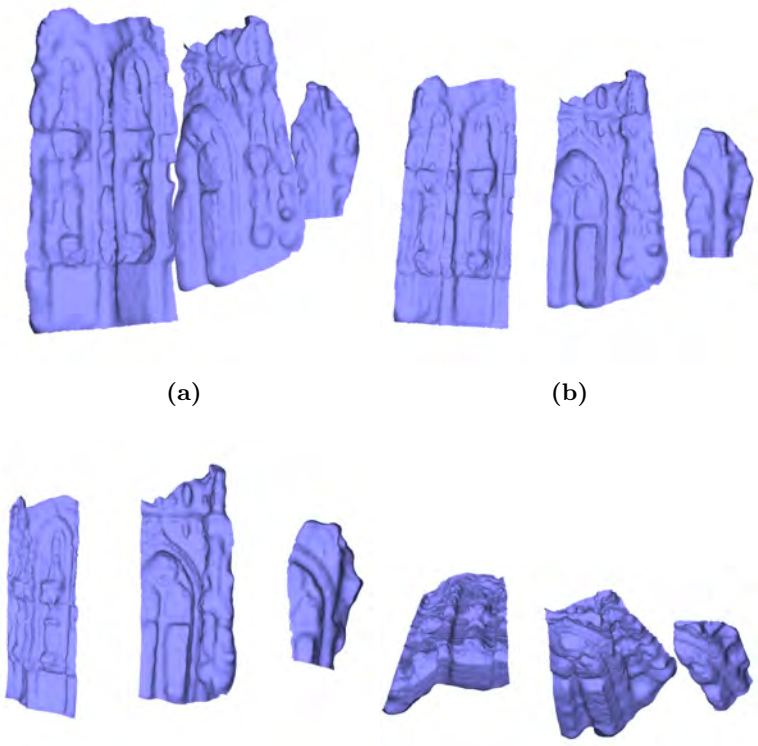


Figure 5.64: Mesh Refinement results: Fountain - Different views of the mesh model for the Fountain dataset after Mesh Refinement.

Table 5.5 shows the number of vertices after mesh refinement on each dataset. Note that the number of vertices increases due to mesh re-sampling and re-meshing operations once every 5 iterations.



(a) **(b)**

(c) **(d)**

Figure 5.65: Mesh Refinement results: City Hall - Different views of the mesh model for the City Hall dataset after Mesh Refinement.

5.3.8 Texture-mapped Models

This section presents the final texture-mapped models for each of the datasets. Mapping the object's texture onto the point models from the Patch reconstruction stage is relatively straightforward. One simply projects the patch centre into its corresponding reference image and samples the image colour. However, the vertices of the mesh models do not inherently have reference images. To determine the reference camera for a particular vertex, the closest camera with the most compatible normal is chosen. That is the camera which is closest to the vertex with camera normal most in-line with the vertex normal. In practice this is achieved by selecting the camera with the smallest[1] dot product between the vertex normal and each of the camera normals.

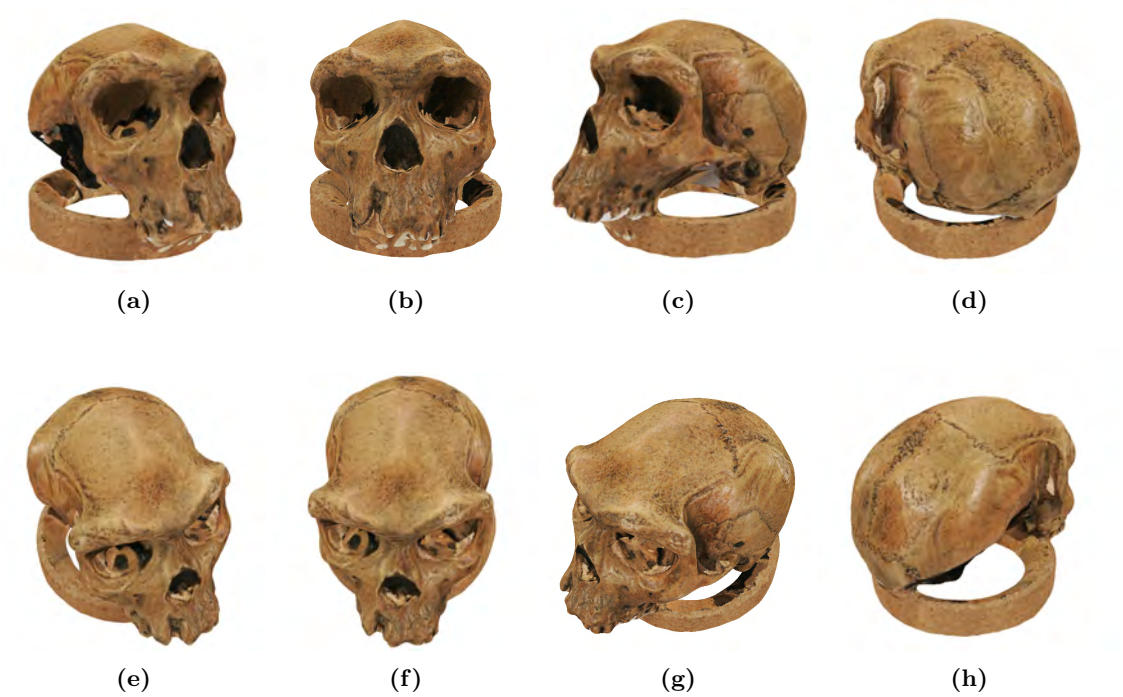


Figure 5.66: Final Texture-mapped model: Skull - Different views of the Final Texture-mapped model for the Skull dataset.

Figure 5.66 shows different views of the texture-mapped Skull model. The texture-mapping process is accurate for most of the model however for regions where the surface normal is not compatible with any of the cameras the mapping process fails and incorrect surface texture is shown.

[1]The smallest value is chosen because the camera normals point toward the object. Hence the most compatible camera will have the largest negative dot product with the vertex normal.

The texture-mapped model for the Dino dataset is shown in Figure 5.67. Here the mapping process is almost perfect, except for small regions near the dinosaur's hind legs (circled in red in Figures 5.67a and 5.67b). For the Temple model, Figure 5.68, the only noticeable flaw is seen in 5.68d (circled in red) and 5.68e, where the incorrect texture is chosen for the square section at the bottom of the model. This occurs because the rays between the closest compatible camera and the points in this region are occluded by the pillar (seen in Figure 5.68d). This type of occlusion is difficult to detect and correct especially if the coverage of the cameras is limited.

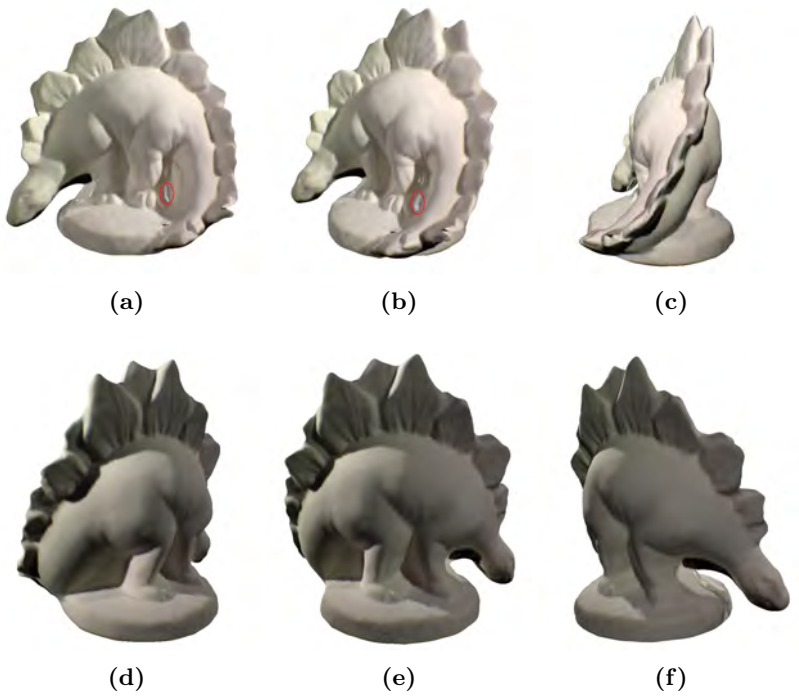


Figure 5.67: Final Texture-mapped model: Dino - Different views of the Final Texture-mapped model for the Dino dataset. Circled regions indicate errors caused by occlusion.

The results for the Fountain, Figure 5.69, and City Hall, Figure 5.70, models show just how accurate the modelling process can be for these large scene datasets. These datasets comprise high resolution images taken at irregular and wide viewing angles. When texture-mapped the models take on a very realistic appearance. Black regions in the models are caused by the surface points projecting outside the reference image i.e. the reference image contains no information for that region. Again the texture-mapping process does suffer from minor occlusion. Regions such as the one to the left of the head of the fish circled in Figures 5.69b and 5.69c are easily identifiable.

However the texture mapping process does perform extremely well and is even able to map the side of the Fountain Figure 5.69e

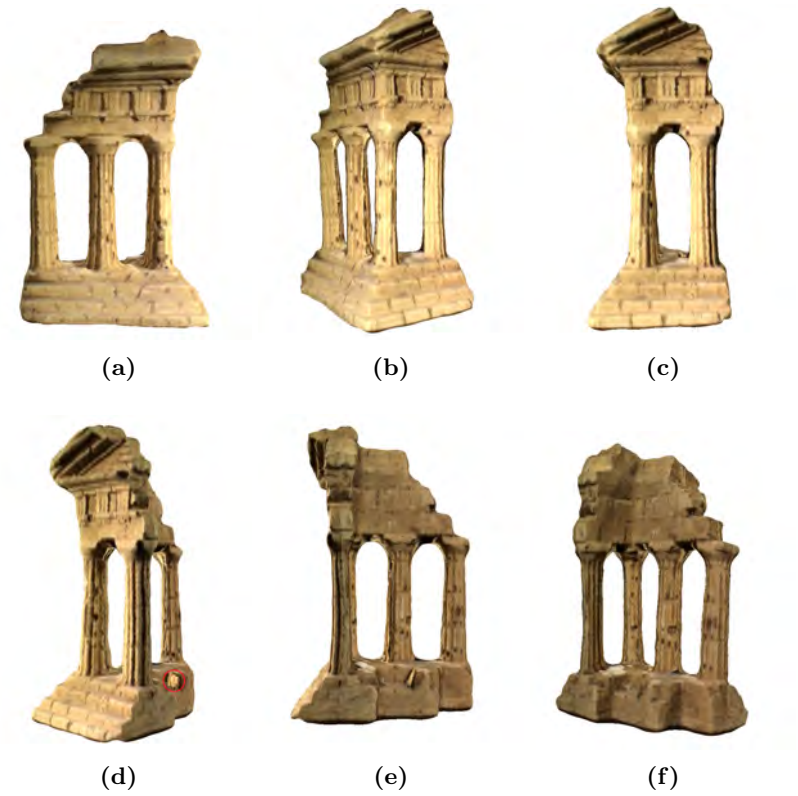
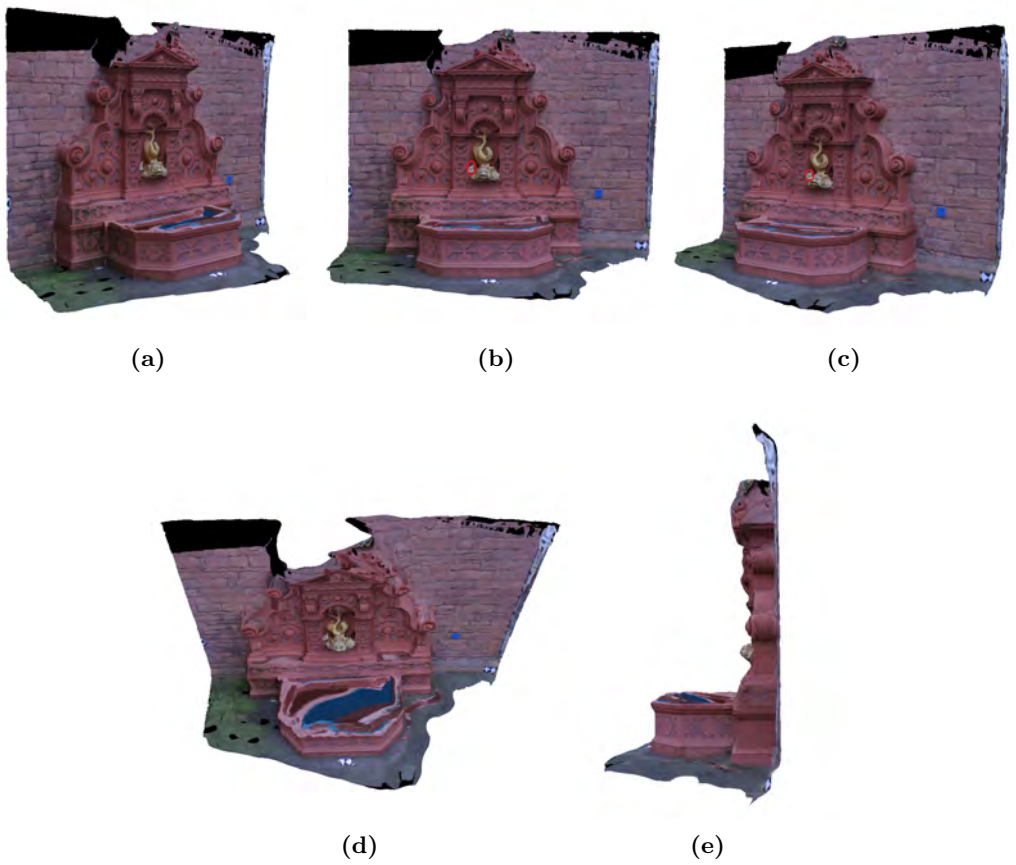


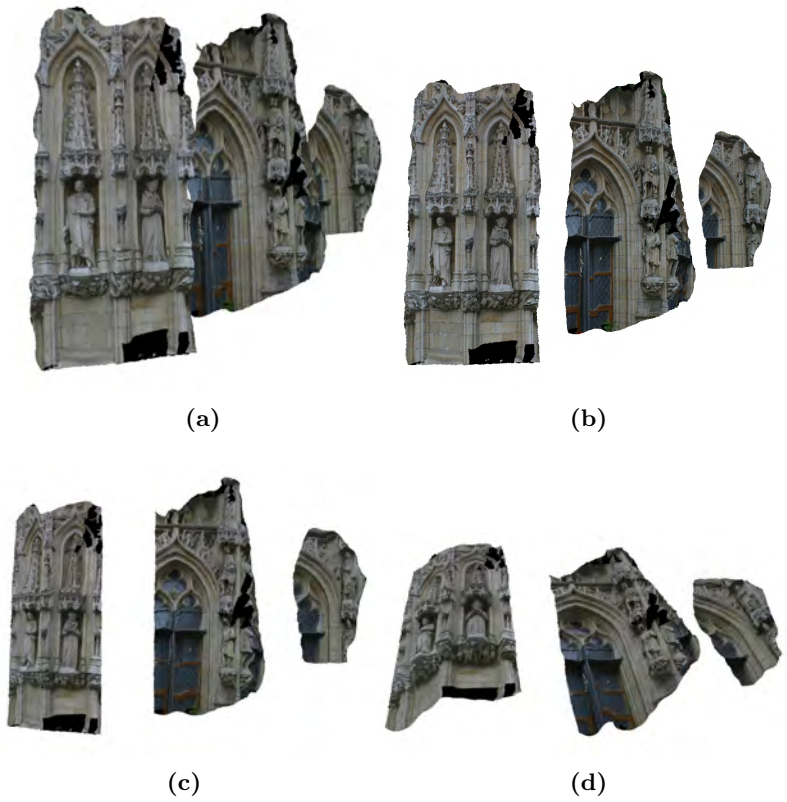
Figure 5.68: Final Texture-mapped model: Temple - Different views of the Final Texture-mapped model for the Temple dataset. Circled regions indicate errors caused by occlusion.



(a) **(b)** **(c)**

(d) **(e)**

Figure 5.69: Final Texture-mapped model: Fountain - Different views of the Final Texture-mapped model for the Fountain dataset. Circled regions indicate errors caused by occlusion.



(a) **(b)**

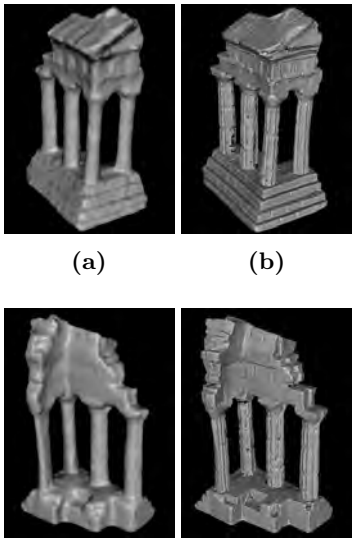
(c) **(d)**

Figure 5.70: Final Texture-mapped model: City Hall - Different views of the Final Texture-mapped model for the City Hall dataset.

5.3.9 System Evaluation

Thus far, the quality of the reconstructed patch and mesh models were assessed visually. This section presents the performance of the reconstruction algorithm against ground truth models. One of the biggest challenges facing multi-view stereo reconstruction is the lack of ground truth data on which to test, evaluate and develop algorithms. The Middlebury evaluation [93] aims to address these issues by providing high quality datasets together with a platform for evaluating multi-view reconstruction results. The evaluation focuses on object datasets that contain challenging characteristics such as both sharp and smooth features, complex topologies with strong concavities and both strong and weak textured surfaces [24]. The first object is a plaster replica (10cm×16cm×8cm) of the "Temple of the Dioskouroi" [94]. The second is a plaster replica (7cm×9cm×7cm) of a stegosaurus dinosaur. For each of these models three types of datasets are available. The *full* dataset contains 312 images for Temple and 363 images for Dino (640×480 pixels) sampled on a hemisphere. The *ring* dataset contains 47 images for Temple and 48 images for Dino, sampled in a horizontal ring around the object and the *sparse ring* datasets have 16 views each. This dissertation uses the *ring* datasets for both models as it provides a good balance between reconstruction time and accuracy.

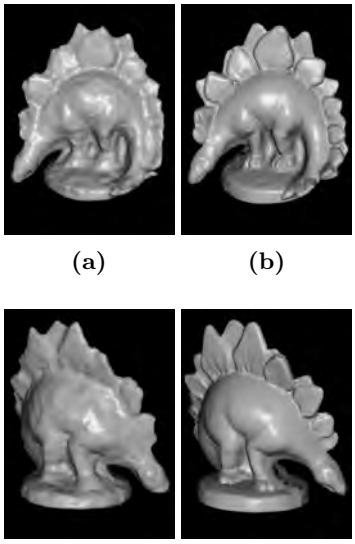
The evaluation measures the performance of reconstruction algorithms using accuracy and completeness as metrics. Accuracy is defined as how close the reconstructed model, R, is to the ground truth, G, and is computed as the signed distance d (in mm) such that 90% of the reconstruction is within d of the ground truth mesh (GTM). Completeness is defined as how much of G is modelled by R and is computed as the percentage of points on the GTM that are within 1.25 mm of the reconstruction, see [24] for details. Figure 5.71 shows the reconstructed Temple model alongside the laser scanned ground truth model. The results of the evaluation on the Temple model show that the reconstruction is accurate to 1.03 mm and 88.8 % complete. This means that 90 % of the reconstructed points are within 1.03 mm of the ground truth model (GTM) and 88.8 % of the points on the GTM are within 1.25 mm of the reconstruction [24]. The results for the Dino model are slightly better with an accuracy of 0.84 mm and completeness of 92.5 %. These results are reasonably accurate considering that one pixel spans approximately 0.25 mm of the objects surface [24].



(a) (b)

(c) (d)

Figure 5.71: Reconstructed Temple model VS ground truth - Two views of the reconstructed Temple model 5.71a, 5.71c and ground truth 5.71b, 5.71d.



(a) (b)

(c) (d)

Figure 5.72: Reconstructed Dino model VS ground truth - Two views of the reconstructed Dino model 5.72a, 5.72c and ground truth 5.72b, 5.72d.

Figure 5.73 shows histograms of the signed errors for the accuracy, Figure 5.73a and 5.73b, and completeness, Figure 5.73c 5.73d of the models. For each vertex in R the sign of the distance is determined by taking the dot product of the outward facing normal of the nearest point on G and the vector from that point to the vertex on R [24]. This gives an indication of whether the reconstruction tends to under- or over-estimate the true shape of the surface. For both models the histograms indicate that the reconstructions tend to slightly over-estimate the surface as the majority of the reconstructed points lie inside the ground truth model. This effect is very slight and is most likely due to the over smoothing effect of the Laplacian filters. Figure 5.71 confirms this as the reconstructed models appear over smoothed and thus some very fine surface details have been lost. An online comparison against other state-of-the-art reconstruction algorithms can be found at [95], the comparison is best viewed online as it is updated regularly and contains many interactive tables and figures.

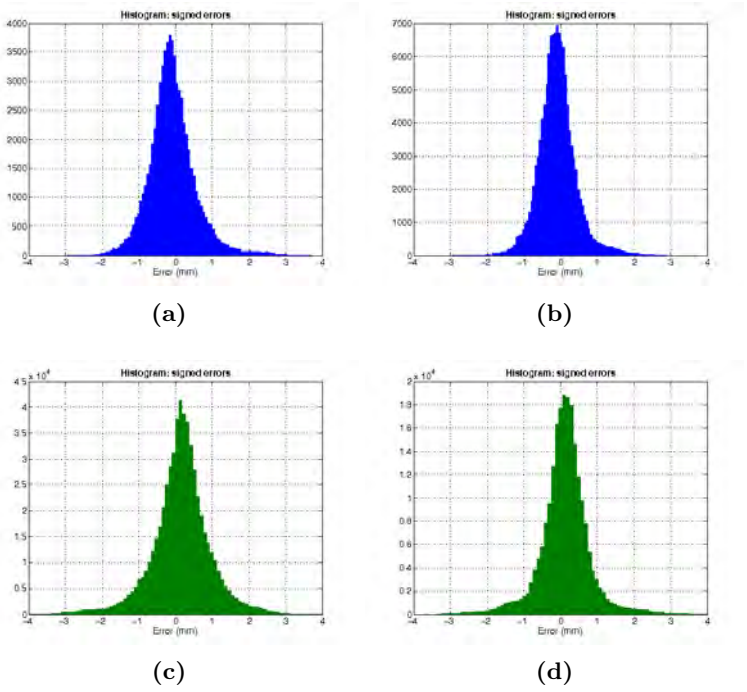


Figure 5.73: Histogram of signed errors - Signed accuracy graphs 5.73a, 5.73b and Signed completeness graphs 5.73c, 5.73d for the Temple and Dino models respectively.

5.4 Summary

The implementation and results of the 3D reconstruction algorithm is the focus of this Chapter. Results are presented in two stages. The intermediate results details the implementation of each stage within the reconstruction pipeline. Thereafter the final results are presented and a quantitative evaluation in terms reconstruction accuracy and completeness is made.

The Initial Feature Matching process consists of two main stages, Feature detection and Feature matching. Feature detection is achieved using the difference of Gaussian and Harris feature detectors. These detectors are able to detect corner, edge and blob features successfully and are robust to changes in rotation and variations in illumination. To achieve uniform coverage of features over the object, η features are selected within a uniform $\beta_1 \times \beta_2$ pixel grid covering the surface. The detected feature points are then matched by the IFM algorithm. For each feature a set of candidate matching features are identified in the subsequent images. To improve the probability of selecting the correct matching feature first, the candidate features are sorted in order of increasing photometric discrepancy. Triangulation between the two features results in a 3D point/patch on the objects surface. For the patch to be successfully reconstructed the 3D properties of the patch are optimised with respect to its visible images. To avoid local minima in the optimisation stage, an improved estimate of the initial orientation of the patch is made whereby the surface normal of the patch is reorientated towards the cameras with low photometric discrepancy. If the number of visible images after optimisation is greater than τ, the patch is said to truly lie on the objects surface. This process is iterated for each feature resulting in a set of patches covering the surface of the model. An Expansion and Filtering stage is then iterated 3 times to increase the density and remove outliers in the patch model. Final results of the patch reconstruction stage indicate that the models are sufficiently dense and contain a minimal number of outliers.

Conversion of the orientated point model to a mesh model is achieved by the mesh reconstruction stage. For object datasets a surface mesh is initialised using the object's visual hull. The initialised mesh is then refined by the iterative snapping technique, which has the effect of moving the mesh vertices towards the reconstructed patches. For scene datasets the surface mesh is initialised using the Poisson surface reconstruction technique. Both initialisation techniques perform well and the final results show that most of the surface structure including deep concavities are recovered. A final refinement stage is then used to recover any fine surface details, and remove any noise that

may have been overlooked by the Iterative snapping or Poisson surface reconstruction algorithms. The final mesh models are dense with most containing more than 100k vertices. To give the models a realistic appearance, a texture mapping process is carried out. Each vertex is projected into its reference image and image colours are sampled. Minor errors in the mapping process stem from incorrect or indeterminable reference images. These errors are unavoidable because the image dataset does not cover every possible view of the surface.

An assessment of the performance of the reconstruction algorithm is made in terms of reconstruction accuracy and completeness. Two reconstructed models, Temple and Dino, were evaluated against laser scanned ground truth models by the Middlebury evaluation. Both objects contain challenging characteristics such as both sharp and smooth features, complex topologies with strong concavities and both strong and weak textured surfaces. The results of the evaluation show that the reconstructed models are accurate to within 1.03 mm and 88.8% complete for the Temple model and 0.84 mm accurate and 92.5 % complete for the Dino model. Signed histograms of the accuracy show that the models tend to slightly under estimate the objects surface. This can be atributed to an over smoothing effect of the Laplacian filters. Nevertheless the results are in line with other state-of-the-art reconstruction algorithms in the evaluation.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

A robust method for estimating the rigid 3D structure of objects and scenes from calibrated images has been presented. Estimation of accurate 3D models is one of several key challenges facing analysis at the encoder of model-based video encoding systems. The proposed approach is founded on state-of-the-art computer vision techniques and combines several methodologies to achieve robustness to occlusions and changes in illumination.

MVS reconstruction algorithms can be classified into four classes according to a number of attributes such as *initialisation requirements*, *scene representation*, *photo-consistency measures* and *visibility models*. Each class has its strengths and the proposed Patch-based algorithm draws on the strengths of Voxel-based methods and Polygonal surface-based methods to achieve its flexibility, robustness and accuracy. A two stage surface representation is employed for the initial modelling and the final surface refinement. Initial surface modelling is performed using an unconnected patch model that enables modelling of arbitrary complex topologies. The patch model is then transformed into a mesh model for final surface refinement using mesh refinement techniques. This approach increases the flexibility and accuracy of the reconstruction algorithm. Robustness to occlusions and changes in illumination are addressed by using a scene space photoconsistency measure in combination with normalized cross correlation. Occlusions are also dealt with by not selecting images for the visibility model that have a photometric score with the reference image that is below a certain threshold.

Feature detection is performed using the Harris and Difference of Gaussian feature detectors which are robust to rotation and variances in illumination. Each identified feature is matched in successive images in an attempt to generate a patch on the object's surface. To ensure that the Feature Matching process is robust to outliers, candidate features are sorted in increasing order of photometric discrepancy. A further improvement is made to avoid local minima in the optimisation stage by re-orientating the patch (surface normal) towards cameras with low photometric discrepancy with the reference image. The density of the patch model is increased and outliers are removed by the Expansion and Filtering stages. The Expansion stage first identifies, for each patch, vacant neighbouring cells that are fit for expansion and attempts to generate a patch for each cell. Three filters are employed by the filtering stage to remove outliers in the patch model. The first two filters rely on visibility consistency to remove outliers near the surface of the model whilst the third filter uses a weak form of regularization and removes a patch if its proportion of neighbours is below 0.25. The Expansion and Filtering stages are iterated 3 times to increase the density and effectively remove outliers. To enable reconstruction of challenging regions the correlation matching threshold α is relaxed by 0.2 after each iteration.

The Patch-Based surface representation is useful for modelling arbitrary complex topologies. However, its unconnected point representation makes performing image based modelling tasks such as smoothing or producing a closed surface, difficult. For this reason, the orientated patch model is converted into a surface mesh model to enable further refinement using mesh based refinement techniques. Two different approaches are taken depending on the availability of accurate segmentation information. Surface meshes for objects datasets are initialised at the boundary of the visual hull and are refined using the Iterative Snapping technique. The position of each vertex is optimised using the sum of a geometric smoothness function and patch reconstruction consistency term. For scene datasets, where segmentation information is unavailable, the Poisson surface reconstruction technique is used. The PSR algorithm directly converts an orientated point model into a surface mesh model, where the size of the mesh triangles depends on the density of the reconstructed points. For both datasets a final mesh refinement process is carried out to recover fine surface details and reduce noise that may still be present.

Results for the feature detectors show that similar feature points are detected in different views of the object approximately 45° apart. Hence, the feature detectors are particularly robust to rotation and variances in illumination. A uniform distribution of features are selected by overlaying

a grid onto each image and selecting η features within each block. The feature point selection process of rearranging candidate matching points in order of increasing photometric discrepancy has proven to avoid reconstruction of incorrect matching points and thus reduce the number of outliers in the model. Results from the patch optimisation stage indicate that the new estimate of surface orientation significantly improves the optimisation stage by allowing it to converge to the correct depth and orientation in a lower number of iterations whilst avoiding local minima. The final results for the initial feature matching stage indicate that the algorithm is able to reconstruct an accurate patch surface model with minimal outliers. The expansion stage is able to significantly increase the density of the reconstructed model, however it is unaware of outliers and will expand any patch that satisfies the expansion conditions. The filtering stage is aggressive and many patches are removed, some possibly true positives. Final results for the patch reconstruction stage indicate that the models are sufficiently dense and contain a minimal amount of outliers. The mesh initialisation stage converts the orientated point model into a surface mesh model. Results indicate that both initialisation techniques perform well and are able to recover most of the surface structure including deep concavities and thin structures. The final refinement stage is then used to recover any fine surface details, and remove any noise that may have been overlooked by the Iterative snapping or Poisson surface reconstruction algorithms. The resulting mesh models are sufficiently dense with most containing more than 100k vertices. Finally a relatively straight forward texture mapping process is used give the models a realistic appearance. A system evaluation in terms of reconstruction accuracy and completeness show that the reconstruction algorithm can achieve an accuracy of 0.84 mm and completeness of 92.5 % for a (7cm×9cm×7cm) object caprured with low resolution 640×480 cameras.

Overall, the 3D reconstruction system has achieved the goals set out in Chapter 1 and Chapter 2 by providing the model-based communications system with an algorithm that is accurate, robust to outliers and that can handle object and scene datasets within the same framework.

6.2 Future Work

The reconstruction system presented in this dissertation is modular, allowing for a variety of algorithms to be inserted at different stages in the reconstruction pipeline. Hence a good framework for future research on 3D reconstruction of objects and scenes is provided. Several aspects of the existing framework can be modified to improve in efficiency, robustness and accuracy of the reconstruction process.

The Matlab development environment within which the reconstruction system is coded, aids visualisation and development of the abstract concepts presented in this dissertation. However, it is not suited to iterative processes [1] and is hence not the best language for this implementation. Thus significant improvements in terms of reconstruction time and efficiency can be obtained by implementing the iterative processes in languages such as C or C++.

Most state-of-the-art reconstruction algorithms rely on region based matching and achieve accurate results through an intense process of combining heuristics with filtering to remove outliers. One of the biggest challenges facing image based modelling when it comes to selecting a camera baseline is that a small baseline results in the estimated depth being less precise due to narrow triangulation. Hence, for a more precise depth estimation, a longer baseline is required [96]. If the underlying image intensity function is periodic, matching becomes more difficult and there is a greater possibility of a false match. This indicates that there is a trade-off between accuracy and precision in matching. Okutomi *et al.* [96] proposed using the Sum of Squared Differences in inverse distance as a matching metric. The Sum of Squared Differences (SSD) for each individual baseline are summed together to produce the SSD-in-inverse-distance. This function is shown to provide a unique and clear minimum at the correct matching position. Hence, Okutomis approach does not search for a match but rather computes the disparity for it. The sum of squared differences is a simple matching technique that was popular in the early 90's due to its low complexity. However due to its simplicity SSD has poor performance with regard to changes in illumination across image sequences. To overcome this failure in matching we propose using normalized-cross-correlation (NCC) as a replacement for the SSD used in the multiple baseline reconstruction technique. NCC has been shown to be invariant to changes in image amplitude such as those caused by changing

[1] Matlab is optimised for matrix manipulation and vectorization and is thus slow for iterative processes.

lighting conditions [97]. Progress has been made in the development of the algorithm and initial testing of the Multiple baseline approach using NCC has provided promising results.

Appendix A

Datasets

Figure A.1: Skull dataset - 10 out of 24 images (Courtesy of Jodi Blumenfeld and Steven R. Leigh)

Figure A.2: Temple dataset - 10 out of 47 images (Courtesy of S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski)

Figure A.3: Dino dataset - 10 out of 48 images (Courtesy of S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski)

Figure A.4: Fountain dataset - 11 out of 11 images (Courtesy of Christoph Strecha)

Figure A.5: City Hall dataset - 7 out of 7 images (Courtesy of Christoph Strecha)

Appendix B

Contents of CD

A number of files associated with this dissertation are provided on the accompanying CD. The reader is encouraged to view them. The files are as follows:

- **Naren Ramchunder - MSc Dissertation.pdf**:
  A soft copy of this dissertation.

- **Reconstruction Results** folder:
  A folder with the reconstruction results for each dataset. A Mesh viewing application (Mesh Lab) is also included. Please note that the models are large and have been compressed into .rar files. Once uncompressed the image dataset and 3D models for each dataset are available and can be viewed with Mesh Lab.

Appendix C

Reconstruction Results

Table C.1 contains the reconstruction parameters used in the experiments. By adjusting these parameters one can control the reconstruction time and accuracy of the models.

Table C.1: Reconstruction Parameters - values used in experiments.

Dataset	Images	Image Size	μ	β_1	β_2	β_3	η	γ	α	ζ_3
Skull	24	1950×1750	5	2	30	35	4	3	0.3	1
Dino	48	640×480	7	2	32	32	4	3	0.3	1
Temple	47	640×480	5	2	32	32	4	3	0.3	1
Fountain	11	3072×2048	7	2	32	32	5	3	0.3	1
City Hall	7	3072×2048	5	2	32	32	5	3	0.3	1

The following figures provide larger illustrations of the reconstruction results for each of the datasets. However, the reader is encouraged to view the models interactively. Mesh models in .ply format are stored on the accompanying CD.

Figure C.1: Final Texture-mapped model: Skull - 1 of 2

Figure C.2: Final Texture-mapped model: Skull - 2 of 2

Figure C.3: Final Texture-mapped model: Dino - 1 of 2

Figure C.4: Final Texture-mapped model: Dino - 2 of 2

Figure C.5: Final Texture-mapped model: Temple - 1 of 2

Figure C.6: Final Texture-mapped model: Temple - 2 of 2

Figure C.7: Final Texture-mapped model: Fountain - 1 of 2

Figure C.8: Final Texture-mapped model: Fountain - 2 of 2

Figure C.9: Final Texture-mapped model: CityHall - 1 of 2

Figure C.10: Final Texture-mapped model: CityHall - 2 of 2

References

[1] K. Aizawa and T.S. Huang. **Model-based image coding advanced video coding techniques for very low bit-rate applications**. *Proceedings of the IEEE*, **83**(2):259–271, 2002.

[2] D.E. Pearson. **Developments in model-based video coding**. *Proceedings of the IEEE*, **83**(6):892–906, 2002.

[3] H. Li, P. Roivainen, and R. Forchheimer. **3-D motion estimation in model-based facial image coding**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **15**(6):545–555, 2002.

[4] H. Li, A. Lundmark, and R. Forchheimer. **Image sequence coding at very low bit rates: A review**. *Image Processing, IEEE Transactions on*, **3**(5):589–609, 2002.

[5] W.J Welsh. **Model-based coding of videophone images**. *Electronics & communication engineering journal*, **3**(1):29–36, 2002.

[6] P. Eisert. **MPEG-4 facial animation in video analysis and synthesis**. *International Journal of Imaging Systems and Technology*, **13**(5):245–256, 2003.

[7] G. Fleishmann B. Girod and R. Heeg. **Modeling for real-time facial animation**. *Workshop on Coding Tech. for Very Low Bit-rate Video, Colchester, UK*, (2.1), 1994.

[8] J. Ostermann. **Animation of synthetic faces in MPEG-4**. In *Computer Animation 98. Proceedings*, pages 49–55. IEEE, 2002.

[9] B. Amberg, A. Blake, A. Fitzgibbon, S. Romdhani, and T. Vetter. **Reconstructing high quality face-surfaces using model based stereo**. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[10] J. Xiao, J. Chai, and T. Kanade. **A closed-form solution to non-rigid shape and motion recovery**. *International Journal of Computer Vision*, **67**(2):233–246, 2006.

[11] Y. Lee, D. Terzopoulos, and K. Waters. **Realistic modeling for facial animation**. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 55–62. ACM, 1995.

[12] F. Pighin, R. Szeliski, and D.H. Salesin. **Resynthesizing facial animation through 3d model-based tracking**. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, **1**, pages 143–150. IEEE, 2002.

[13] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. **Mesh optimization**. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 19–26. ACM, 1993.

[14] Nello Zuech. **Noncontact 3D-based Machine Vision in Metrology**. *http://www.machinevisiononline.org/vision-resources-details.cfm?content$_i$d = 1194id = 2newsType$_i$d = 0*, **1**, 2005.

[15] B. Scholz-Reiter, M. Lutjen, H. Thamer, and D. Dickmann. **Towards Machine Vision based Surface Inspection of Micro-Parts**.

[16] M.L. Kari, S. Rusinkiewicz, M. Ginzton, J. Ginsberg, K. Pulli, D. Koller, S. Anderson, J. Shade, L. Pereira, and J. Davis. **The digital Michelangelo project: 3D scanning of large statues**. 2000.

[17] 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.

[18] P.J. Besl and N.D. McKay. **A method for registration of 3-D shapes**. *IEEE Transactions on pattern analysis and machine intelligence*, pages 239–256, 1992.

[19] D.F. Huber and M. Hebert. **3d modeling using a statistical sensor model and stochastic search**. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, **1**. IEEE, 2003.

[20] **Breuckmann- 3D metrology**. *http://www.breuckmann.com/en/home.html*, **1**, 2010.

[21] **konica minolta VIVID 910**. *http://www.konicaminolta.com/instruments/products/3d/non-contact/vivid910/index.html*, **1**, 2010.

[22] **The VIUscan color laser scanner**. *http://www.creaform3d.com/en/handyscan3d/products /viuscan.aspx*, **1**, 2010.

[23] SM Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. **A comparison and evaluation of multi-view stereo reconstruction algorithms**. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, **1**, 2006.

[24] G.G. Slabaugh, W.B. Culbertson, T. Malzbender, M.R. Stevens, and R.W. Schafer. **Methods for volumetric reconstruction of visual scenes**. *International Journal of Computer Vision*, **57**(3):179–199, 2004.

[25] S.M. Seitz and C.R. Dyer. **Photorealistic scene reconstruction by voxel coloring**, March 26 2002. US Patent 6,363,170.

[26] K.N. Kutulakos and S.M. Seitz. **A theory of shape by space carving**. *International Journal of Computer Vision*, **38**(3):199–218, 2000.

[27] R. Bhotika, D. Fleet, and K. Kutulakos. **A probabilistic theory of occupancy and emptiness**. *Computer VisionECCV 2002*, pages 112–130, 2002.

[28] C. Hernández Esteban and F. Schmitt. **Silhouette and stereo fusion for 3D object modeling**. *Computer Vision and Image Understanding*, **96**(3):367–392, 2004.

[29] G. Vogiatzis, PHS Torr, and R. Cipolla. **Multi-view stereo via volumetric graph-cuts**. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, **2**, 2005.

[30] Y. Furukawa and J. Ponce. **High-fidelity image-based modeling Univ. of Illinois at Urbana-Champaign**. Technical report, Tech. Rep, 2006.

[31] T. Fromherz and M. Bichsel. **Shape from multiple cues: Integrating local brightness information**. In *Proceedings of the Fourth International Conference for Young Computer Scientists, ICYCS*, **95**, pages 855–862. Citeseer, 1995.

[32] P. Ratner. **Modeling a POLYGON Head**. *http://www.highend3d.com/maya/tutorials /modeling/ polygon/.*, 2005.

[33] S. Soatto, AJ Yezzi, and H. Jin. **Tales of shape and radiance in multiview stereo**. In *Ninth IEEE International Conference on Computer Vision, 2003. Proceedings*, pages 974–981, 2003.

[34] 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*, page 83. IEEE, 2003.

[35] H. Jin, S. Soatto, and AJ Yezzi. **Multi-view stereo beyond Lambert**. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings*, **1**, 2003.

[36] J.P. Pons, R. Keriven, and O. Faugeras. **Modelling dynamic scenes by registering multi-view image sequences**. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, **2**, 2005.

[37] PJ Narayanan, PW Rander, and T. Kanade. **Constructing virtual worlds using dense stereo**. In *Computer Vision, 1998. Sixth International Conference on*, pages 3–10. IEEE, 2002.

[38] S.B. Kang, R. Szeliski, and J. Chai. **Handling occlusions in dense multi-view stereo**. 2001.

[39] G. Vogiatzis, C.H. Esteban, P.H.S. Torr, and R. Cipolla. **Multi-view Stereo via Volumetric Graph-cuts and Occlusion Robust Photo-Consistency**.

[40] O. Faugeras and R. Keriven. **Complete dense stereovision using level set methods**. *Computer VisionECCV'98*, page 379, 1998.

[41] R. Keriven. **A variational framework for shape from contours**. *Ecole Nationale des Ponts et Chaussees, CERMICS, France, Tech. Rep*, 2002.

[42] W.E. Lorensen and H.E. Cline. **Marching cubes: A high resolution 3D surface construction algorithm**. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM, 1987.

[43] S.N. Sinha and M. Pollefeys. **Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation**. 2005.

[44] S. Tran and L. Davis. **3d surface reconstruction using graph cuts with surface constraints**. *Computer Vision–ECCV 2006*, pages 219–231, 2006.

[45] A. Hornung and L. Kobbelt. **Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding**. 2006.

[46] S. Paris, F.X. Sillion, and L. Quan. **A surface reconstruction method using global graph cut optimization**. *International Journal of Computer Vision*, **66**(2):141–161, 2006.

[47] Y. Furukawa and J. Ponce. **Carved visual hulls for image-based modeling**. *International Journal of Computer Vision*, **81**(1):53–67, 2009.

[48] M. Goesele, B. Curless, and SM Seitz. **Multi-view stereo revisited**. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, **2**, 2006.

[49] C. Strecha, R. Fransens, and L. Van Gool. **Combined depth and outlier estimation in multi-view stereo**. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, **2**, 2006.

[50] C. Strecha, T. Tuytelaars, and L. Van Gool. **Dense matching of multiple wide-baseline views**. 2003.

[51] T.K. Moon. **The expectation-maximization algorithm**. *Signal Processing Magazine, IEEE*, **13**(6):47–60, 1996.

[52] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S.M. Seitz. **Multi-view stereo for community photo collections**. In *Proc. ICCV*, pages 1–8. Citeseer, 2007.

[53] R. Kolluri, J.R. Shewchuk, and J.F. O'Brien. **Spectral surface reconstruction from noisy point clouds**. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 11–21. ACM, 2004.

[54] M. Habbecke and L. Kobbelt. **A surface-growing approach to multi-view stereo reconstruction**. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[55] M. Habbecke and L. Kobbelt. **Iterative multi-view plane fitting**. In *Vision, modeling, and visualization 2006: proceedings, November 22-24, 2006, Aachen, Germany*, page 73. IOS Press, 2006.

[56] M. Lhuillier and L. Quan. **A quasi-dense approach to surface reconstruction from uncalibrated images**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 418–433, 2005.

[57] Y. Furukawa and J. Ponce. **Accurate, dense, and robust multi-view stereopsis**. *IEEE transactions on pattern analysis and machine intelligence*, 2009.

[58] M. Kazhdan. **Reconstruction of solid models from oriented point sets**. In *Proceedings of the third Eurographics symposium on Geometry processing*, page 73. Eurographics Association, 2005.

[59] Y. Furukawa and J. Ponce. **Accurate, dense, and robust multiview stereopsis**. In *Proc. CVPR*, 2008.

[60] L. McMillan and W. Matusik. **Image-Based Visual Hulls**. 2001.

[61] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan. **Image-based 3D photography using opacity hulls**. *ACM Transactions on Graphics*, **21**(3):427–437, 2002.

[62] S.N. Sinha and M. Pollefeys. **Visual-hull reconstruction from uncalibrated and unsynchronized video streams**. 2004.

[63] Yasutaka Furukawa and Jean Ponce. **3D Photography Dataset**. *http://www.cs.washington.edu/homes/ furukawa/research/mview/index.html*, 2009.

[64] Strecha. **Multi-view evaluation**. *http://cvlab.epfl.ch/ strecha/multiview/denseMVS.html*, 2010.

[65] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision* . Prentice Hall, 1998.

[66] B. Cyganek and J.P. Siebert. *An introduction to 3D computer vision techniques and algorithms*. Wiley, 2009.

[67] K.G. Derpanis. **The Harris corner detector**. *York University http://www. cse. yorku. ca/kosta/CompVis Notes/harris detector. pdf*, 2004.

[68] C. Harris and M. Stephens. **A combined corner and edge detector**. In *Alvey vision conference*, **15**, page 50. Manchester, UK, 1988.

[69] K. Mikolajczyk. **Detection of local features invariant to affine transformations**. *PhD hesis, Institut National Polytechnique de Grenoble, France*, 2002.

[70] T. Lindeberg. **Scale-space theory: A basic tool for analyzing structures at different scales**. *Journal of applied statistics*, **21**(1-2):225–270, 1994.

[71] D.G. Lowe. **Distinctive image features from scale-invariant keypoints**. *International journal of computer vision*, **60**(2):91–110, 2004.

[72] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge Univ Pr, 2003.

[73] Robyn Owens. **Full OpenCV Wiki** . *http://homepages.inf.ed.ac.uk/rbf/CVonline/*, **1**, 1997.

[74] R.I. Hartley. **In defense of the eight-point algorithm**. *IEEE Transactions on pattern analysis and machine intelligence*, **19**(6):580–593, 1997.

[75] W.H. Press, SA Teukolsky, WT Vetterling, and BP Flannery. **Numerical Recipes in C: The Art of Scientific Computing**, 1992.

[76] R.Y. Tsai and T.S. Huang. **Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**(1):13–27, 1984.

[77] P.H.S Torr and A.W Fitzgibbon. **Invariant fitting of two view geometry**. *IEEE transactions on pattern analysis and machine intelligence*, **26**(5):648–650, 2004.

[78] G. Xu and Z. Zhang. *Epipolar geometry in stereo, motion, and object recognition: a unified approach*. Springer, 1996.

[79] H.C. Corben and P. Stehle. *Classical mechanics*. Dover Pubns, 1994.

[80] N. Andrei. **Scaled conjugate gradient algorithms for unconstrained optimization**. *Computational Optimization and Applications*, **38**(3):401–416, 2007.

[81] I.M Navon, P.K.H Phua, and M. Ramamurthy. **Vectorization of conjugate-gradient methods for large-scale minimization**. In *Proceedings of the 1988 ACM/IEEE conference on Supercomputing*, pages 410–418. IEEE Computer Society Press, 1988.

[82] The Georgia institute of technology :SLR-Optimization in Engineering Design. **NLP-Unconstrained-Multivariable**. *http://www.srl.gatech.edu/education/ME6103/NLP-Unconstrained-Multivariable.ppt*, **1**, 2004.

[83] L. Kobbelt and M. Botsch. **A survey of point-based techniques in computer graphics**. *Computers & Graphics*, **28**(6):801–814, 2004.

[84] M. Kazhdan, M. Bolitho, and H. Hoppe. **Poisson surface reconstruction**. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, page 70. Eurographics Association, 2006.

[85] A.I. Bobenko and B.A. Springborn. **A discrete laplace–beltrami operator for simplicial surfaces**. *Discrete and Computational Geometry*, **38**(4):740–756, 2007.

[86] M. Belkin, J. Sun, and Y. Wang. **Discrete Laplace operator on meshed surfaces**. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 278–287. ACM, 2008.

[87] G. Taubin. **Geometric signal processing on polygonal meshes**. *Eurographics State of the Art Reports*, 2000.

[88] Yasutaka Furukawa. **3D Photography Datasets**. *http://www.cs.washington.edu/homes/furukawa/research/mview/index.html*, **1**, 2009.

[89] M. Kazhdan. **Poisson surface reconstruction Algorithm**. *http://www.cs.jhu.edu/ misha/Code/PoissonRecon/*, **1**, 2010.

[90] Ben Tordoff. **Space Carving Demo** . *http://www.mathworks.com/matlabcentral/fileexchange/26160-carving-a-dinosaur*, 2011.

[91] P.O. Persson and G. Strang. **A simple mesh generator in MATLAB**. *SIAM review*, pages 329–345, 2004.

[92] Wikipedia. **Suture (joint) Human skull**. *http://en.wikipedia.org/wiki/Sutures$_o$f$_s$kull*, 2012.

[93] S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. **A comparison and evaluation of multi-view stereo reconstruction algorithms**. *http://vision.middlebury.edu/mview/eval/*, 2008.

[94] Photograph by Charles G. Hollins S. Akragas. **Temple of the Dioskouroi**. *http://www.perseus.tufts.edu/hopper/image?img=2003.05.0038redirect=true*, **1**, 2011.

[95] S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. **Middlebury evaluation results** . *http://vision.middlebury.edu/mview/eval/doAuth.php?aid=ramchundernov-30-2011*, 2011.

[96] M. Okutomi and T. Kanade. **A multiple-baseline stereo**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**(4):353–363, 1993.

[97] JP Lewis. **Fast normalized cross-correlation**. In *Vision Interface*, **10**, pages 120–123. Citeseer, 1995.