Statistical Framework for Multi Sensor Fusion and 3D Reconstruction

by

Jonathan Ruttle, BA, BAI, MSc

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Doctor of Philosophy

University of Dublin, Trinity College

November 2012

Declaration

Permission to Lend and/or Copy

Acknowledgments

Firstly, I would like to thank my supervisor Rozenn Dahyot, who showed me how to combine the methods of statistical modelling and the harsh reality of computer vision to create something intriguing. Her guidance and encouragement over the years have been invaluable.

To the many current and former members of the Graphics, Vision and Visualisation (GV2) group in Trinity College, I offer my sincere gratitude. Michael Manzke and John O'Kane deserve special mention as they are partly responsible for this endeavor. I am also very grateful to Donghoon Kim for our collaboration and sharing of insights into computer vision.

This work was financed in part by an Innovation Partnership between Sony-Toshiba-IBM and Enterprise Ireland, without which this project would not have been possible.

Finally, my deepest thanks go to my family and friends, who have supported me in so many ways while I have been carrying out this work.

Jonathan Ruttle

University of Dublin, Trinity College
November 2012

Abstract

Multi-view 3D reconstruction is an area of computer vision where multiple images are taken of an object and information in those images is used to generate a 3D model describing the shape and size of that object. The ability to automatically generate 3D models of objects has many uses from content creation for games to object recognition and is a first step in many other computer vision tasks like markerless motion capture.

In this thesis a new framework to achieve 3D reconstruction is presented. This framework is based on the generalised Radon transform and is linked to kernel density estimation. A new smooth differentiable function is defined that can be optimised using gradient ascent algorithms. The framework is applied to two applications; firstly to computing the visual hull, a 3D reconstruction from multiple silhouettes and secondly, to generate a 3D reconstruction from depth information. The framework is capable of overcoming the considerable noise present in depth data to generate an accurate 3D reconstruction.

The framework is extended to optimise camera alignment parameters in a multi-camera system. Existing techniques for calculating camera parameters can be prone to error. This extension optimises these initial estimates of the camera parameters to facilitate accurate 3D reconstructions in real environments.

Finally two data-sets were generated and captured to test and evaluate all the algorithms developed.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

3D reconstruction is one of the most basic and important tasks in a multi-camera system. It is the process of combining multiple 2D images together to form a 3D model of the object of interest. It serves as a foundation for numerous higher-level applications in many domains including motion capture, 3D recognition and 3D modelling. An example of a 3D reconstruction can be seen in Figure 1.1. In this thesis a novel framework for merging data from a multi-camera system to produce a flexible system for 3D reconstruction is proposed. Examples are given for how the framework performs using both synthetic data and real world data and show how the same framework can be used to help optimise camera alignment.

1.1 3D Reconstruction Challenges

3D reconstruction can be a very challenging problem. The reason for this can be broadly broken down into three main factors: converting from 2D information to 3D information, the nature of the object, and the considerable noise in the acquisition system.

- The information available is generally 2D images. To perform a 3D reconstruction it is necessary to calculate the 3rd dimension with no other information. This is not a trivial task. Depth cameras can be used to help in this respect.
- The object that is being reconstructed is unknown. A human that sees a teapot knows in general the 3D shape of a teapot and can use that prior information to

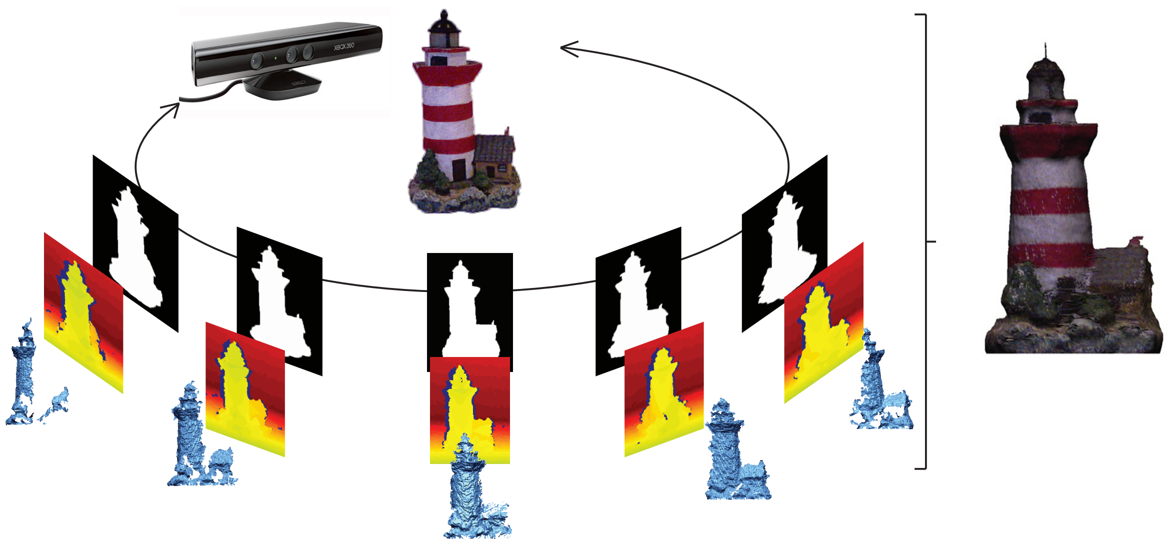


Figure 1.1: An example of a 3D reconstruction. The Microsoft Kinect camera is rotated around an object (Lighthouse). Multiple images are captured and silhouette and depth information is extracted. All this information is merged together to generate a 3D reconstruction (right).

infer the shape of the object. Here no such prior knowledge is available, only the information from the images is available.

- Noise increases the difficulty of this problem considerably. Noise altering the quality of the images and uncertainty in the camera parameters can cause errors in the reconstruction. Noise in the appearance of the object due to light variations and other noises can all cause problems when attempting to reconstruct an object.

1.2 Motivations

- Inexpensive creation of 3D models - 3D models are widely used in many areas including computer games, animation movies, personal avatars (human representations) and computer simulations. Traditionally 3D model creation is done manually on a computer and requires considerable skill. In the games sector alone millions are spent paying for 3D model content to be created. Considerable savings can be made if this process could be done automatically, without requiring hours of work by expensive skilled workers.

- Object Recognition - Identifying and recognising objects in ordinary images is a highly active area of research in computer vision. While this is normally done using 2D images there has been a move towards using 3D reconstructions as they can help with the recognition process even with variations in pose and illumination that can be problematic when only using 2D images. This requires firstly generating the 3D reconstruction before it can be recognised.
- Stereo - Recent years have shown an increase in 3D movies. These movies use stereo camera rigs, or multi-camera rigs to capture the movie to create the 3D effect. It is possible to use these multi-camera setups to also do partial or full 3D reconstructions of the scene. This is useful for postprocessing where artifacts have to be removed, like stunt man wires, or corrected, or to add special effects into the scene.
- First step towards other problems - Many higher level computer vision tasks use 3D reconstruction as a first step towards their goal, for example Markerless Motion Capture which is the tracking and capturing of the movement of a human and their limbs. One method to achieve this is to do repeated 3D reconstructions at regular time intervals. Another area is Human Computer Interface where a person can pose or gesture and it will be recognised by the computer to perform an action. 3D reconstruction can be used to help determine the persons pose or gesture.

1.3 Goal

The goal of this thesis is to develop a new multi-view 3D reconstruction framework with the following characteristics:

- Be flexible and capable of utilising information available from multiple sources of many types, weighting appropriately the importance of the information and converting it into an easy form to infer and create a 3D meshed reconstruction of the object of interest.

- A statistical approach capable of modelling directly the noise of all the input data from the sensors at a fundamental level, without the need for multiple stages to remove outliers or to smooth data.

- Be suitable to take advantage of the latest parallel computer architectures to allow for fast reconstructions.

1.4 Contributions and Thesis Outline

The main contribution of this thesis is a new statistical framework for 3D reconstruction which is based on the generalised Radon transform. An explanation of the generalised Radon transform and an overview of the current state of the art on 3D reconstruction methods is presented in Chapter 2. The framework is designed to do a 3D reconstruction and infer the shape of an object of interest from multi-view data and is presented at the beginning of Chapter 3. The generalised Radon transform is extended to become the generalised relaxed Radon transform which can be linked to kernel density estimates (also explained in Chapter 2).

Chapter 3 continues with two applications of this framework. The first application is a rethinking of the inference of the visual hull from silhouettes. The visual hull is a geometric entity or 3D model which defines the shape and size of an object of interest. A silhouette is the set of pixels in an image which represents the object. These pixels will generally be white and the rest of the pixels in the image (the background and the other objects not of interest) will be black. Silhouette extraction (also called foreground/background segmentation) is an ongoing area of research which is not considered in this thesis; it is assumed that silhouette images are available. By projecting these silhouettes (white pixels) out of the images and intersecting the silhouettes from multiple views it is possible to create a 3D reconstruction of the object called a visual hull. Instead of an objective function that is discrete (as currently proposed in the literature using a 3D histogram to construct the visual hull as described in Chapter 2), here a smooth differentiable function is defined that can be optimised using a gradient ascent algorithm which is suitable for parallel computer architecture. The second application of the framework is to generate a 3D reconstruction from multi-view depth information. Special hardware is now capable of not only capturing an image of a scene but also capturing the depth at each pixel in the image from the camera to the corresponding point in the scene. This represents a considerable increase in information that can be used in the reconstruction but depth information captured by current hardware can be exceptionally noisy and this noise must be considered when generating a 3D reconstruction from depth information.

The framework is extended in Chapter 4 to find the optimal value of the camera parameters. Techniques exist which allow for a good initial estimation of the camera parameters to be calculated. Camera parameters are the position and orientation of the camera in space and the internal variables which allow a point in space to be projected into the image plane. While the internal variables can be accurately calculated the position and orientation can be prone to error. Accurate camera parameters are essential for recovering fine detail in the reconstruction. This extension of the framework allows for these uncertainties to be taken into account and helps improve the aligning of the cameras together.

Chapter 5 applies the reconstruction techniques to a variety of objects and compares and contrasts the different methods and effects of different parameters. Two data-sets were created to test and evaluate these reconstruction methods. The first data-set is a synthetic data-set created using software. This allows full control of all the parameters and settings. The synthetic data-set was created from 3D models. These models can therefore be used as ground truths to allow full quantitative evaluations to be performed. The second data-set was created using the new Microsoft Kinect camera, which is capable of also capturing depth data. Data for numerous objects was captured, the objects reconstructed and a full evaluation was performed.

Chapter 6 summarises all the work done in this thesis and concludes this research by outlining possible future directions of investigation.

1.5 Publications to Date

During the course of this thesis the following publications were produced:

- Jonathan Ruttle, Michael Manzke and Rozenn Dahyot. **Estimating 3D scene flow from multiple 2D optical flows**, in Irish Machine Vision and Image Processing conference, pages 6-11, 2009 [85]. This work is presented in Appendix

E.

- Jonathan Ruttle, Michael Manzke, Martin Prazak and Rozenn Dahyot. **Synchronized real-time multi-sensor motion capture system**. In SIGGRAPH Asia sketch Poster, page 50:1-50:1, 2009 [87]. This work is presented in Appendix F.

- Donghoon Kim, Jonathan Ruttle and Rozenn Dahyot. **3D shape estimation from silhouettes using mean-shift**, in IEEE International Conference on Acoustics, Speech and Signal Processing, pages 1430-1433, 2010 [51]. This work is summarised in the state of the art Section 2.3.3.

- Jonathan Ruttle, Michael Manzke and Rozenn Dahyot. **Smooth kernel density estimate for multiple view reconstruction**, in Conference on Visual Media Production, pages 74-81, 2010 [86]. This work is presented in Section 3.2.

The next chapter presents the current state of the art.

Chapter 2

State of the Art

In this chapter, the past works on 3D multi-view reconstruction (Section 2.1), camera calibration (Section 2.2), inference with kernel modelling (Section 2.3) and the generalised Radon transform (GRT) (Section 2.4) are reviewed. Visual hull, a common approach for 3D shape inference, is an essential step in 3D reconstruction that is often modelled using a discrete objective function. This objective function corresponds to a 3D histogram and can be changed to a smooth differentiable objective function using kernel modelling. It is proposed that 3D volumes are inferred by using the gradient ascent method optimising a kernel mixture (Section 2.3.2). Finally the GRT is introduced as it is at the core of the work presented later in this thesis.

2.1 3D Multi-View Reconstruction

Seitz *et al.* [90] provides a good summary of the field where they split the reconstruction up into four categories. The first category of techniques computes a discontinuous cost function on a 3D volume as a 3D histogram [91, 13, 7]. The second category involves iteratively evolving a surface to decrease or minimise a cost function [28, 55, 4]. The third category uses depth information computed on each pair of images which it then merges together to produce the end result [14]. The fourth category of algorithms does matching across images from which surface points can be estimated [78].

A number of 3D reconstruction methods use a combination of some or all of the algorithms. For example, some methods start off computing a cost function on a 3D volume to produce an initial estimate of the shape and size of the object as in the first category. This would give an estimate of the surface which can be improved using algorithms from the second category which are based on either depth estimation or surface point matching from the third and fourth categories. This results in a pipeline of processes, each with the goal of providing further raw data for a further stage of the pipeline or refining the current mesh to produce higher accuracy [109, 88].

Another way to categorise the different algorithms at a more basic level is in the different information used behind the method. The next set of subsections describe some 3D reconstruction algorithms; firstly from silhouette information, secondly from colour information and thirdly from depth information.

2.1.1 Shape from Silhouette

The visual hull is a geometric entity created by shape-from-silhouette 3D reconstruction techniques introduced by Martin *et al.* [67]. This method assumes the foreground object in an image can be separated from the background. This foreground object, also known as a silhouette, is the 2D projection of the corresponding 3D foreground object. Along with the camera viewing parameters, the silhouette defines a back-projected generalised cone that contains the object. The intersection of two or more such cones is called the visual hull (Figure 2.1).

In practice this method is accomplished by dividing the world up into regular 3D volumes called voxels (volume elements). In turn each voxel is projected onto each image plane and if the corresponding pixels in each of the image planes are foreground pixels then the voxel is assumed to be part of the visual hull. If the projected pixels are background pixels then the voxel is assumed to be transparent.

An octree representation of the space to improve the efficiency of the algorithm was first proposed by Potmesil in 1987 [79] and later optimized by Szeliski in 1993 [100] (Figure 2.2). The space is divided into 8 voxels and each voxel is projected onto the image plane that can see that pixel. If the projected voxel contains no foreground pixels then it is made transparent. If all the pixels in the projected voxel are foreground then the voxel is kept but if some of the pixels are foreground and some are background then the voxel is subdivided into 8 more voxels. This process is repeated until a minimum voxel size is reached.

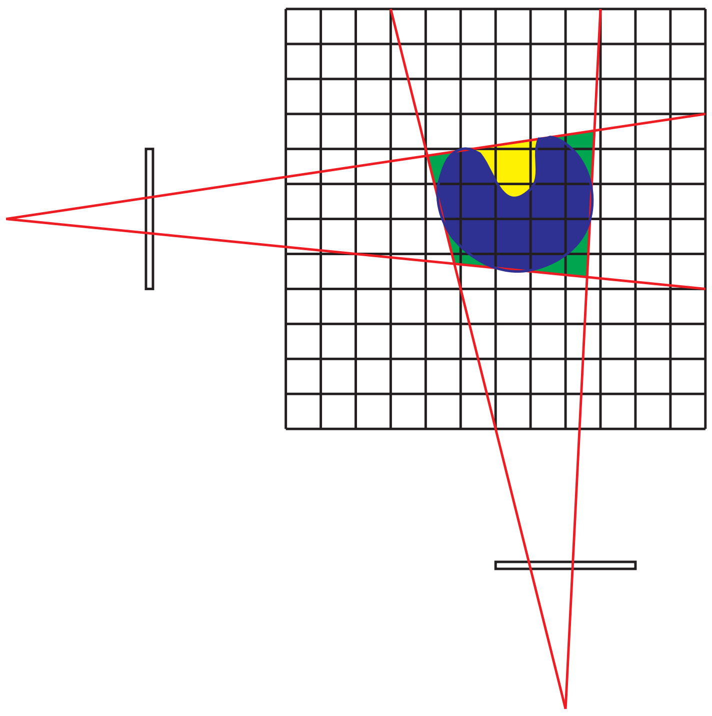


Figure 2.1: An example of a silhouette cone intersection on a slice. The object (blue) is seen by two cameras. With more cameras the green areas will be refined, but the concavity (yellow) cannot be recovered with only silhouette information.

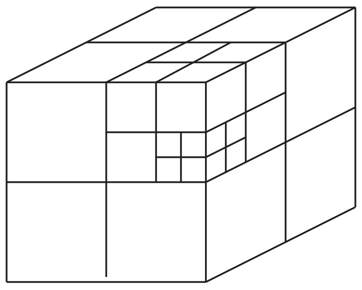


Figure 2.2: To improve efficiency an octree representation can be used when doing a voxel reconstruction. The space is represented as 8 voxels, each voxel can be split into a further 8 voxels until a minimum voxel size is reached.

This type of shape-from-silhouette method can produce a reconstruction that is guaranteed to enclose the true object but is unable to recover surface concavities in the object [56]. The quality of the resulting 3D reconstruction greatly depends on the number of viewpoints and the accuracy of segmenting the image into foreground and background pixels. The 3D reconstruction can suffer from self occlusions but this problem can be partially resolved with greater number of viewpoints. The minimum size of the voxel also greatly affects the end quality of the reconstruction; the smaller the voxel the greater the quality. Unfortunately the computation time and memory requirement are directly linked with the size of the voxel and as the voxel is made smaller, the time and memory requirement increase. Many 3D reconstruction algorithms are variations on this method, all designed to generate a visual hull [27, 11, 57, 97, 69, 99, 32]. Once the solid voxel representation is generated it is generally required to convert it into a 3D mesh. This can be done using a method called Marching Cubes which will discover the edge voxels so that they can be converted into mesh vertices [62].

2.1.2 Shape from Photo-Consistency

Seitz *et al.* [91] proposed an extension to the visual hull called the photo hull. The idea behind the photo hull is to use colour consistency instead of just geometric intersection of silhouette cones. If a voxel has consistent colour across all the image planes that can see it, then it is deemed to be part of the surface of the object. If the colours are inconsistent, then the voxel is deemed to be transparent. This method of repeated classification of voxels is called space carving and is used until all the voxels are deemed colour consistent, transparent, or not visible (inside the object). The end result of the method is a set of coloured voxels which is called the photo hull.

By using colour consistency, surface concavities can be recovered if the concavity has a different colour to the rest of the object and if the surface point is visible to more than one viewpoint (Figure 2.3). In general the photo hull requires more viewpoints than the visual hull. This is because the photo hull requires multiple cameras to see each surface point around the object for it to be verified as a surface point, while the visual hull only requires the silhouette cones to intersect to determine the visual hull.

This method requires a decision metric which determines whether a voxel is colour consistent or not. An underlying assumption is made that the object surface follows

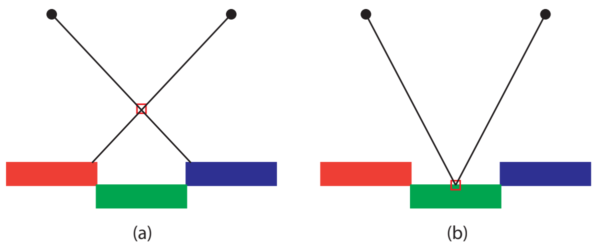


Figure 2.3: Given a Lambertian reflection model the colour of the surface of an object should be consistent from multiple views. This can be used to recover concavities, which can be difficult using only silhouette information.

a Lambertian reflection model, meaning light falling on it is scattered such that the apparent brightness of the surface to an observer is the same regardless of the observer's angle of view. Therefore, the colour of the surface point should look the same from any viewpoint that can see it. A number of variations of this method have been proposed that use different methods of determining colour consistency and employ optimisations in the data structure to suit the algorithm [55, 96, 95, 1, 53].

2.1.3 Shape from Stereo Pairs

Stereopsis is another highly active field of computer vision research. The idea behind stereopsis is similar to that of the human vision system. Two nearby views of the same scene are taken at the same time. By matching points in one image to the other image it is possible using the two cameras' geometry to calculate the 3D position of that point in space. Repeating this for as many points as possible in the scene gives a point cloud that can represent the scene. This can be used to generate a 3D reconstruction of one side of the object. By using multiple sets of stereo pairs of cameras it is possible to produce a point cloud from each stereo pair and merge them together to produce a full reconstruction.

Scharstein and Szeliski [89] produced a comprehensive review of dense two-frame stereo algorithms in 2002. The following is a very short outline of stereo algorithms which have been used to do full 3D reconstructions. For further information [89] is a good place to start.

The basis of all the methods is the matching process where a point, patch or feature from the left image is matched to the same point, patch or feature in the right image. Corner and edge detection as proposed by Harris and Stephens was one of the first feature based matching methods; edges and corners in one image should correspond to corners and edges in another image [42]. More advanced feature methods then were developed such as the Scale-Invariant Feature Transform (SIFT) method [63] which provides a way to detect and describe features in a scale-invariant manner to make it easier to match. The distance between the camera centres (baseline) is an important factor in stereo vision; the further away the cameras are, the harder it is for good robust feature matches to be made. Matas *et al.* proposed Maximally Stable Extremal Regions (MSER) [68] are suitable for this type of feature matching. In order to speed up stereo matching Bay *et al.* proposed Speeded Up Robust Feature (SURF) [2] which is based on 2D Haar wavelets and which are very fast to calculate. Other features exist as seen in [70, 104]. Mikolajezyk and Schmid gave a good review of common local descriptors [71]. To improve the task of matching it is possible to rectify the images. This is where the images are aligned in such a way that each line in one image corresponds to a line in the other image [35]. This means that a 2D search of the whole image for a match can be reduced to a 1D search of the corresponding line in the other image.

Once the matches are made and the camera geometry is used to calculate real world position of the points, it is then required to merge the points together. A number of methods have been proposed to accomplish this. It is important to be robust to outliers. When points are matched incorrectly those points will be useless for the reconstruction and must be removed before continuing [14, 58]. A number of methods use an initial step of generating a visual hull from silhouette images. This assists with merging the stereo information and removing outliers [29, 16, 48].

A large number of variations exist which try to deal with different scenarios from a large scene to small objects using a variety of techniques from multi-resolution to multi-stage approaches. Some of the best methods are evaluated and compared on the Middlebury web-site as described by Seitz *et al.* [90] these include [45, 12, 61, 108, 34].

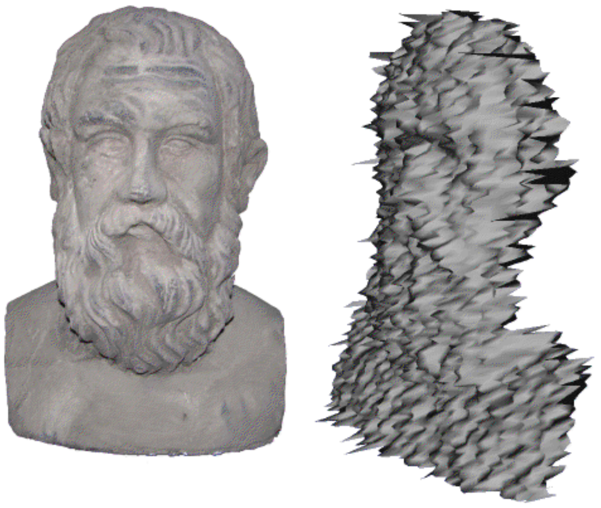


Figure 2.4: The right image gives an example of the raw data from a time-of-flight depth camera of the object on the left as presented in [20]

2.1.4 Shape from Depth

If a depth sensor is available, then the stereo matching stage from the previous section can be skipped and the problem becomes focused on merging together the point clouds to create the 3D reconstruction. The problem that arises in this situation is that the depth data from the sensors is generally very noisy, as seen in Figure 2.4, and must be smoothed to some extent before being used. Another problem that can arise is aligning the depth data together to the same world space so that all the depth data aligns correctly. Both Cui *et al.* [20] and Reynolds *et al.* [82] have proposed methods which help solve these problems both on an object scale and in a scene scale.

With the introduction of the Microsoft Kinect sensor in November 2010 there has been a large increase in the number of methods introduced for 3D reconstruction from depth information. This is because the Kinect sensor is the first low cost (∼€150), mass produced and relatively high resolution (640×480 pixels)depth sensor to be introduced into the market. Previous to the Kinect sensor, depth cameras were extremely expensive (>€1000) and had relatively low resolution ($< 320 \times 240$ pixels). More information on the Kinect sensor can be found in Appendix A.

KinectFusion is one such method for doing real-time 3D reconstruction using the

Kinect sensor that has been proposed in papers by Izadi *et al.* [47] and Newcombe *et al.* [72]. The KinectFusion method continuously tracks the six degrees of freedom pose of the camera and fuses new viewpoints of the scene into a global surface-based representation. An iterative closest point algorithm is used to calculate the relative six degrees-of-freedom transformation from the current view-point to that of the previous frame to perform the continuous tracking. The newly aligned data is then added to the growing reconstruction which is a voxel based volumetric representation. The method uses a GPU implementation to be able to give real-time results. The method relies on small movement between frames to calculate a good transformation from the current view-point to that of the next and has considerable memory requirements in terms of it's voxel based volumetric representation.

2.1.5 Shape from Other Information

There are a number of other methods which can be used to generate 3D reconstructions. For example, some methods use the normals of the silhouette edges in the image plane. The normal of the silhouette edges are the 2D projections of the 3D normal of the surface of the object and they can be used to generate a 3D reconstruction [60]. Different methods of using multi-resolution representation of the raw data are proposed by Manson *et al.* [66] which uses wavelets to help reconstruct a surface mesh. Paris *et al.* propose a method based on graph cuts which is used to minimise the distance between the estimate of the reconstruction and the data from the images [74].

2.1.6 Remarks

In the majority of the methods outlined above a discrete form of mathematics is generally used when analysing the space being considered. The world or space is divided into discrete regular cubes or partitions. This can have its advantages but it also has some major disadvantages, particularly in terms of scalability, as the computation time and memory requirements generally scale exponentially as dimensions are added, or massively, as resolution is increased. Another component of a number of the methods is the pipeline approach, where a reconstruction algorithm will have multiple stages, each stage either providing information to the next or refining the previous. These methods have the advantage of combining different methods and different information from different sources together, to produce a more accurate and complete reconstruction than any one stage or method could on their own. The disadvantage of this type of pipeline is that in each stage decisions have to be made in terms of matching points or threshold levels, and with each decision errors can be made. Information can be lost and bad information can be propagated through to the next stage. Extra stages can be added to the pipeline to remove any outlier or error to reduce their further effect, but very little can be done to recover any information that may be lost.

These two areas are considered in this thesis as the two major bottlenecks or problem areas for the future of 3D reconstruction methods. The goal, therefore, of this thesis is to develop a framework which tackles the 3D reconstruction problem from a different direction. Instead of an objective function that is discrete, here a smooth continuous and differentiable function is defined that can be optimised using a gradient ascent algorithm. This framework has the flexibility to allow all the same information to be used, but combined all together at the same time, with the accuracy of each sensor modelled at a fundamental level and includes tools and techniques that will make the problem of scaling in dimensionality and resolution a feasible and affordable cost. The future of computer processing architecture is highly focused on multi-core solutions and many of the latest techniques are focusing on massively parallel architectures like multi-core CPUs and GPUs. Therefore, it is also important that the framework should be suitable and designed to take advantage of such architectures.

2.2 Camera Calibration

The procedure for determining the intrinsic and extrinsic camera parameters is called camera calibration. Intrinsic parameters are particular to the internals of a specific camera including the focal length, center point and radial distortion coefficients. These intrinsic parameters may change as a camera's focus and magnification is changed. The extrinsic parameters place the camera at a particular point in the world relative to a known origin. Many methods are available for determining the camera parameters [43]. The choice of the method generally depends on the prior information available and the situation for which the camera calibration is required.

A common approach is to capture several images of a known geometric pattern such as a planar checkerboard (Figure 2.5). The knowledge of the relative positions of

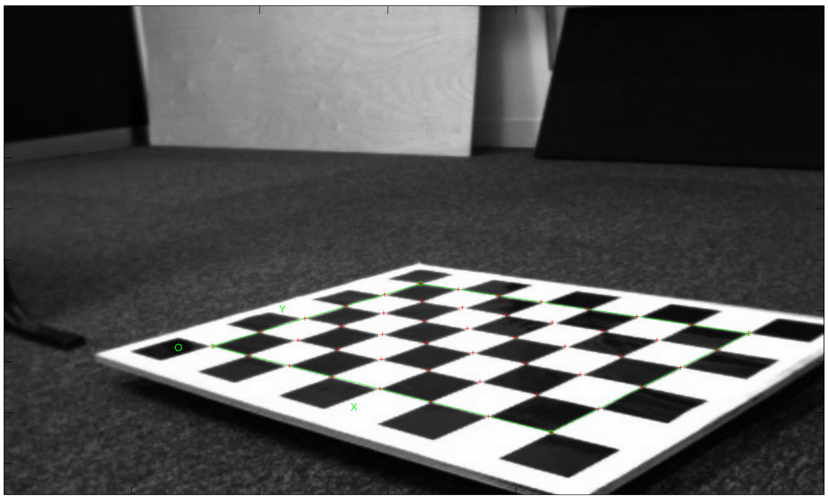


Figure 2.5: The corners (red) of a standard checkerboard pattern can be used to calibrate a standard camera.

the corners of the checkerboard in the real world and the corresponding coordinates of the projection of those corners in the image plane can be used to solve for the camera parameters. Since the projection matrix contains 12 entries and 11 degrees of freedom, a minimum of five and a half 3D to 2D projections are needed, since each projection contains two relationships x and y. To overcome problems due to noise, normally many more points are used and an optimisation process is used to solve for the unknown parameters both extrinsic and intrinsic. Coefficients for radial distortion can also be calculated using this method. For a standard camera left in the one configuration it is possible to only calculate the intrinsic parameters once and then only one image of a checkerboard is needed to calculate the extrinsic parameters for each viewpoint required.

For a multi-camera system, common points should be used to guarantee that all the cameras have a common world origin and are correctly positioned relative to each other. A MATLAB toolbox has been developed by Bouguet and has been used for all the camera calibration in this work [9].

An alternative method for calculating the extrinsic parameters of a camera is by using the optical flow between images [73]. By measuring the apparent movement of the objects in the image space from one viewpoint to another, it is possible to work backwards (with the assumption that the scene stays static) and calculate the movement of the camera relative to the objects. This method requires large overlap between viewpoints of the scene, so good optical flow can be calculated. It works best with a slowly moving camera with a high frame rate resulting in very little movement which can be accurately tracked over time. The accuracy of this method depends on the accuracy of the optical flow calculations. To improve this, the optical flow and camera movement can be calculated over numerous frames and the result smoothed and interpolated to correct for any noise.

2.3 Inference using Kernel Density Estimates

2.3.1 Kernel Density Estimation

Kernel density estimation is a method for estimating a probability density function. The method was first proposed by Rosenblatt [84] and later expanded by Parzen [75], who added multiple different kernel types. The basis of kernel density estimation is that given a sequence of independent identically distributed random variables $x_1, x_2, \ldots, x_n$, with a common probability density function $f(x)$ then the kernel density estimate $\hat{f}(x)$ will be:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=0}^{n} k\left(\frac{x - x_i}{h}\right) \tag{2.1}$$

where k is the kernel, and $h > 0$ is a smoothing parameter called the bandwidth. A number of different kernels can be used. The requirements for a kernel are that it should be symmetric and that it should integrate to one. The most common kernel used is the Gaussian kernel as seen in Equation 2.2, and while there are many alternative kernels, the Gaussian kernel is the one that is used throughout this thesis.

$$k(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \tag{2.2}$$

The bandwidth of the kernel is a very important parameter. If it is too small, noise in the sampled data map produces spurious artifacts. If it is too large, the result will be over-smoothed and fine detail and structure will be lost. The goal, therefore, is to have the bandwidth as small as possible to preserve as much detail and structure as possible but large enough to overcome any noise in the input.

2.3.2 Stochastic Exploration

There is a great deal of literature covering the area of stochastic exploration, but two books in particular give an excellent detailed explanation of the subject; chapter 5 of Monte Carlo Statistical Methods [83], and chapter 9 of Convex Optimization [10]. The following section gives an overview of stochastic exploration as it is applied in this thesis. For more information, please refer to these two books.

In general the goal is to solve and optimise the problem of:

$$\arg\max_x f(x) \tag{2.3}$$

It is assumed that f is convex and twice differentiable. It is also assumed that there exists an optimal point x^*; a solution to the problem in Equation 2.3. In certain situations the problem can be solved analytically by solving the set of equations:

$$\nabla f(x^*) = 0 \tag{2.4}$$

where ∇f is the gradient of the function. In all cases the maxima or minima of the function f should have a gradient equal to zero. Therefore, solving for x when the gradient equals zero should solve the problem.

Realistically, in most cases of interest, the problem is too complex to be solved analytically. An alternative to solving the problem in Equation 2.3 is called regular sampling. Regular sampling is a method where the domain of the function is broken up into small regular intervals and the value calculated per interval. The result can be thresholded to give a solution to the problem. In practice, regular sampling can be computationally very expensive and the accuracy of the result depends highly on the resolution of the regular sampling. Therefore, to increase the accuracy has a considerable effect on the computational cost.

Another alternative to solving the problem is by using an iterative algorithm. In this situation an algorithm is used to compute a sequence of points $x^{(0)}, x^{(1)}, ..., x^{(k)}$, such that as $k \to \infty$ then $x^{(k)} \to x^*$. This sequence of points is called a maximising sequence. Many iterative algorithms exist. One such algorithm is the mean-shift algorithm. While not a major focus in this work, an overview of its use is given in the section below. Another area of algorithms is the gradient ascent algorithms which includes Newton's method. Newton's method is widely used in the work in this thesis and is fully explained in the sections below.

Mean-Shift

A mean-shift method is a non-parametric mode-seeking algorithm, which climbs the gradient of a probability distribution to find the nearest domain mode (peak). The algorithm was originally proposed by Fukunaga and Hostetler [33]. Cheng [15] generalised the method and introduced it to the image analysis field with a mode-seeking process on the density function surface. More recently, Comaniciu and Meer [18] proposed practical applications such as segmentation, tracking [19, 17], etc. The convergence of the mean-shift is proven for the Gaussian kernel in [15] and for any kernels with a convex and monotonically decreasing profile, such as the Epanechnikov kernel in [18]. It has been shown that mean-shift can be used on high dimension problems such as texture classification [36].

Gradient Ascent

The general ascent algorithm is as follows:

Algorithm 1: General Ascent Algorithm

```
given a starting point $x^{(0)}$
repeat
    1. Determine ascent direction $\Delta x$
    2. Choose a step size $t > 0$
    3. Update $x^{(n+1)} = x^{(n)} + t\Delta x$
until stopping criterion is satisfied
```

This algorithm contains a number of elements which will now be explained further including; the starting point $x^{(0)}$, the ascent direction Δx, the step size t and the stopping criterion.

Ascent Direction The first ascent direction that can be used is the gradient $\nabla f(x)$ of the function f at the point x. This method exhibits approximately linear convergence. Many different ascent directions can be used such as steepest ascent. In practice the Newton Step has been shown to converge faster than most methods with a quadratic convergence rate. The Newton direction is:

$$\Delta x_{nt} = [\mathbf{H}f(x)]^{-1}\nabla f(x) \tag{2.5}$$

where $\mathbf{H}f(x)$ is the Hessian matrix of the function f.

While the gradient will always give a direction that increases, the Newton direction gives the direction that may point in either the increasing or decreasing direction. This depends on whether it is closer to a maxima or a minima and on the underlying function being maximised. It is possible to do a dot product between the gradient and the Newton step to determine if it is heading in the correct direction. A positive result is correct; a negative result means the direction needs to be inverted. This can be done by multiplying the direction by minus one. One alternative to this is to compare the value of the function before and after the jump. If the value of the function has decreased then the direction of the jump can be reversed.

Step Size An important part of any gradient ascent method is the size of the step. The goal is to have the step as large as possible, therefore reaching the target as soon as possible, but not too big for fear of overshooting the target. Generally the step size will be 1. This does not mean the size of the jump will be 1, it will just be 1 times the size of the ascent direction calculated above. The size of the ascent direction will depend greatly on the underlying function and the gradient method being used. For example, the Newton step with a quadratic function or polynomial underneath will in general produce very reliable results. In this work the framework is based on a Gaussian kernel at its core. However, because of the structure of the Gaussian kernel, this can cause some extremely large jump sizes if the point lies around the turning point of the curve. A number of methods can be used to alleviate this problem. One such solution is to specify a maximum jump size and limit any jump to that size. Some more complicated solutions involve line searching. This is where the line of the ascent direction is explored and sampled and the step size that maximises the problem is used.

One such line searching algorithm used is called backtracking. Backtracking uses two constants α and β where $0 < \alpha < 0.5$ and $0 < \beta < 1$.

Algorithm 2: Backtracking line search

```
given a descent direction $\Delta x$ and $\alpha \in (0, 0.5)$ and $\beta \in (0, 1)$
$t = 1$
while $f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$ do
    $t = \beta t$
end
```

Stopping Criterion A very important part of using Newton's method is the stopping criteria. A number of methods can be used to determine when to stop jumping towards the maxima. The most common is to define a threshold and when the jump distance is smaller than this threshold, stop jumping. A similar method of thresholding can be applied to the change in value of the function between one point and the next. If the value decreases this can be a sign that it has gone beyond the maxima, but this might also be due to the problem mentioned above where Newton's method will just as easily converge towards a minima. Another method also used is to define a maximum number of jumps. This can be used bearing in mind that if the point has not converged towards the maxima within that number of jumps it may be deemed to be following some random noise in a random direction. Therefore, it is better to stop after that maximum number of jumps and start again somewhere else.

Starting Point Another important point for any iterative gradient ascent algorithm is the starting point. The goal is to have the starting point as close to the end target as possible. This is because the closer it is to the end point the faster it will reach it. Also the further away the starting point is from the maxima the more problems that can occur, for example, if the point is closer to a minima than a maxima, as discussed above. Another similar problem is it can't differentiate between a local maxima and a global maxima; it will just converge to the closest maxima. One method to alleviate this problem is to increase the bandwidth of the underlying kernel. This will smooth the function and reduce the number of local maxima as they all amalgamate into the one maxima.

Simulated Annealing

Simulated Annealing is a method where the the scale of the problem is changed to allow for faster convergence. It also helps avoid the trapping attraction of local maxima and focus on global maxima. In this case the scale of the problem can be changed by increasing the bandwidth of the underlying kernel. Increasing the bandwidth can incur some other problems, as it can remove the fine detail of an object. Therefore, as the point converges towards the global maxima the bandwidth can be decreased; similar to the method used by Shen *et al.* [92]. It is possible to use a simulated annealing scheme which will start with a large bandwidth h_{max} and decrease it iteratively towards a minimum bandwidth h_{min} as the point converges to a global maxima. This is generally done at a geometric rate.

$$h_t = \alpha^t h_{max} \quad \text{until} \quad h_t = h_{min} \quad \text{with} \quad 0 < \alpha < 1 \tag{2.6}$$

It has been shown that the speed of mean-shift can be improved by using an annealing method [52, 22].

Overall Newton's Method of optimisation has been shown to be more efficient (requires fewer steps to converge) than mean-shift and makes fewer assumptions on the form of the underlying kernel structure [39]. In general mean-shift requires a linear kernel structure but a Newton-style only requires the kernel to be differentiable to the second order to fit the equation.

A number of papers have used these types of gradient ascent algorithm to solve a number of problems in computer vision. The most popular of which are tracking algorithms [26, 30, 31, 92].

Parallel Computing

The mean-shift algorithm has been shown to be easily parallelled and can be programmed for a GPU as shown by Li and Xiao [59].

Srinvasan and Duraiswami [98] have shown that kernel based functions are very parallelizable and suitable for multi-core graphical processors (GPU) which can result in substantial computational speedups. They have also provided a library (called GPUML) which is for a C/C++ and MATLAB interface for speeding up the computation of weighted kernel summations and kernel matrix construction on GPU.

2.3.3 Inference of Visual Hull using Kernel Modelling

As noted in Section 2.1.6 most 3D reconstruction methods focus on discrete objective functions. By using kernel modelling it is possible to generate smooth objective functions. Work that has been done towards this goal was to define a continuous objective function for an orthographic camera model [1].

$$\hat{p}(\mathbf{x}) \propto \frac{1}{n}\sum_{i=1}^{N}\frac{1}{\sqrt{2\pi}h}\exp\left(\frac{-\left(\rho_i - x\cos\theta_i - y\sin\theta_i\right)^2}{2h^2}\right) \tag{2.7}$$

In this work $\mathbf{x} = (x, y)$ the latent point of interest on a 2D slice and $\{\rho_i\theta_i\}_{i=1,\dots,N}$ are a set of N independent observations which represent the rays corresponding to silhouette pixels from all the orthographic cameras.

This results in a continuous representation of the space which can not only be explored using regular sampling techniques but can also be explored using mean-shift gradient ascent techniques to reconstruct the object of interest. This work allowed all information to be considered in a continuous likelihood, where all information is weighted and combined together.

Limitations exist in respect of the orthographic camera model used as the basis of the algorithm which is mainly used for medical images as compared to the pin-hole camera model which is more widely used. Also the mean-shift algorithm used is not suitable to work with the more complex pin-hole camera model as it requires a linear mathematical model or a linear approximation to work. Kim's work [50] shows that the framework can be extended to include colour and prior information which improves the accuracy and completeness of the reconstruction.

2.4 Generalised Radon Transform

The framework which is presented in the next chapter has at its core the generalised Radon transform. Therefore, this section describes the classic Radon transform and the generalised Radon transform and some of the applications to which they have been applied.

[1]This work was explored with Donghoon Kim and published in [51].

2.4.1 Radon Transform

In 1917 Johann Radon proposed the Radon transform [80] for a function $f(x)$ as:

$$\mathcal{R}[f(\mathbf{x})](p,\xi) = \int f(\mathbf{x})\delta(p - \xi \centerdot \mathbf{x})d\mathbf{x} \tag{2.8}$$

where:

- $f(\mathbf{x}) \in \mathcal{S}(\mathbb{R}^m)$ is a function that is infinitely differentiable
- $d\mathbf{x} = dx_1, \ldots, dx_m$ is a volume element
- $\xi = (\xi_1, \ldots, \xi_m)$ is a unit vector that defines the orientation of a hyperplane with equation $p = \xi \centerdot x$
- $\delta(\centerdot)$ is the Dirac delta function

This equation provides the mathematical framework for a great number of reconstruction problems in physics, geology, medical imaging and other areas. A 2D example of the Radon transform can be seen in Figures 2.6 and 2.7. The Radon transform is widely applicable to tomography and tomographic reconstruction, for example, computed axial tomography (CAT scan) and magnetic resonance imaging (MRI). An explicit and computationally efficient inversion algorithm exists for 2D Radon transforms called filtered back-projection. Many algorithms have been developed which use the Radon transform [65, 81, 49, 24].

The Hough transform is conceptually very close to the two-dimensional Radon transform; they can be seen as different ways of looking at the same transform [23]. The Hough transform can be seen as a discrete form of the Radon transform and it is mainly used for detecting lines in a 2D image [24, 21].

Tomographic reconstruction reconstructs 3D volume from density data, and silhouette images can be used as rough approximations of hollow objects. Consequently, similar to the visual hull, Pintavirooj and Sangworasil showed that using the inverse Radon transform on silhouette images of an object taken from different points of view, allows reconstruction of an approximation of the 3D shape [77]. The reconstruction is achieved by using the inverse Radon transform on all the corresponding silhouette lines of a slice, which like a conventional x-ray, gives a cross-sectional picture of the object.

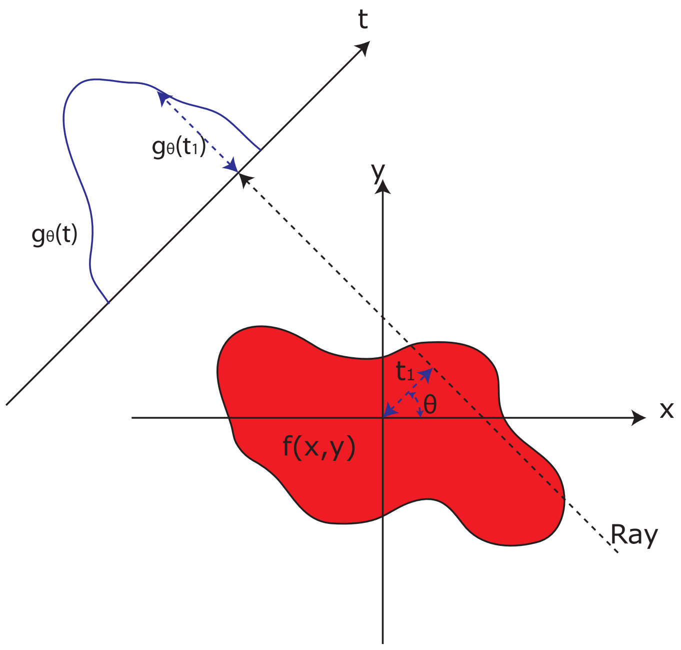


Figure 2.6: An example of a Radon transform at an angle θ of a distribution; the distribution is projected onto the plane with the magnitude related to the width of the distribution at that point. Similarly an X-Ray would have a greater magnitude depending on the absorption which is greater where there is a bone.

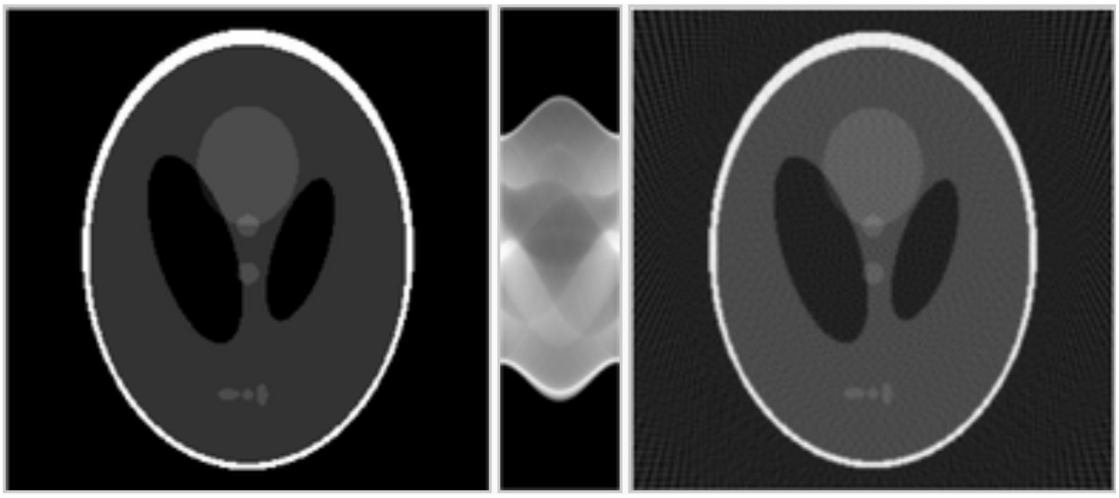


Figure 2.7: The left image is the original image. The middle image shows the Radon transform of that image with each column of the image corresponding to a different angle. The right image shows the reconstruction using the inverse Radon transform. This image is produced using MATLAB's radon and iradon commands with 180 views.

Whereas in the x-ray situation, density information is used which enables the inside of the object to be seen, using only the silhouette information, the surface of the object can be reconstructed. Kim *et al.* [51], mentioned above, compared their method to this approach due to the similar assumption of an orthographic camera model.

2.4.2 Generalised Radon Transform

Over the years the Radon transform has been generalised [25]; it can be defined in many ways. The generalised Radon transform associated with the equation $\lambda + F(\mathbf{x}, \Theta) = 0$ is:

$$\mathcal{R}[f(\mathbf{x})](\Theta, \lambda) = \int_{\mathbb{R}^{d_\mathbf{x}}} A(\mathbf{x}, \Theta)\ f(\mathbf{x})\ \delta\left(\lambda + F(\mathbf{x}, \Theta)\right)\ d\mathbf{x} \tag{2.9}$$

$$\text{with } \mathbf{x} \in \mathbb{R}^{d_\mathbf{x}},\ \Theta \in \mathbb{R}^{d_\Theta},\ \text{and } \lambda \in \mathbb{R}$$

The functions f, A and F are defined such that the integral 2.9 can be computed at (Θ, λ) [25]. The integral 2.9 can be equivalently rewritten as:

$$\mathcal{R}[f(\mathbf{x})](\Theta, \lambda) = \int_{\mathcal{D}} A(\mathbf{x}, \Theta) f(\mathbf{x}) d\mathbf{x} \tag{2.10}$$

where the domain $\mathcal{D} \subset \mathbb{R}^{d_{\mathbf{x}}}$ is defined as:

$$\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^{d_{\mathbf{x}}} | \lambda + F(\mathbf{x}, \Theta) = 0\}. \tag{2.11}$$

The generalised Radon transform can be used as a technique for detecting parameterised shapes, for example, curves, circles and even hyper-sphere in larger dimensions. The way this works is, if given a mathematical model of the shape, the generalised Radon transform can be used to derive a mapping from an image or object space onto a parameter space. The axes of the parameter space corresponds to the parameters of the model. Applying the transform to an image, any shape fitting the model will result in peaks in the parameter space corresponding to the parameters of the shapes in the image [44].

Once this mapping is derived then the problem shifts to finding the peaks in the parameter space. The majority of techniques divide up the parameter space into discrete bins then, either choosing a point in the parameter space and computing its value by integrating the image or object space over all the points that belong to it, or choosing a point in the image or object space and adding its contribution to the appropriate bin in the parameter space; a process known as voting. Then either by exploring the entire parameter space or iterating over all the points in the image or object space, it is possible to detect the peaks in the parameter space.

The generalised Radon transform is widely used for pattern detection. Toft used a discrete form of the generalised Radon transform to detect curves in noisy images [103]. Hansen and Toft used a similar discrete generalised Radon transform to identify hyperbolas of particular interest from seismic data [41].

2.5 Conclusion

In this chapter the current state of the art in multi-view 3D reconstruction methods has been outlined in Section 2.1. The core principles and base methods on how different information can be used to produce a reconstruction have been detailed in the Sections

2.1.1 to 2.1.5. Methods for performing camera calibration have been outlined in Section 2.2. It is noted that there are some limitations and drawbacks to the current state of the art. The work in this thesis presents an entirely new framework with the goal of tackling these issues. By combining kernel density estimation as described in Section 2.3.1 and stochastic exploration techniques as described in Section 2.3.2, with the basic principles behind 3D reconstruction, it is possible to produce a framework which can model more accurately all the components and information available to produce a 3D reconstruction. This new framework is an extension of the generalised Radon transform which is explained in Section 2.4. The next chapter details this new framework and describes how it is applied to accomplish real 3D reconstructions.

Chapter 3

Generalised Relaxed Radon Transform for 3D Reconstruction

This chapter presents an overview of the statistical framework used as the foundation of the work being presented. Firstly a generic derivation of the framework is described. Once the generic framework is outlined then the two specific applications of the framework which have been developed and tested are presented.

3.1 Generalised Relaxed Radon Transform

The equation associated with the generalised Radon transform in the state of the art was $\lambda + F(\mathbf{x}, \Theta) = 0$ as seen in Equation 2.9. A random variable $\boldsymbol{\varepsilon}$ can be introduced and that equation can be rewritten as:

$$\boldsymbol{\lambda} + F(\mathbf{x}, \Theta) = \boldsymbol{\varepsilon} \sim \delta(\boldsymbol{\varepsilon}) \tag{3.1}$$

F is a known link function that models the relationship between a latent random vector of interest Θ, and a random vector $\mathbf{x}$ for which N independent observations $\mathbf{x}^{(i)}$ have been collected. In the generalised Radon transform the distribution of $\boldsymbol{\varepsilon}$ was modelled as a Dirac function centered on 0. Here that hypothesis is **relaxed** and $\boldsymbol{\varepsilon}$ is now a random vector modelling the perturbation or noise and has a known distribution

$p_{\boldsymbol{\varepsilon}}(\boldsymbol{\varepsilon})$ other than the Dirac function.

$$\boldsymbol{\lambda} + F(\mathbf{x}, \Theta) = \boldsymbol{\varepsilon} \sim p_{\boldsymbol{\varepsilon}}(\boldsymbol{\varepsilon}) \tag{3.2}$$

For now, no explicit meaning is given to any of the variables and they are just considered random variables. The meaning of the variables will be explained later in the context of the applications (see Sections 3.2 and 3.3).

In summary:

- $\Theta = [\theta_1, \theta_2, \cdots]$ is the latent random vector of interest.
- $\mathbf{x} = [x_1, x_2, \cdots]$ is a random vector for which N independent observations have been collected $\{\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \cdots]\}_{i=1,\cdots,N}$
- $\boldsymbol{\lambda}$ is an additive variable. For statistical reasons it is useful to have an additive variable in this situation as it helps with the derivation of the framework. For the applications considered in this thesis, the only case of interest will be when $\boldsymbol{\lambda} = 0$. Other application may use this additive variable if it fits in with the modelling.
- $\boldsymbol{\varepsilon}$ is an explicit random variable modelling the perturbation or noise. It has a known distribution $p_{\boldsymbol{\varepsilon}}(\boldsymbol{\varepsilon})$.
- F is a given function that models the relationship between the variables $\mathbf{x}$, Θ, $\boldsymbol{\varepsilon}$. F is defined as $F : \mathbb{R}^{dim(\mathbf{x})} \times \mathbb{R}^{dim(\Theta)} \rightarrow \mathbb{R}^{dim(\boldsymbol{\varepsilon})}$ (note that $dim(\boldsymbol{\varepsilon}) = dim(\boldsymbol{\lambda})$).

Given the observation variable $\mathbf{x}$, the latent variable of interest Θ and the Equation 3.2, the probability density function of $\boldsymbol{\lambda}$ can be written as:

$$p_{\boldsymbol{\lambda}|\mathbf{x},\Theta}(\boldsymbol{\lambda}|\mathbf{x}, \Theta) = p_{\boldsymbol{\varepsilon}}(\boldsymbol{\lambda} + F(\mathbf{x}, \Theta)) \tag{3.3}$$

The joint probability density function of Θ and $\boldsymbol{\lambda}$ can be computed by first considering the joint probability density function of Θ, $\mathbf{x}$ and $\boldsymbol{\lambda}$, expanded using Bayes' Theorem and then integrating with respect to $\mathbf{x}$.

$$p_{\Theta,\mathbf{x},\boldsymbol{\lambda}}(\Theta, \mathbf{x}, \boldsymbol{\lambda}) = p_{\boldsymbol{\lambda}|\mathbf{x},\Theta}(\boldsymbol{\lambda}|\mathbf{x}, \Theta)\ p_{\mathbf{x},\Theta}(\mathbf{x}, \Theta) \tag{3.4}$$

$$p_{\Theta,\boldsymbol{\lambda}}(\Theta,\boldsymbol{\lambda}) = \int p_{\boldsymbol{\lambda}|\mathbf{x},\Theta}(\boldsymbol{\lambda}|\mathbf{x},\Theta)\ p_{\mathbf{x},\Theta}(\mathbf{x},\Theta)d\mathbf{x} \tag{3.5}$$

Equation 3.3 can be substituted in for $p_{\boldsymbol{\lambda}|\mathbf{x},\Theta}(\boldsymbol{\lambda}|\mathbf{x},\Theta)$ and again Bayes' Theorem can be used to expand $p_{\mathbf{x},\Theta}(\mathbf{x},\Theta)$.

$$p_{\Theta,\boldsymbol{\lambda}}(\Theta,\boldsymbol{\lambda}) = \int p_{\boldsymbol{\varepsilon}}(\boldsymbol{\lambda}+F(\mathbf{x},\Theta))\ p_{\Theta|\mathbf{x}}(\Theta|\mathbf{x})\ p_{\mathbf{x}}(\mathbf{x})d\mathbf{x} \tag{3.6}$$

It should be noted that this is very similar to the generalised Radon transform as seen in the state of the art Section 2.4 Equation 2.9, with $p_{\mathbf{x}}(\mathbf{x})$ equivalent to $f(\mathbf{x})$, $p_{\Theta|\mathbf{x}}(\Theta|\mathbf{x})$ equivalent to $A(\mathbf{x},\mathbf{u})$, $\boldsymbol{\lambda}$ equivalent to c and $F(\mathbf{x},\Theta)$ equivalent to $I(\mathbf{x},\mathbf{u})$. The major difference is where originally the distribution of $\boldsymbol{\varepsilon}$ was modelled by a Dirac function, now this assumption is relaxed and the perturbation or noise is modelled with a known distribution $p_{\boldsymbol{\varepsilon}}(\boldsymbol{\varepsilon})$, therefore, giving rise to the generalised relaxed Radon transform.

Solving the integration gives the expectation with respect to $\mathbf{x}$.

$$p_{\Theta,\boldsymbol{\lambda}}(\Theta,\boldsymbol{\lambda}) = \mathbb{E}_{\mathbf{x}}[\ p_{\boldsymbol{\varepsilon}}(\boldsymbol{\lambda}+F(\mathbf{x},\Theta))\ p_{\Theta|\mathbf{x}}(\Theta|\mathbf{x})] \tag{3.7}$$

For the moment, within the framework the only known knowledge considered is Equation 3.2. Thus, with no other knowledge, we assume that Θ is independent of $\mathbf{x}$, therefore, $p_{\Theta|\mathbf{x}}(\Theta|\mathbf{x}) = p_{\Theta}(\Theta)$ and can therefore be taken out of the expectation. Thus $p_{\Theta}(\Theta)$ is now a prior on the latent variable. If additional information were available linking Θ and $\mathbf{x}$, then this could be included in the framework. For example, in the Statistical Hough Transform proposed by Dahyot, additional information from the gradient of the image is used to provide an additional link between the latent variable of interest and the observation [21].

$$p_{\Theta,\boldsymbol{\lambda}}(\Theta,\boldsymbol{\lambda}) = p_{\Theta}(\Theta) \times \mathbb{E}_{\mathbf{x}}[\ p_{\boldsymbol{\varepsilon}}(\boldsymbol{\lambda}+F(\mathbf{x},\Theta))] \tag{3.8}$$

The joint probability density function $p_{\Theta,\boldsymbol{\lambda}}(\Theta,\boldsymbol{\lambda})$ can be approximated using the observations. The Law of Large Numbers can be used in this situation; the average of the results obtained from a large number of observations should be close to the expected value and will tend to become closer as more observations are added. Therefore, the expected joint probability density function $\hat{p}_{\Theta,\boldsymbol{\lambda}}(\Theta, \boldsymbol{\lambda})$ can be seen as:

$$\hat{p}_{\Theta,\boldsymbol{\lambda}}(\Theta, \boldsymbol{\lambda}) = \underbrace{p_{\Theta}(\Theta)}_{\text{prior}} \times \underbrace{\frac{1}{N}\sum_{i=1}^{N} p_{\boldsymbol{\varepsilon}}(\boldsymbol{\lambda} + F(\mathbf{x}^{(i)}, \Theta))}_{\text{Averaged likelihood}} \tag{3.9}$$

In the case $\boldsymbol{\lambda} = 0$, results in:

$$\hat{p}_{\Theta,\boldsymbol{\lambda}}(\Theta, \boldsymbol{\lambda} = 0) = p_{\Theta}(\Theta) \times \overline{lik}(\Theta) \tag{3.10}$$

where

$$\overline{lik}(\Theta) = \frac{1}{N}\sum_{i=1}^{N} p_{\boldsymbol{\varepsilon}}(F(\mathbf{x}^{(i)}, \Theta)) \tag{3.11}$$

In particular the averaged likelihood $\overline{lik}(\Theta)$ as defined above is what is used throughout this work for inferring the latent variable Θ.

3.2 Application 1: 3D Reconstruction using Silhouette Information

The averaged likelihood $\overline{lik}(\Theta)$ can be used to infer the 3D shape of an object from multiple silhouette images of that object to create a Visual Hull [1].

The Stanford Bunny is a computer graphics 3D test model and it will be used as an example object throughout this work. Four images of the object can be seen in Figure 3.1.

The input in this application is a set of silhouette images where the image has been segmented and the foreground pixels are white and the background pixels are black. Four sample silhouette images of the Stanford Bunny can be seen in Figure 3.2.

For each silhouette image there is also a known 4×3 camera projection matrix. In this case our observations are the white foreground pixels across all the images, in total N foreground pixels. Each pixel has a pixel coordinate (x_1, x_2) and an associated

[1]This application has been published in 2010 in the Conference on Visual Media Production (CVMP)[86].

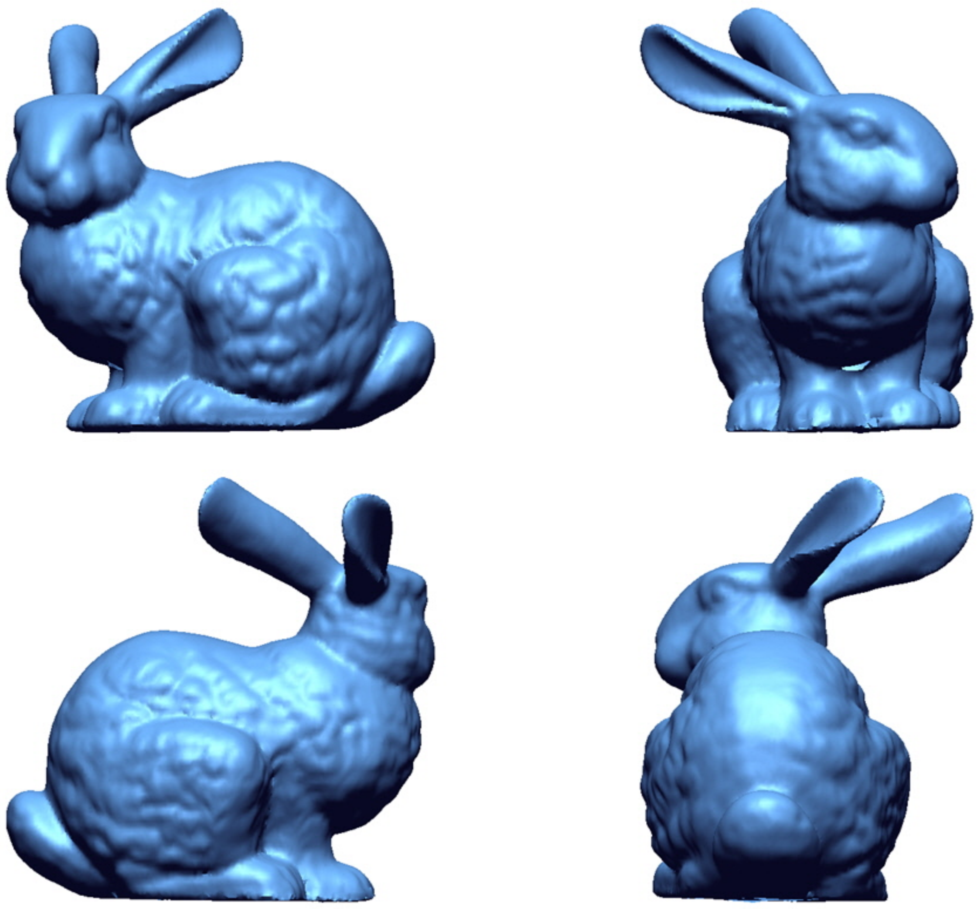


Figure 3.1: Four images of the Stanford Bunny Object.

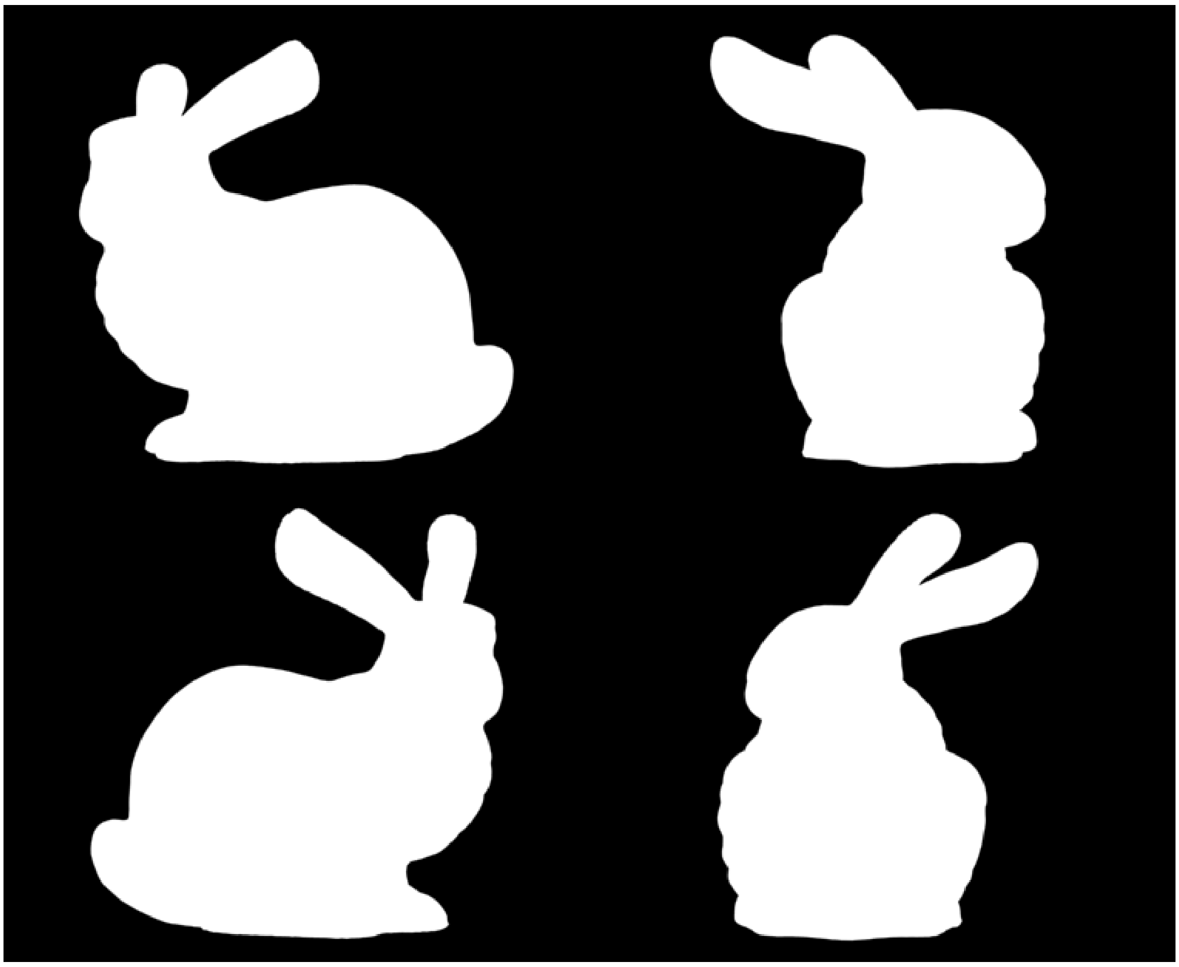


Figure 3.2: Four silhouette images of the Stanford Bunny Object, the foreground pixels are white and the background pixels are black.

4×3 camera projection matrix:

$$\begin{bmatrix} x_3 & x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 & x_{10} \\ x_{11} & x_{12} & x_{13} & x_{14} \end{bmatrix} \tag{3.12}$$

The latent variable of interest $\Theta = (\theta_1, \theta_2, \theta_3)$ is the 3D coordinate in the real world.

Each foreground pixel can be projected out into the world using the projection matrix. This can be seen as a ray starting from the camera centre and travelling through that pixel in the image plane and out into the world. As the ray travels further away from the camera centre it diverges from its neighbour's pixels. Therefore, it is better to visualise each pixel as a cone. The pointed end of the cone starts at the camera centre and travels through the pixel and out into the real world, getting bigger as it does. In fact it is best modelled as a fuzzy cone of probability that matches the perspective projection of the camera. The likelihood, therefore, of a point in space belonging to an object of interest can be calculated by summing contributions from all the cones from all the images. A point with a high likelihood will, therefore, belong to the object and a point with a low likelihood will not.

One method to calculate the contribution to the likelihood of a single point from one cone requires knowledge of the shortest distance from the point to the cone and then the size of the cone at that point along the ray. This can be a very complicated method. A slightly easier method is to project the point back into the image plane. Again using the projection matrix the 2D pixel distance is calculated between the cone's pixel and the point's pixel, as can be seen in Figure 3.3

3.2.1 Mathematical Formulation

In this case Equation 3.2 becomes:

$$\underbrace{\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix}}_{\boldsymbol{\lambda}} + \underbrace{\begin{pmatrix} x_1 - \frac{\theta_1 x_3 + \theta_2 x_4 + \theta_3 x_5 + x_6}{\theta_1 x_{11} + \theta_2 x_{12} + \theta_3 x_{13} + x_{14}} \\ x_2 - \frac{\theta_1 x_7 + \theta_2 x_8 + \theta_3 x_9 + x_{10}}{\theta_1 x_{11} + \theta_2 x_{12} + \theta_3 x_{13} + x_{14}} \end{pmatrix}}_{F(\mathbf{x},\Theta)=(F_1(\mathbf{x},\Theta),F_2(\mathbf{x},\Theta))^T} = \underbrace{\begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \end{pmatrix}}_{\boldsymbol{\varepsilon}} \tag{3.13}$$

The error $\boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2)$ is modelled as a normal (or Gaussian) density function centred on 0 with a diagonal covariance matrix with standard deviations h_1 and h_2.

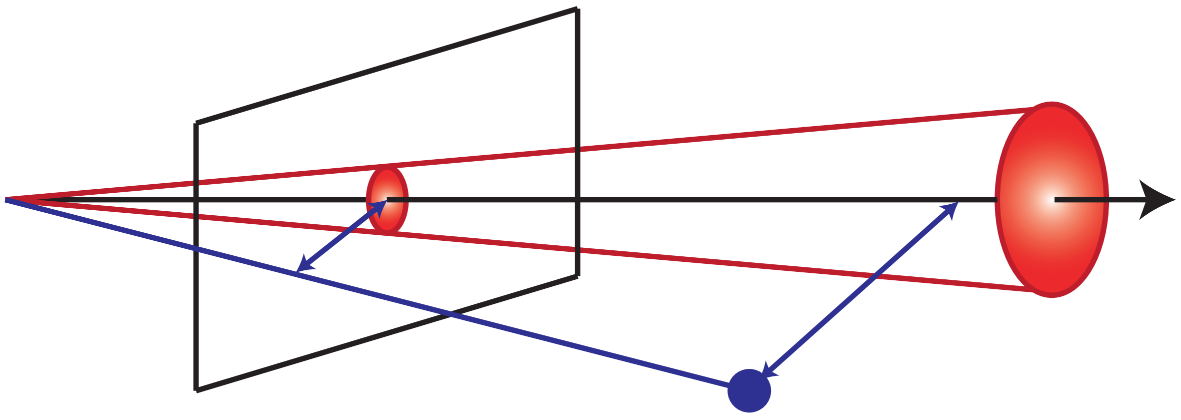


Figure 3.3: Image plane and pixel ray (black) with cone like distribution (red), the point (blue) can be projected back to the image plane using the cameras projection matrix.

$h_1 = h_2 = 1$ was chosen to model the uncertainty about the pixel position. This standard deviation can also be understood as the bandwidth of the kernel. Silverman shows that the choice of the bandwidth for kernel density estimation is more important then the choice of kernel [94]. Here a Gaussian distribution is chosen because under mild conditions the mean of a large number of random variables drawn from the same distribution is distributed approximately normally. A number of other distributions were considered and tested including the Epanechnikov, cosine, triangular and triweight. The Gaussian performed the best in the initial tests. The other distributions did not perform the gradient ascent step as well as the Gaussian distribution which had much better convergence success and rates. In the end a truncated Gaussian was used as outlined in Section 3.2.2 below.

In summary:

- $\boldsymbol{\lambda} \in \mathbb{R}^2$ - Auxiliary Variable
- $\mathbf{x} \in \mathbb{R}^{14}$ - Pixel coordinates and projection matrix
- $\Theta \in \mathbb{R}^3$ - Real world 3D coordinates
- $\boldsymbol{\varepsilon} \in \mathbb{R}^2$ - Uncertainty about pixel positions. $\varepsilon_1 \sim \mathcal{N}(0, h_1)$ and $\varepsilon_2 \sim \mathcal{N}(0, h_2)$. $h_1 = h_2 = 1$.

- $F(\mathbf{x}, \Theta) : \mathbb{R}^{14} \times \mathbb{R}^{3} \rightarrow \mathbb{R}^{2}$ - The link function which describes the distance between the observed pixel's coordinates and Θ's pixel's coordinates after projecting into the image plane.

Therefore the averaged likelihood using the framework developed can then be seen as:

$$\overline{lik}(\Theta) = \frac{1}{N \prod_{j=1}^{2} \left(\sqrt{2\pi} h_j \right)} \sum_{i=1}^{N} \prod_{j=1}^{2} \exp \left(\frac{- \left(F_j(\mathbf{x}^{(i)}, \Theta) \right)^2}{2h_j^2} \right) \tag{3.14}$$

In particular:

$$\arg\max_{\Theta} \ \overline{lik}(\Theta) \tag{3.15}$$

is the set of points which represent the points which belong to the volume of the object of interest. For reconstruction purposes only the points belonging to the surface of the object is required. The methods described in Sections 3.2.3 and 3.2.4 show how this can be achieved.

3.2.2 Truncated Gaussian

In practice using all the foreground pixels from all the cameras can be computationally very expensive. It can be seen though, that only very few of the pixels in each camera contribute to the likelihood for a point in space. The majority of the other pixels contribute very little. Using this information, it can be seen that a truncated Gaussian will give practically the same result for a fraction of the computational cost.

The point of interest is projected into each camera, the pixels closest in each camera to this projected point will make the largest contribution to the likelihood. Radiating out from this point results in less and less of a contribution. In fact when the radius is 4-5 times the standard deviation h (or bandwidth) from the centre, $> 99.99\%$ of the likelihood is accumulated. Therefore, in practice, only the pixels within a radius of the pixels corresponding to the point of interest are used. This results in a likelihood which is practically the same as considering all pixels, but at a significant fraction of the computational cost.

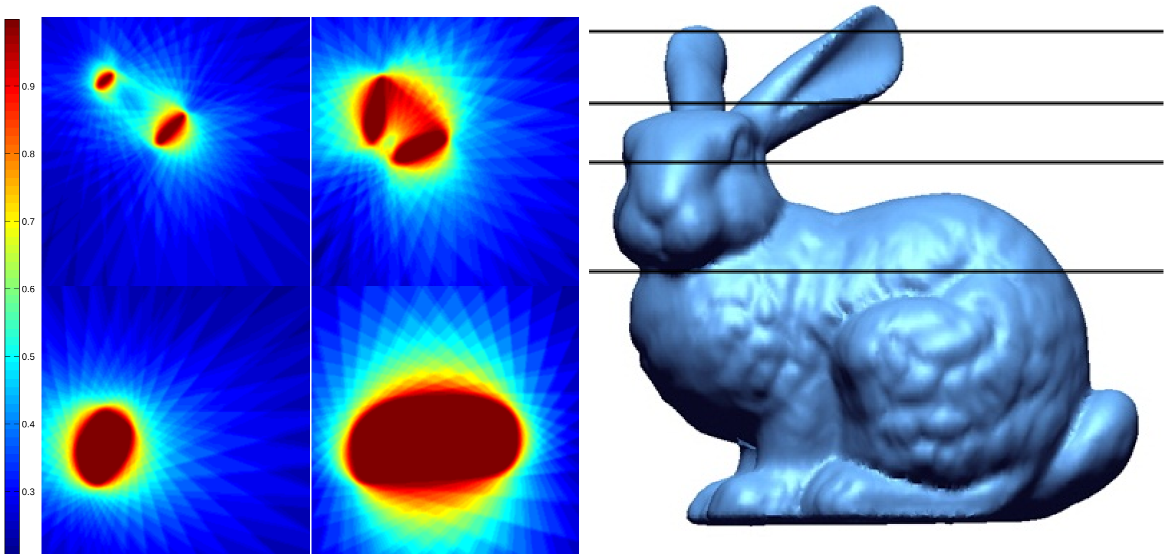


Figure 3.4: The left image shows example slices of regularly sampled likelihood of the Stanford Bunny object using 36 cameras and $h = 1$. The right image shows what slices of the object are being visualised. Top left: Top of the ears, Top right: Middle of the ears, Bottom left: Head, Bottom right: Middle of the body.

3.2.3 3D Reconstruction using Regular Sampling

Now that it is possible to determine the likelihood of a particular point in space belonging to an object of interest, it is possible to directly generate a 3D reconstruction by regular sampling. The world is broken up into a regular 3D grid and the likelihood at each point on the grid is calculated. Slices of the Stanford Bunny object can be seen in Figure 3.4

The number of cameras used greatly effects the quality of the result; four examples can be seen in Figure 3.5, where 36, 12, 9 and 3 cameras were used. It can be seen that as the number of cameras decreases the smoothness of the shape also decreases. It was observed that when using only silhouette information, cameras directly opposite each other provide little additional information to each other. It is therefore recommended to use an odd number of equally spaced cameras fully around an object or use cameras from just one 180 degree side rather then a full 360 degree.

The bandwidth or standard deviation used also has an effect on the likelihood.

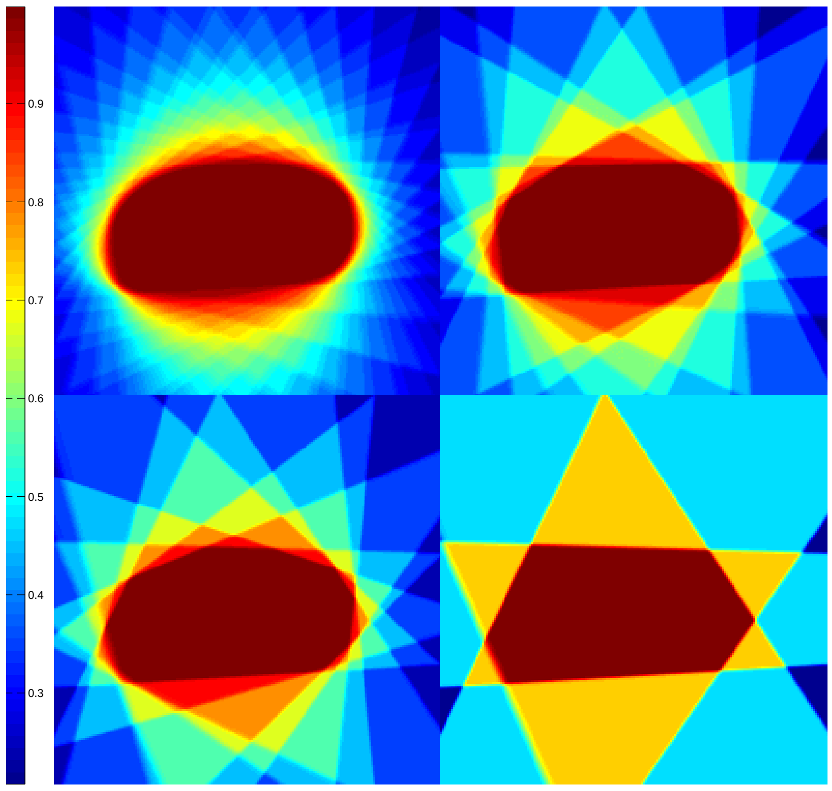


Figure 3.5: Example slices of regularly sampled likelihood of the Stanford Bunny object using different numbers of cameras and $h = 1$. Top left: 36 cameras, Top right: 12 cameras, Bottom left: 9 cameras, Bottom right: 3 cameras.

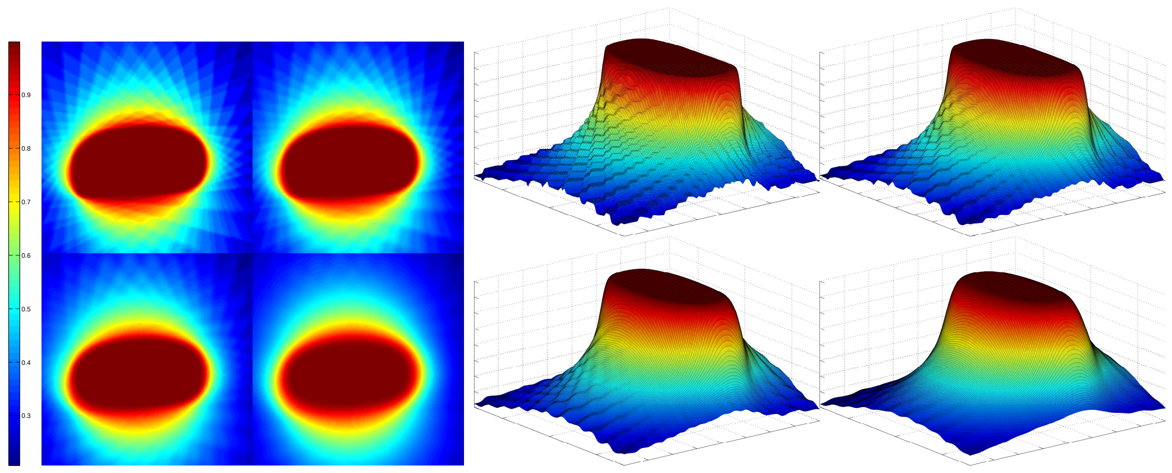


Figure 3.6: Two different visualisations of four example slices of regularly sampled likelihood of the Stanford Bunny object using different bandwidths, number of cameras = 36. Top left: $h = 1$, Top right $h = 2$, Bottom left $h = 4$, Bottom right: $h = 8$.

Bandwidth	Time(s)
1	36.8
2	108.9
4	372.7
8	2312.7

Table 3.1: Computation times for a single 200×200 slice of a reconstruction with 36 cameras for different bandwidths.

Figure 3.6 shows four different bandwidths, 1, 2, 4 and 8. It can be seen that as the bandwidth is increased the overall smoothness of the likelihood increases. This can be used to remove noise from any input data, although too large a bandwidth will result in loss of detail. A bandwidth of 1 is generally used as this models the uncertainty in the pixel location itself. It should also be noted that as the bandwidth is increased the truncated gaussian is expanded to be 4-5 times the size of the bandwidth. This results in a considerable increase in computation time as can be seen in Table 3.1. The computation time is also dependent on the number of cameras. A decrease in the number of cameras corresponds to a decrease in the computation time as seen in Table 3.2.

A full 3D reconstruction can be achieved by calculating the likelihood for multiple

No. of Cameras	Time(s)
36	36.8
12	23.2
9	21.2
3	17.9

Table 3.2: Computation times for a single 200 $\times$ 200 slice of a reconstruction with a bandwidth of 1 for different numbers of cameras.

regularly spaced slices and thresholding the likelihood. This results in a binary volume of the object of interest. The surface of the binary volume can be meshed to produce the result as seen in Figure 3.7. The reconstruction is a convex hull of the original object as it is not possible to recover concave regions with only silhouette information. It should be noted that the quality of the result depends on the quality and number of silhouette images available and the resolution of the regular sample. Increasing the number of images and the resolution of the sampling can produce better results but can greatly increase the computation cost.

This approach can be computationally very expensive, although in the advent of multi processor CPU's and highly parallel computing possibilities of GPU's, it is becoming more feasible as each point in space can be separately computed by a different core. This, along with space hierarchial techniques like octrees [100], can be used to make the reconstruction method more computationally feasible in time-constrained situations. However, using GPU's and octree structures can become very programmatically complicated.

3.2.4 3D Reconstruction using Iterative Gradient Ascent

As can be seen in the previous section, regular sampling can be computationally very expensive. Overall the goal is to find $\arg\max_{\Theta} \overline{lik}(\Theta)$. The likelihood $\overline{lik}(\Theta)$ is twice differentiable and therefore meets the requirements for using Newton's Method. It has been shown that Newton's Method can find the maxima faster and is more memory efficient than regular sampling. An example of using Newton's method for converging a point to the maxima can be seen in Figure 3.8. Newton's method includes a number of considerations as outlined in the state of the art Section 2.3.2 including; ascent direction, step size, stopping criterion and starting point. Most of these con-

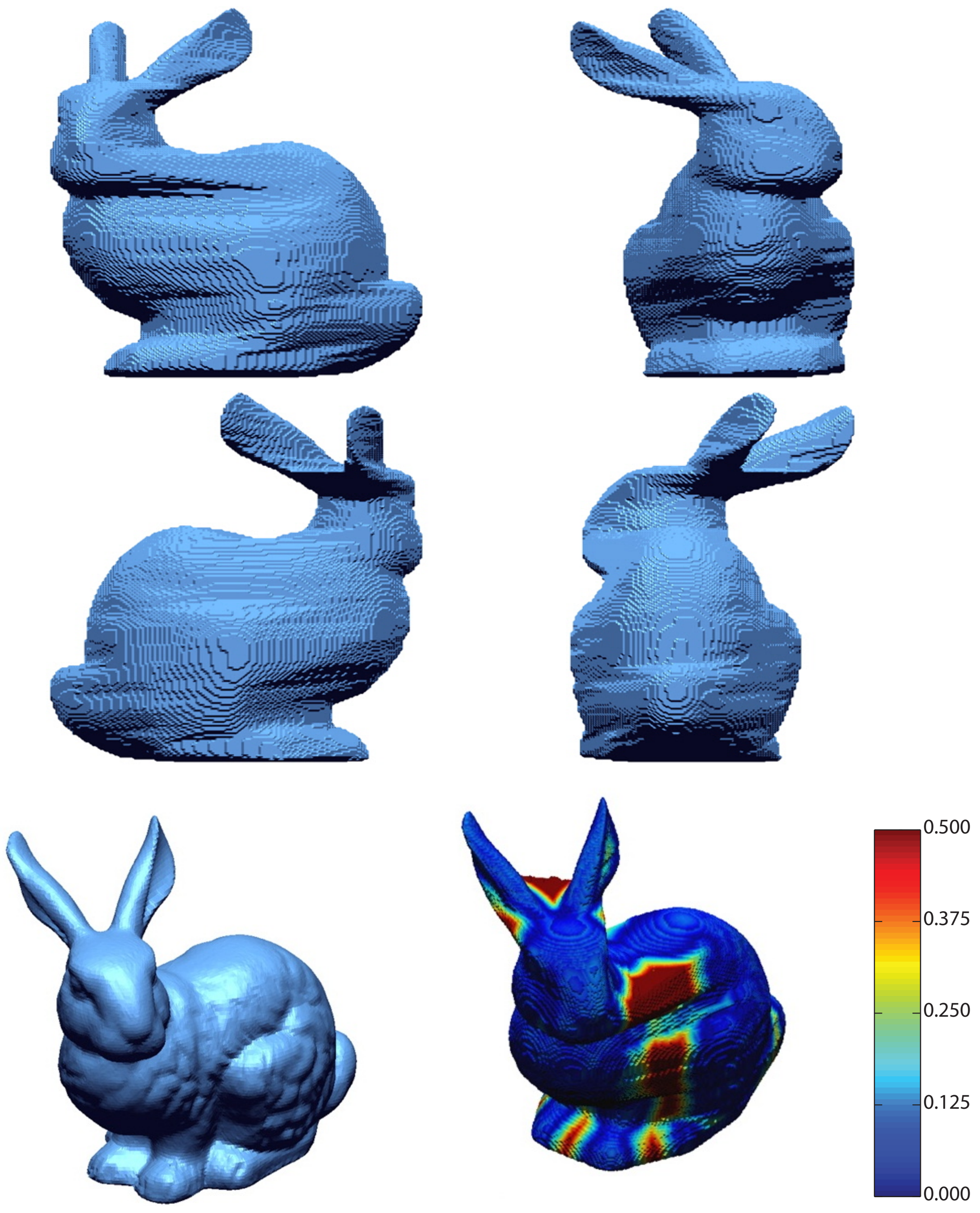


Figure 3.7: The top four images are views of the Reconstruction of the Stanford Bunny object using silhouette information and regular sampling. The bottom left is the ground truth, the bottom right is an error plot. The scale is 0-0.5 cm on an object of 10 × 13 × 13 cm. The average error is 1.23 mm. It can be seen that the worst of the error is in concavities of the object, which is expected, as using silhouette information alone can only recover a convex hull of the shape.

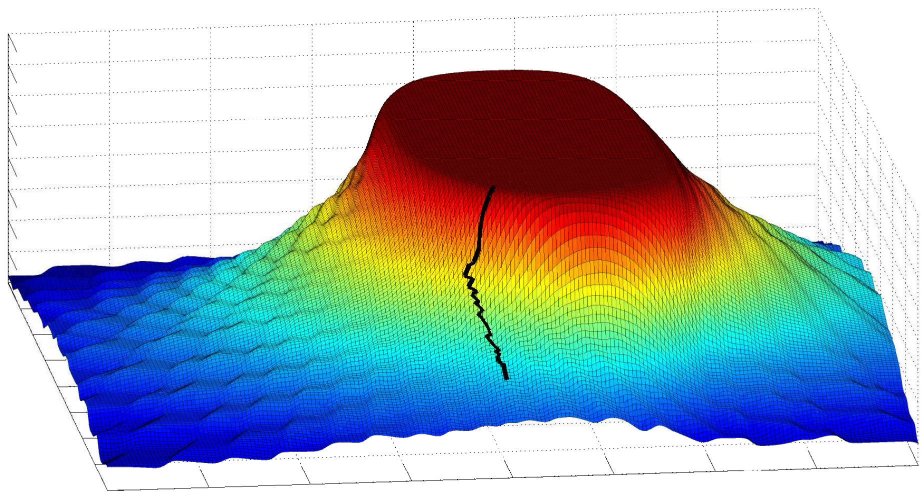


Figure 3.8: Starting at the bottom of the black line, a point is converged towards higher likelihood along the black line until it reaches the plateau where it stops at the plateau.

siderations can be solved using the standard approach. The gradient can be used to determine ascent direction. Backtracking line search can be used to determine step size. A stopping criterion can be determined by monitoring the change in likelihood and stopping when it reduces to below a small threshold. The starting point requires more consideration. There are three main situations that can occur when a random starting point is converged using Newton's Method:

- First the random point is too far away from the object with 0 likelihood. In this case sometimes increasing the bandwidth can help but if not Newton's Method will fail and the point will not converge towards the object. In fact it will most likely not move at all.
- In the second situation the point will be near the object or on the gradient. In this situation the point should move towards the object and stop at the edge of the plateau.
- In the third situation the point is inside the object. As the object is a plateau,

the point will not move because the plateau is perfectly flat and the gradient is 0. While a point of maximum likelihood is found, it is not ideal. It cannot be used to create the surface mesh as it is not on the surface but inside the object.

The goal, therefore, is to have lots of points in the second situation all the way around the object. A few methods have been tried to achieve this:

- Firstly, random points scattered throughout the space. But this method causes many points to appear in situation one and three above which then have to be culled before meshing. See Figure 3.9.
- A second method is to use prior knowledge of the scene. Having an idea where the object is and a rough estimate of the size of the object, then start with an equally spaced ring or sphere of points around the object. This works well for some shapes that are fairly simple and have a consistent average size. When an object is very irregular, with sections splitting off into bits or multiple objects this sometimes causes problems, as the radius of the ring of points has to be very wide and parts end up too far away and sections of the surface of the object end up with no points. See Figure 3.10.
- One method to overcome this problem is to do a sparse regular sampling with a very low resolution. This gives a very good idea of the object size and a first estimation of the surface of the object. Points are then placed along this estimated surface and converged to the best solution possible. While there is a cost to doing the regular sampling first, it does not need to be too expensive as it can be of very low resolution. This cost can also be offset because the closer the starting points are to the surface the faster they converge to the maxima. See Figure 3.11.
- Although not explored in this work, another method which would work in a tracking application, is to use the knowledge from the previous frames in time to predict good possible starting points for future frames.

Newton's method converges a point to the nearest maxima. In this situation the point can be converged in 3D space, moving in all three dimensions, towards the nearest

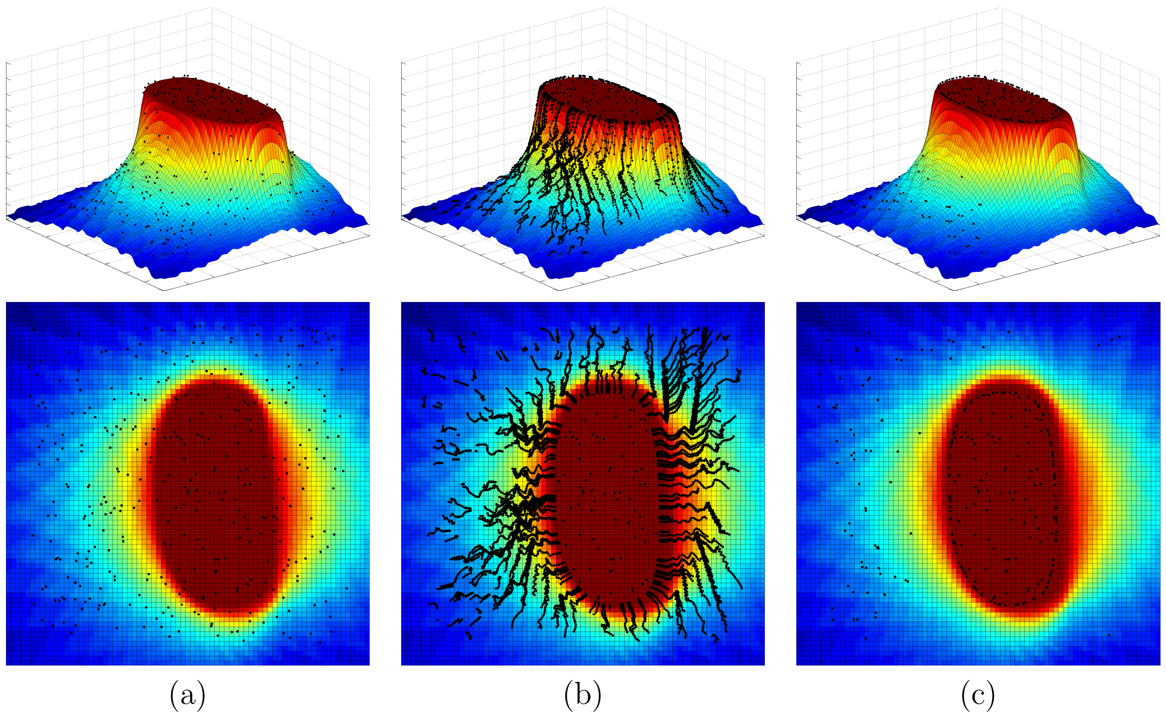


Figure 3.9: Starting with a random selection of points (a), the points are converged to maxima, their path is shown in (b) and finish at the final point locations as shown in (c).

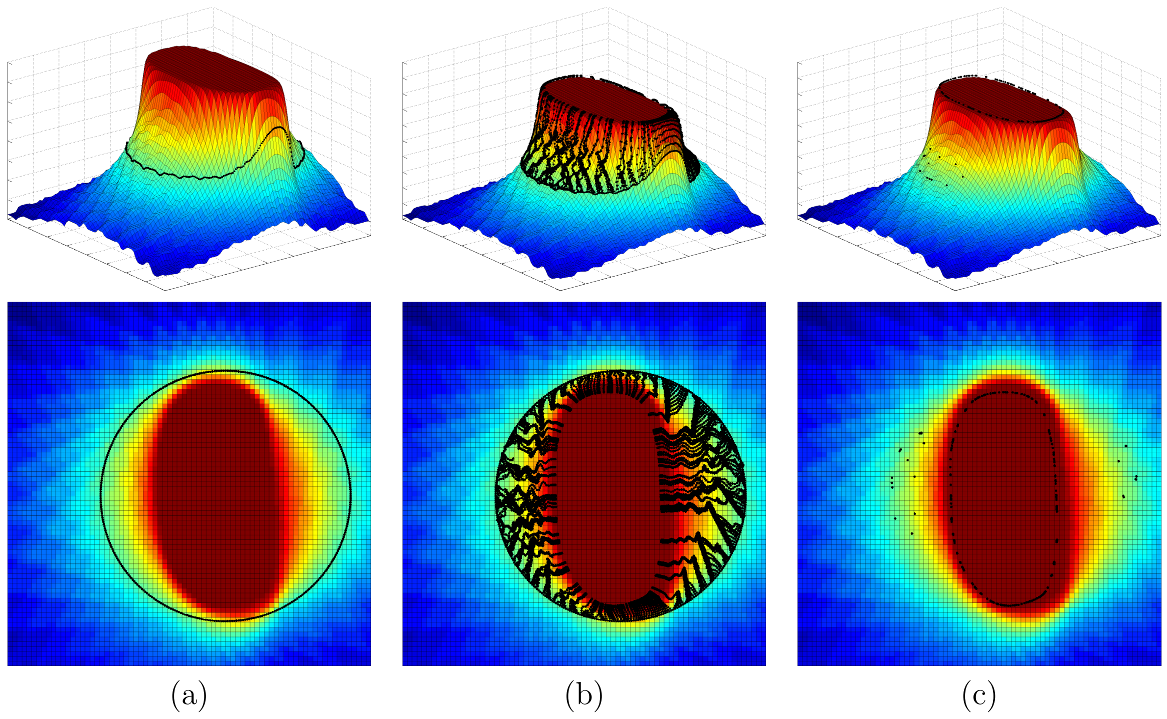


Figure 3.10: Starting with a ring of points (a), the points are converged to maxima, their path is shown in (b) and finish at the final point locations as shown in (c).

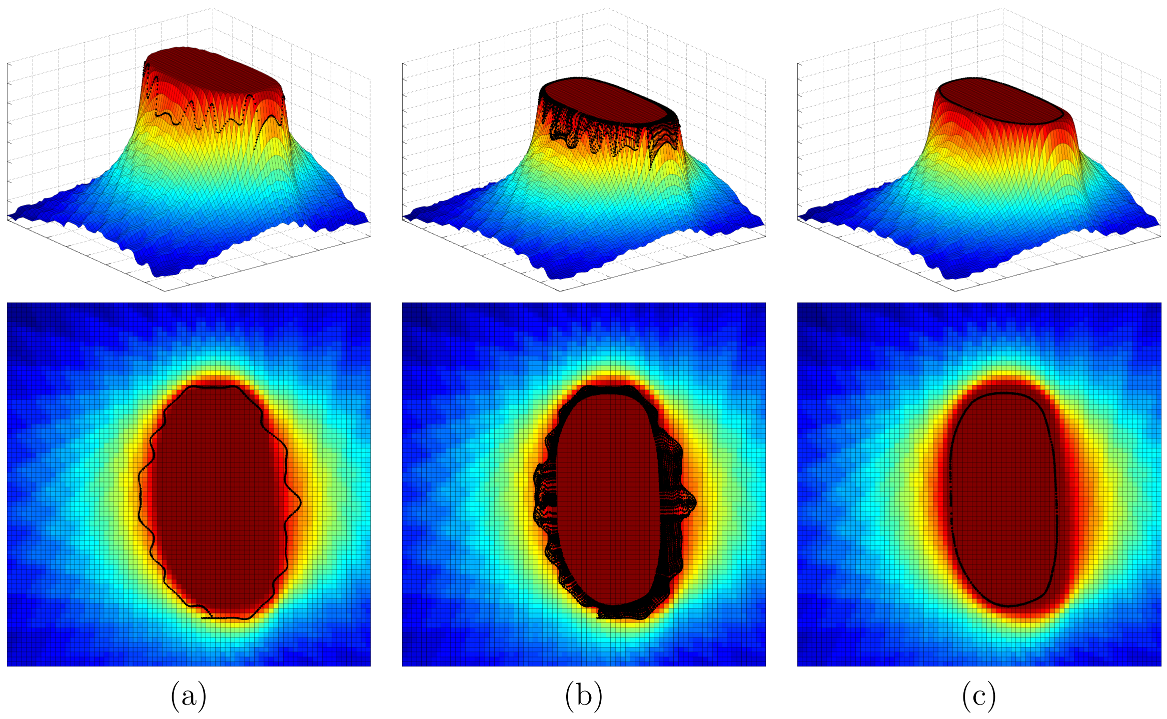


Figure 3.11: Firstly the slice is sparsely sampled. The boundary of this is then turned into points (a), the points are converged to maxima, their path is shown in (b) and finish at the final point locations as shown in (c).

3D point with the highest likelihood. To ensure a greater coverage of the surface a constraint can be placed on this movement to limit it to the 2D slice. This means that points started in that slice will stay in that slice and not move vertically. This results in a more even spread of points across the whole of the object and reduces the chance of holes in the final mesh.

Repeating the process as seen in Figure 3.11 for multiple slices of the object and combining them together can produce a complete 3D reconstruction as seen in Figure 3.12.

Kernel Bandwidth

As noted in the State of the Art Section 2.3.2 the bandwidth of the kernel is highly important for Newton's method to perform correctly. Too large a bandwidth and the result will be overly smoothed, and fine detail will be lost, but too small, will cause the noise to appear in the likelihood and result in lots of local maxima and the stochastic exploration will not be able to converge towards the required global maxima. As outlined in the state of the art, simulated annealing can be used to deliver faster convergence and helps avoid the trapping attraction of local maxima and focus on global maxima. Simulated Annealing requires two variables to be set: h_{max} and h_{min}. h_{max} can be an arbitrarily large number; in this case 10 was chosen. h_{min} should be as small as possible but still larger then the noise or accuracy of the sensor being used. In this case the sensor is a camera capturing a 2D image, which is represented by a collection of pixels, therefore, the level of accuracy of the sensor and also the optimal h_{min} for h_1 and h_2 above is 1 pixel unit.

3.3 Application 2: 3D Reconstruction using Depth Information

In many 3D Reconstruction algorithms depth information plays a vital role. Depth information can come from a number of different places. Different forms of specialist hardware can be used to capture depth information. There are also computer vision techniques solely devoted to calculating depth from 2D images, such as stereo pairs. This depth information then forms a triplet of pixel co-ordinates and depth values. In

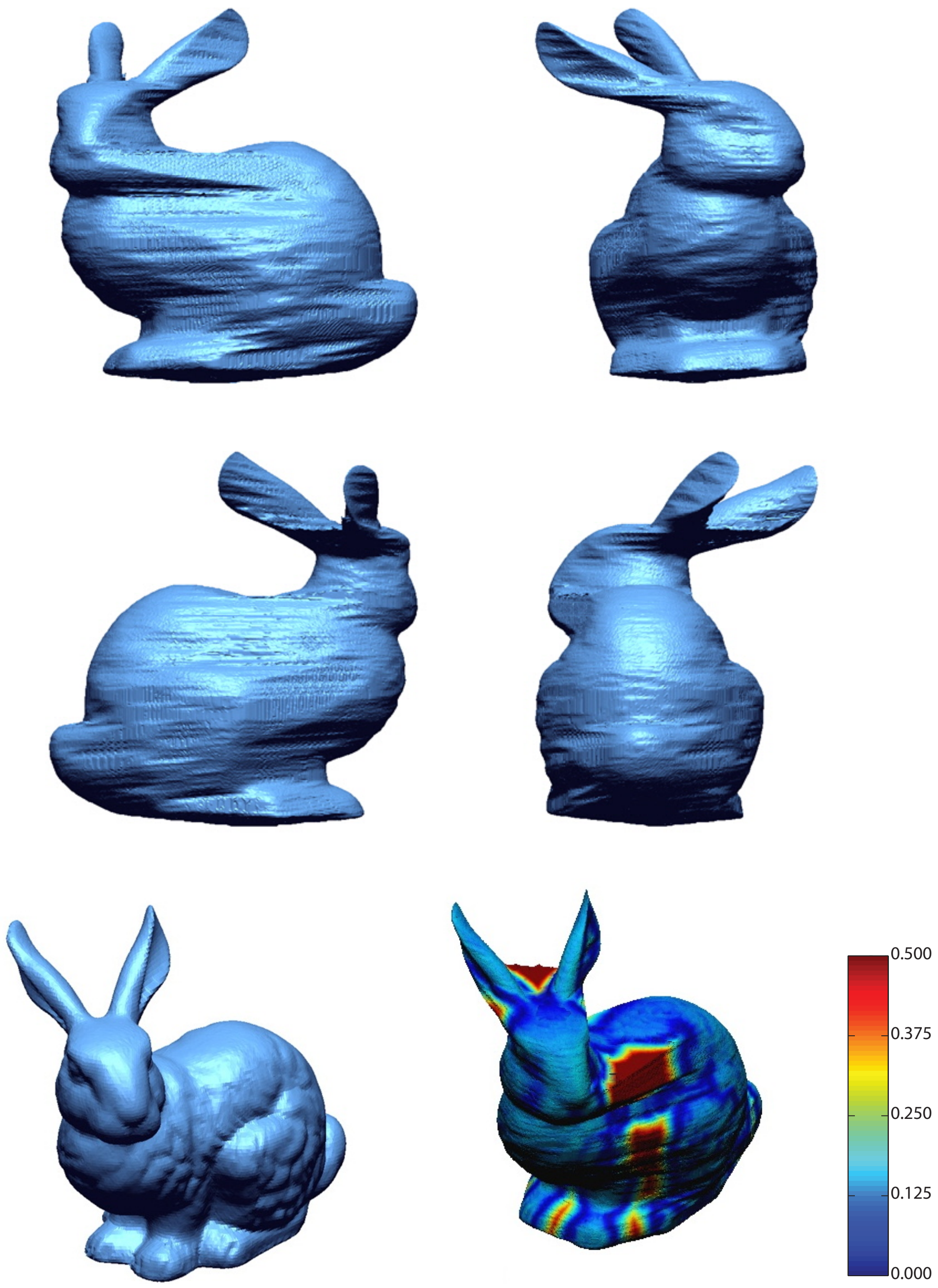


Figure 3.12: The top four images are views of the Reconstruction of the Stanford Bunny object using silhouette information and Newton's Method. The bottom left is the ground truth, the bottom right is an error plot. The scale is 0-0.5 cm on an object of 10 $\times$ 13 $\times$ 13 cm. The average error is 1.37 mm. When compared with the regular sampling method, it is possible to see the result is much smoother.

either of these situations, whether using a hardware or software solution to calculate depth, current technipques result generally in very noisy depth data. Many techniques have been developed in the past to reduce this noise by merging multiple depth data together to produce cleaner information. Depth data still presents problems with self occlusion and in general multiple images from multiple angles are required to achieve good reconstructions.

In this work use was made of a Microsoft Kinect Camera. This relatively cheap camera captures both a colour image and a depth image. The depth image is captured using a technique called structured light. This technique works by projecting a regular matrix of infrared dots. A CMOS sensor specifically tuned to see the infrared dots is used to pick up their position. Knowing the relative position of the projected dots and the sensor's dots, it is then possible to calculate the depth of each dot. In practice this results in reasonably accurate depth data, although it has some limitations, for example, seeing through glass and being unable to see mirror surfaces. The sensor has a range of roughly 0.7 - 6m. The depth data is provided in a 640 x 480 depth image with an 11 - bit channel which provides 2048 levels of sensitivity.

Autodesk 3DS Max can be used to generate synthetic data, producing the example object of the Stanford Bunny as seen in Figure 3.13.

3.3.1 Mathematical Formulation

The exact same framework as with the silhouette version can be used in the depth version, with the addition of a third component, as shown in Equation 3.16. While the first two components give the distance between the observed pixel position and the point's projected pixel's position, same as the silhouette version, the third component gives the difference between the observed depth at the point and the distance computed between the camera centre and the 3D point. It should also be noted that in the silhouette version only silhouette pixels were used, whereas this time only a depth image is available and all pixels can be used. If a silhouette image is available in addition to the depth image, only depth pixels corresponding to silhouette pixels can be used. The advantage of using this extra information greatly depends on the quality

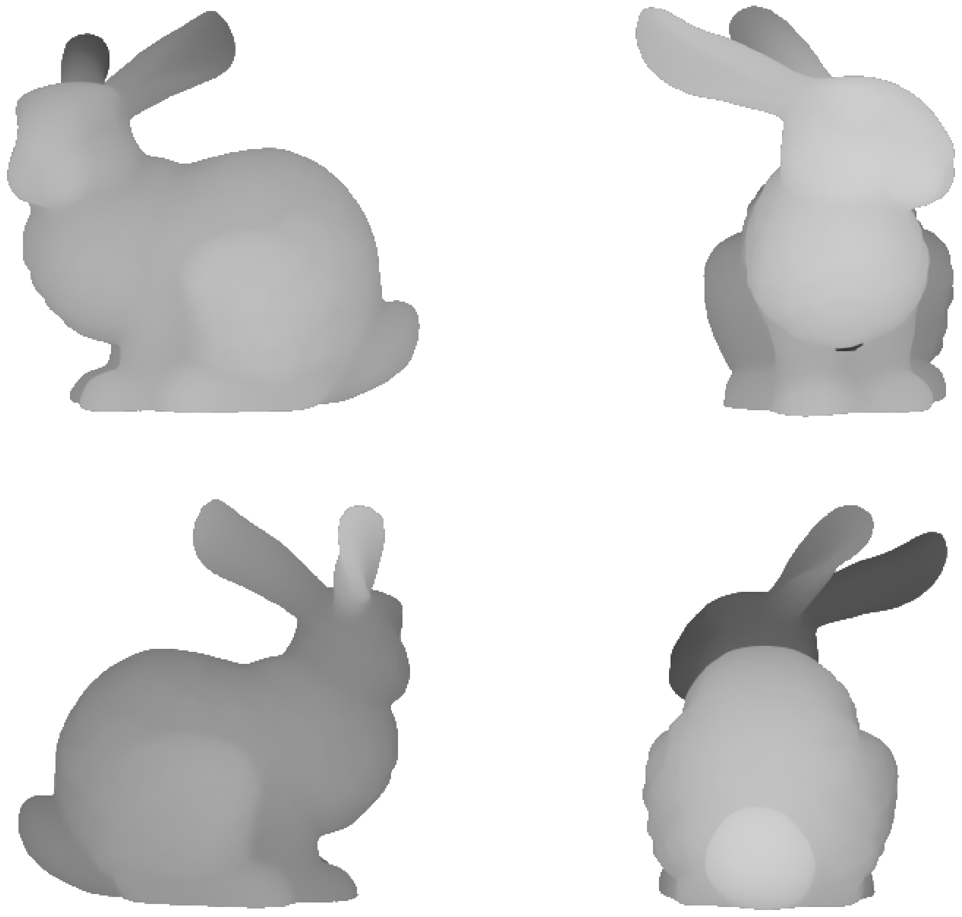


Figure 3.13: Four depth images of the Stanford Bunny Object, the darker the pixel the further away the pixel is from the camera.

of both the depth data and the silhouette data.

$$\underbrace{\begin{pmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{pmatrix}}_{\boldsymbol{\lambda}} + \underbrace{\begin{pmatrix} x_1 - \frac{\theta_1 x_3 + \theta_2 x_4 + \theta_3 x_5 + x_6}{\theta_1 x_{11} + \theta_2 x_{12} + \theta_3 x_{13} + x_{14}} \\ x_2 - \frac{\theta_1 x_7 + \theta_2 x_8 + \theta_3 x_9 + x_{10}}{\theta_1 x_{11} + \theta_2 x_{12} + \theta_3 x_{13} + x_{14}} \\ x_{15} - \sqrt{(x_{16}-\theta_1)^2 + (x_{17}-\theta_2)^2 + (x_{18}-\theta_3)^2} \end{pmatrix}}_{F(\mathbf{x},\Theta)=(F_1(\mathbf{x},\Theta),F_2(\mathbf{x},\Theta),F_3(\mathbf{x},\Theta))^T} = \underbrace{\begin{pmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_3\end{pmatrix}}_{\boldsymbol{\varepsilon}} \tag{3.16}$$

In summary:

- $\boldsymbol{\lambda} \in \mathbb{R}^3$ - Auxiliary Variable
- $\mathbf{x} \in \mathbb{R}^{18}$ -

(x_1, x_2)	Pixel Coordinate in image space
$\begin{bmatrix} x_3 & x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 & x_{10} \\ x_{11} & x_{12} & x_{13} & x_{14} \end{bmatrix}$	Camera Projection Matrix
x_{15}	Depth value observed
(x_{16}, x_{17}, x_{18})	Camera centre coordinate

- $\Theta \in \mathbb{R}^3$ - Real world 3D coordinates.
- $\boldsymbol{\varepsilon} \in \mathbb{R}^3$ -

$\varepsilon_1 \sim \mathcal{N}(0, h_1)$	Uncertainty about pixel positions
$\varepsilon_2 \sim \mathcal{N}(0, h_2)$	
$\varepsilon_3 \sim \mathcal{N}(0, h_3)$	Uncertainty about depth data

- $F(\mathbf{x}, \Theta) : \mathbb{R}^{18} \times \mathbb{R}^3 \to \mathbb{R}^3$ - Distance between observed pixel's coordinate and Θ's pixel's coordinates after projecting into the image plane.

Therefore, the averaged likelihood using the framework developed can then be seen as:

$$\overline{lik}(\Theta) = \frac{1}{N \prod_{j=1}^{3} \left(\sqrt{2\pi} h_j\right)} \sum_{i=1}^{N} \prod_{j=1}^{3} \exp\left(\frac{-\left(F_j(\mathbf{x}^{(i)}, \Theta)\right)^2}{2h_j^2}\right) \tag{3.17}$$

The resulting likelihood can be explained in two parts: the first part is the same silhouette cone from the first likelihood as seen the first diagram of Figure 3.14. The second part is the depth information. This creates a fuzzy doughnut shaped ring of

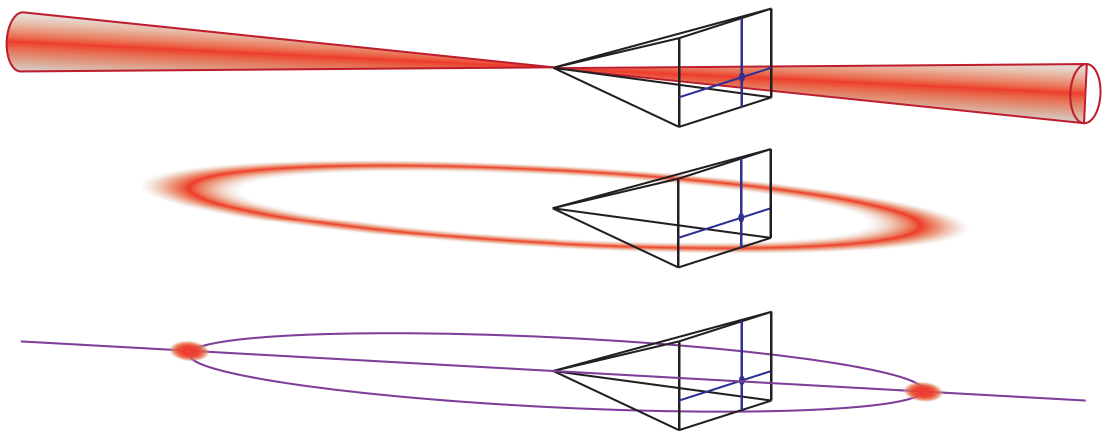


Figure 3.14: The combination of the cone from the pixel and the depth information is combined to give the likelihood of where the corresponding surface point of the object is in the space.

probability around the camera centre with a radius equal to the depth value observed at the pixel location, as seen in the second diagram of Figure 3.14. These two parts combined give the final probability as seen in the third diagram of Figure 3.14. It can be seen that there are actually two areas of probabilities created. This is because the cone is actually mirrored behind the camera as well as in front. Only the area in front of the camera should be considered. In practice using the camera positions and orientation, limits can be set on the area of interest. This allows the areas behind the cameras to be ignored.

In this situation :

$$\arg\max_{\Theta} \; \overline{lik}(\Theta) \tag{3.18}$$

becomes the set of points which represent points on the surface of the object.

It is possible to use a simpler link function. This involves preprocessing the depth images and converting them into a point cloud. The link function would then place a fuzzy sphere around each point. This produces a much simpler and very similar end likelihood but in practice the more complicated version works better. It models the noise more accurately where the depth and pixel size errors are accounted for separately, whereas in the simpler version the two are combined and so does not perform as well. The same section of the object can be seen in Figure 3.15 with the two models, the

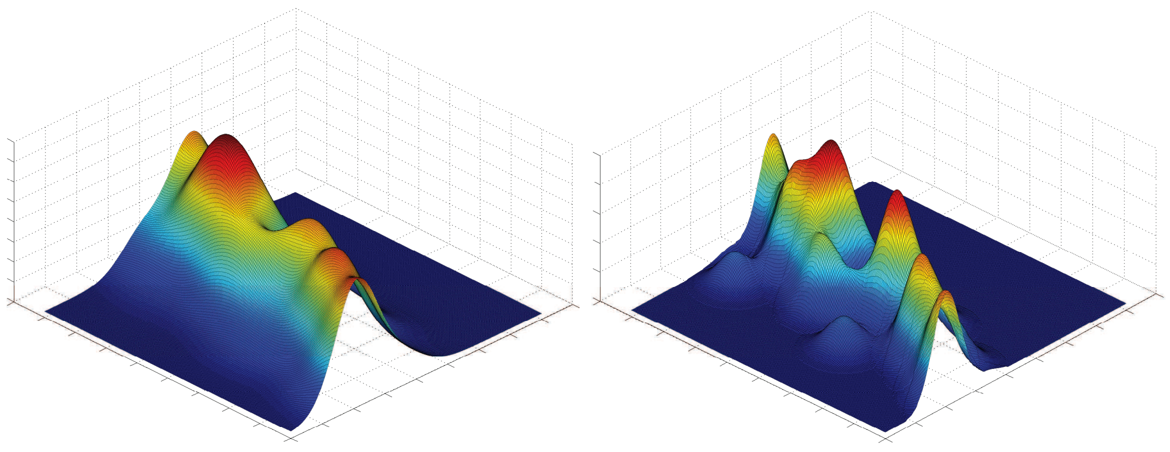


Figure 3.15: Comparison of the two depth models, original model (left) with a pixel bandwidth of 1 and a depth bandwidth of 0.0002. The second model (right) just has a depth bandwidth of 0.0002. A ridge can be seen in the first model as the input data connects together but in the second model the input data separates out into individual peaks which do not connect and the ridge disappears.

first one is the original model with a pixel bandwidth of 1 and a depth bandwidth of 0.0002 (while this may appear very small the unit is in meters and the object is only 10 × 13 × 13 cm in size). The second model just has a depth bandwidth of 0.0002. A ridge can be seen in the first model as the input data connects together but in the second model the input data separates out into individual peaks which do not connect and the ridge disappears. An alternative visualisation of the two models can be seen in Figure 3.16. Again the different bandwidths allow for a more precise model of the data to be projected, which can change depending on the pixel position.

3.3.2 Regular Sampling

Using the same methods of regular sampling, as with the silhouette version, can be done with the depth version. Four slices of the example object can be seen in Figure 3.17. The first major difference that can be seen is that now the maxima represents only the surface of the object and not the whole volume of the object as seen in the silhouette version . The other thing that can be seen is that the maxima is not level the whole way around the object. Some parts of the surface of the object have a greater

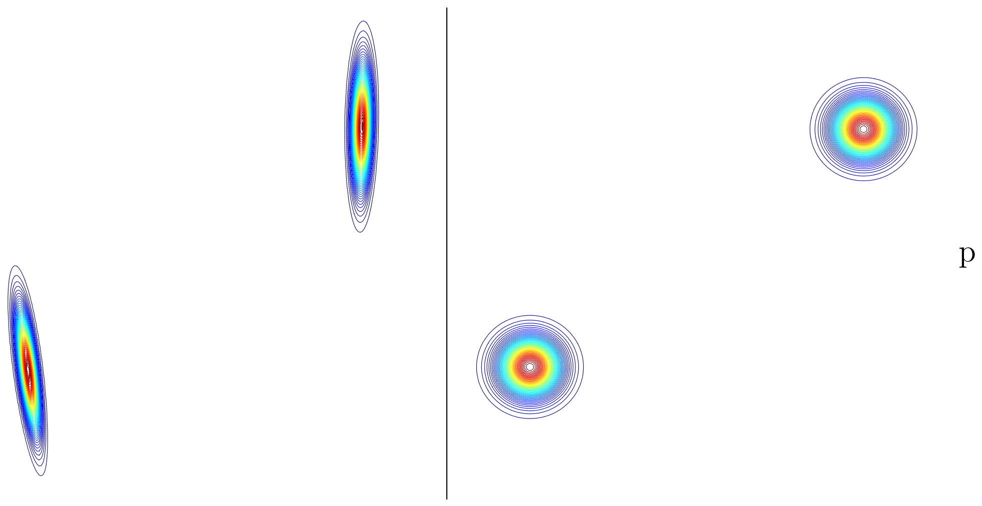


Figure 3.16: An alternative visualisation of the comparison of the two depth models. Here just two data points are in the source data, and this image shows the likelihood model for their locations in the space. The original model (left) with a pixel bandwidth of 1 and a depth bandwidth of 0.0002. The second model (right) just has a depth bandwidth of 0.0002. The different bandwidths allow for a more precise model of the data to be projected, which can change depending on the pixel position.

likelihood than other parts. Indeed the magnitude of the likelihood is directly related to the number of views which can see it. For example, if part of the surface is only visible by a few cameras then it will have a lower likelihood than part of the surface that is visible by a larger number of cameras.

3.3.3 Iterative Gradient Ascent

In the same way as with the silhouette version, an iterative gradient ascent algorithm, Newton's Method, can be used to explore the space of interest faster than Regular Sampling. Unfortunately an additional problem occurs unlike before. While the maxima of the likelihood was a plateau, it is now an irregular ridge. Points will converge up the slope to the top of the ridge, but do not stop there. They continue along the ridge until they reach a maxima on the ridge. This means that even with an equally spaced set of starting points, the points end up converging to a small set of ridge maxima, which is useless for generating a 3D mesh form, as seen in Figure 3.18.

A method is therefore required which will reach the ridge and then generate points

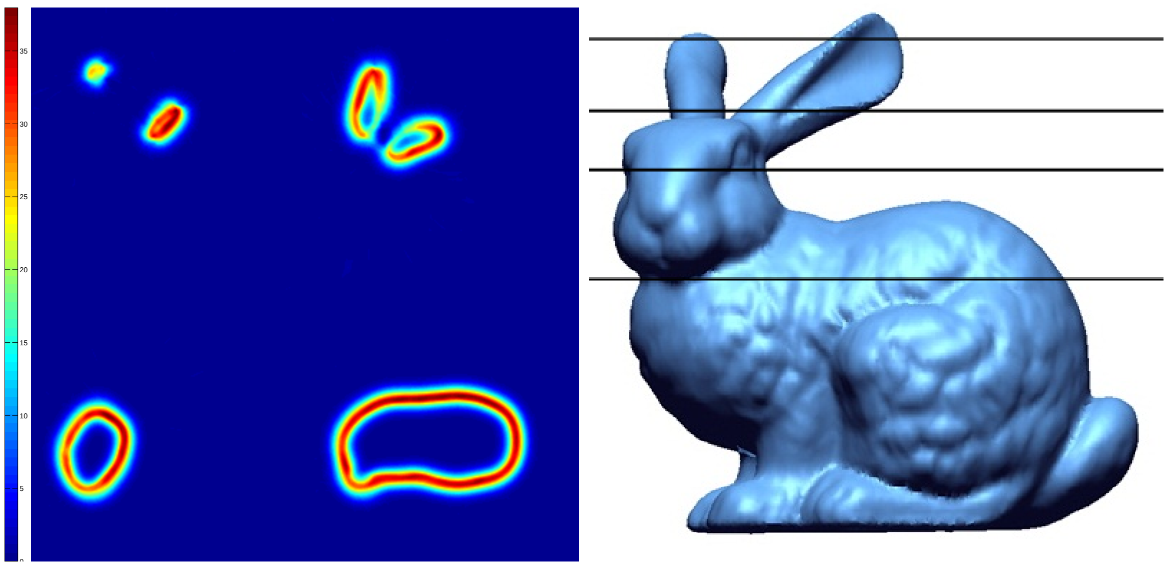


Figure 3.17: The left image shows example slices of regularly sampled likelihood with depth information of the Stanford Bunny object using 36 cameras and $h = 1$. The right image shows what slices of the object are being visualised. Top left: Top of the ears, Top right: Middle of the ears, Bottom left: Head, Bottom right: Middle of the body.

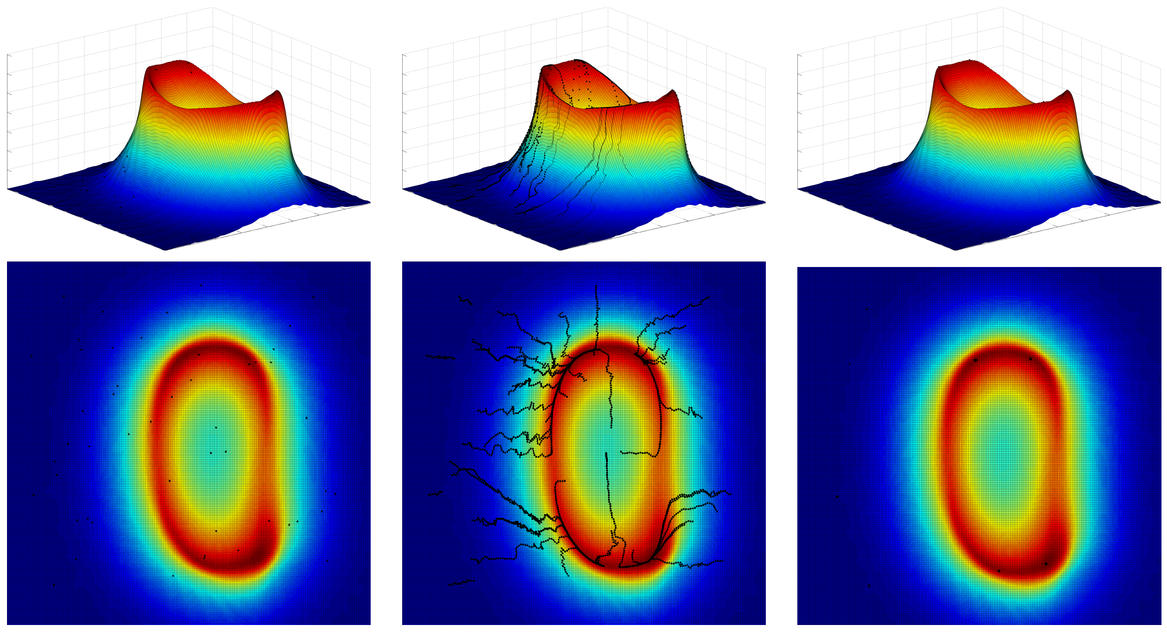


Figure 3.18: Similarly to the silhouette version, 50 random points are selected for starting positions (left image), they converge to the nearest maxima using Newton's algorithm. The final locations can be seen in the right image, the problem is the majority of the 50 points have converged to just 4 points on the ridge, not equally spaced as is required for good reconstruction.

as it moves along the ridge.

3.3.4 Surface Explore

When the likelihood of a slice of the surface can be seen as an irregular ridge, the goal of 'Surface Explore' is to travel along that ridge and not gets trapped at maxima along the ridge.

Figure 3.19 shows first an example section of a ridge. All the points so far have converged to the peak of this ridge (black dot). A doughnut shaped likelihood (middle image of Figure 3.19) is added to achieve the effect seen in the final image of Figure 3.19. This doughnut shaped likelihood zeros the majority of the ridge except for two points either side of the centre point on the ridge. It is then possible to converge to either of these points, which gives a point on the ridge that would not be the maxima of the ridge, only another point on the ridge roughly the size of the doughnut away from the last point. The process is repeated travelling the whole way around the object until the start point is reached again. Adding this modification to the framework results in Equation 3.19. This 'Surface Explore' method can be applied to the full 3D volume of the likelihood to explore the surface in any direction, but to simplify the inference, it is applied to 2D slices of the likelihood.

$$\underbrace{\begin{pmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\\ \lambda_4\end{pmatrix}}_{\boldsymbol{\lambda}} + \underbrace{\begin{pmatrix} x_1 - \frac{\theta_1^{(n+1)}x_3+\theta_2^{(n+1)}x_4+\theta_3^{(n+1)}x_5+x_6}{\theta_1^{(n+1)}x_{11}+\theta_2^{(n+1)}x_{12}+\theta_3^{(n+1)}x_{13}+x_{14}} \\ x_2 - \frac{\theta_1^{(n+1)}x_7+\theta_2^{(n+1)}x_8+\theta_3^{(n+1)}x_9+x_{10}}{\theta_1^{(n+1)}x_{11}+\theta_2^{(n+1)}x_{12}+\theta_3^{(n+1)}x_{13}+x_{14}} \\ x_{15} - \sqrt{(x_{16}-\theta_1^{(n+1)})^2+(x_{17}-\theta_2^{(n+1)})^2+(x_{18}-\theta_3^{(n+1)})^2} \\ x_{19} - \sqrt{(\hat{\theta}_1^{(n)}-\theta_1^{(n+1)})^2+(\hat{\theta}_2^{(n)}-\theta_2^{(n+1)})^2+(\hat{\theta}_3^{(n)}-\theta_3^{(n+1)})^2} \end{pmatrix}}_{F(\mathbf{x},\hat{\Theta}^{(n)},\Theta^{(n+1)})=(F_1(\mathbf{x},\hat{\Theta}^{(n)},\Theta^{(n+1)}),F_2(\mathbf{x},\hat{\Theta}^{(n)},\Theta^{(n+1)}),F_3(\mathbf{x},\hat{\Theta}^{(n)},\Theta^{(n+1)}),F_4(\mathbf{x},\Theta))^T} = \underbrace{\begin{pmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_3\\ \varepsilon_4\end{pmatrix}}_{\boldsymbol{\varepsilon}} \tag{3.19}$$

In summary:

- $\boldsymbol{\lambda} \in \mathbb{R}^4$ - Auxiliary Variable
- $\mathbf{x} \in \mathbb{R}^{22}$ -

(x_1, x_2)	Pixel Coordinate in image space
$\begin{bmatrix} x_3 & x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 & x_{10} \\ x_{11} & x_{12} & x_{13} & x_{14} \end{bmatrix}$	Camera Projection Matrix
x_{15}	Depth value observed
(x_{16}, x_{17}, x_{18})	Camera centre coordinate
x_{19}	Radius of doughnut

- $\hat{\Theta}^{(n)} \in \mathbb{R}^3$ - Estimated real world 3D coordinates of previous point on ridge.
- $\Theta^{(n+1)} \in \mathbb{R}^3$ - Real world 3D coordinates of next point on ridge.
- $\boldsymbol{\varepsilon} \in \mathbb{R}^4$ -

$\varepsilon_1 \sim \mathcal{N}(0, h_1)$	Uncertainty about pixel positions
$\varepsilon_2 \sim \mathcal{N}(0, h_2)$	
$\varepsilon_3 \sim \mathcal{N}(0, h_3)$	Uncertainty about depth data
$\varepsilon_4 \sim \mathcal{N}(0, h_4)$	The radius of the doughnut tube

- $F(\mathbf{x}, \hat{\Theta}^{(n)}, \Theta^{(n+1)}) : \mathbb{R}^{19} \times \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^4$

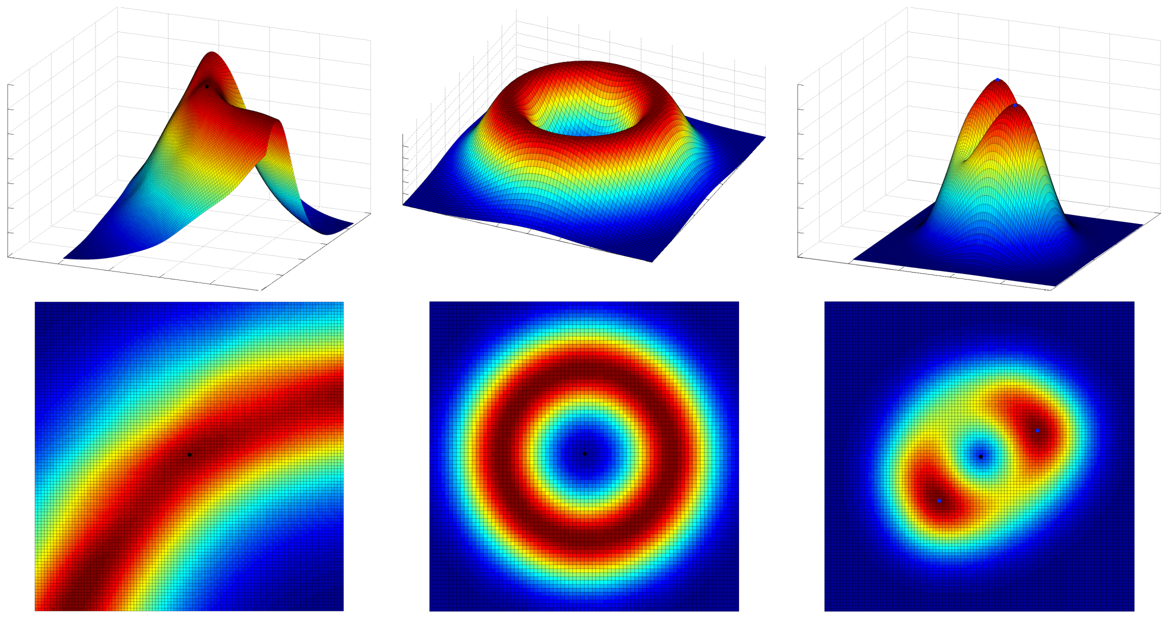


Figure 3.19: The likelihood of the surface of an object can be seen as an irregular ridge (left most image). Due to the irregular height of the ridge, points when converged go to a maxima on the ridge and are not evenly spread along the ridge. By combining a doughnut shaded likelihood (middle image) with the ridge likelihood it is possible to determine a next and previous point along the ridge (right image). The current point $\hat{\Theta}^{(n)}$ can be seen in black. The possible next points $\Theta^{(n+1)}$ can be seen in blue on the top of the peaks in the right image.

Therefore, the averaged likelihood using the framework developed can then be seen as:

$$\overline{lik}(\Theta^{(n+1)}|\hat{\Theta}^{(n)}) = \frac{1}{N\prod_{j=1}^{4}\left(\sqrt{2\pi}h_j\right)} \sum_{i=1}^{N}\prod_{j=1}^{4} \exp\left(\frac{-\left(F_j(\mathbf{x}^{(i)},\hat{\Theta}^{(n)},\Theta^{(n+1)})\right)^2}{2h_j^2}\right) \tag{3.20}$$

The first point $\hat{\Theta}^{(1)}$ can be found using:

$$\hat{\Theta}^{(1)} = \arg\max_{\Theta}\ \overline{lik}(\Theta) \qquad \text{with } \overline{lik} \text{ defined by Equation 3.17} \tag{3.21}$$

This point can then be used to find the next point along the ridge using Equation

3.20:

$$\hat{\Theta}^{(n+1)} = \arg\max_{\Theta^{(n+1)}} \ \overline{lik}(\Theta^{(n+1)}|\hat{\Theta}^{(n)}) \tag{3.22}$$

This is repeated until a stopping criteria is reached.

The goal is to create a chain of points around the object and stop when the chain returns back to the first point to form a ring of points. This has a number of problems, for example, if the object does not loop or form a complete ring. Another situation is when there are multiple rings as seen around the ears of the bunny object; the object splits into multiple parts.

Starting Points

To make sure as much of the object is covered and as many parts of the object are covered (if it is split into multiple sections), multiple starting points are used. A regular 5×5 or 6×6 grid of points is used in the proximate area where the object should be. Each point is converged using Newton's method to its nearest maxima. At this point any duplicated points are thrown away, that is, where 2 or more points have converged to the same maxima, there is no need to keep multiple copies. Any points with really low likelihood are also discarded. These are generally points that are not close enough to any maxima to converge at all. The points that are left are now the starting points for any chain or chains along the ridges in the likelihood. The points are sorted in order of highest likelihood, and the first starting point used is the one with the highest likelihood.

Next Starting Point

Once one point on the ridge has been found, the next point is found using Equation 3.22. The next point should be roughly x_{19} (the radius of the doughnut) away from the first point. For the second point along the ridge it is unknown in what direction that next point should be found, so multiple directions can be used, and then converged using Newton's method. After two points on the ridge have been found the relative direction from the previous two positions can be used to give an estimate of the direction to the next point. This can be used to give very close starting positions for the next point along the ridge. The closer the start position is to the maxima the faster it will converge and so the faster the overall reconstruction will take.

Stopping Criteria

Ideally there would be one chain of points along the ridge and that would be stopped when it looped back around to the beginning, but now there can be multiple points on the ridge, and the ridge may not form a closed loop. If the likelihood of a ridge drops dramatically, two problems may occur; the next point will start following some low level noise or it will follow the same ridge back and repeat roughly the previous points in reverse order. Either situation is not good. So if either the likelihood drops below a threshold or the next point comes within a minimum distance (generally the same as the radius of the doughnut) of any of the previous points excluding the immediately previous point, then the chain is stopped. If the chain has not formed a ring looping back to the beginning, then the chain has two ends. Both ends can be continued until either stopping criteria is met. Once both ends of the first starting point have been stopped, the process is repeated with the next starting point. In this case the stopping criteria is based on both the current chain and the other chains, as it is not necessary to repeat exploration of any ridge that has already been explored.

The process is repeated for multiple slices of the object to achieve a full reconstruction. It is possible to repeat the process again, now using slices in a different orientation to improve surface completeness.

3.3.5 Result

The final result of using depth information and surface explore can be seen in Figure 3.20. Immediately it can be seen that the result is of much higher quality which is expected when using depth information, and now the concave regions are also discovered more accurately. A more in depth analysis of these results can be found in Chapter 5 and a full set of results for different objects and camera numbers can be found in Appendix B.

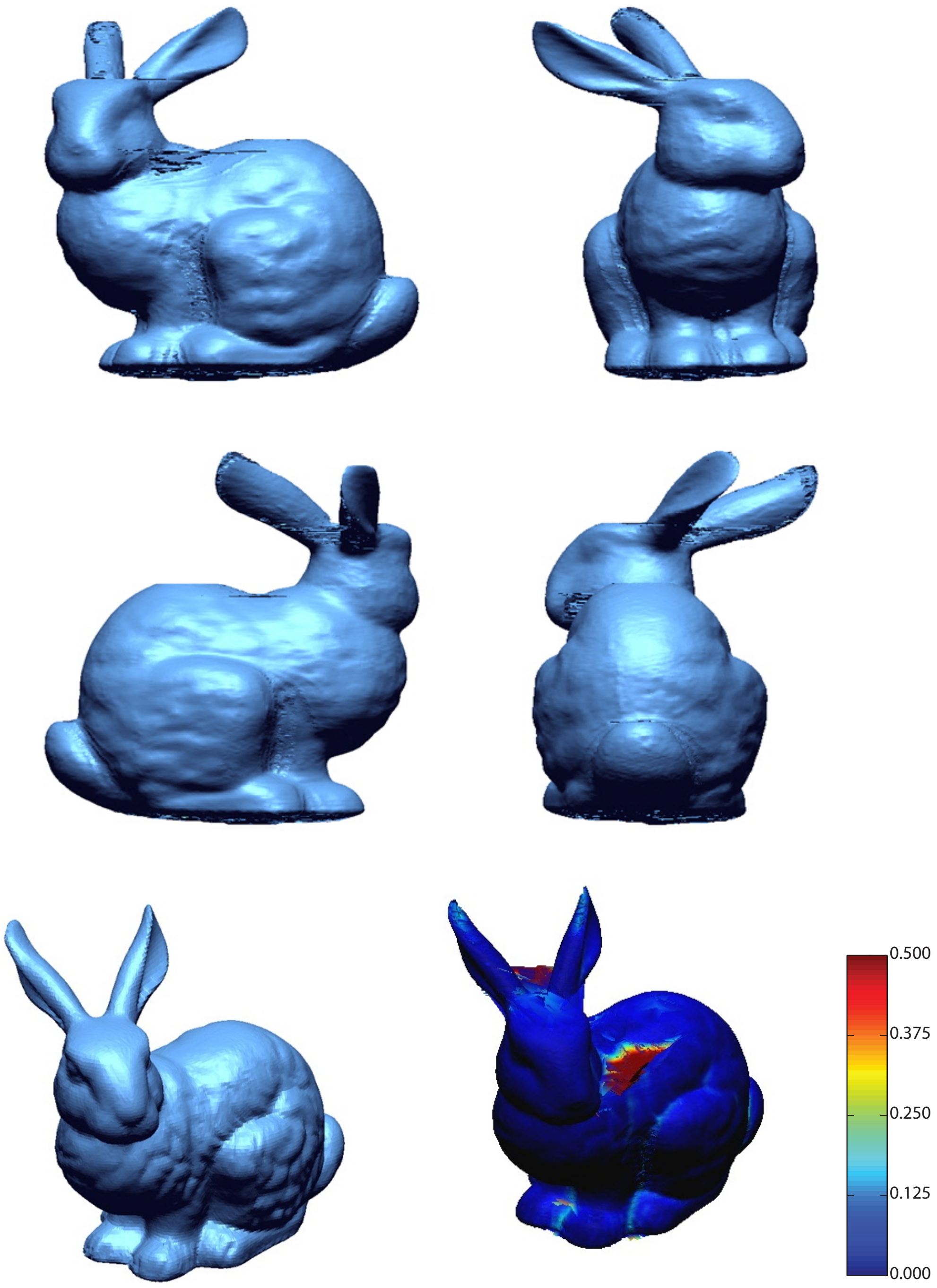


Figure 3.20: Four views of the Reconstruction of the Stanford Bunny object using depth information and surface explore. The scale is 0-0.5 cm on an object of 10 $\times$ 13 $\times$ 13 cm. The average error is 0.5 mm. When compared with the regular sampling method and Newton's method, it is possible to see the result has greatly improved with greater detail visible.

3.4 Conclusion

In this chapter a new novel framework for merging data has been presented in Section 3.1. Two examples of where this framework can be applied to real 3D reconstruction problems has been presented in Sections 3.2 and 3.3 with final results seen in Figures 3.12 and 3.20.

The work done with Donghoon Kim can be shown to be an application of this new modelling where the cameras are orthographic [51]. The preliminary work has been extended by Kim to use colour as observation and infers the photo hull [50]. Additionally, Kim showed that prior information, when available, can also be added in the modelling to improve the reconstruction [50]. While these applications were limited to the simpler orthographic camera model, it is proof of concept that the overall framework is very flexible and can allow any information available to be used to generate the likelihood model, which can then be further explored using gradient ascent methods.

Up to this point all the methods were only applied to synthetic data which was generated using Autodesk 3DS Max which had perfect camera calibration and alignment. When real data was used from a real camera (Microsoft Kinect for XBox), it was found that camera calibration is very important, especially for depth data. While noise in any individual image would be modelled and overcome in the modelling, poor alignment between different images would create competing results for the same surface at different points in space. It is possible to overcome this problem with an extremely high bandwidth but the object gets completely over-smoothed to the point where it has no recognizable features. An alternative is to improve the alignment of the cameras before attempting the reconstruction. The next chapter presents a new framework which is an extension of the framework presented in this chapter for aligning the cameras.

Chapter 4

Generalised Relaxed Radon Transform with Nuisance Variables

In this chapter the framework presented in the last chapter (see Section 3.1) is extended to encompass and optimise any nuisance variables which may be present in the probabilistic model. A nuisance variable is a random variable appearing in the modelling, but that is not of direct interest in the application (as opposed to the latent variable Θ).

4.1 Introduction

In all methods and algorithms presented in this thesis, it is assumed that highly accurate camera parameters are known and are part of the observation. While in some highly controlled environments and in synthetic environments this level of accuracy is possible, in other situations it can be very expensive, time consuming and hard to achieve. The intrinsic parameters, like focal length and pixel centre, can be determined by standard camera calibration techniques. The extrinsic camera parameters, such as camera rotation and position relative to a real world co-ordinate system, can be harder to accurately determine. In general a checkerboard pattern is used to determine these parameters. All cameras in the one set up should be able to see the one calibration pattern, although multiple calibration patterns can be used if the translation between the patterns is known. However, with the pattern in awkward angles and mapping from pattern to pattern, a problem arises with a propagation of error, which can mean data can be out of alignment by up to 2 or 3cms. This means that fine detail can be impossible to recover for small objects.

In this instance the extrinsic camera parameters are the nuisance variable. The parameters are a necessary component for the statistical models to work, but beyond that there is no requirement to know them.

It should be noted that the objective here is to align all the cameras to one space and that there is no requirement to align to some particular real world origin. Therefore, one camera can be chosen as a reference camera and all other cameras can be aligned to that camera. Given that only rotation and translation of the cameras is being refined, this use of a reference camera should not affect the shape of the final reconstruction and should only affect the local origin depending on the error in that original reference camera.

The basic relationship between a point in space and a pixel coordinate for a pin-hole camera is the *projection matrix*, P. This is a 4×3 matrix which has 11 degrees of freedom. A very basic explanation of the projection matrix as it applies to this situation can be found in the Appendix C, Section C.1.2. (For a more in-depth explanation see [43]). The projection matrix has two basic types of parameters: extrinsic parameters and intrinsic parameters. In this work only the extrinsic parameters are being considered.

Both sets of parameters can be estimated with different degrees of accuracy with existing techniques. Intrinsic parameters can generally be estimated to a high degree of accuracy. Extrinsic parameters generally have a greater level of inaccuracy. This framework aims to take an initial estimate of the extrinsic parameters and refine them to produce highly accuracy parameters.

4.2 Framework

The extrinsic camera parameters being considered are the Roll ψ_1, Pitch ψ_2, and Yaw ψ_3 of the rotation matrix and the three components of the translation vector ψ_4, ψ_5, ψ_6 which go into the projection matrix. The intrinsic camera parameters (focal length in horizontal and vertical axis f_x, f_y and the coordinates of the centre pixel (u_0, v_0)) can be robustly estimated using standard techniques so these parameters are left alone and deemed to be correct and accurate. The main relationship is now extended to include these nuisance variables Ψ.

$$\boldsymbol{\lambda} + F(\mathbf{x}, \Theta, \Psi) = \boldsymbol{\varepsilon} \sim p_{\boldsymbol{\varepsilon}}(\boldsymbol{\varepsilon}) \tag{4.1}$$

The problem of aligning all the cameras is broken down into aligning just two cameras at a time. One camera is aligned with a reference camera and then the process is repeated, aligning just the next camera until all the cameras are aligned.

Here $\Psi = [\psi_1 \ \psi_2 \ \psi_3 \ \psi_4 \ \psi_5 \ \psi_6]^T$ is the set of extrinsic camera parameters for a camera. With the first camera these parameters Ψ are left unaltered and therefore the situation becomes identical to the premise in the previous chapter where the parameters can be considered part of the observation.

Therefore,

$$\boldsymbol{\lambda} + F(\mathbf{x}, \Theta, \Psi) = \boldsymbol{\varepsilon} \sim p_{\boldsymbol{\varepsilon}}(\boldsymbol{\varepsilon}) \tag{4.2}$$

is the same as:

$$\boldsymbol{\lambda} + F(\mathbf{x}, \Theta) = \boldsymbol{\varepsilon} \sim p_{\boldsymbol{\varepsilon}}(\boldsymbol{\varepsilon}) \tag{4.3}$$

From this setup with one camera, the 3D point cloud is computed and is then used as a sample of the likelihood $\overline{lik}(\Theta)$ (as defined in Equation 3.17) this is then used as the input to refine the parameters for the next camera. A sample likelihood can be seen in the top right image of Figure 4.1.

The second camera should have a good overlap with the first camera in terms of the scene they are viewing, therefore, the likelihood of camera 1, $\overline{lik}_1(\Theta)$, should be similar to the likelihood of camera 2, $\overline{lik}_2(\Theta)$.

With this in mind the following framework is derived in the same way as in the previous chapter but in this case the objective is to calculate the likelihood of a set of camera parameters Ψ.

$$\boldsymbol{\lambda} + F(\mathbf{x}, \Theta, \Psi) = \boldsymbol{\varepsilon} \sim p_{\boldsymbol{\varepsilon}}(\boldsymbol{\varepsilon}) \tag{4.4}$$

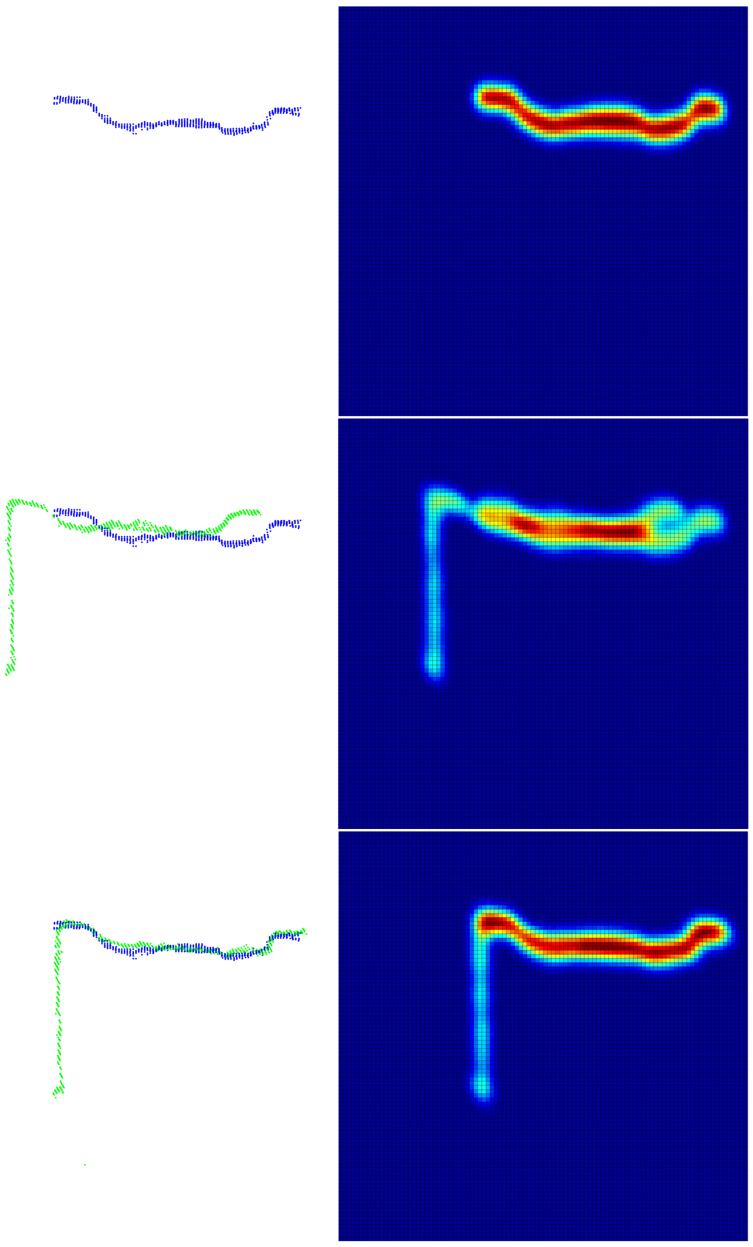


Figure 4.1: In the top pair of images the raw data (left) and equivalent likelihood (right) can be seen for a single camera. The middle pair of images shows the addition of a second camera which is not quite aligned properly to the initial data. The bottom pair of images shows the two cameras aligned after finding the best Ψ for the second camera using the Equation 4.18. In this case the points are sampled from the likelihood of the top right image and used to align the second camera. It can be seen that the result in the bottom image has a much better alignment in the raw data and a stronger likelihood in the bottom right image.

In the same way as before the probability density function of $\boldsymbol{\lambda}$ can be written as:

$$p_{\boldsymbol{\lambda}|\mathbf{x},\Theta,\Psi}(\boldsymbol{\lambda}|\mathbf{x},\Theta,\Psi) = p_{\boldsymbol{\varepsilon}}(\boldsymbol{\lambda} + F(\mathbf{x},\Theta,\Psi)) \tag{4.5}$$

Now the joint probability density function of $\boldsymbol{\lambda}$ and this time, Ψ, can be computed:

$$p_{\boldsymbol{\lambda},\mathbf{x},\Theta,\Psi}(\boldsymbol{\lambda},\mathbf{x},\Theta,\Psi) = p_{\boldsymbol{\lambda}|\mathbf{x},\Theta,\Psi}(\boldsymbol{\lambda}|\mathbf{x},\Theta,\Psi)\ p_{\mathbf{x},\Theta,\Psi}(\mathbf{x},\Theta,\Psi) \tag{4.6}$$

Substituting $p_{\boldsymbol{\varepsilon}}(\boldsymbol{\lambda} + F(\mathbf{x},\Theta,\Psi))$ in and assuming independence between $\mathbf{x}$, Θ and Ψ.

$$p_{\boldsymbol{\lambda},\mathbf{x},\Theta,\Psi}(\boldsymbol{\lambda},\mathbf{x},\Theta,\Psi) = p_{\boldsymbol{\varepsilon}}(\boldsymbol{\lambda} + F(\mathbf{x},\Theta,\Psi))\ p_{\mathbf{x}}(\mathbf{x})\ p_{\Theta}(\Theta)\ p_{\Psi}(\Psi) \tag{4.7}$$

$$p_{\boldsymbol{\lambda},\Psi}(\boldsymbol{\lambda},\Psi) = \iint p_{\boldsymbol{\varepsilon}}(\boldsymbol{\lambda} + F(\mathbf{x},\Theta,\Psi))\ p_{\mathbf{x}}(\mathbf{x})\ p_{\Theta}(\Theta)\ p_{\Psi}(\Psi)\ d\mathbf{x}\ d\Theta \tag{4.8}$$

Solving the integration gives the expectation:

$$p_{\boldsymbol{\lambda},\Psi}(\boldsymbol{\lambda},\Psi) = p_{\Psi}(\Psi)\mathbb{E}_{\Theta}[\mathbb{E}_{\mathbf{x}}[\ p_{\boldsymbol{\varepsilon}}(\boldsymbol{\lambda} + F(\mathbf{x},\Theta,\Psi))]] \tag{4.9}$$

The joint probability density function of $p_{\boldsymbol{\lambda},\Psi}(\boldsymbol{\lambda},\Psi)$ can be approximated using both the observations $\left\{\mathbf{x}^{(i)}\right\}_{i=1\ldots N}$and the points of interest $\left\{\Theta^{(j)}\right\}_{j=1\ldots M}$ sampled from the $\overline{lik}_1(\Theta)$ from the reference camera.

$$\hat{p}_{\boldsymbol{\lambda},\Psi}(\boldsymbol{\lambda},\Psi) = \underbrace{p_{\Psi}(\Psi)}_{\text{prior}} \times \underbrace{\frac{1}{M.N}\sum_{j=1}^{M}\sum_{i=1}^{N} p_{\boldsymbol{\varepsilon}}(\boldsymbol{\lambda} + F(\mathbf{x}^{(i)},\Theta^{(j)},\Psi))}_{\text{Averaged likelihood}} \tag{4.10}$$

The case of interest is when $\boldsymbol{\lambda} = 0$ therefore:

$$\hat{p}_{\boldsymbol{\lambda},\Psi}(\boldsymbol{\lambda} = 0,\Psi) = p_{\Psi}(\Psi) \times \frac{1}{M.N}\sum_{j=1}^{M}\sum_{i=1}^{N} p_{\boldsymbol{\varepsilon}}(F(\mathbf{x}^{(i)},\Theta^{(j)},\Psi)) \tag{4.11}$$

$$\overline{lik}(\Psi) = \frac{1}{M.N}\sum_{j=1}^{M}\sum_{i=1}^{N} p_{\boldsymbol{\varepsilon}}(F(\mathbf{x}^{(i)},\Theta^{(j)},\Psi)) \tag{4.12}$$

Now this likelihood can be used to find the best camera parameters for the next camera.

$$\arg\max_{\Psi} \ \overline{lik}(\Psi) \tag{4.13}$$

4.3 Application Detail

The conversion from Roll, Pitch and Yaw to a rotation matrix can be achieved in many ways depending on how they are calculated. The following is the convention used in this work.

$$R(\Psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi_1) & \sin(\psi_1) \\ 0 & -\sin(\psi_1) & \cos(\psi_1) \end{bmatrix} \begin{bmatrix} \cos(\psi_2) & 0 & -\sin(\psi_2) \\ 0 & 1 & 0 \\ \sin(\psi_2) & 0 & \cos(\psi_2) \end{bmatrix} \begin{bmatrix} \cos(\psi_3) & \sin(\psi_3) & 0 \\ -\sin(\psi_3) & \cos(\psi_3) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

The intrinsic and extrinsic camera parameters can be combined to create the projection matrix P as seen in the following equation:

$$P(\Psi) = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} & \psi_4 \\ R(\Psi) & \psi_5 \\ & \psi_6 \end{array} \right] \tag{4.15}$$

The rotation matrix and the translation vector can be used to determine the coordinate of the centre of the camera using the following equation:

$$C(\Psi) = -\begin{bmatrix} \\ R(\Psi) \\ \\ \end{bmatrix}' \begin{bmatrix} \psi_4 \\ \psi_5 \\ \psi_6 \end{bmatrix} \tag{4.16}$$

Using the functions $P(\Psi)$ and $C(\Psi)$ the formulation from the previous chapter can be rewritten in terms of the nuisance variable. Where the projection matrix and the camera centre were previously observations $x_4 - x_{18}$, they are now written in terms of the nuisance variable. The subscript for these parameters is now used as an index into the relevant matrix ($P(\Psi)_{ij}$ refers to the element at ith row and jth column of the projection matrix determined using parameters Ψ).

$$\underbrace{\begin{pmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{pmatrix}}_{\boldsymbol{\lambda}} + \underbrace{\begin{pmatrix} x_1 - \frac{\theta_1 P(\Psi)_{11}+\theta_2 P(\Psi)_{12}+\theta_3 P(\Psi)_{13}+P(\Psi)_{14}}{\theta_1 P(\Psi)_{31}+\theta_2 P(\Psi)_{32}+\theta_3 P(\Psi)_{33}+P(\Psi)_{34}} \\ x_2 - \frac{\theta_1 P(\Psi)_{21}+\theta_2 P(\Psi)_{22}+\theta_3 P(\Psi)_{23}+P(\Psi)_{24}}{\theta_1 P(\Psi)_{31}+\theta_2 P(\Psi)_{32}+\theta_3 P(\Psi)_{33}+P(\Psi)_{34}} \\ x_3 - \sqrt{(C(\Psi)_1-\theta_1)^2+(C(\Psi)_2-\theta_2)^2+(C(\Psi)_3-\theta_3)^2} \end{pmatrix}}_{F(\mathbf{x},\Psi,\Theta)=(F_1(\mathbf{x},\Psi,\Theta),F_2(\mathbf{x},\Psi,\Theta),F_3(\mathbf{x},\Psi,\Theta))^T} = \underbrace{\begin{pmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_3\end{pmatrix}}_{\boldsymbol{\varepsilon}} \tag{4.17}$$

In summary:

- $\boldsymbol{\lambda} \in \mathbb{R}^3$ - Auxiliary Variable
- $\mathbf{x} \in \mathbb{R}^3$ -

(x_1, x_2)	Pixel Coordinate in image space
x_3	Depth value observed

- $\Theta \in \mathbb{R}^3$ - Real world 3D coordinates.
- $\boldsymbol{\varepsilon} \in \mathbb{R}^3$ -

$\varepsilon_1 \sim \mathcal{N}(0, h_1)$	Uncertainty about pixel positions
$\varepsilon_2 \sim \mathcal{N}(0, h_2)$	
$\varepsilon_3 \sim \mathcal{N}(0, h_3)$	Uncertainty about depth data

- $\Psi \in \mathbb{R}^6$ -

ψ_1	Roll component of rotation matrix
ψ_2	Pitch component of rotation matrix
ψ_3	Yaw component of rotation matrix
(ψ_4, ψ_5, ψ_6)	3D translation vector

- $P(\Psi) : \mathbb{R}^6 \to \mathbb{R}^{3\times 4}$ - Function defined in Equation 4.15, which converts the camera parameters into the projection matrix.
- $C(\Psi) : \mathbb{R}^6 \to \mathbb{R}^3$ - Function defined in Equation 4.16, which converts the camera parameters into the coordinates of the camera centre.
- $F(\mathbf{x}, \Psi, \Theta) : \mathbb{R}^3 \times \mathbb{R}^6 \times \mathbb{R}^3 \to \mathbb{R}^3$ - link function between $\mathbf{x}$, Θ and Ψ.

Therefore, the averaged likelihood using the framework developed can then be seen as:

$$\overline{lik}(\Psi) = \frac{1}{N\prod_{k=1}^{3}\left(\sqrt{2\pi}h_k\right)} \sum_{j=1}^{M}\sum_{i=1}^{N}\prod_{k=1}^{3} \exp\left(\frac{-\left(F_k(\mathbf{x}^{(i)}, \Psi, \Theta^{(j)})\right)^2}{2h_k^2}\right) \tag{4.18}$$

This process can lead to a propagation of error around the cameras. This error can be reduced by having a good initial estimation of the camera parameters by using standard camera calibration techniques outlined in the Appendix C, Section C.1.4. The error can be further reduced by repeating the process with a different reference camera and by calibrating the cameras in reverse order.

The camera calibration method works well on irregular surfaces with plenty of curves and corners. On large flat plain surfaces the method may be less accurate. A camera with a view of just a flat plane section has trouble uniquely matching it to another flat plain section. This is because two flat plain sections will give a good match as long as they are near each other in the same plane, so quite often they will be offset from each other in the plane. A sample alignment can be seen in Figure 4.1. Figure 4.2 shows the end alignment of a set of cameras. The red cameras give the original position and orientation of the cameras. The blue cameras show the end aligned position and orientation.

4.4 Conclusion

In this chapter a new method for aligning camera views has been presented. The method uses a framework which is an extension of the framework presented in the previous chapter. The method has been used to align all the real world images taken with the Kinect camera across multiple objects without which the reconstructions could not be generated. The reconstruction following this camera alignment process can be found in Appendix B. In the next chapter a more in-depth explanation of how the synthetic data was generated, how the Kinect data was captured and a detailed comparison between all the methods and parameters chosen is given.

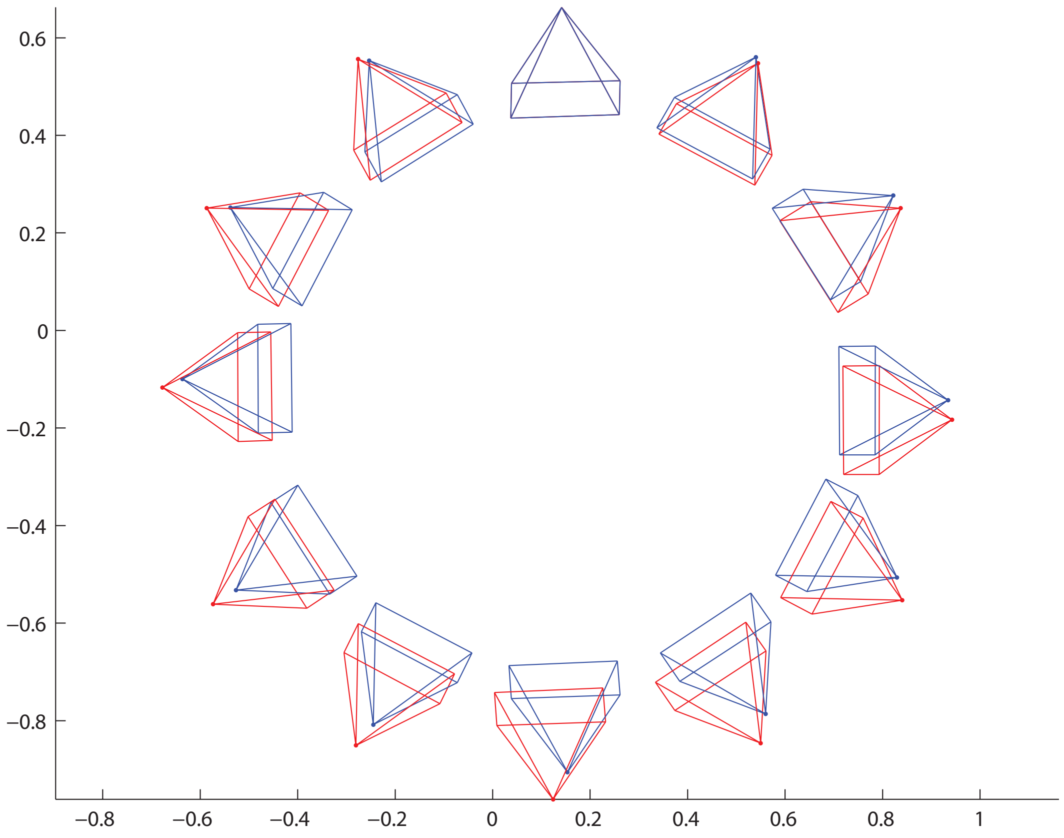


Figure 4.2: Example camera alignment. The red cameras show the original position and orientation of the cameras. The blue cameras show the position and orientation after performing the camera alignment process.

Chapter 5

Experimental Details and Result Comparisons

In this chapter the applications as explained in the previous two chapters are used to generate more reconstructions. Initially additional synthetic objects are reconstructed so that the results can be compared to their ground truths and full quantitative evaluations can be performed. To show that these methods work in real-world environments and can work on real objects they are applied to a new data-set which was captured in-house using the Microsoft Kinect camera. The methods are also applied to the Middlebury multi-view reconstruction data-set [90], which is a 3rd party data-set which allows some of the methods to be compared to other state of the art algorithms.

Section 5.1 explains the methods used to create and capture the synthetic and real data-sets used in this thesis and also gives a quick overview of the Middlebury data-set. Section 5.2 shows example reconstructions from each of these data-sets. A full set of reconstructions can be found in Appendix B. The section continues with examples of different scenarios and shows what effects different parameters have on the end reconstructions and highlights any problems that can arise and how they can be resolved or avoided.

5.1 Data Acquisition

In this work, three data-sets of objects are used, synthetic, real and Middlebury.

Synthetic objects are objects which have been computer generated using software (Autodesk 3DS Max) and so provide a synthetic data-set. Synthetic objects are used because they are the easiest to obtain, modify and evaluate. They are generated from a 3D Mesh and that same 3D Mesh can be used to exactly evaluate the results of the reconstruction.

Real objects are ones which actually exist and of which images have been taken during this work. Real objects are used to show that the methods developed can also work in real life with real noise, under real situations.

Middlebury objects are a set of two real objects which the vision group in Middlebury College have captured and provided as a benchmark and evaluation data-set for multi-view reconstruction algorithms. The Middlebury objects are used so that the results can be compared to other state of the art methods for multi-view reconstruction.

In this work, two different types of data are required for the different applications, specifically silhouette data and depth data. Silhouette data is a set of images where background subtraction has been performed. This results in a set of images which have white for the object of interest and black everywhere else. Depth data is a set of images which provide a per pixel distance from the camera centre to the real world point in space represented by that pixel. Colour images are also useful to have as they can be used to generate the silhouette data and can also be used to texture the object once the reconstruction is completed. The following sections describe how the data was generated or captured for each of the data-sets and each of the data types.

5.1.1 Synthetic Object Acquisition

To generate the data for the synthetic objects a software program called Autodesk 3DS Max as seen in Figure 5.1 was used. A 3D mesh of the object required is imported into the software, where it can be lit, coloured, orientated, scaled and moved to wherever it is required. A virtual camera is then created. The focal length, position and orientation of the camera can be controlled, adjusted and set. In general the camera is rotated around the object to give a 360 degree view of the object. The camera can then be set to save these views, as either perfect silhouette images, colour images or depth buffer images. The result is camera calibration data and a set of silhouette and depth images for generating the reconstruction and a ground truth 3D mesh for evaluating the result.

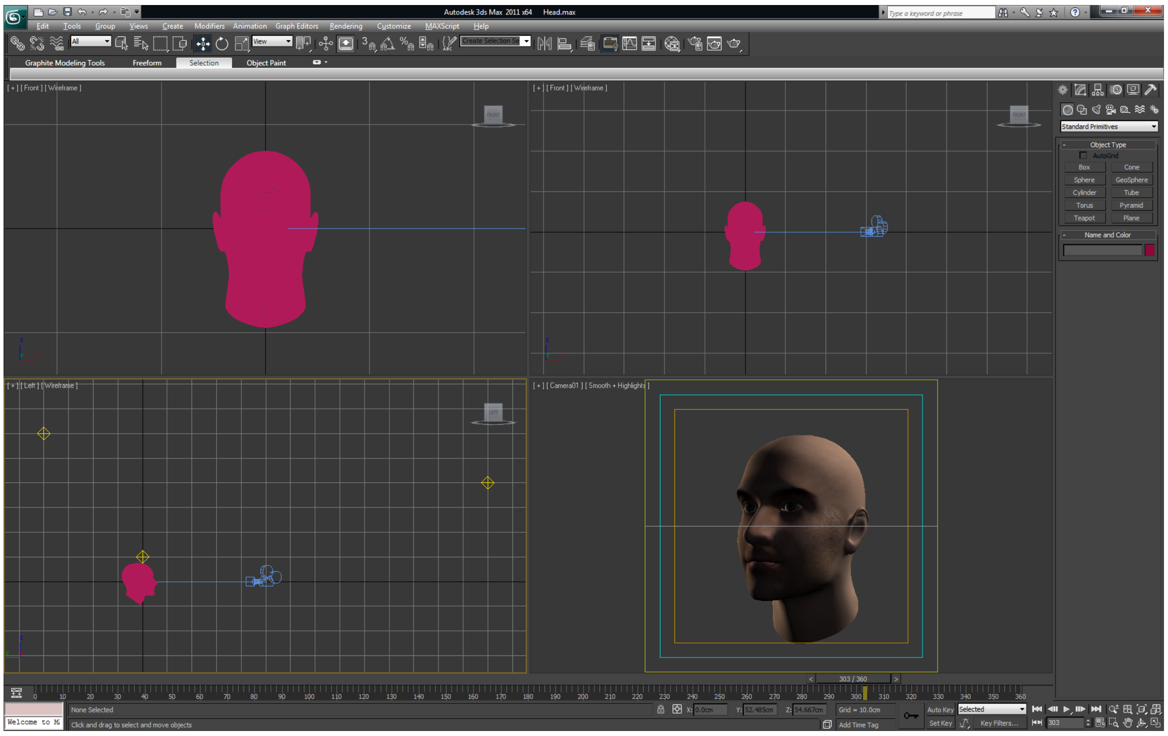

Autodesk 3ds Max 2011 x64 Head.max
Edit Tools Group Views Create Modifiers Animation Graph Editors Rendering Customize MAXScript Help
Graphite Modeling Tools Freeform Selection Object Paint
[+] [Front] [Wireframe]
[+] [Left] [Wireframe]
[+] [Camera01] [Smooth + Highlights]
Standard Primitives
Object Type
AutoGrid
Box
Cone
Sphere
GeoSphere
Cylinder
Tube
Torus
Pyramid
Teapot
Plane
Name and Color
303 / 360
None Selected
Click and drag to select and move objects
Grid = 10.0cm
Add Time Tag
Auto Key
Selected
Set Key
Key Filters...

Figure 5.1: A screen shot of the software program Autodesk 3DS Max which was used to generate the synthetic data.

Five synthetic objects have been used in this thesis and can be seen in Figure 5.2. They include the Stanford Bunny which was used as an example object in the previous chapters. The other objects used include a 3D mesh model of the Stanford Dragon, the Utah Teapot, a generic head and a generic human girl. The range of objects is designed to test a range of scenarios from the simple object of the teapot to the more complex object of the Dragon. The use of the head and human girl objects are designed to test how the algorithms perform in more common practical scenarios where human pose tracking and face recognition may be of interest.

5.1.2 Real Object Acquisition

For the real objects, a single Microsoft Kinect Camera was used to capture both colour images and depth data of the object.

In practice it was found to be easier to rotate the object and leave the camera stationary than to rotate the camera around the object. This method of rotating the object meant that the images were more consistent in terms of:

- position of the object within the image
- distance of the object from the camera
- measurement of rotation between consecutive images

And it also required less space than rotating the camera around the object.

As seen in Figure 5.3 a board is placed on top of a large sheet of paper, which is marked every 10 degrees around in a full circle. The board and the marked sheet of paper allow the board to be placed in the 36 different orientations around a full circle. Initially a set of images is taken with the board in all 36 positions with no object included, and only a checkerboard pattern on top of the board. This allows all 36 positions to be calibrated, and an initial estimate of the camera parameters to be obtained.

In turn an object is placed on the board and the board is placed in all 36 positions, every 10 degrees on a full horizontal circle, with a capture taken at each position. A selection of objects was used from highly complex to relatively plain to test the algorithms in different ways. The selection of 6 objects can be seen in Figure 5.4

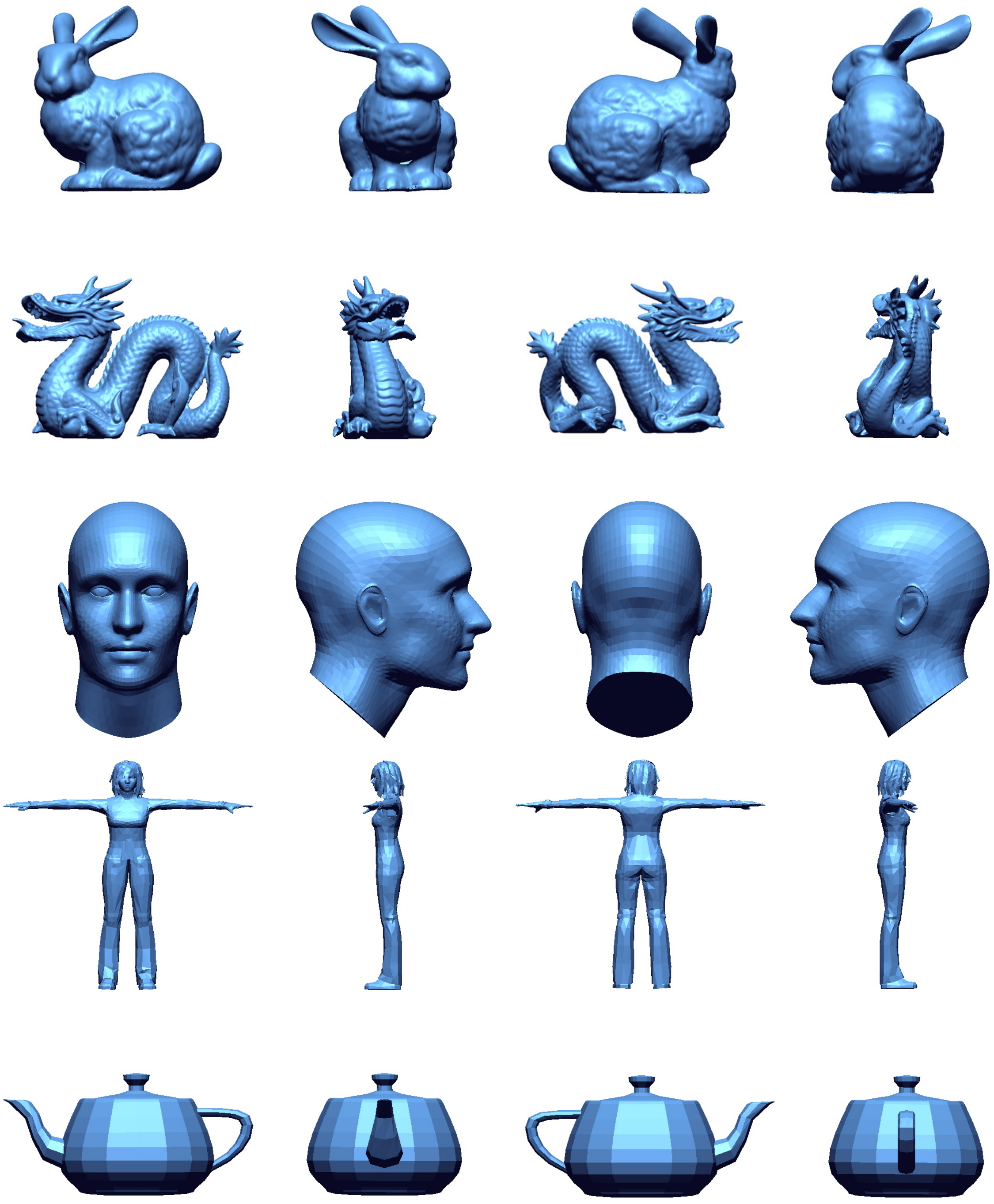


Figure 5.2: The five synthetic objects.

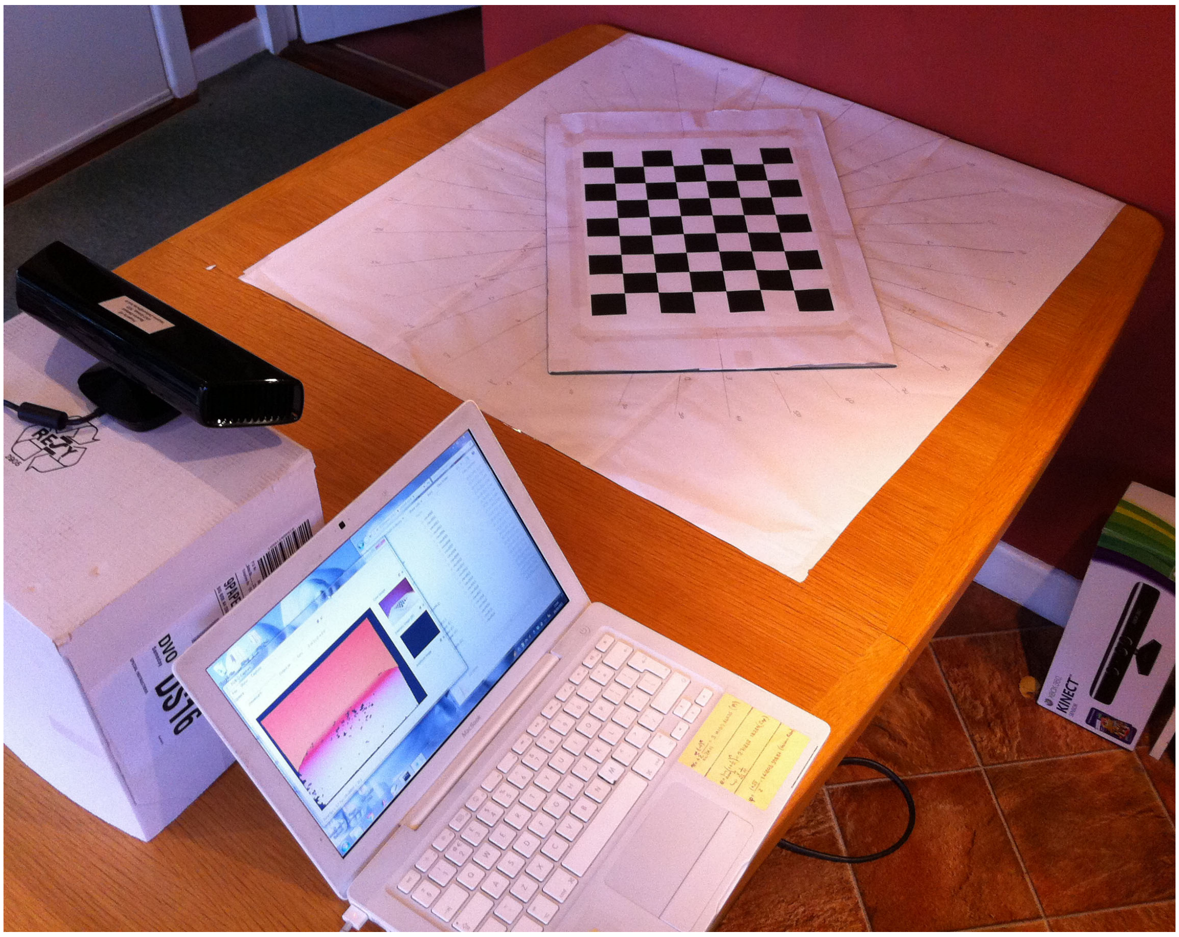


Figure 5.3: A photograph showing an example setup using the Kinect Camera and rotating platform used to generate the real object data-set.

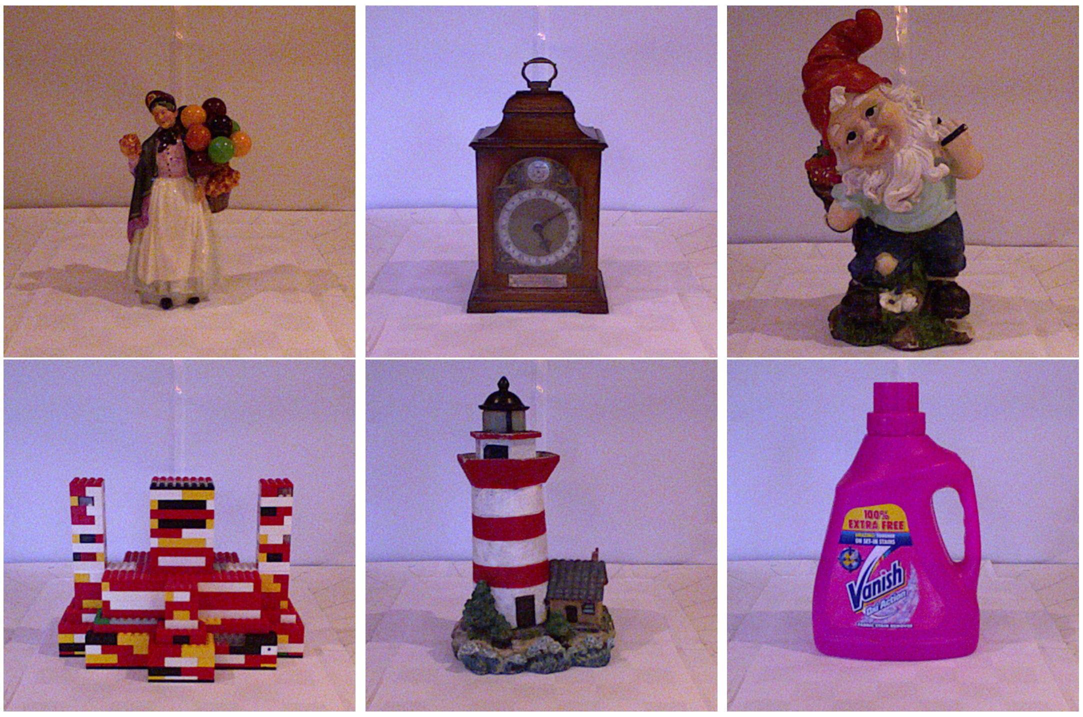


Figure 5.4: The six real objects acquired using the Kinect Camera.

The Kinect camera outputs a colour image and a depth image. A silhouette image can be generated using standard background subtraction techniques to segment the foreground and background pixels. The silhouette images were improved using the depth data and when necessary any obvious errors in the segmentation were corrected by hand. The Kinect camera which uses a technique called structured light generates the depth data. A full description of the Kinect camera can be found in Appendix A

No extraordinary measures were taken to get perfect images; one of the goals of this work was to have a method that could be used by anyone without having to go to extraordinary lengths to either set up an environment or have very specialized equipment. The Kinect camera is the only piece of equipment needed. The rest of the setup can be done by anyone on their kitchen table. No special lights, background or rig was used and the calibration step is only designed to give a good first estimate of the initial parameters as it will later be optimized by the algorithm in Chapter 4.

Depth Bandwidth

As highlighted in Chapter 3, the bandwidth is a very important factor for the reconstruction method. The goal is to set the minimum bandwidth as small as possible to preserve as much detail from the raw data, but it still needs to be larger then the noise within the data. Within the image plane the raw data has a maximum accuracy of 1 pixel, so this links directly to the bandwidth for raw data from the image plane. To determine the depth data bandwidth requires measuring the accuracy of the raw data. To achieve this the Kinect camera was pointed at a flat surface (a wall) and repeated depth images were captured from different distances. A calibration pattern was also placed on the wall so that the distance of the camera from the wall could be determined using standard extrinsic camera calibration. Figure 5.5 shows the relationship between the raw data captured by the Kinect camera and the standard deviation within each capture. Different conversions are available to convert the observed Kinect depth value to real distance values; these are also included in the graph. Polynomial fitting is used to give a best fit from the observed Kinect depth data to measured distance. For all the experiments carried out in this work the object was placed between 0.6 and 0.8 meters away from the camera. For this section of the four degree polynomial curve there is a near linear conversion. The standard deviation along this section relates to the noise of each depth data capture and results in a depth bandwidth of 0.0002m.

If different sensors are used, the different noise characteristics would need to be measured. Noisier sensors will need larger bandwidths which will result in greater smoothing being applied to the raw data. This will reduce the impact of the sensor. More accurate sensors will have smaller bandwidths and will therefore have a greater impact at creating reconstructions with greater accuracy.

5.1.3 Middlebury Objects

The computer vision group in the University of Middlebury provide a multi view data-set of two objects; a Dino and a Temple as seen in Figure 5.6. The data-set consists of three setups, a full hemisphere of cameras, a dense ring of cameras and a sparse ring of cameras. As the data-sets for the synthetic and real objects were images from a dense ring of cameras, the similar data-set will be used for the Middlebury Objects. All camera parameters are provided and instruction for extracting silhouette images are also

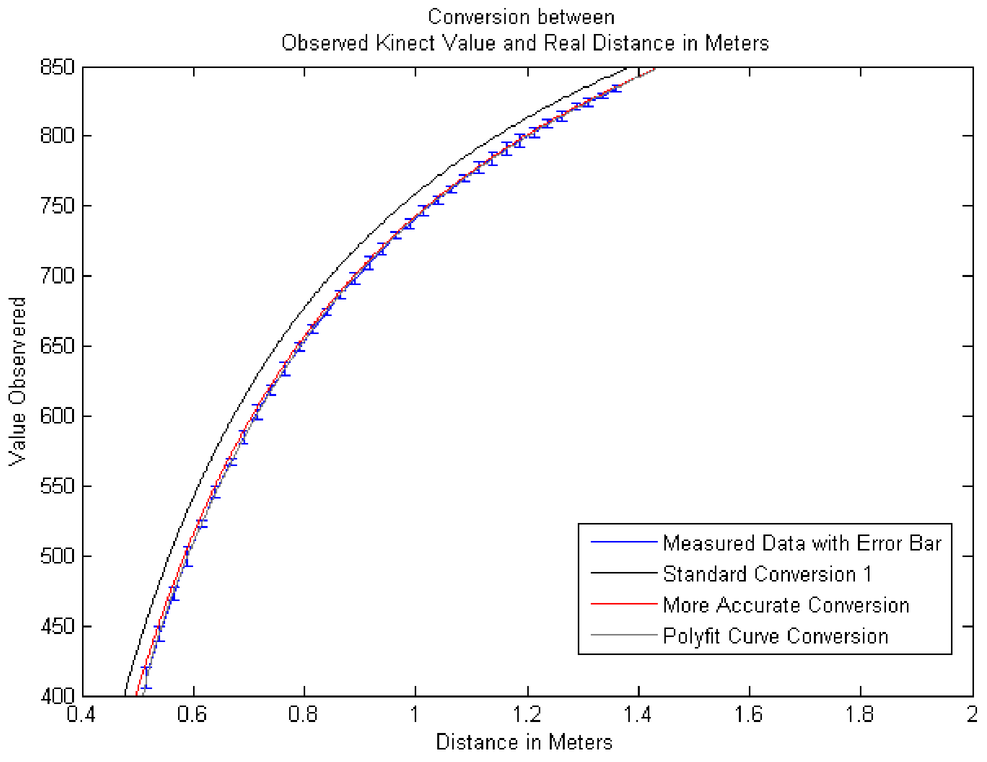


Figure 5.5: Graph showing the relationship between the raw depth data and the real world depth units. All the experiments carried out in this work have the object placed between 0.6 and 0.8 meters away from the camera. For this section of the four degree polynomial curve there is a near linear conversion. The consistent standard deviation along this section allows the depth bandwidth to be calculated. This works out at 0.0002m.

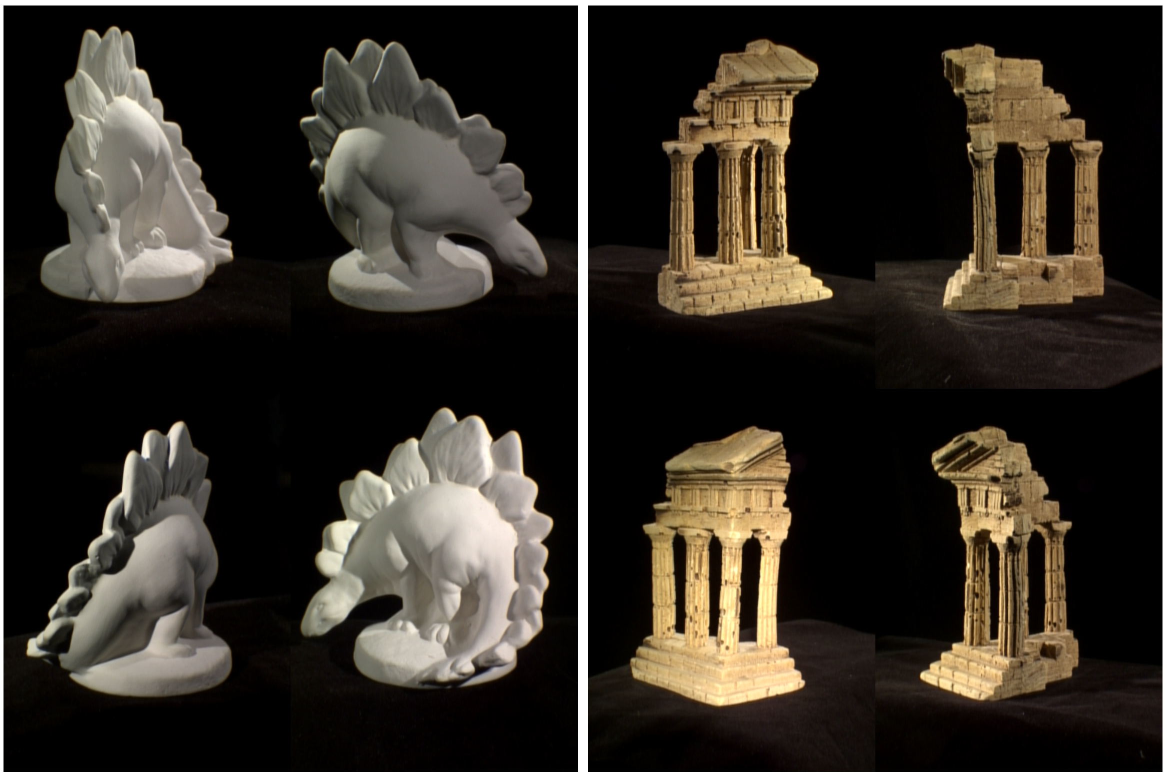


Figure 5.6: The two Middlebury objects, the Dino on the left and the Temple on the right.

included. No depth data is available for the Middlebury project so only the silhouette version is applied when generating the reconstructions. Most of the methods on the Middlebury evaluation web-site use colour information to generate their reconstructions. As this can be more informative than only silhouette information, the results obtained in this work can not be fully compared. More information on the Middlebury data-set and alternative data-sets can be found in Appendix D. More information on the sizes of the silhouette images and where applicable the depth images and number of cameras can be found in Table 5.1. Table 5.2 gives the rough size of each of the objects used across the three data-sets.

	Silhouette Image	Depth Image	Number of Images
Synthetic data-set	512×512	512×512	36, 12, 9, 6, 3
Real data-set	1280×1024	640×480	36, 12, 9, 6, 3
Middlebury data-set	480×640	n/a	48/47, 12, 10, 9, 6, 3

Table 5.1: Comparison of image size and number of images for each of the data-sets used

Object	Object Size (cm)
Bunny	$10 \times 13 \times 13$
Dragon	$7 \times 15 \times 10$
Head	$15 \times 10 \times 17$
Human	$12 \times 3 \times 13$
Teapot	$10 \times 16 \times 8$
Balloon Lady	$13 \times 9 \times 22$
Clock	$15 \times 11 \times 24$
Gnome	$18 \times 15 \times 32$
Lego	$26 \times 18 \times 18$
Lighthouse	$18 \times 16 \times 26$
Vanish	$18 \times 10 \times 27$
Dino	$8 \times 9 \times 8$
Temple	$11 \times 16 \times 8$

Table 5.2: Comparison of the object sizes across each of the data-sets used

5.1.4 Implementation

All the applications have been programmed in MATLAB. This was chosen as it provides the best platform for developing and testing the methods. All programs run on a single thread on an Intel (R) Core (TM) 2 Quad CPU Q9450 processor @ 2.66GHz running Windows 7.

It should be noted that a C++ multi-threaded version of these programs could be written and would result in considerably faster computation times, potentially near 20-30 times or even over a 100 times improvement on a GPU processor. However, as it was not the object of this thesis to explore programming on different architectures, these alternatives were not explored. Therefore, while the reconstruction times displayed in the following sections may appear high, it is theoretically possible to greatly reduce them to a much faster speed.

5.2 Results

Figures 5.7, 5.8 and 5.9 show example reconstructions of all the synthetic, real and Middlebury objects used in this work. The synthetic and real objects are reconstructed from silhouette information using regular sampling, from silhouette information using Newton's method and from depth information using regular sampling and surface explore. As no depth information is available for the Middlebury objects only the silhouette methods are used to reconstruct these objects. A full set of reconstructions can be found in Appendix B.

5.2.1 Evaluation Methodology

To assess the quality of a reconstruction, it is compared to a ground truth, when available. One main metric used to quantify this quality is called *Accuracy*. For every point on the reconstructed mesh the shortest distance from that point to the surface of the ground truth is calculated. This distance is computed by finding the closest point on each triangle of the ground truth mesh to the point being considered, then the shortest distance is taken as the accuracy of that point, see Figure 5.10. This is repeated for every point in the reconstruction. The point is given as a positive or negative accuracy depending on whether it is inside the ground truth or outside the

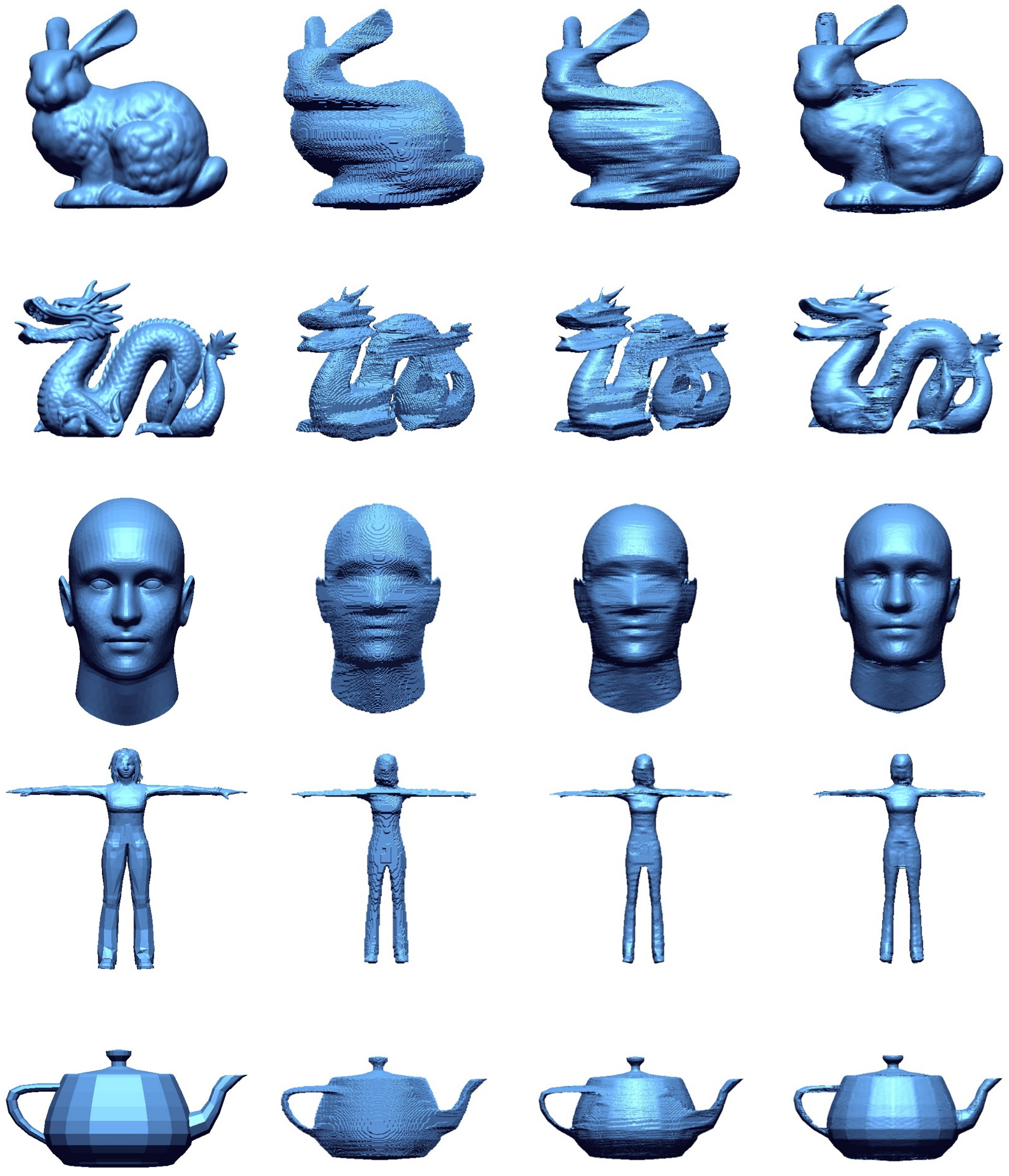


Figure 5.7: Example reconstructions of the synthetic objects all using 36 camera views. The columns from left to right are the original object, the reconstruction from silhouette data using regular sampling, the reconstruction from silhouette data using Newton's method and the reconstruction from depth data using Newton's method and surface explore.

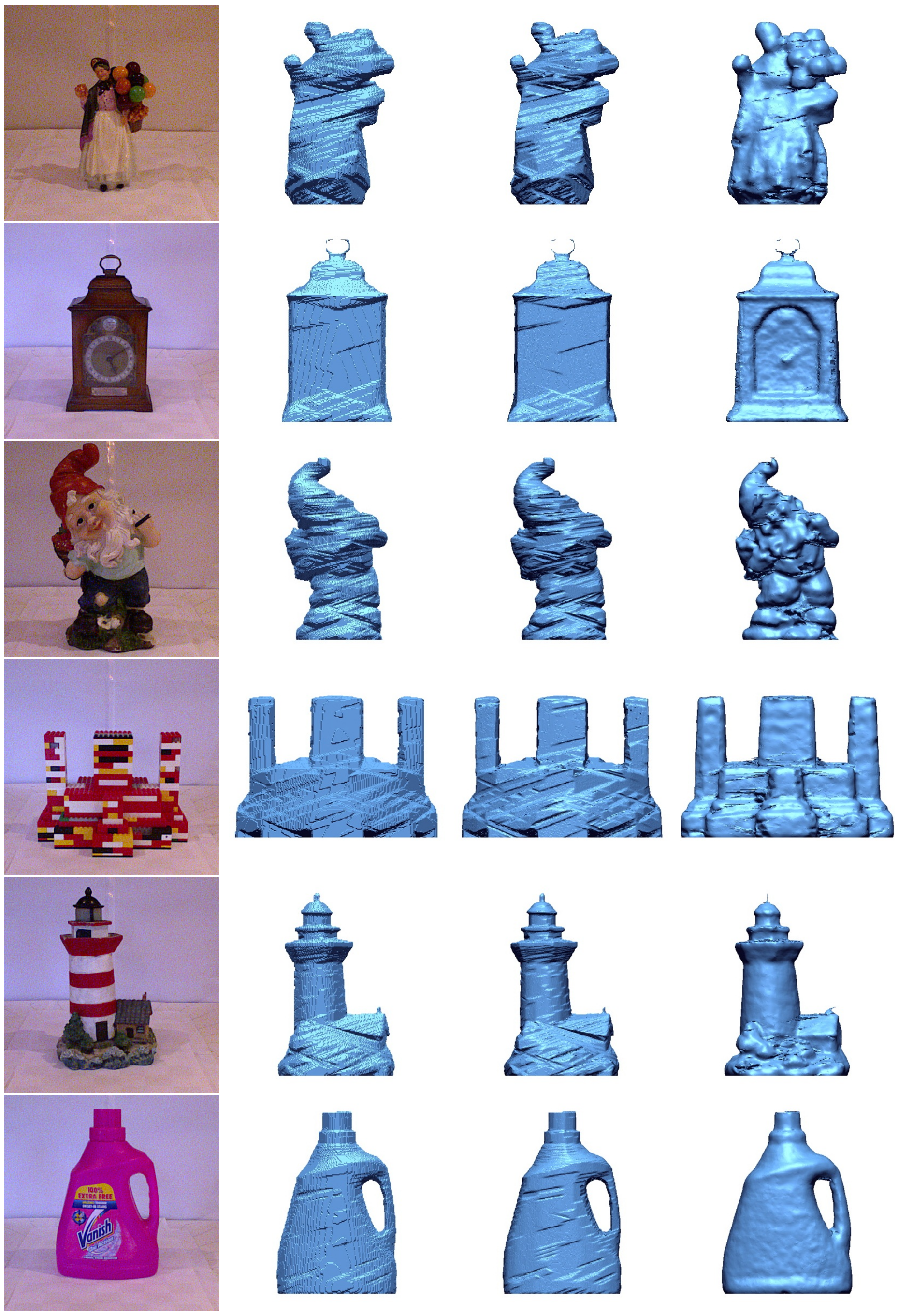


Figure 5.8: Example reconstructions of the real objects all using 36 camera views. The columns from left to right are the original object, the reconstruction from silhouette data using regular sampling, the reconstruction from silhouette data using Newton's method and the reconstruction from depth data using Newton's method and surface explore.

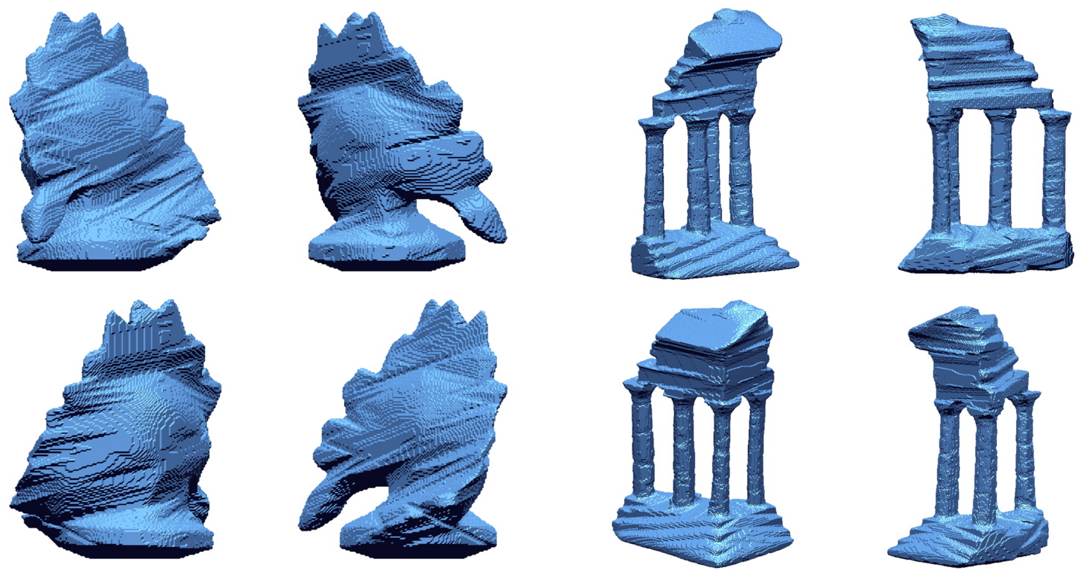


Figure 5.9: Results for Middlebury objects using regular sampling method. These reconstructions use only silhouette information as no depth information is available for the Middlebury objects.

ground truth. This therefore allows analysis on whether under-fitting or over-fitting is taking place. Absolute accuracy is used when plotting the results on the model and the mean value is quoted as the accuracy of the object; the smaller the accuracy value, the better the reconstruction. The other method used to assess the quality of the reconstruction is a visual examination and inspection of the end result.

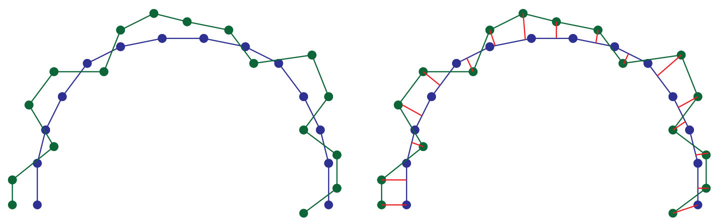


Figure 5.10: The accuracy of the reconstruction (green) relative to the ground truth model (blue) is the shortest distance from each point on the reconstruction to the ground truth as shown with the red lines.

	Accuracy	Completeness
Silhouette Version	2.25 mm	75.5 %
Standard Visual Hull	2.41 mm	77.0 %
Current Best Method on Middlebury [34]	0.28 mm	99.8 %
Current Worst Method on Middlebury [54]	2.81 mm	86.0 %

Table 5.3: Comparison of Middlebury results with other methods. These results were published in [86].

5.2.2 Middlebury Comparison

The Dino object from the Middlebury web-site was reconstructed using the silhouette version of the framework and the result was submitted to Middlebury for evaluation. The results of this evaluation along with that of the standard visual hull (as presented in Section 2.1.1 of the state of the art) and the current best and worst results on the web-site can be seen in Table 5.3. As most of the reconstruction methods being compared on the Middlebury web-site use colour information, and the reconstruction submitted as part of this work uses only silhouette information the results are not fully comparable, as colour information can be much more informative than silhouette information. The results are better then the standard visual hull results and some of the other methods presented on the web-site.

Middlebury use two metrics to measure the quality of the reconstruction. The first one called accuracy is similar to the metrics in this work and is calculated in a similar way. The figure quoted in this work is a mean value of all the distances calculated between the reconstruction and the ground truth. The value used on the Middlebury web-site is based on what distance 90 % of distances encompasses i.e. 90 % of the distances calculated will be under that value. The other metric used is the reverse, called completeness, where the distance from the ground truth to the reconstruction is calculated and the percentage within 1.25 mm is given.

5.2.3 Effects of Regular Sampling Resolution

When generating a result from regular sampling one of the main concerns is the size/resolution of the sampling. In this work 3 different resolutions are used: 50, 100 and 200. This is where the space of interest is split up into $50 \times 50 \times 50$, $100 \times$

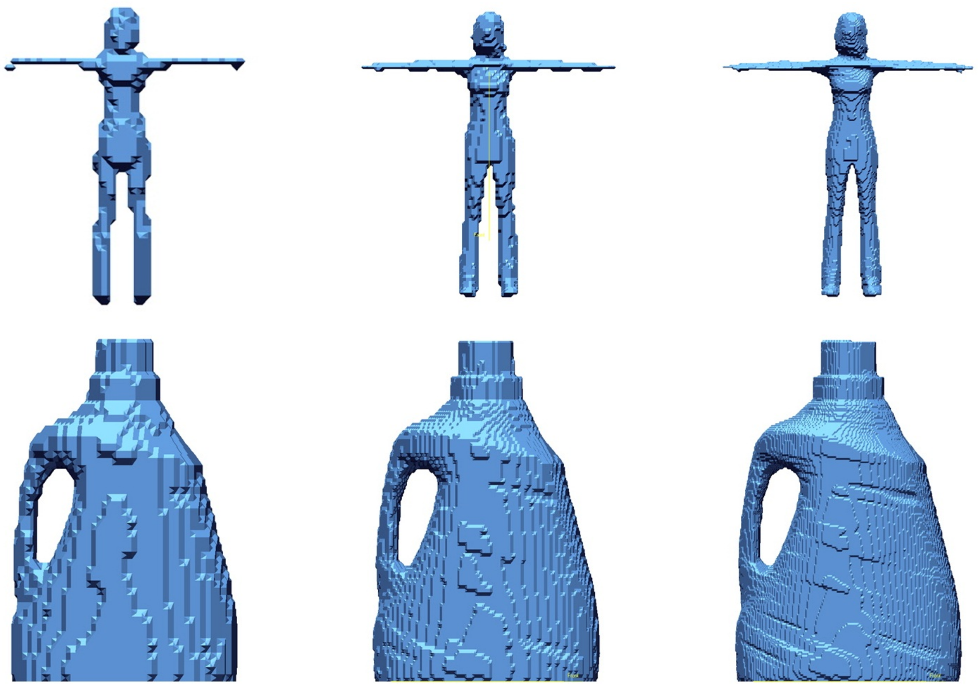


Figure 5.11: Going from left to right the resolution of the regular sampling is double each time, 50, 100 and 200

100 × 100 or 200 × 200 × 200 regularly sampled points.

In the following examples (Figure 5.11) the regular sampling is very noticeable in the blocky nature of the reconstructions. This blocky nature is reduced, but is still visible as the resolution is increased.

As can be seen in the graph in Figure 5.12 this doubling of the resolution results in a roughly 8 times increase in the computation time required.

Only a small improvement in the accuracy of the reconstruction can be seen as the

Resolution	No. of Points Sampled	Vertices	Mean Accuracy (mm)
50	125,000	7,458	1.310
100	1,000,000	30,872	1.245
200	8,000,000	125,248	1.230

Table 5.4: Comparison of accuracy and number of vertices of end reconstruction with different resolutions.

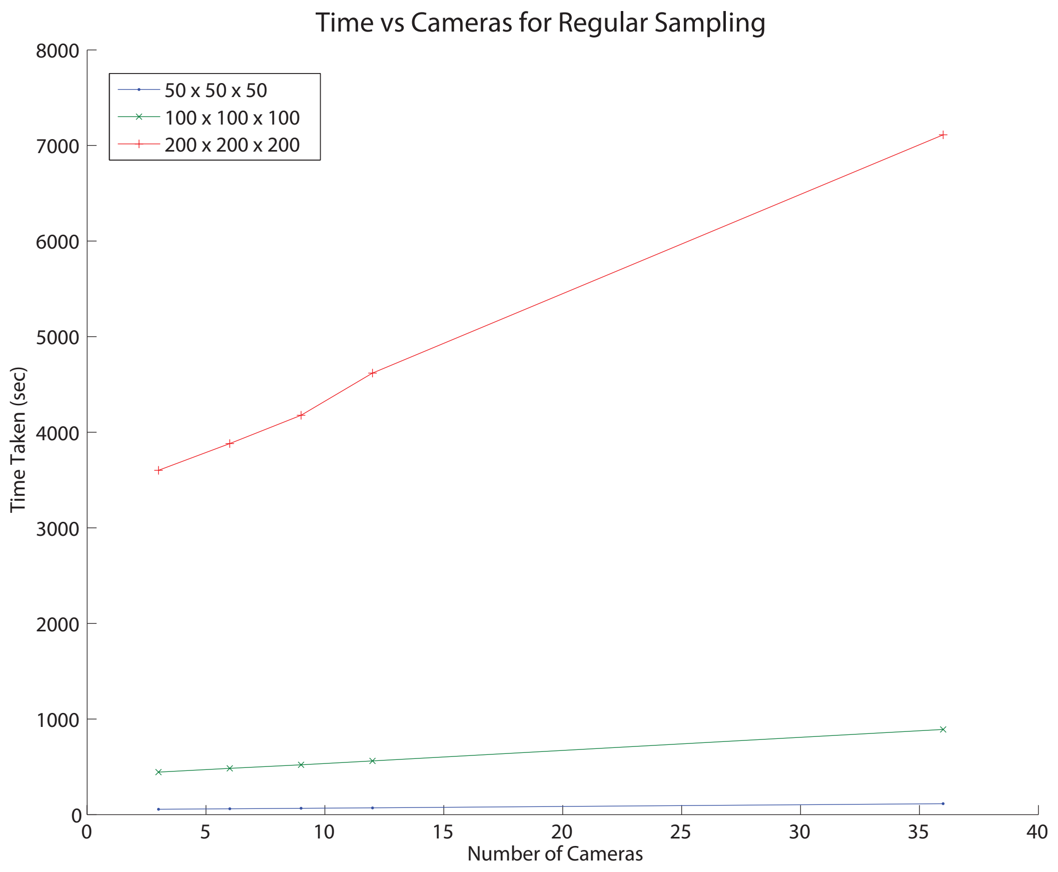


Figure 5.12: Comparison of different regular sampling resolutions and camera numbers on the time taken to complete full 3D reconstruction.

No. of Vertices	Mean Accuracy (mm)
7,458	1.401
30,872	1.371
125,248	1.373

Table 5.5: Comparison of accuracy and number of vertices of end reconstruction with numbers of reconstructed points using the Newton's Method approach.

resolution is increased, as shown in Table 5.4. The difference is very slight as the major error in accuracy is due to concavities which are not reconstructed. With the increased resolution the number of points sampled is exactly 8 times larger which means the memory requirement is also 8 times larger to store the value of each point sampled. (The number of vertices in the end reconstruction has a roughly 4 times increase each time the resolution is doubled. This is understandable as the number of sample points within the object will have roughly an 8 times increase and therefore the corresponding surface area of the object will have a 4 times increase).

5.2.4 Improvement from Regular Sampling to Newton's Method

A noticeable improvement in the visual quality of the reconstruction can be seen between the regular sampling approach and the Newton's Method approach.

First the blocky nature of the reconstruction has been smoothed out to produce a better looking result as seen in Figure 5.13.

To ensure fair comparison, the number of starting points used in the Newton's method approach is made the same as the number of vertices in the end reconstruction from the regular sampling approach. Again the difference between the accuracy of the two methods is over-shadowed by the error produced by the concavities but overall the accuracy is very similar and can be seen in Table 5.5 as compared to Table 5.4.

Computation time is nearly linear with the number of points and is a fraction of the computation time, as compared to the regular sampling approach, as seen in the graph in Figure 5.14. There is now a direct link between the number of points and the computation time, unlike the regular sampling computation time which was based solely on the resolution of the sampling which was completely unrelated to the end number of vertices of a reconstruction.

Memory requirement is considerably less as only the points themselves need to be

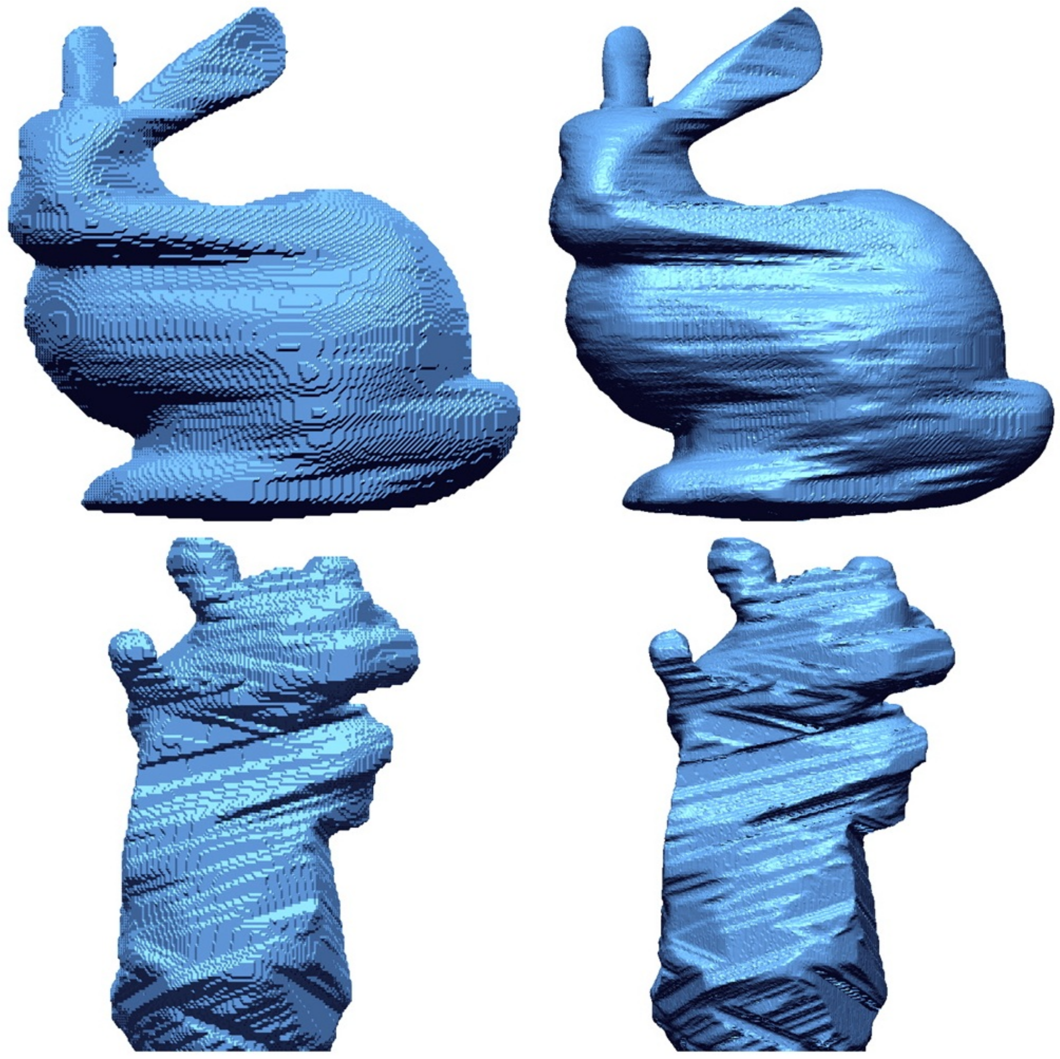


Figure 5.13: The left side shows the regular sampled object and the right side shows the Newton's method approach

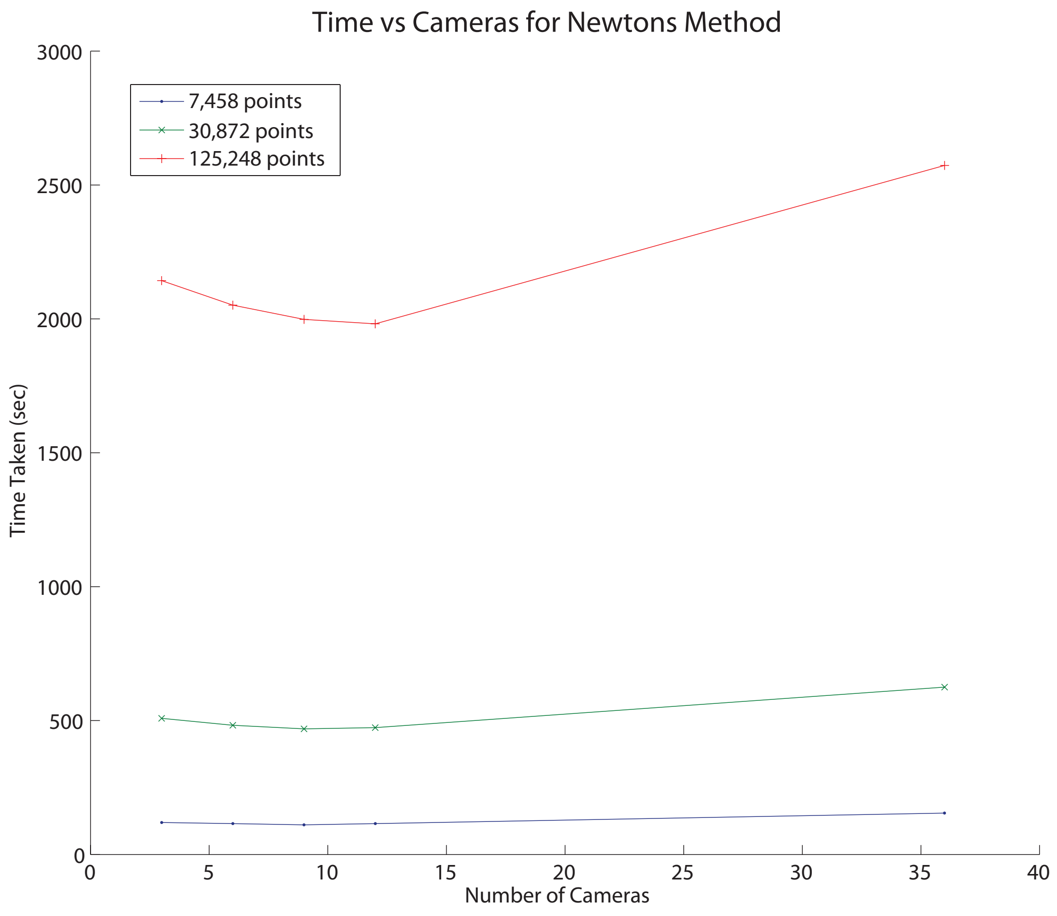


Figure 5.14: Comparison of different numbers of reconstruction points and camera numbers on the time taken to complete full 3D reconstruction.

stored. There is no requirement to store a large memory block to cover every point sampled in the space, so only 125,248 points are stored as compared to the regular sampling where 8,000,000 point values are stored until they are converted to a set of mesh points.

5.2.5 Improvement from silhouette to depth data

When depth data is used in the modelling instead of silhouette data a dramatic improvement in accuracy can be seen in the graph in Figure 5.17. There is also a similar improvement in visual appearance as seen in Figure 5.15. This is mainly due to the fact that concavities can now be reconstructed.

Computation time is similar, in that it is directly linked to the number of points, but it is increased as the modelling is different. There is no difference in the memory requirement as the same type and number of points are stored in both instances.

The ability of this framework to work on real environments with noisy data is demonstrated on the real objects. The raw data from the Kinect camera can be seen in Figure 5.16. No smoothing is done to the original data or to the end reconstruction. The framework is designed to take into account the accuracy and noise of each input and model it, therefore, the framework does the smoothing without losing any important information.

5.2.6 Impact of number of cameras

The number of cameras used can greatly affect the quality of the result. In terms of pure accuracy it can be seen from the graph in Figure 5.17 that with under 9 cameras there is a big drop in accuracy for the pure silhouette version, but more then 9 cameras does not result in any significant increase in accuracy.

Visually the results as the number of cameras is increased improves, smoothing out any blocky nature resulting from lack of views.

With the depth version the number of cameras required is less important, as there is more useful data in each camera view.

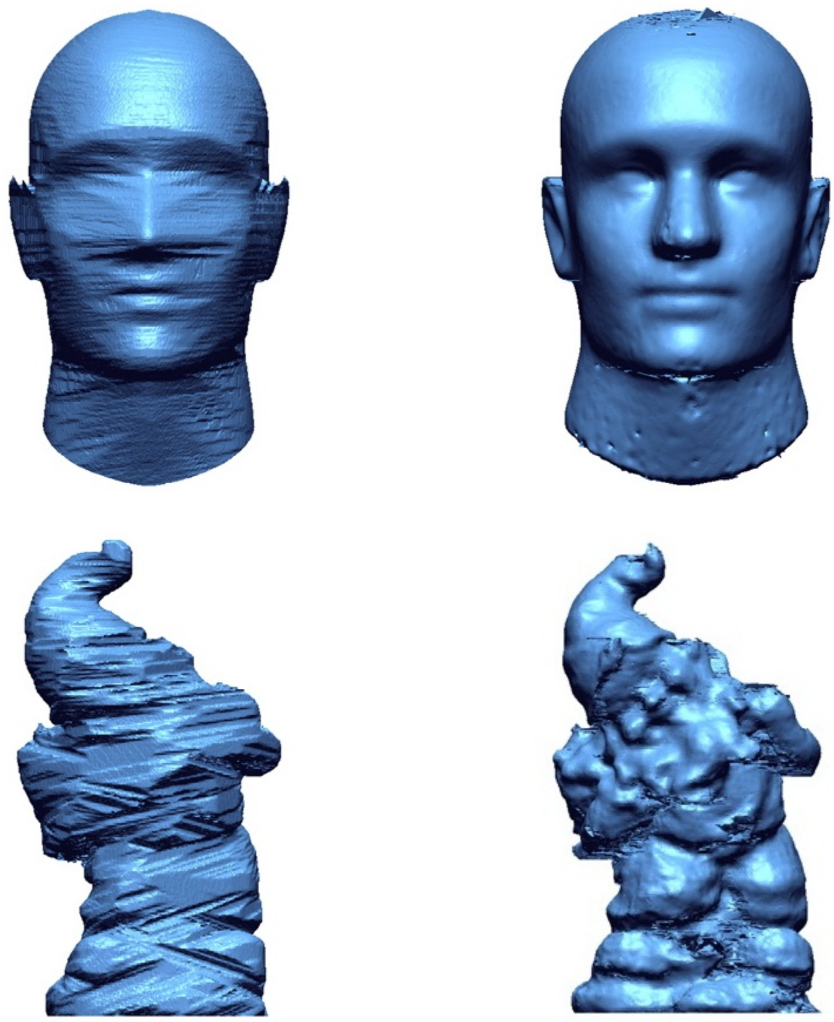


Figure 5.15: The left side shows the object using silhouette information only and the right side includes depth information

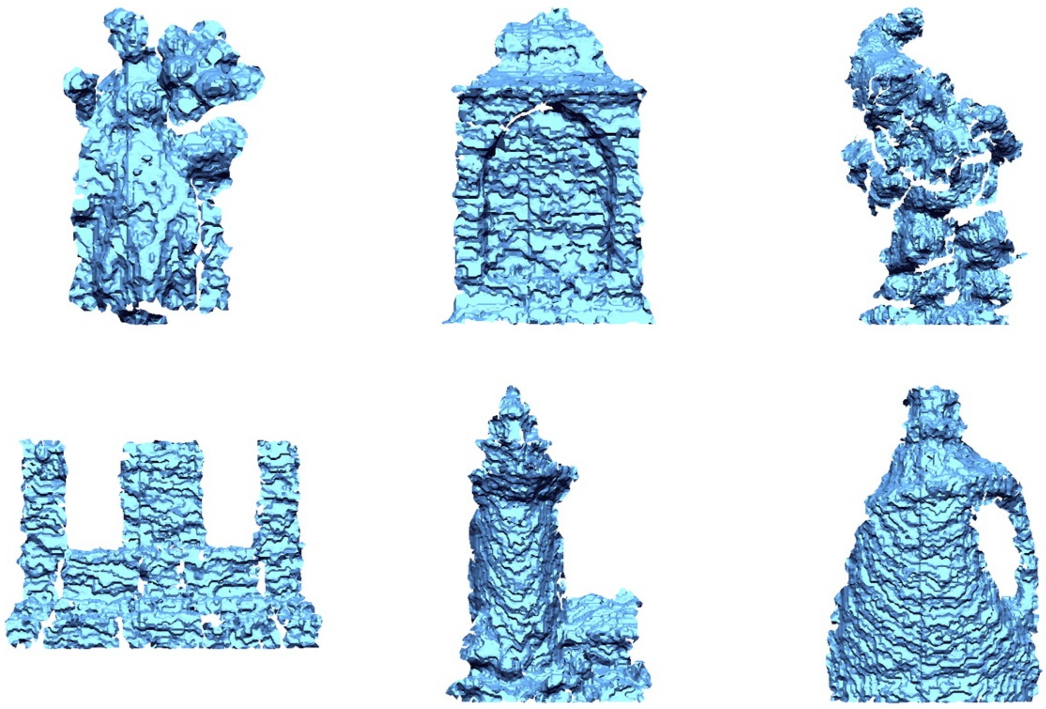


Figure 5.16: Meshing the raw depth data from the Kinect camera illustrates the extent of noise in the raw data. No smoothing is applied to this data before it is used. The framework is designed to take this type of noise into consideration.

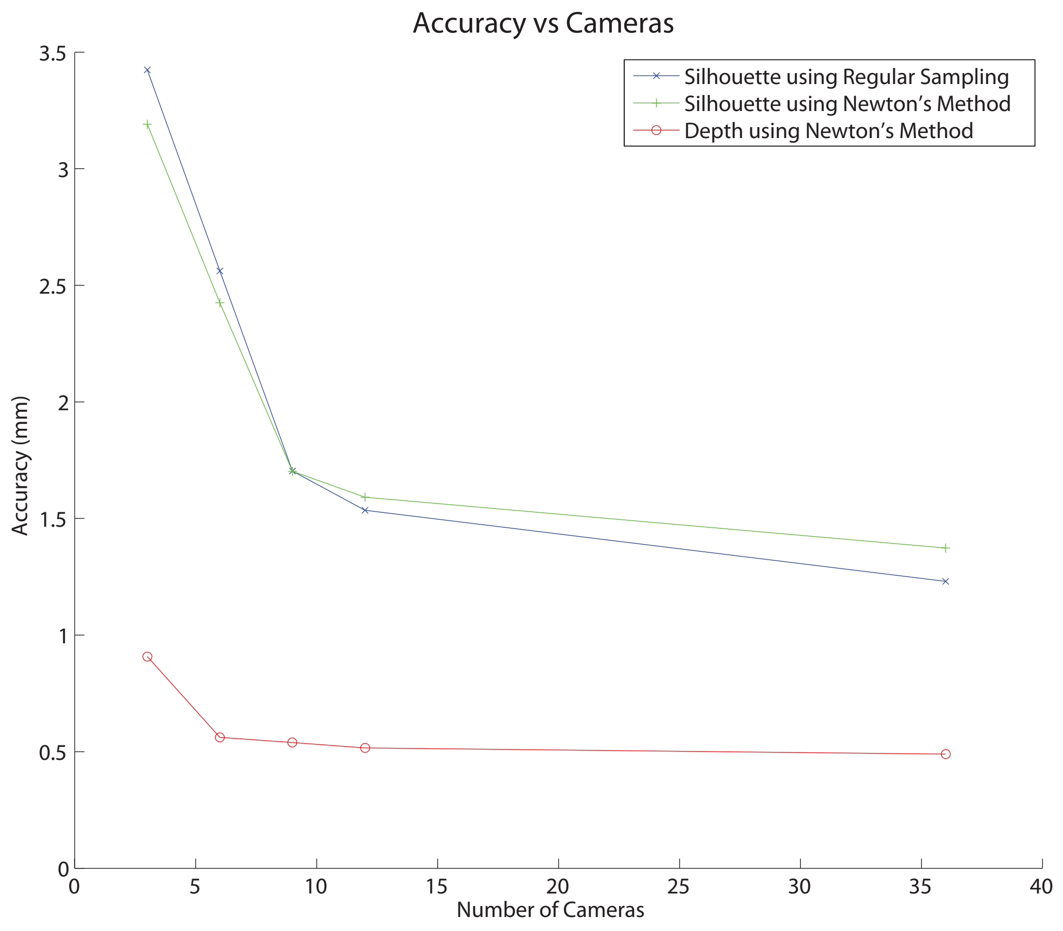


Figure 5.17: Comparison of camera numbers on accuracy across the three methods.

5.2.7 Impact of quality of the silhouettes

Background subtraction in uncontrolled environments is still a very hard problem and so the quality of the silhouettes can vary. For example, if the object is the same colour as one of the walls of the room, then from certain views the background subtraction might fail and give a poor silhouette or potentially no silhouette at all. In this case, for the silhouette version, the affect will be a lower likelihood across the whole space. This will affect the effective threshold used in the regular sampling approach greatly, but shouldn't affect the Newton's method approach much, as it is still just looking for the maxima of the space and not a maxima above a certain value. The end result of a damaged or missing silhouette depends on the scenario. For example, if there are only 3 camera views and one of the silhouettes has errors, this will result in a very poor reconstruction, but if a few views have errors out of 36 views, the quality of the reconstruction will not be unduly affected. Figure 5.18 shows what happens when the silhouettes used to reconstruct an object are replaced with that of another object. The figure shows a likelihood slice of the middle section of the Bunny object. In the figure, increasing numbers of the silhouettes are replaced with that of another object (the Head object). It can be seen that up to nine silhouettes can be contaminated before the overall shape of the slice begins to change.

The effects of 'salt and pepper' noise on the silhouette images was also explored [86]. It was shown that it did not adversely affect the final results and had only a minor decrease in accuracy. It is debateable how realistic 'salt and pepper' noise is on silhouette images given that it can generally be easily removed using morphology techniques.

5.2.8 False Limbs

For the silhouette version, if the number of cameras is very low, there can be a problem with some reconstructions called false limbs. This is where a false limb will appear in the overlap of the shadows of two other parts of the object. This is noticeable in the teapot object. The handle appears three times and so does the spout when there are only 3 cameras used, as seen in Figure 5.19. When using depth data, false limbs do not occur.

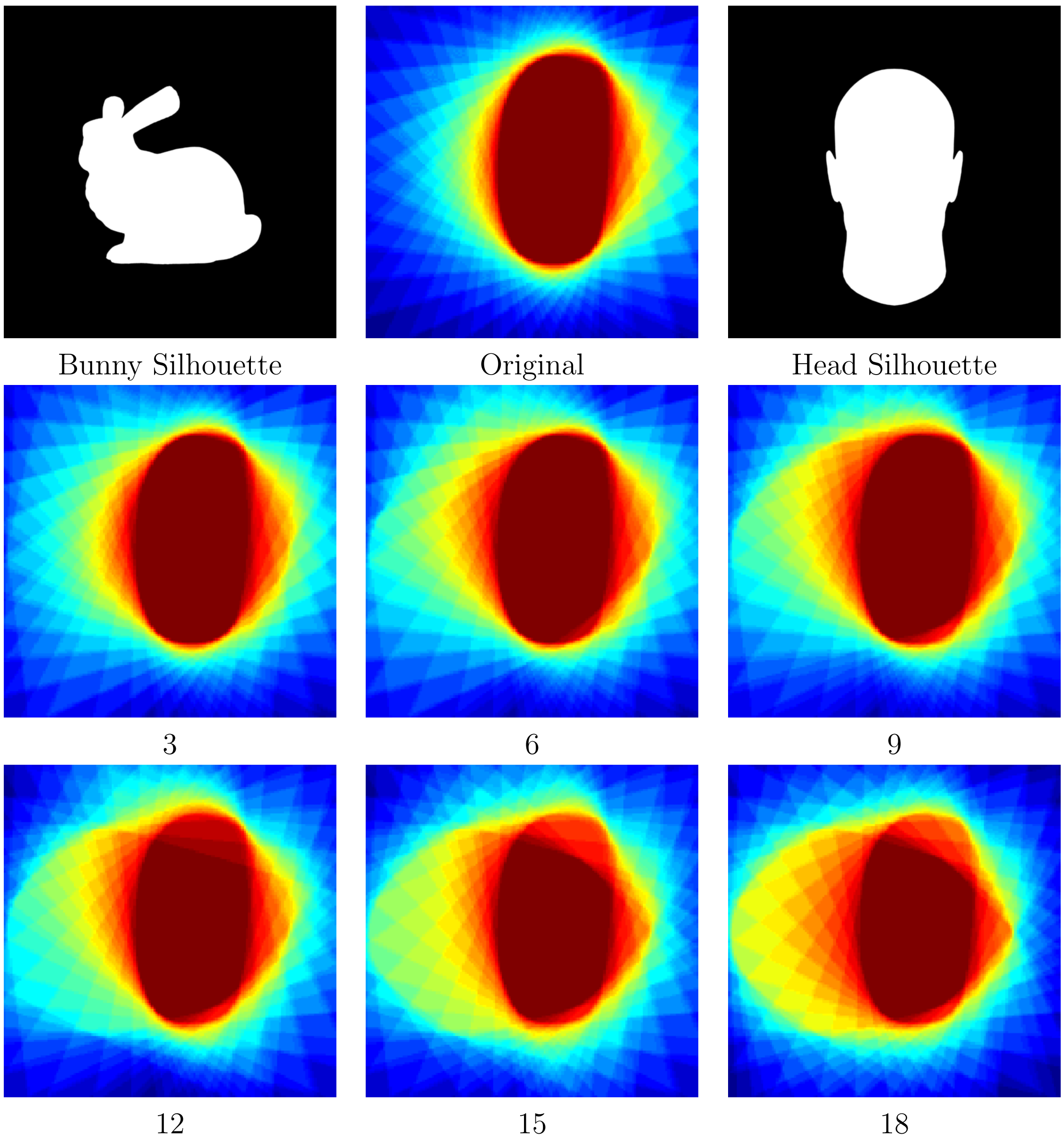


Figure 5.18: For a single slice of a silhouette reconstruction from 36 camera views of the Bunny object (An example silhouette can be seen in the top left image). Different numbers of silhouettes are replaced by silhouettes from another object (the Head object an example silhouette of this object can be seen in the top right image). This is to assess how the model handles contaminated data. Up to nine silhouettes present very little change, however, over that number and it can be seen that the slices are starting to change shape.

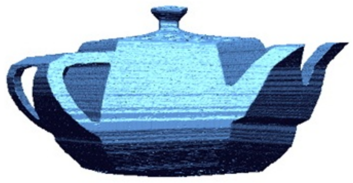


Figure 5.19: Example of false limbs being produced. Here only 3 cameras are used to reconstruct the teapot object using silhouette information alone. Due to overlapping shadows false spouts appear around the object.

5.2.9 Texturing

The colour images also taken of the objects can be used to add colour to the final reconstructions. Each of the vertices of the reconstruction are projected back into the image planes and the corresponding pixels are used to colour the vertices. Examples of the coloured reconstructions can be seen in Figure 5.20.

5.2.10 Comparison with Laser Scan

With the synthetic objects it was possible to compare the results with a ground truth model. This was possible because the raw data was synthesized from an original model, therefore, this original model could be used to determine the accuracy of the reconstruction. With the real objects no such ground truth exists which makes it harder to determine the accuracy of the reconstruction method in real life situations. To provide an estimate of the the accuracy of the reconstruction a laser scanner was used as seen in Figure 5.21.

The laser scanner used is a Minolta Vivid-700. It provides a scan of the front of the object. No post processing is done to the scan in terms of filling the holes or merging multiple scans together. While the laser scan is not complete and has holes in it, it can be seen to be slightly more detailed than the reconstruction, in particular the detail of

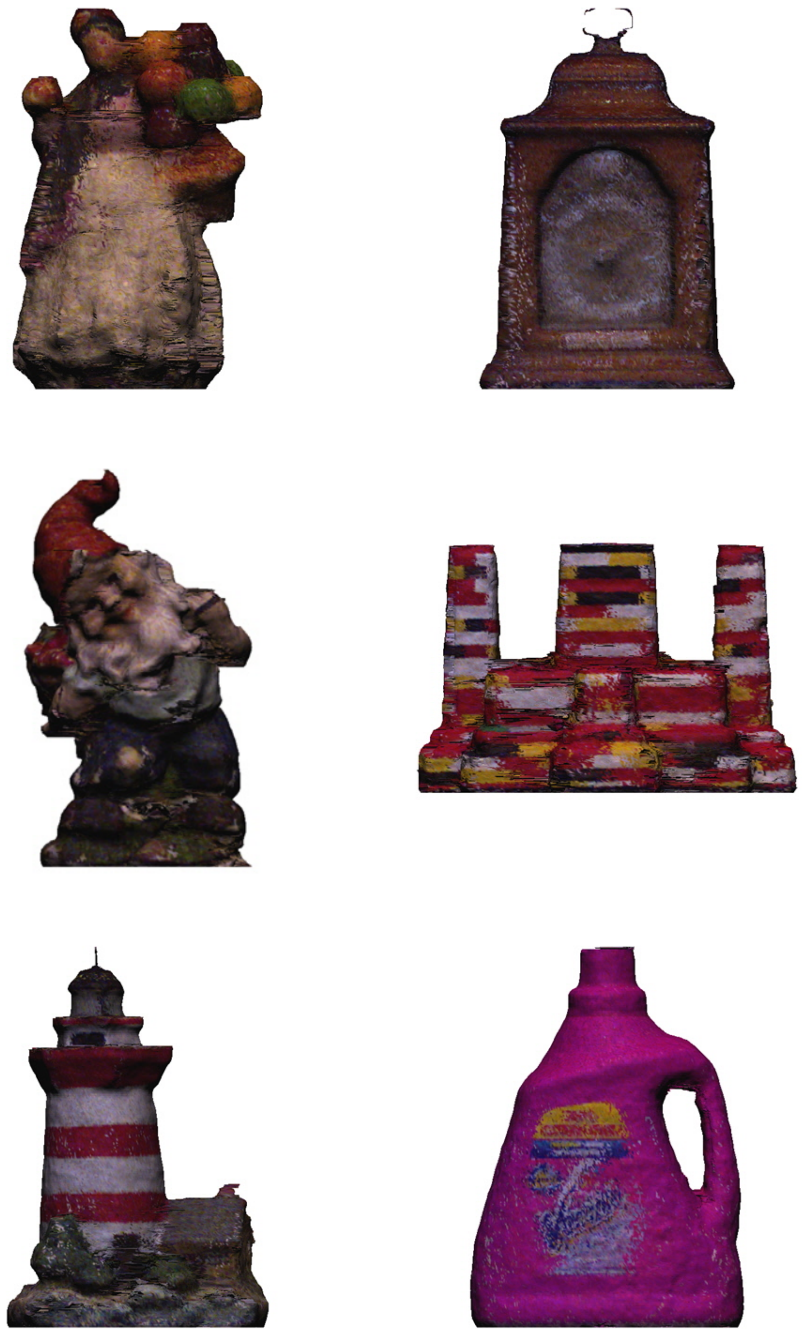


Figure 5.20: Example reconstructions with added colour information

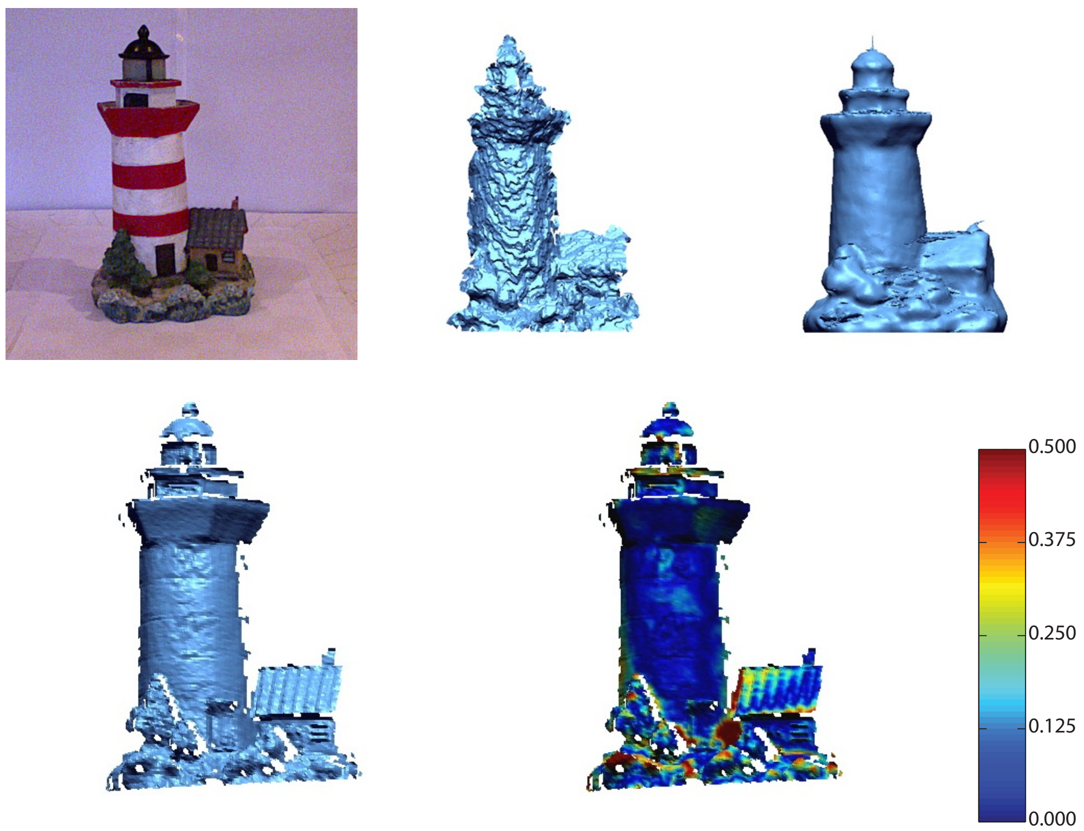


Figure 5.21: Comparison between reconstruction method and laser scan. The top left image shows an image of the object, the top middle image shows the raw data from the Kinect camera, the top right image shows the reconstruction of the object. The bottom left image shows the raw data from the laser scan of the object and the bottom right image shows the error between the reconstruction and the laser scan. The error bar scale is 0-0.5 cm and the object is 18 $\times$ 16 $\times$ 26 cm. The average error is 1.4 mm.

the roof tiles can be clearly seen on the laser scan where on the reconstruction the roof of the house is smoothed over. This is for two reasons; the raw data from the Kinect is too noisy to identify that level of detail and the reconstruction method results in a certain level of smoothing to reduce the impact of noise on the reconstruction. Overall the reconstruction is shown to be very faithful to the laser scanner with an average accuracy of 1.4mm. This is an excellent result given the Kinect camera is only a few hundred euro while the laser scanner is a couple of thousand euro.

While the lighthouse is an example of a complex object, better results can be seen with the simpler vanish bottle object. The laser scan has trouble capturing the colour blue so the scan has some gaps in it as seen in Figure 5.22. Otherwise the results shows an accuracy result of 0.765mm.

The laser scan provides a close approximation to a ground truth as it provides a mesh which has is more accurate than the raw data from the Kinect, but it is not perfect. For the Vanish object, parts of the blue label haven't been recovered and some specular reflection during the scanning process caused some noise around the center of the lid.

5.3 Conclusions

In this chapter, the reconstruction methods are assessed over several data-sets. The methods to generate the data-sets used in this work are described. A set of example reconstructions are displayed to show that the methods described in the previous chapters work for a range of shapes. The improvements between methods are shown and the impact of a number of parameters are explained. The methods have been proven to work with real objects under real-world conditions, where there is real-world noise (noise on the depth data recorded on the Kinect, and inaccurate camera parameters) and could potentially be used by anyone with a Kinect camera to reconstruct everyday objects. The next chapter gives an overview of all the contributions presented in this thesis along with potential future avenues for research.

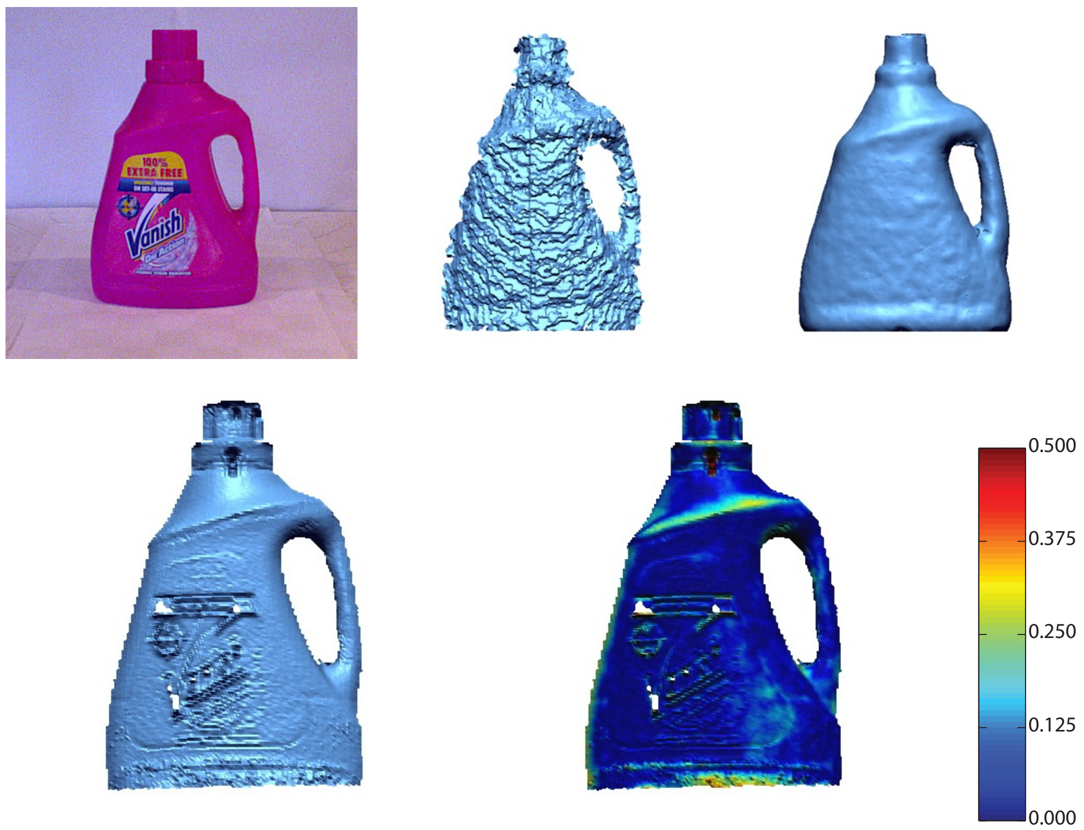


Figure 5.22: Comparison between reconstruction method and laser scan. The top left image shows an image of the object, the top middle image shows the raw data from the Kinect camera, the top right image shows the reconstruction of the object. The bottom left image shows the raw data from the laser scan of the object and the bottom right image shows the error between the reconstruction and the laser scan. The error bar scale is 0-0.5 cm and the object is 18 × 10 × 27 cm. The average error is 0.765 mm.

Chapter 6

Conclusions

This thesis focuses on developing a new framework for merging data from multiple sources in a more statistically sound method to reconstruct a 3D object. This chapter gives an overview of all the work presented in this thesis and outlines the main contributions achieved. Section 6.2 then describes potential future avenues of research, that could be explored next.

6.1 Summary

Chapter 2, Section 2.1 presents an overview of multi-view reconstruction methods and highlights a number of areas of limitations. Specifically these are the discrete nature of a number of the methods which in terms of computation and memory requirements do not scale well and the multi-stage pipelines which require decisions be made that can result in loss of information and propagation of error.

Initial work was carried out with Donghoon Kim on a method to develop a continuous framework to tackle these problems. The work was based on an orthographic camera model and used only silhouette information [51].

This initial work has been continued in this thesis and formalised. Chapter 3, Section 3.1 presents **a new flexible statistical framework for multi-view reconstruction** which is based on the generalised relaxed Radon transform. It has the ability to merge data from multiple sources and of multiple types together. It has the ability to model the individual noise of each type of data based directly on the sensor capturing the data. It then allows all the data captured to be used for the inference of the reconstruction in one step with no requirement for multi-stage decisions.

Section 3.2 presents **a modelling based on silhouette information** which is placed in the framework. This modelling is based on the pinhole camera model as opposed to the orthographic camera model presented by Kim *et al.* [51]. Section 3.2.3 then presents **a reconstruction method based on regular sampling from silhouette information** which uses the modelling to infer the shape of the object. This method has similar limitations in terms of memory requirements and computation time but is then extended in Section 3.2.4 to **a reconstruction method based on Newton's method from silhouette information**. This method overcomes many of the limitations and produces similar results. As only silhouette information is used it is not possible to recover concavities but the convex shape of the object can be accurately reconstructed.

Relatively cheap devices such as the Microsoft Kinect camera are allowing more investigation of depth data, therefore, Section 3.3 incorporates **a modelling based on depth information**. Sections 3.3.3 and 3.3.4 then present **a reconstruction method based on Newton's method and a developed technique called surface explore from depth information** to infer the shape of the object. This method can merge both silhouette and depth data and models the separate error in each to produce accurate reconstructions. The added depth data allows concavities to be reconstructed.

It was found that camera alignment is very important for depth data, therefore, Chapter 4 presents **an extension of the framework to optimize for nuisance variables, in particular the camera parameters**. This extension is applied to a real world data set captured with the Kinect camera and allows real objects to be reconstructed.

Finally Chapter 5 gives an overview of the generation and capture of three data-sets used in this thesis. All the reconstruction methods are then compared and contrasted showing their relative memory requirements and computation time to achieve full reconstructions. The methods are also applied to real-world objects in which the cameras are first fully aligned using the methods described in Chapter 4 and then the objects are reconstructed using the methods described in Chapter 3. The Kinect depth data is very noisy by itself but experimental results show that the framework presented in this thesis is able to cope with that noise and produce accurate results. The results are also compared to laser scans of the same objects which are used as near ground truth representations. The accuracy of the reconstructions when compared to the laser scans is similar to that of the synthetic objects accuracies.

6.2 Future Directions

There are many areas of future research possible; this section outlines some of them.

Parallel Implementation

The mathematical framework is designed with the potential for parallel computation, both at the kernel density estimation level as described by Srinvasan and Duraiswami [98] and at the higher level of reconstruction. Each point being reconstructed is independent of all the other points and can be processed in parallel. The computer industry is moving towards greater multi-core processing at both the CPU and GPU levels and taking advantage of this can result in considerable speed ups in generating reconstructions.

Extending the Model

The current modelling presented in this thesis is capable of utilising silhouette information, depth information and also a combination of the two. Kim has shown that colour information can also be modelled using very similar techniques which results in a higher dimensional solution [50]. It would be useful to extend the modelling further to include other information such as feature points and descriptors.

Modelling Prior Information

Face and human body reconstruction and tracking are very active areas of research. The results can be greatly improved by including prior information based on the fact that the type of object being reconstructed is known. 3D deformable human face models exist [76] which can be used as prior information when reconstructing faces. Kim showed that the mathematical basis of this work can be extended to include this prior information. Kim did this with an orthographic camera model and on 2D slices [?]. This type of prior modelling could be extended to work with 3D data and the more common pinhole camera model. Also different types of prior knowledge such as over-complete dictionaries or parametric dictionaries for object classes, could be used to improve the accuracy of the result and speed of reconstruction.

By introducing constraints on what type of object is being reconstructed, for example, a face or human pose, then with the help of that prior knowledge, it is possible to go further and work on dynamic faces and tracking the human pose, similar to Shotton *et al.* [93].

Over Time Estimation

At the moment it is assumed that all the images are taken at the same point in time for one instance of the object, with no concept of previous and future moments in time. It may be possible to extend the framework to use past and present reconstructions to help with future reconstructions especially for tracking applications. The starting points for the current reconstruction can be based on the reconstruction of a previous time. This could potentially improve the reconstruction as more information is added over time and it could also improve the speed of generating the reconstruction. This over time estimation could be combined with prior model information to help reconstruct dynamic objects.

Multiple Objects in the Same Scene

It should be possible to reconstruct multiple objects in the one scene. Just like the legs of the human or the towers of the Lego object, parts of objects can be reconstructed separately and so multiple objects in the one scene can be reconstructed. It should be noted that if one object obscures another object (blocks a camera view), the reconstruction of that obscured object will be hampered. The ability of the algorithm to cope with obscured objects partly depends on how robust it is to outliers, as the obscuring object will present a data outlier in the input which should be ignored. Hansen and Toft show that the discrete generalised Radon transform is robust when estimating curves in noisy images [41]. This work uses an algorithm based on a continuous form of the generalised Radon transform which should therefore also be robust, but the tools and techniques for extracting the reconstruction may not be as robust to these types of outliers and may need improvement to avoid them.

Meshing Algorithm

The current approach produces a set of points scattered over the surface of the object of interest. A post process is then used to mesh the point cloud to generate the final 3D reconstruction. The surface explore technique presented in Section 3.3.4 results in connected points. It also allows the user to define the rough distance between points on the surface of an object. It would be useful to directly generate the meshed structure of the points directly from this data. The underlying likelihood information could also be used in weighting points to improve the overall mesh structure. An addition to this would be to vary the point sampling across the surface of the object based on the complexity of the surface. Flat areas only require a few points to be accurately reconstructed whereas areas which are highly complex require numerous points to be accurately reconstructed. By detecting the complexity of the surface and varying the number of points used would result in a more efficient approach.

Moving Camera with Calibration on the Fly

The current setup assumes stationary cameras with known parameters. The techniques explained in Chapter 4 allow these camera parameters to be refined to improve alignment. Potentially this work could be extended to allow a hand held camera move dynamically around the object and perform the calibration of the camera on the fly.

Appendix A

Kinect

Kinect for XBox 360 is a cheap (∼€150) web-cam style add-on peripheral for the Microsoft XBox 360 console introduced in November 2010. It is designed as an alternative input controller which allows the user to control the system using human gestures and voice command. To achieve this the sensor captures an RGB image, depth data and an array of audio inputs. Software can then be used to interpret specific human gestures, track a human skeleton and perform voice recognition. Its main use is aimed at gaming. The research community see this device as an inexpensive way to capture depth information as traditional tools to capture depth information have been very expensive well exceeding €1,000 or even €10,000 in cost.

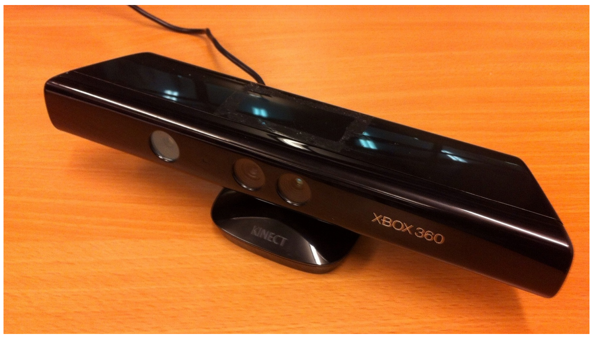


Figure A.1: The Microsoft Kinect sensor

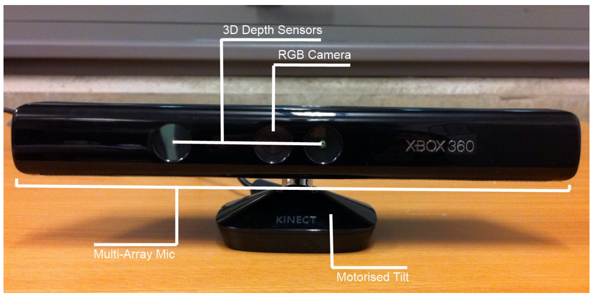


Figure A.2: The parts of the Kinect sensor

A.1 Hardware

The Kinect sensor is made up of a number of components including:

- an RGB camera
- an infrared laser projector
- a monochrome CMOS sensor
- a multi-array microphone
- a motorised tilt mechanism

The RGB camera is, by default, set at a resolution of 640×480 and runs at 30 frames per second. It records 8-bits per pixel using a Bayer filter to encode the colour information. It is possible to increase the resolution of the RGB camera to 1280×1024 and while it is able to capture images at 30 frames per second the USB interface is only able to transmit at 10 frames per second at that higher resolution.

The infrared laser projector and the monochrome CMOS sensor form the two parts of the 3D depth sensor. A dedicated chip on-board the Kinect converts the input from these two components and computes a depth per pixel. The result is an output depth image with a resolution of 640×480 with a depth resolution of 11-bits providing 2048 levels of depth.

The multi-array microphone is used to capture audio data. The time difference of sound captured between different microphones in the array can be used to determine the location of the audio source relative to the microphone array. It should be noted that the audio component was not used in this work.

The motorised tilt mechanism allows the Kinect sensor to automatically or manually be tilted up to 27° either up or down to improve its field of view.

A.2 Depth Sensor / Structured Light

The method the Kinect sensor uses to determine the depth information is called structured light. Structured light is a method whereby a known regular pattern of light is projected onto the scene or object of interest. In the case of the Kinect sensor this light is an infrared light and so is not visible to the naked eye. A separate sensor, in this case the monochrome CMOS sensor, is used to detect the location and shape of the projected pattern. Knowing the exact relative position between the projector, the sensor and the deformation of the regular pattern, it is possible to calculated the 3D coordinate of each projected point using complex geometry.

Their are a number of limitations to this method of extracting depth information. One major limitation is that the relative position of the projector and the sensor will always create a shadow to one side of the object, in this case the left, where the CMOS sensor can see but, the projector can't hit, as it is blocked by the object itself.

Another limitation to this method is due to the surface materials in the scene. If the material of an object is reflective or transparent this will result in major difficulties as the projected pattern will be distorted due to reflections, or will be distorted by passing through the surface due to its transparency. This will result in missing or inaccurate depth data on those surfaces.

The range of depth the sensor can determine is also limited. If the object or surface is too close to the sensor it will not be able to detect the pattern correctly and conversely if the object is too far away the pattern will also disappear. In the case of the Kinect sensor this depth range is between 0.5 meters to roughly 6/7 meters. Lighting conditions can also effect this method. If the light pattern is obscured by another light source it can become undetectable, and too much natural sunlight will obscure the infrared pattern.

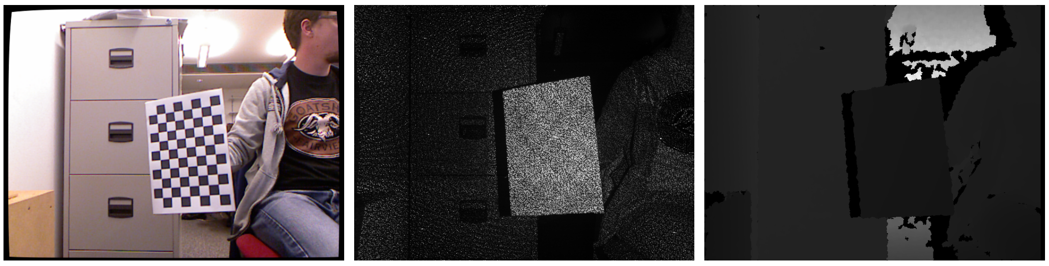


Figure A.3: Example Kinect Capture. Left is the RGB image, the middle is the projected pattern as seen by the CMOS sensor and the right image is a grayscale representation of the depth with white being far away and the dark being close to the camera.

A.3 Software / Drivers

A number of software packages and drivers have been officially and unofficially released for the Kinect. As the original device was only designed to work with the XBox 360 console, the use of third party software and drivers was required to access the data from the Kinect for other uses. In late June 2011 Microsoft released an official software package and drivers to access the data.

Three drivers have been examined during this work:

- Microsoft Kinect for Windows SDK [1]
- OpenNI [2]
- libfreenect [3]

Each of the different packages has their own advantages and disadvantages, for example, the official Microsoft SDK has very robust skeleton tracking [93] but doesn't support the higher resolution RGB image that the unofficial ones can offer. The unofficial software packages have a greater depth range which is slightly limited in the Microsoft SDK but only the Microsoft SDK supports full audio features.

[1]`http://kinectforwindows.org/`
[2]`http://openni.org/`
[3]`http://openkinect.org/wiki/Main_Page`

One very useful source found during this work was Nicolas Burrus' web-site [4]. He has developed a number of software programs for the Kinect which are very useful to get started when using the alternative packages OpenNI and libfreenect.

A.4 Calibration

A number of different parameters must be considered to fully calibrate the Kinect:

- the intrinsic parameters of the RGB camera
- the intrinsic parameters of the Depth camera
- the rotation and translation between the RGB camera and the depth camera
- the extrinsic parameters with respect to a world origin
- the depth value recording to a real world depth distance

The intrinsic parameters of the RGB camera can be found using standard calibration techniques using a chessboard pattern. For the depth camera this is a bit more difficult. Ideally the same chessboard pattern would be used but the chessboard corners of a standard chessboard would not be visible in the depth image. To overcome this problem a pane of glass is used with a chessboard pattern masked out as seen in Figure A.4. This means the depth camera would see through the unmasked squares and be stopped at the masked squares, resulting in a chessboard pattern seen in the depth image. This can then be used to calibrate the intrinsic parameters of the depth camera. Viewing this same glass chessboard from both the RGB camera and the depth camera allows a standard stereo calibration to be done to determine the rotation translation between the two cameras. The relationship allows a real world origin to be calculated in the RGB image and from that the extrinsic parameters of the depth camera can be determined, or the depth camera data can be re-projected into the RGB camera space to provide a pixel to pixel match.

Once all these other parameters have been calibrated it is possible to calibrate the depth data itself. A chessboard pattern was placed on a wall in front of the Kinect

[4]`http://nicolas.burrus.name/index.php/Research/Kinect`

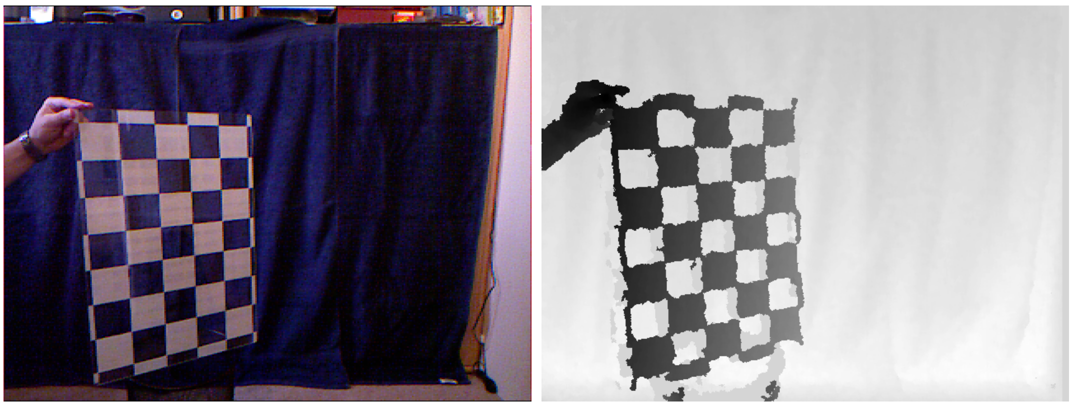


Figure A.4: The glass chessboard used to calibrate the Kinect Camera, both the colour and the depth sensor can see the pattern.

camera and repeated pictures of the wall were taken while moving the camera further backwards. Using the RGB image of the chessboard the distance from the wall to the RGB camera centre was calculated and was then translated to the centre of the depth camera. This depth value was then compared to the average depth value observed in the depth image. Using the two sets of values, depth measured and depth observed, a polynomial function was calculated using least squares to give a conversion between the two values.

Appendix B

Further Results

This appendix displays the rest of the results obtained using the method described in the main body of this thesis.

All the results are displayed in the same layout, the first row shows; a raw colour image of the object being reconstructed, then a silhouette image and then the raw depth data from one view. The subsequent rows have the reconstructions for increasing numbers of cameras generally 3, 6, 9, 12, and 36. The first column has the silhouette reconstruction using regular sampling as described in Section 3.2.3. The second column has the silhouette reconstruction using Newton's Method as described in Section 3.2.4. The third column has the depth reconstruction version using both Newton's method and the surface explore algorithm as described in Sections 3.3.3 and 3.3.4. Figures B.1 to B.6 show the results for the real objects captured using the Kinect sensor. The extrinsic camera parameters for these objects were optimised using the method described in Section 4.3. Figures B.7 to B.11 show the results for the synthetic objects, and Figures B.12 and B.13 show the results for the Middlebury objects.

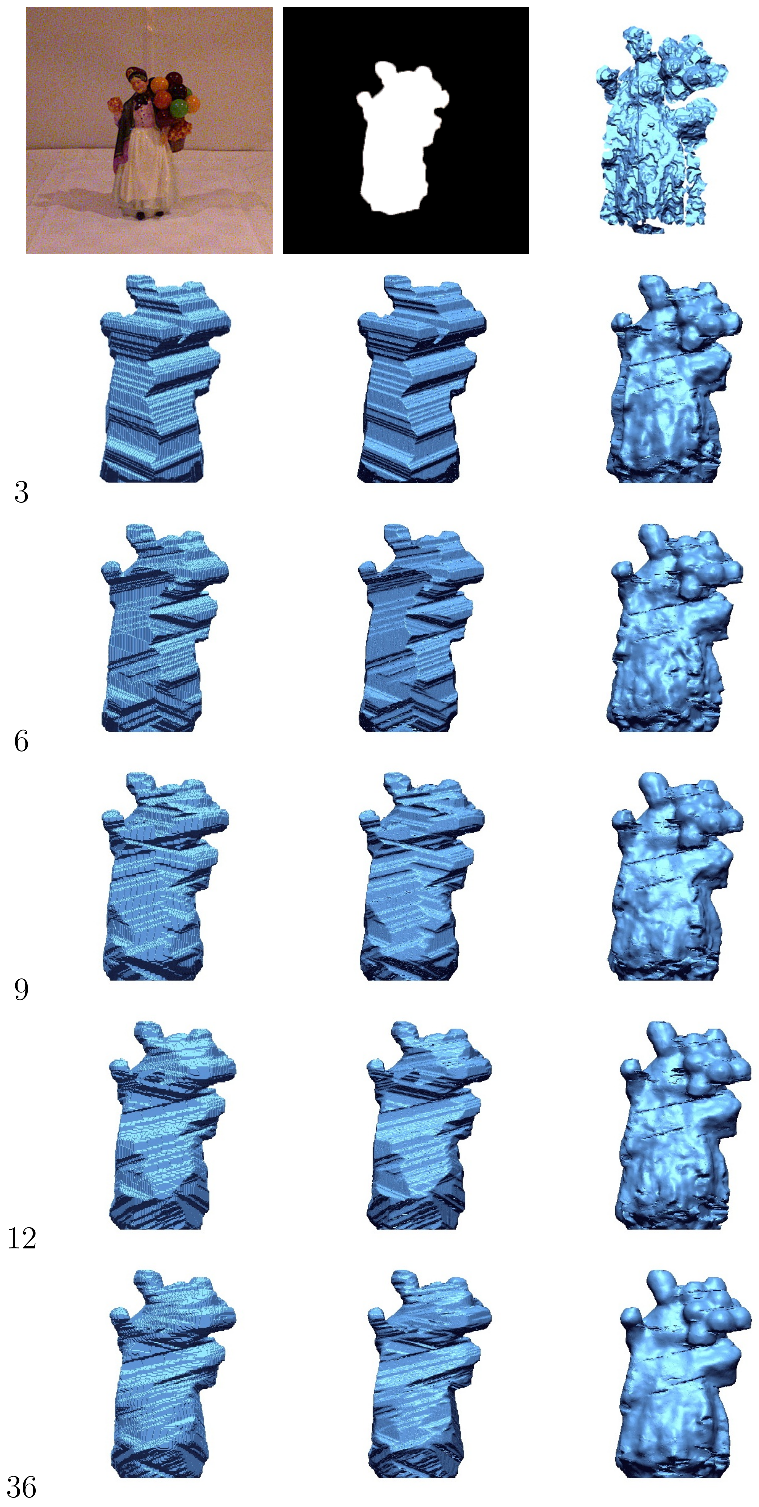


Figure B.1: Balloon Lady

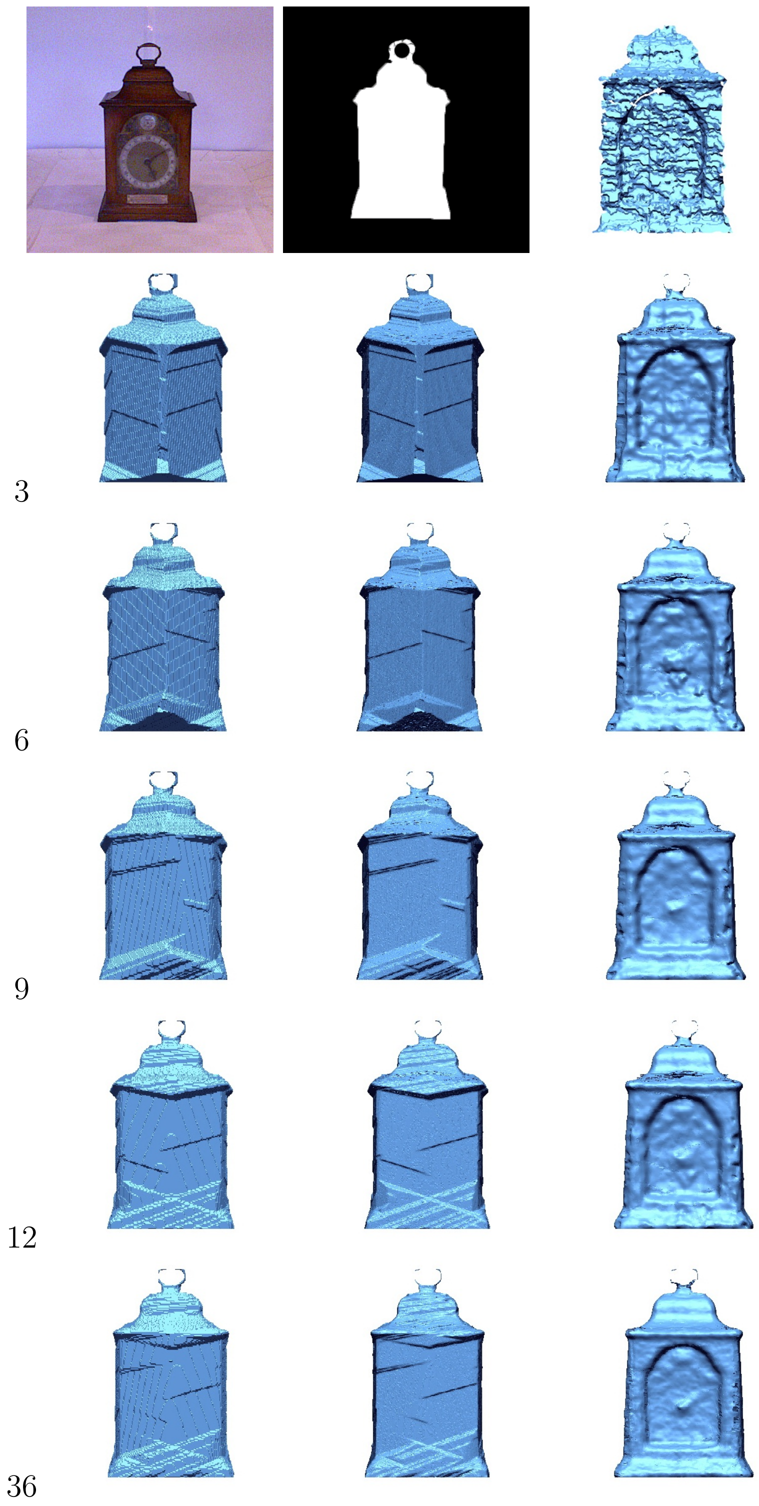


Figure B.2: Clock

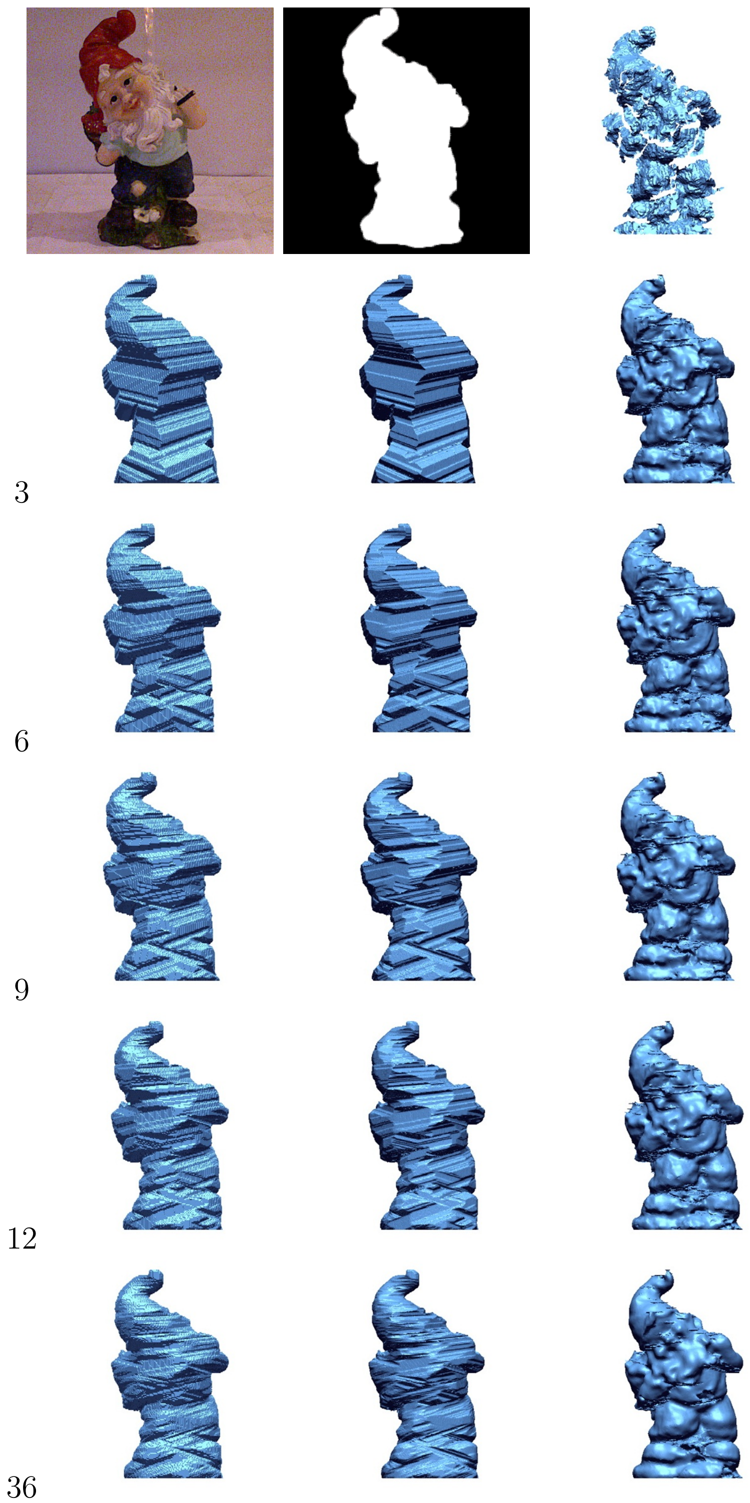


Figure B.3: Gnome

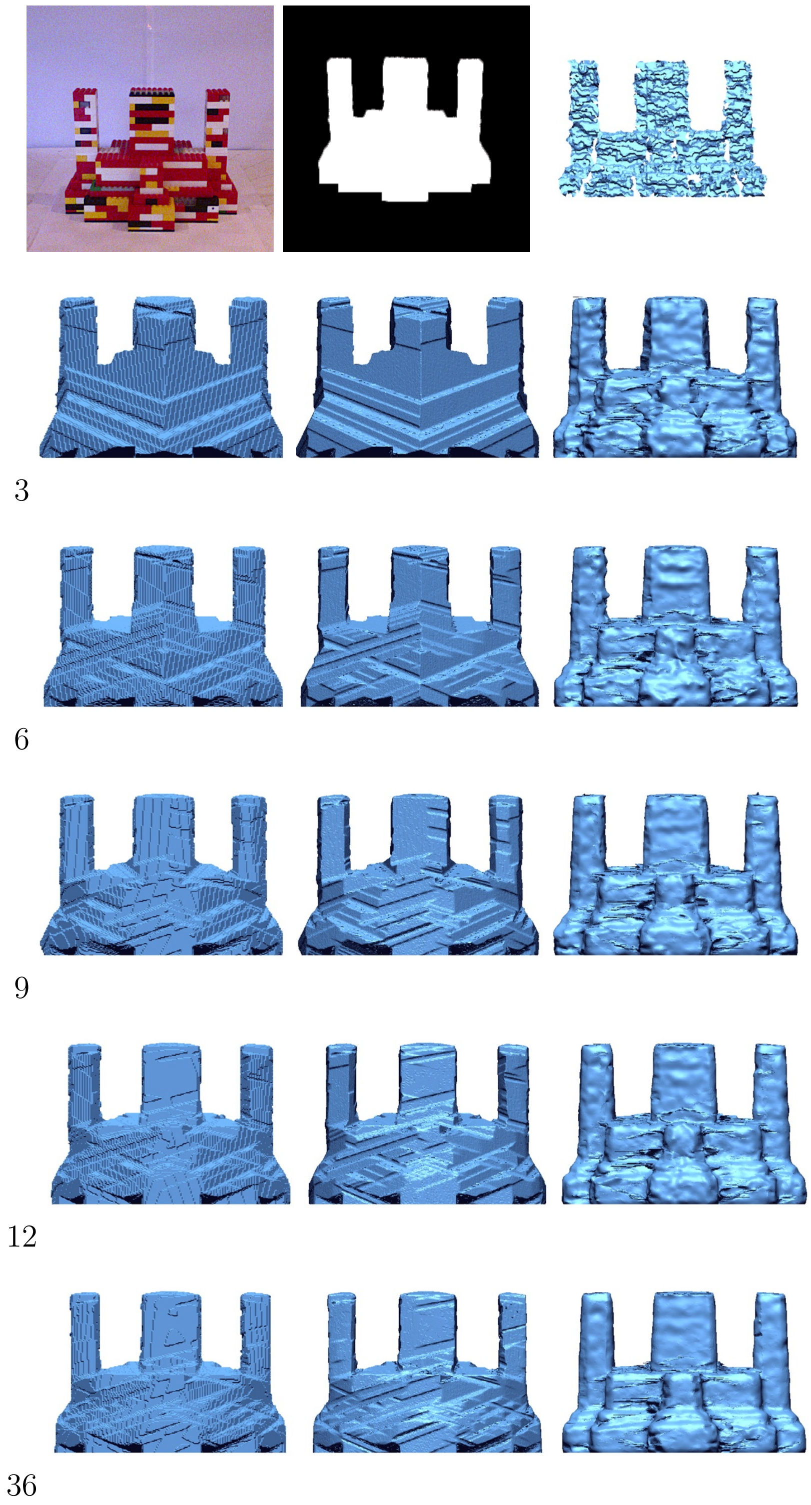


Figure B.4: Lego

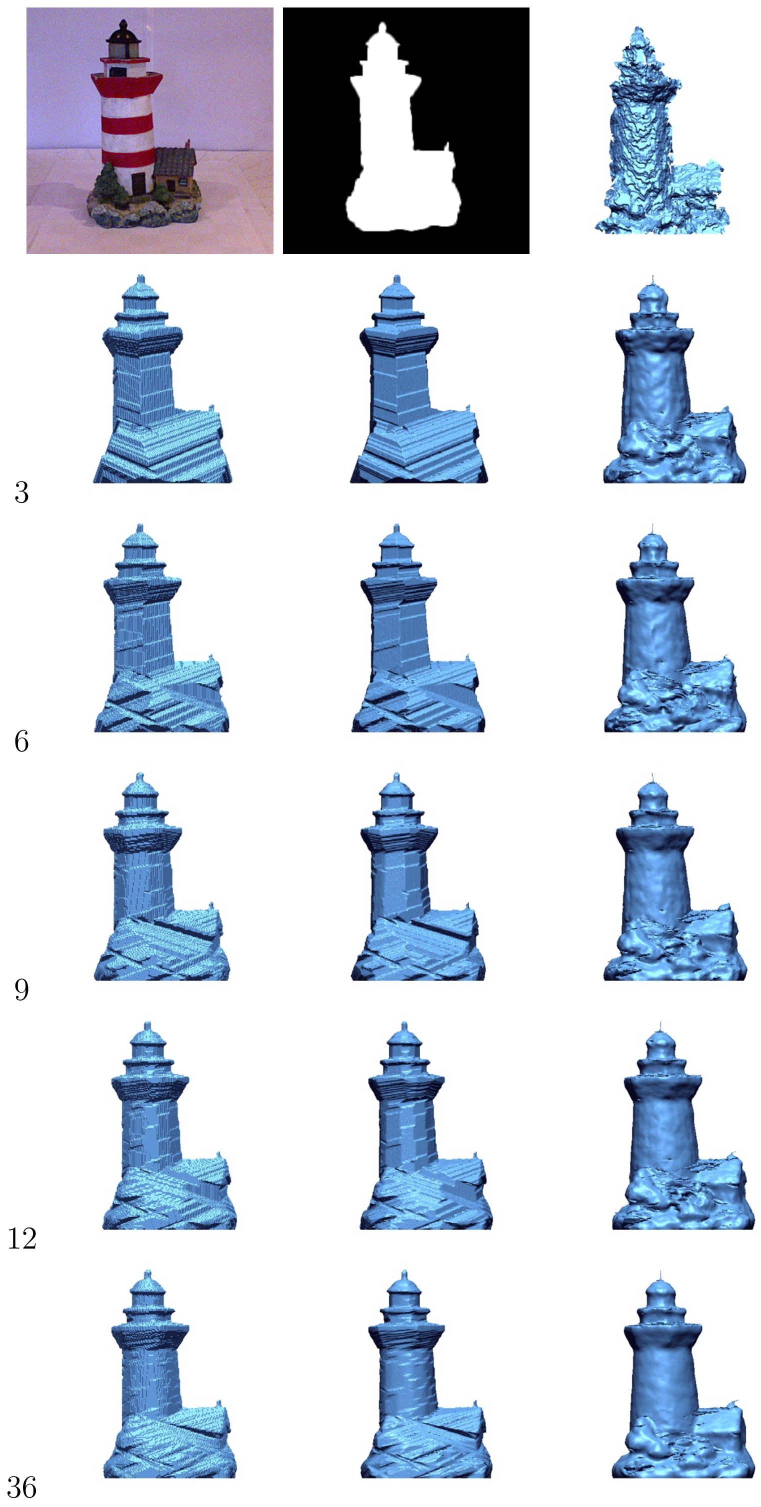


Figure B.5: Lighthouse

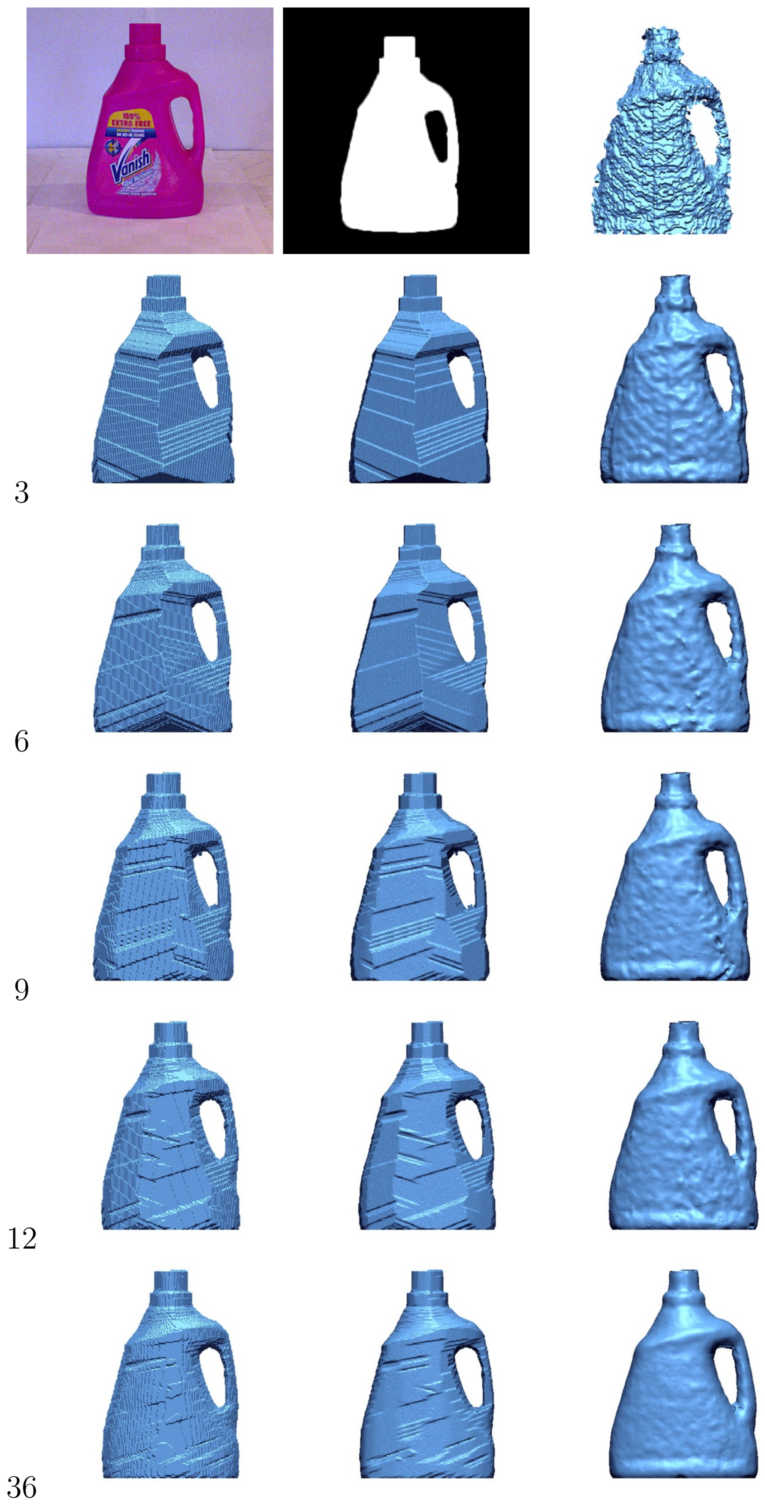


Figure B.6: Vanish

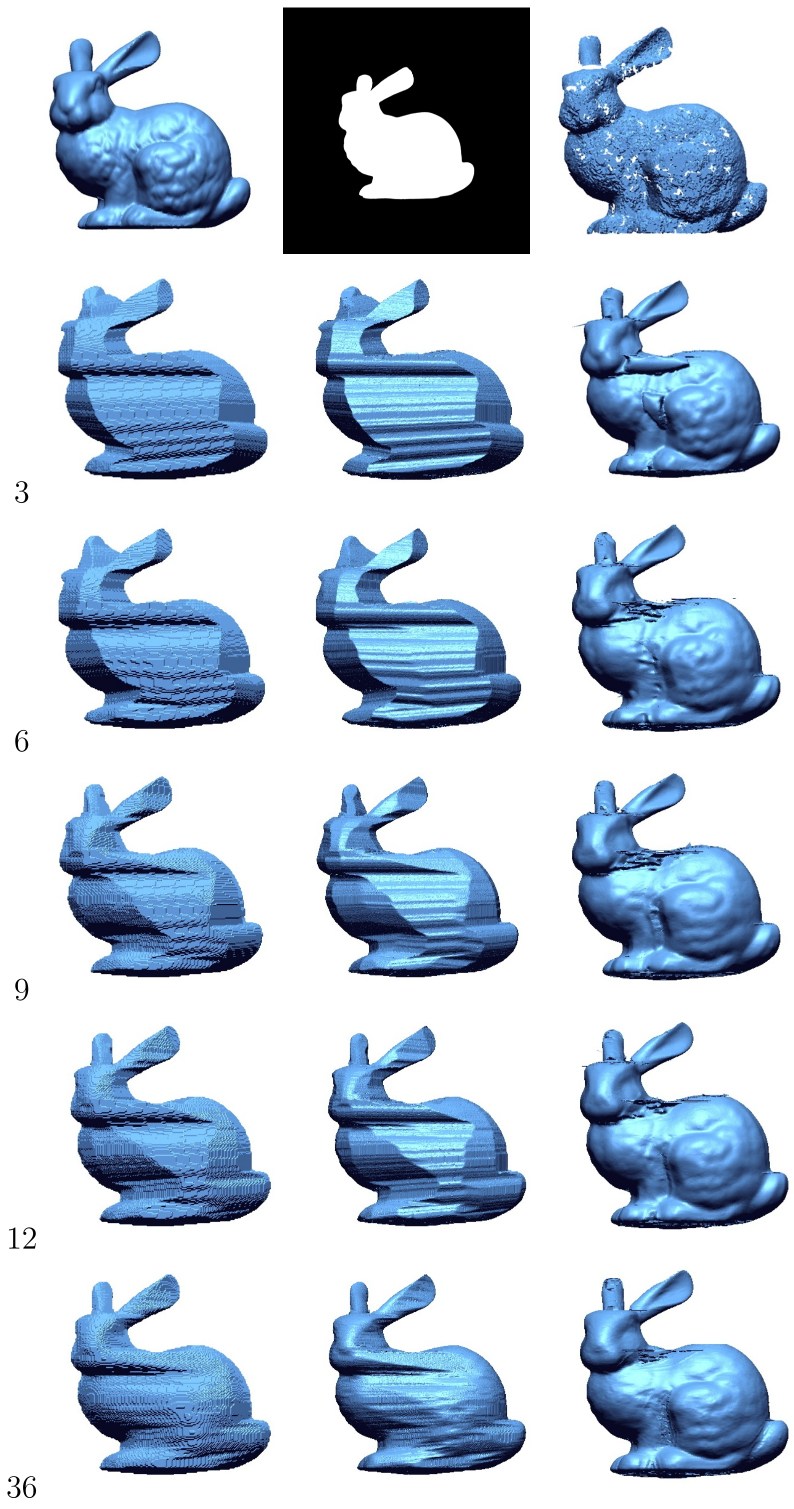


Figure B.7: Bunny

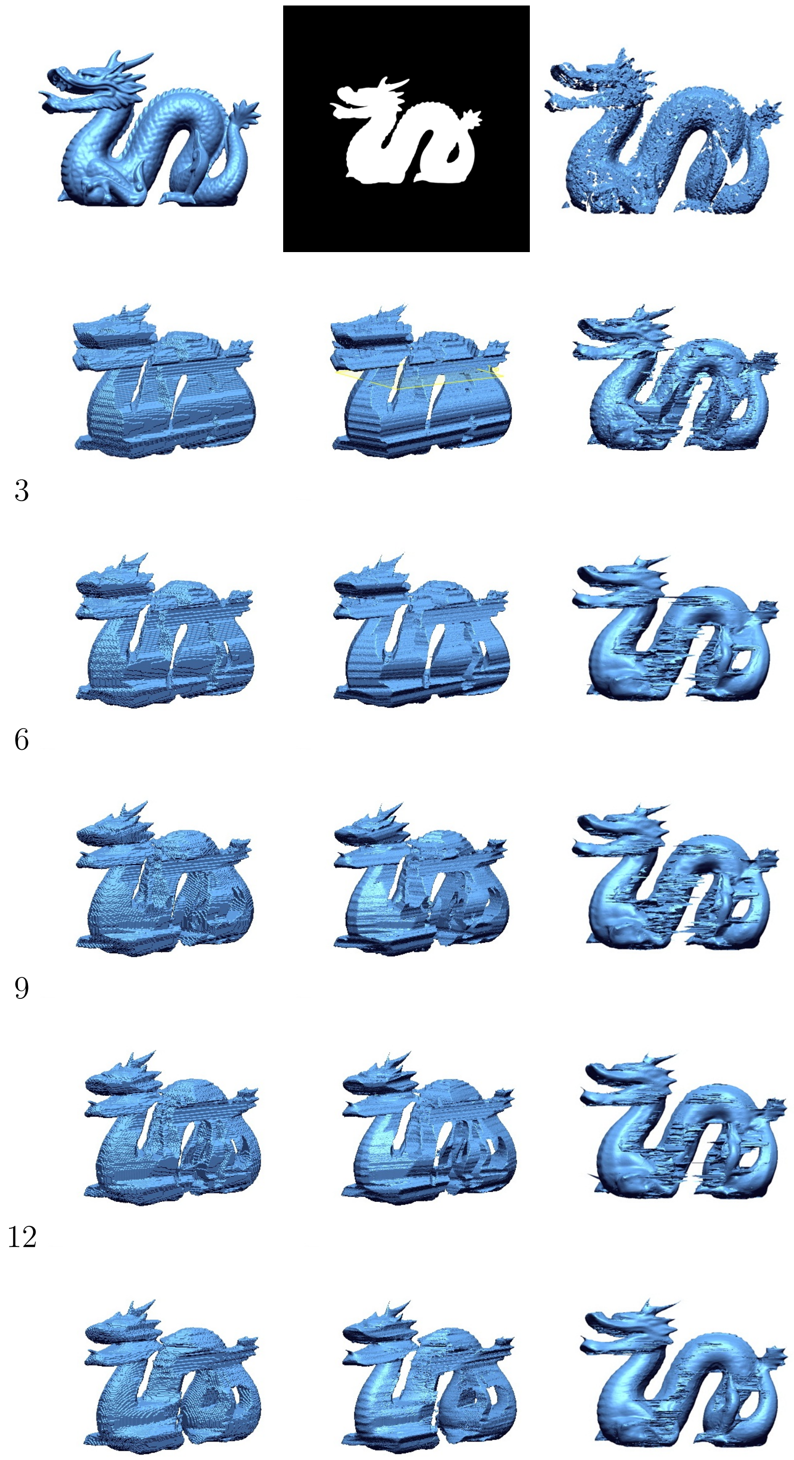


Figure B.8: Dragon

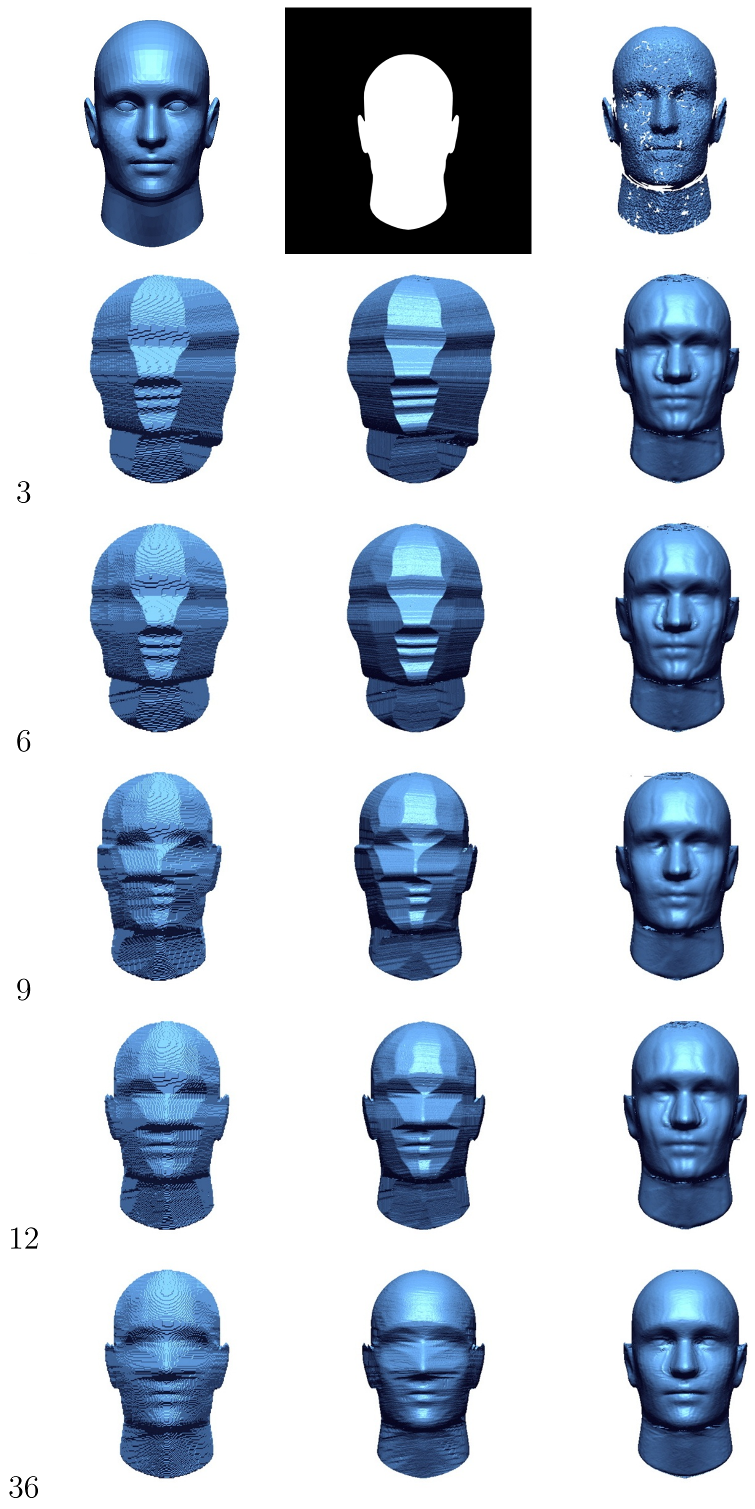


Figure B.9: Head

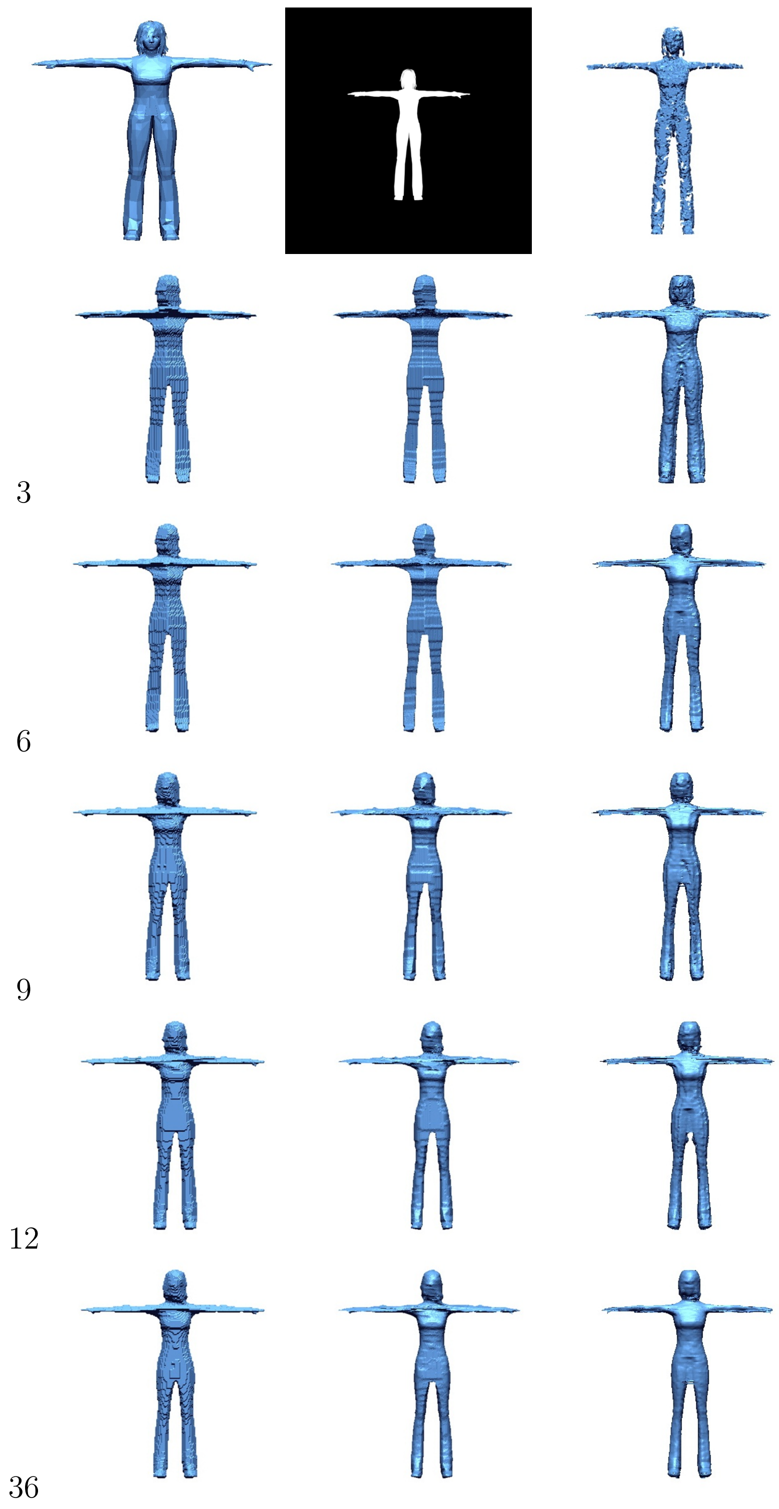


Figure B.10: Human

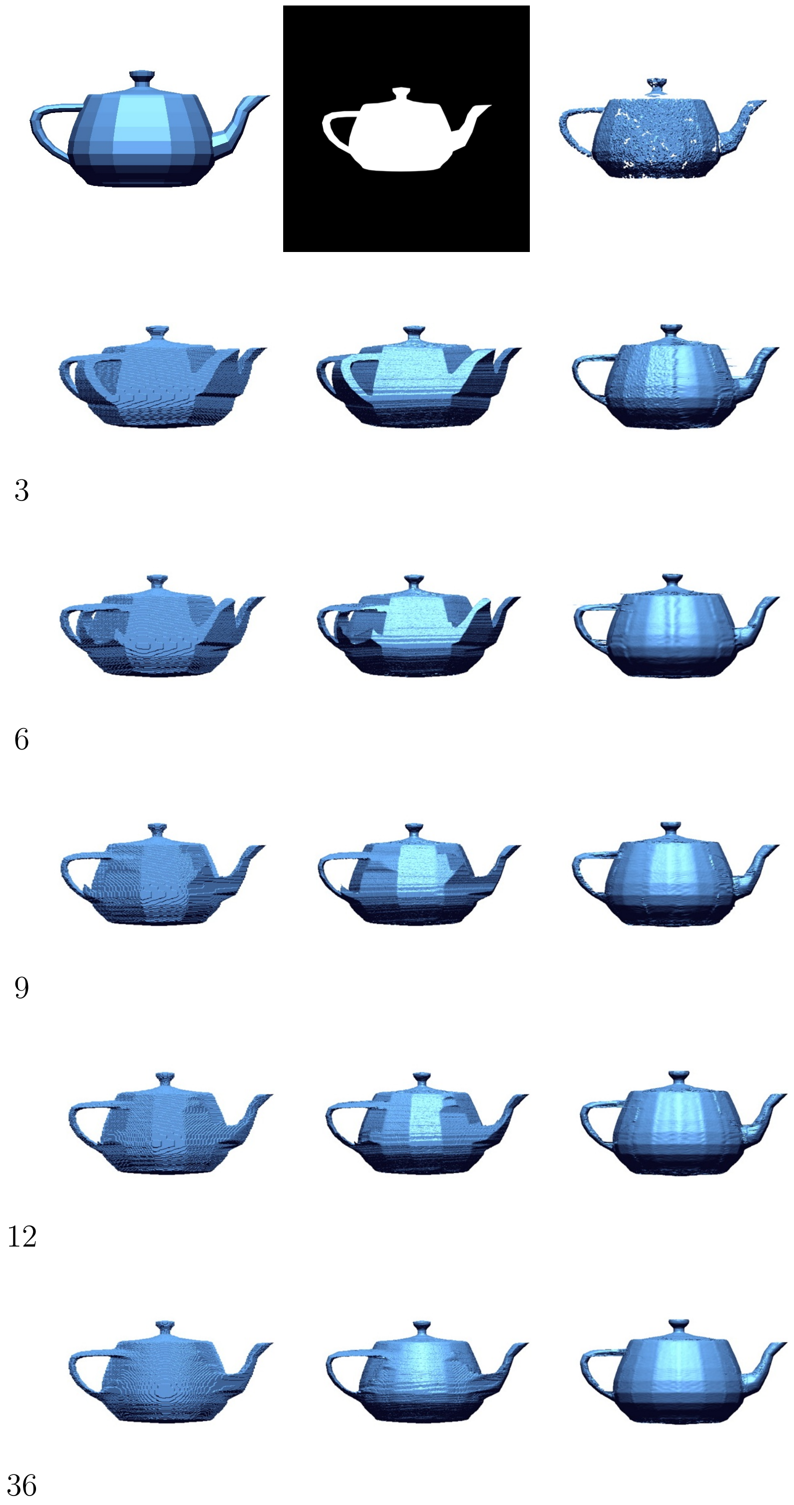


Figure B.11: Teapot

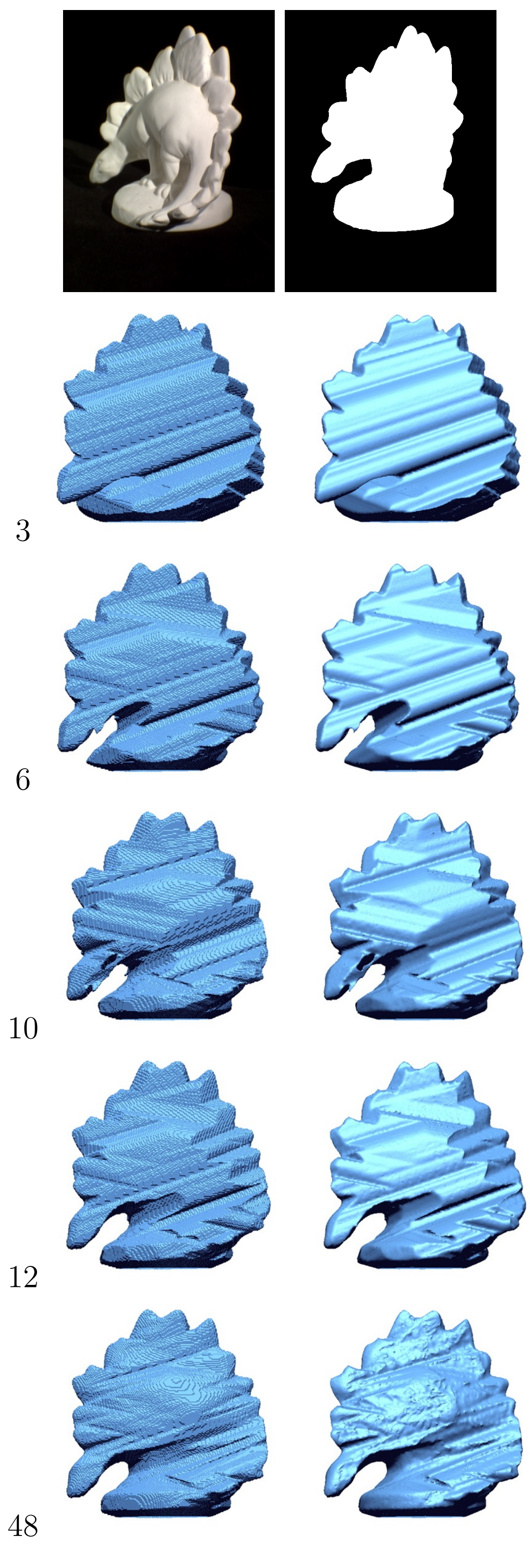


Figure B.12: Dino

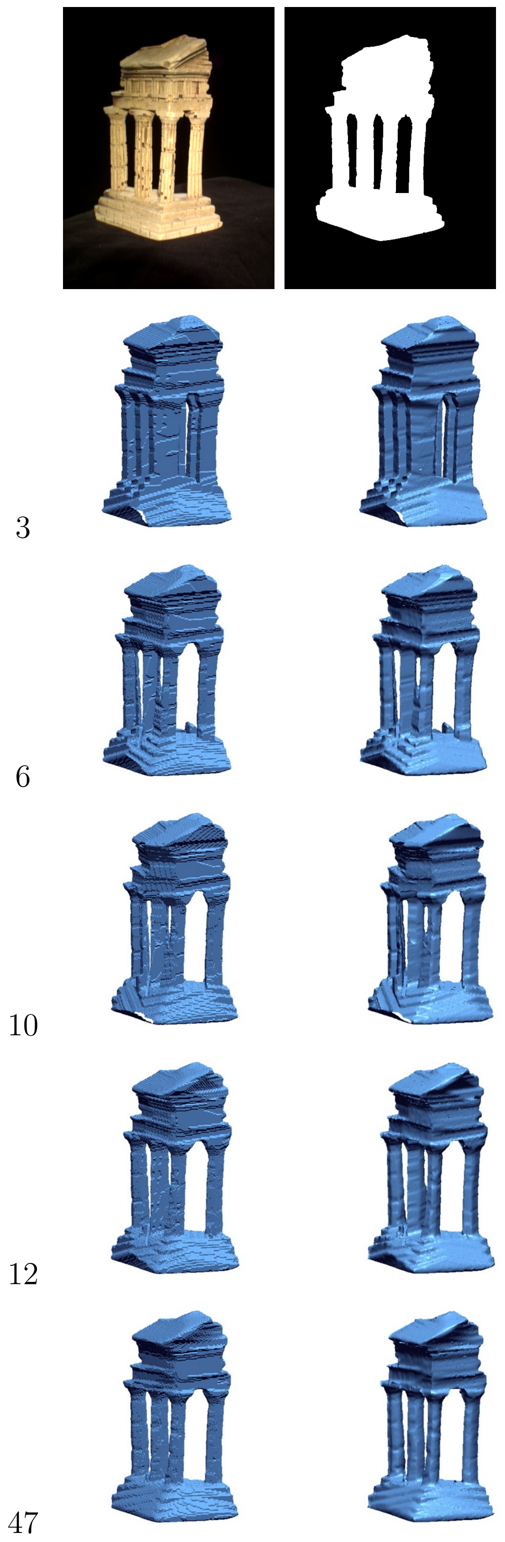


Figure B.13: Temple

Appendix C

Cameras

C.1 Cameras

C.1.1 Orthographic Camera

A simple camera model is the orthographic projection camera model. While in a pinhole camera model all the rays intersect at the camera centre, the rays of an orthographic projection model are all parallel. The advantage of this is that the 3D to 2D relationship is linear and so the mathematical relationship is simpler.

C.1.2 Pin-hole Camera

The pinhole camera model (Figure C.1) describes the relationship between the coordinates of a 3D point in space and its projection onto the image plane [43]. In mathematics the pinhole camera model can be represented by the camera matrix (projection matrix). This is a 3 $\times$ 4 matrix which can be used to calculate the 2D homogeneous pixel coordinate projected position of a 3D homogeneous point coordinate.

The camera matrix is made up of a number of components. These components can be broken into two categorises; intrinsic camera parameters and extrinsic camera parameters. The intrinsic camera parameters encompass the focal length (f), image format (m_x, m_y, γ) and principle point (u_0, v_0) of the inner workings of the camera.

The extrinsic camera parameters define the position and rotation of the camera within a world coordinate system. A 3 $\times$ 3 rotation matrix $\mathbf{R}$ rotates the camera into

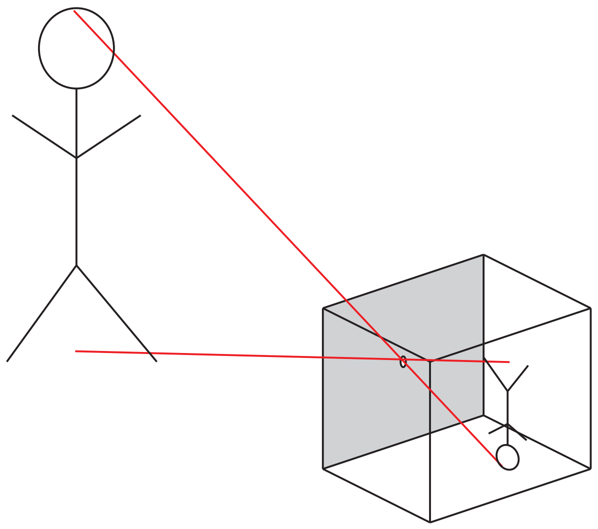


Figure C.1: The pinhole camera model

the correct direction and a 1×3 translation vector $\mathbf{T}$ moves the camera centre into the correct location.

The relationship can then be defined between the 3D homogeneous point $(X\ Y\ Z\ 1)^T$ and the 2D homogeneous pixel coordinate $(u\ v\ 1)^T$ as:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f \times m_x & \gamma & u_0 \\ 0 & f \times m_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{C.1}$$

C.1.3 Depth Camera

There are a number of different types of depth cameras or more commonly called 3D scanners. The most common 3D scanner technology would be time-of-flight which probes the scene using laser light and measures the round-trip time of a pulse of light to calculate the distance between the object and the camera. These are generally very expensive pieces of equipment. What is becoming more common is the depth sensors that calculate depth using structured light. The most common example of this is the Microsoft Kinect for the XBox 360 (Appendix A gives more detailed information

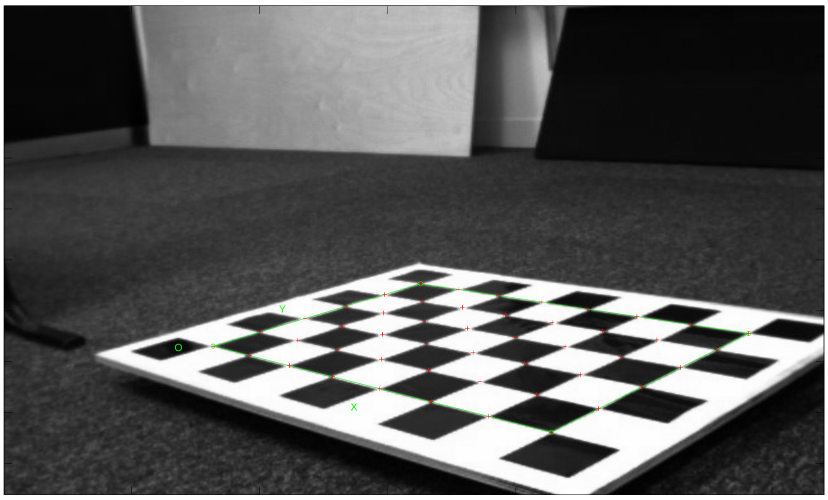


Figure C.2: The corners (red) of a standard checkerboard pattern can be used to calibrate a standard camera.

on the Kinect sensor). Structured light is a technique whereby a regular pattern is projected onto a scene and the observed deformation of that pattern is used to calculate the distance. Depth cameras can also be emulated in software using two views close together and using stereo matching where each point in one image is matched to a point in the second image. Using the camera geometry a distance can then be calculated. In all these examples noise is generally a problem and has to be accounted for in any method that attempts to use such depth information.

C.1.4 Camera Calibration

The procedure for determining the intrinsic and extrinsic camera parameters is called camera calibration. Many methods are available for determining the camera parameters [43]. The choice of the method generally depends on the prior information available.

A common approach is to capture several images of a known geometric pattern such as a planar checkerboard (Figure C.2). The knowledge of the relative positions of the corners of the checkerboard in the real world and the corresponding coordinates of the projection of those corners in the image plane can be used to solve for the camera parameters. Since the projection matrix contains 12 entries and 11 degrees of freedom, a minimum of five and a half 3D to 2D projections are needed, since each projection contains two relationships x and y. To overcome problems due to noise, normally many more points are used and an optimisation process is used to solve for the unknown parameters. Coefficients for radial distortion can also be calculated using this method.

For a multi-camera system, common points should be used to guarantee that all the cameras have a common world origin and are correctly positioned relative to each other. A MATLAB toolbox has been developed by Bouguet and has been used for all the camera calibration in this work [9].

Appendix D

Data-sets

An important aspect in 3D reconstruction and in human motion capture is the input data used and the methods of validating the results. There are a number of data sets published with ground truths and test metrics which anyone can use.

The Human Eva data set contains two configurations [6]. Human Eva I consists of a seven camera system with three colour cameras and four grey scale cameras. These cameras were software synchronised. Human Eva II improved this configuration with four colour cameras which were hardware synchronised. An optical marker-based motion capture was obtained with a ViconPeak MoCap commercial system. This means that a skeleton ground truth is provided.

This dataset was used in some indirectly related work which is covered in appendix E. Other datasets such as the Amsterdam Library of Object Images (ALOI) [37] and the i3Post [38] were considered but not used.

The Middlebury multi-view stereo dataset by Seitz et al. [90] represents one of the most used data sets to quantitativly compare and evaluate 3D reconstruction algorithms. The data set was used to compare six methods when the data set was originally published in 2006. Over the last four years over 40 more methods have used the data set and published the evaluation results on the web-site [1]. The data set consists of 2 objects each with 3 camera configurations of 16, 47/48 and 312/363 camera views of a static frame. This dataset was the one mostly used in this work with the algorithm explained in Section 3.2 and the results displayed in appendix B.

[1]http://vision.middlebury.edu/mview/

Appendix E

Estimating 3D Scene Flow from Multiple 2D Optical Flows

Scene flow is the motion of the surface points in the 3D world. For a camera, it is seen as a 2D optical flow in the image plane. Knowing the scene flow can be very useful as it gives an idea of the surface geometry of the objects in the scene and how those objects are moving. Four methods for calculating the scene flow given multiple optical flows have been explored and detailed in this appendix along with the basic mathematics surrounding multi-view geometry. It was found that given multiple optical flows it is possible to estimate the scene flow to different levels of detail depending on the level of prior information present.

E.1 Introduction

Optical flow is the 2D projection of the 3D scene flow onto the image plane. Optical flow has been widely described over the last three decades [64], [46] and in that time the methods and techniques for calculating optical flow have improved greatly [8], [3]. It has been shown that it is possible to back-project the optical flow to compute the scene flow [107].

Knowledge of scene flow can help with a number of applications including human motion capture [101] and can also help in determining the structure of the scene [110]. Combining multiple optical flows in a single scene flow could possibly detect and correct error in individual optical flows. This corrected optical flow can then be used for better video compression.

Traditionally calculating scene flow is done by matching pixels across two views of the same scene. This allows the depth information to be calculated and the two optical flows to be combined to generate the scene flow.

Vedula et al. presents three different scenarios for calculating scene flow [107]. The first scenario occurs when there is complete knowledge of the surface of the object in the scene. In this case, Vedula shows that it is possible to calculate scene flow from a single camera. The second scenario occurs when there is knowledge of corresponding information between two image views and in this case it is possible to determine the scene flow from these two views. The third scenario is when there is no knowledge of the scene. In this situation, Vedula shows that, by using a large number of cameras, possible scene flows can be generated and the results can be thresholded to reveal the moving objects in the scene.

Tian and Shah present a method for modelling the translational and rotational motion of the objects in the scene and the scene flow is estimated by using a 5D histogram [102]. Torr et al. calculate matching feature points in each image to calculate the scene flow for the matched points and the scene structure [105].

This appendix summarises some of the basic projection mathematics and multi-view geometry as presented in [43] and customises it to optical flow and scene flow situations. It is shown how it is possible, given a surface point in a scene and two camera views of that point, to calculate the scene flow. Four methods for combining multiple optical flow images to generate a 3D scene flow are presented. The first method uses background subtraction to determine the surface points of the objects in space and then uses projection matrix to calculate the scene flow for each surface point given the corresponding optical flow from each pixel. The second method estimates the background subtraction by thresholding the optical flow, under the assumption that only the moving objects in the scene are of interest. The third method performs the thresholding on the scene flow instead, again assuming that the areas of largest scene flow are the areas of interest. The last method defines a 6D histogram space where all possible point scene flows are considered. This histogram is searched to reveal the point scene flows where there is the most concurrence between the multiple views.

E.2 Notation

All calculations assume knowledge of all extrinsic and intrinsic camera parameters. This includes the camera projection matrix P_i and the camera centre $\mathbf{c}_i$ for each camera i.

$$\mathrm{P}_i = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \tag{E.1}$$

$$\mathbf{c}_i = \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} \tag{E.2}$$

Homogeneous coordinates are used for all 2D pixels $\mathbf{u}_i^t$ in the image plane of camera i at time t and all 3D points $\mathbf{x}^t$ in world space at time t.

$$\mathbf{u}_i^t = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{E.3}$$

$$\mathbf{x}^t = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{E.4}$$

Optical flow $\dot{\mathbf{u}}_i^t$ is the movement of a pixel over a time interval from some previous time t' to the current time t where the interval $|t - t'|$ is assumed to be of short duration.

$$\dot{\mathbf{u}}_i^t = \frac{\mathbf{u}_i^t - \mathbf{u}_i^{t'}}{t - t'} = \begin{pmatrix} \dot{u} \\ \dot{v} \\ 0 \end{pmatrix} \tag{E.5}$$

Similarly the scene flow $\dot{\mathbf{x}}^t$ is:

$$\dot{\mathbf{x}}^t = \frac{\mathbf{x}^t - \mathbf{x}^{t'}}{t - t'} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ 0 \end{pmatrix} \tag{E.6}$$

The time is usually the frame index in the video stream and the optical flow is computed with successive frames $t - t' = 1$.

E.2.1 Determining a 2D Pixel given a 3D Point

A 3D point $\mathbf{x}^t$ in world space can be projected onto a 2D pixel $\mathbf{u}_i^t$ in the image plane by multiplication with the projection matrix P_i.

$$\mathbf{u}_i^t = \mathrm{P}_i \; \mathbf{x}^t \tag{E.7}$$

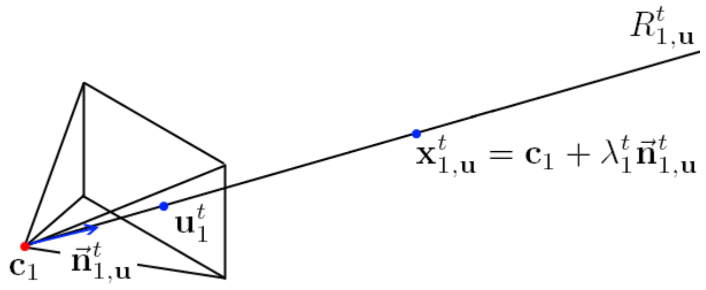
E.2.2 Determining a 3D Ray given a 2D Pixel

Given a single 2D pixel $\mathbf{u}_i^t$, it is not possible to calculate a single 3D point $\mathbf{x}_{i,\mathbf{u}}^t$ without depth information λ_i^t. It is possible to calculate a 3D ray of points where any point on the ray will be projected onto the given 2D pixel $\mathbf{u}_i^t$. Multiplying the pseudo-inverse of the projection matrix P_i^+ (defined by $\mathrm{P}^+ = \mathrm{P}^\top \left(\mathrm{P}\mathrm{P}^\top\right)^{-1}$) by the homogenous pixel coordinate will results in some point $\mathbf{x}_{i,\mathbf{u}}^t$ on the ray in the scene in front of the camera.

$$\mathbf{x}_{i,\mathbf{u}} = \mathrm{P}_i^+ \; \mathbf{u}_i \tag{E.8}$$

Therefore a ray starting at the camera centre $\mathbf{c}_i$ and continuing on through this new point $\mathbf{x}_{i,\mathbf{u}}$ describes all possible points that could be projected onto the pixel $\mathbf{u}_i^t$. A more convenient way to describe this ray is Equation E.10, where $\tilde{\mathbf{n}}_{i,\mathbf{u}}^t$ is the unit vector from $\mathbf{c}_i$ to $\mathbf{x}_{i,\mathbf{u}}$ and λ is some positive number which represents the distance between the camera centre and the point that is being searched for. As seen in Figure E.1.

$$\tilde{\mathbf{n}}_{i,\mathbf{u}}^t = \frac{\mathbf{x}_{i,\mathbf{u}} - \mathbf{c}_i}{\|\mathbf{x}_{i,\mathbf{u}} - \mathbf{c}_i\|} \tag{E.9}$$



$R^t_{1,\mathbf{u}}$
$\mathbf{x}^t_{1,\mathbf{u}} = \mathbf{c}_1 + \lambda^t_1 \vec{\mathbf{n}}^t_{1,\mathbf{u}}$
$\mathbf{u}^t_1$
$\mathbf{c}_1$
$\vec{\mathbf{n}}^t_{1,\mathbf{u}}$

Figure E.1: In camera 1, a pixel $\mathbf{u}^t_1$ and a camera centre $\mathbf{c}_1$ are used to calculate a 3D ray of all possible points $\mathbf{x}^t_{1,\mathbf{u}}$ in the 3D world that project onto pixel $\mathbf{u}^t_1$.

$$\mathbf{x}^t_{i,u} = \mathbf{c}_i + \lambda^t_i \; \tilde{\mathbf{n}}^t_{i,\mathbf{u}} \tag{E.10}$$

E.2.3 Determining a 3D Point given two corresponding Pixels

When two cameras are used and a pixel in each image $\mathbf{u}^t_i$ and $\mathbf{u}^t_j$ are the projection of the same point $\mathbf{x}^t$, it is possible to calculate the 3D position of that point. The points position will be the intersection of the two rays $R^t_{i,\mathbf{u}}$ and $R^t_{j,\mathbf{u}}$ formed by the pixels as described in Section E.2.2. Unfortunately due to the quantisation of the size of the pixels and floating-point arithmetic it is quite unlikely that the rays will actually intersect, but they will pass close to each other. Therefore, the midpoint of the two closest points $\hat{\mathbf{x}}^t_{i,\mathbf{u}}$ and $\hat{\mathbf{x}}^t_{j,\mathbf{u}}$ on the two rays can be used to give an approximate solution $\mathbf{x}^t$. If the distance between the two closest points $\hat{\mathbf{x}}^t_{i,\mathbf{u}}$ and $\hat{\mathbf{x}}^t_{j,\mathbf{u}}$ is very large, for example, larger than the size of a pixel, then this may indicate that the two pixels do not refer to the same point and there may be an error in the pixel matching.

E.2.4 Determining a 3D Scene Flow given two corresponding Optical Flows

Similar to the situation in Section E.2.3 two 2D pixels $\mathbf{u}^t_i$ and $\mathbf{u}^t_j$ with optical flows $\dot{\mathbf{u}}^t_i$ and $\dot{\mathbf{u}}^t_j$ both visualising the same 3D point and scene flow can be used to calculate the position of that point $\mathbf{x}^t$ and the direction and magnitude of that scene flow $\dot{\mathbf{x}}^t$.

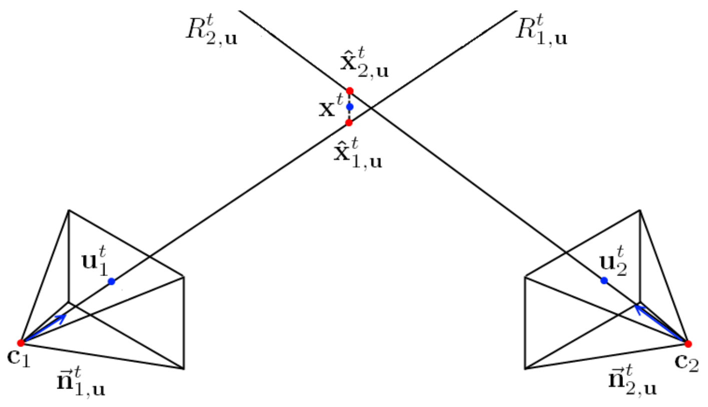

$R^t_{2,\mathbf{u}}$
$R^t_{1,\mathbf{u}}$
$\hat{\mathbf{x}}^t_{2,\mathbf{u}}$
$\mathbf{x}^t$
$\hat{\mathbf{x}}^t_{1,\mathbf{u}}$
$\mathbf{u}^t_1$
$\mathbf{u}^t_2$
$\mathbf{c}_1$
$\mathbf{c}_2$
$\vec{\mathbf{n}}^t_{1,\mathbf{u}}$
$\vec{\mathbf{n}}^t_{2,\mathbf{u}}$

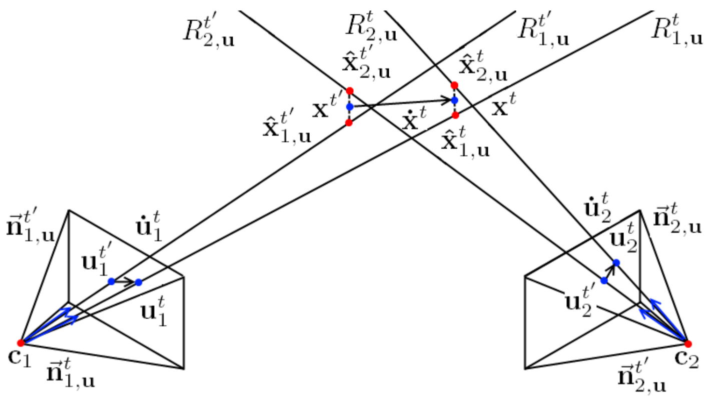
Figure E.2: Two pixels $\mathbf{u}^t_i$ and $\mathbf{u}^t_j$ used to calculate point $\mathbf{x}^t$ by intersection the 2 rays $R^t_{i,\mathbf{u}}$ and $R^t_{j,\mathbf{u}}$. Where camera $i = 1$ and camera $j = 2$.

As shown in Figure E.3 the pixels can be converted into rays $R^t_{i,\mathbf{u}}$ and $R^t_{j,\mathbf{u}}$ and the midpoint of the two closest points reveal the position of the 3D point $\mathbf{x}^t$. The optical flow can be used to calculate the last position of each of the pixels in the previous frame $\mathbf{u}^{t'}_i$ and $\mathbf{u}^{t'}_j$, by subtracting the optical flow from the current position. These new pixels can be converted into rays $R^{t'}_{i,\mathbf{u}}$ and $R^{t'}_{j,\mathbf{u}}$ and the previous 3D point $\mathbf{x}^{t'}$ can be calculated in the same way as in Section E.2.3. The vector from this previous point $\mathbf{x}^{t'}$ to the current point $\mathbf{x}^t$ is the scene flow $\dot{\mathbf{x}}^t$.

E.2.5 Determining a range of possible points and scene flows from a single pixel

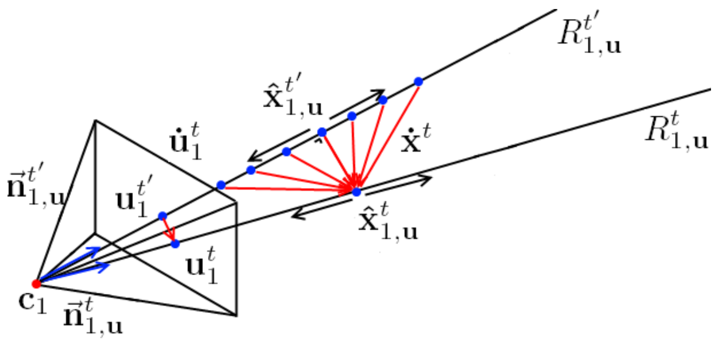
Given a single pixel $\mathbf{u}^t_i$ with optical flow $\dot{\mathbf{u}}^t_i$ it is possible to calculate a range of possible points $\mathbf{x}^t$ and scene flows $\dot{\mathbf{x}}^t$ which would project onto that pixel and optical flow. Using the optical flow it is possible to calculate the position of the pixel in the previous frame $\mathbf{u}^{t'}_i = \mathbf{u}^t_i - \dot{\mathbf{u}}^t_i$. The 3D rays $R^t_{i,\mathbf{u}}$ and $R^{t'}_{i,\mathbf{u}}$ can be calculated for both the current and previous pixel using the technique described in Section E.2.2.

The two rays can be seen in Figure E.4, basically any vector that connects $R^{t'}_{i,\mathbf{u}}$ to $R^t_{i,\mathbf{u}}$ is a possible scene flow. The Equation E.10 gives you a range of possible points $\mathbf{x}^t = \mathbf{c}_i + \lambda^t_i\, \tilde{\mathbf{n}}^t_{i,\mathbf{u}}$ by exploring all valid λ^t_i. For each point $\mathbf{x}^t$ on $R^t_{i,\mathbf{u}}$ a range of possible previous points $\mathbf{x}^{t'}$ can be found the same way on $R^{t'}_{i,\mathbf{u}}$. The time between frames will be very small, therefore it is possible to use this prior knowledge to place a maximum



$R^{t'}_{2,\mathbf{u}}$
$R^{t}_{2,\mathbf{u}}$
$R^{t'}_{1,\mathbf{u}}$
$R^{t}_{1,\mathbf{u}}$
$\hat{\mathbf{x}}^{t'}_{2,\mathbf{u}}$
$\hat{\mathbf{x}}^{t}_{2,\mathbf{u}}$
$\mathbf{x}^{t'}$
$\mathbf{x}^{t}$
$\hat{\mathbf{x}}^{t'}_{1,\mathbf{u}}$
$\dot{\mathbf{x}}^{t}$
$\hat{\mathbf{x}}^{t}_{1,\mathbf{u}}$
$\vec{\mathbf{n}}^{t'}_{1,\mathbf{u}}$
$\dot{\mathbf{u}}^{t}_{1}$
$\mathbf{u}^{t'}_{1}$
$\mathbf{u}^{t}_{1}$
$\mathbf{c}_1$
$\vec{\mathbf{n}}^{t}_{1,\mathbf{u}}$
$\dot{\mathbf{u}}^{t}_{2}$
$\vec{\mathbf{n}}^{t}_{2,\mathbf{u}}$
$\mathbf{u}^{t}_{2}$
$\mathbf{u}^{t'}_{2}$
$\mathbf{c}_2$
$\vec{\mathbf{n}}^{t'}_{2,\mathbf{u}}$

Figure E.3: Two optical flows used to calculate the scene flow of a point using two sets of intersection rays. Where camera $i = 1$ and camera $j = 2$.



$R^{t'}_{1,\mathbf{u}}$
$\hat{\mathbf{x}}^{t'}_{1,\mathbf{u}}$
$R^{t}_{1,\mathbf{u}}$
$\dot{\mathbf{u}}^{t}_{1}$
$\dot{\mathbf{x}}^{t}$
$\vec{\mathbf{n}}^{t'}_{1,\mathbf{u}}$
$\mathbf{u}^{t'}_{1}$
$\hat{\mathbf{x}}^{t}_{1,\mathbf{u}}$
$\mathbf{u}^{t}_{1}$
$\mathbf{c}_1$
$\vec{\mathbf{n}}^{t}_{1,\mathbf{u}}$

Figure E.4: A range of possible scene flows $\dot{\mathbf{x}}^t$ given one optical flow $\dot{\mathbf{u}}^t_i$ after picking one possible point $\mathbf{x}^t$. Where camera $i = 1$.

magnitude on the scene flow. It is possible to calculate the closest point $\hat{\mathbf{x}}^{t'}_{i,\mathbf{u}}$ on the previous ray $R^{t'}_{i,\mathbf{u}}$ to the currently selected point $\mathbf{x}^t$ on $R^t_{i,\mathbf{u}}$, then possible previous points $\mathbf{x}^{t'} = \hat{\mathbf{x}}^{t'}_{i,\mathbf{u}} + \lambda^{t'}_i \ \tilde{\mathbf{n}}^{t'}_{i,\mathbf{u}}$ by varying $\lambda^{t'}_i$ between plus and minus the maximum magnitude of the scene flow.

E.3 Calculating 3D Scene Flow

Previously, to calculate scene flow required knowledge of corresponding pixels. This in general requires pixel matching. The following four algorithms try to avoid this costly approach and use alternative methods by either finding surface points which can be projected into the cameras to find corresponding pixels or by overlapping all possible scene flows to find areas of concurrence.

E.3.1 Background Subtraction

Background subtraction is performed on each of the images to determine which pixels belong to foreground objects and which pixels belong to the background. A dense selection of equally distributed points is generated in the 3D world to be explored. Each point is projected into each of the cameras using Equation E.7. If all the pixels relating to a point belong to a foreground object then the point is kept, otherwise the point is discarded. The remaining points form the solid shape of the object in the scene. The more cameras and the more points used the better the shape will be in approximating the object, but the more computationally expensive it will be.

To extract the surface points or visual hull, the remaining points are examined and inside points are discarded and surface points are kept. This is done by stepping through all the points and each time a border is crossed the point is kept otherwise it is discarded. At this point it is also noted which relative side of the object the point is on; this allows the relative cameras to be selected in determining the scene flow. The remaining points can be projected into the appropriate cameras using the method described in Section E.2.1 and then the method described in Section E.2.3 can be used to determine the scene flow at that point.

E.3.2 Thresholding Optical Flow

Assuming that the only thing moving in the scene is the object then the background subtraction can be approximated by thresholding the optical flow results. The same method described in Section E.3.1 can be used but thresholded optical flow is used instead of the background subtraction results. This solution suffers from problems when the scene flow is in-line with one of the cameras. When this is the case then very little optical flow is seen in that camera and so it falls below the threshold, and therefore disappears in the final estimation of the scene flow.

E.3.3 Thresholding Scene Flow

This time the scene flow is calculated for all the points in the scene. The magnitude of each scene flow is measured and the ones below a threshold are discarded.

E.3.4 Histogram Data

In this case a 6D histogram is created with the first three dimensions as world position x y z, and the last three dimensions as scene flow components $\dot{x}$ $\dot{y}$, $\dot{z}$. For every pixel in every camera the method described in Section E.2.4 is used to generate possible scene flows for every scene flow explored and the relevant entry in the histogram is incremented.

The histogram is reduced to a 3D histogram by taking the maximum scene flow for each point; this is because no single point can have more than one scene flow. This new histogram is then thresholded reducing the points to the ones with the highest values. These should be the surface points of the object as that is where the maximum concurrence should be.

E.4 Results

All four methods where implemented on the Human Eva II dataset [6]. Black et al, [5] optical flow algorithm was used to generate the optical flow for each of the four cameras. The two frames for the four cameras can be seen in Figure E.5, this includes the result of a background subtraction and the thresholded optical flow. The results of the four methods can be seen in Figure E.6.

E.4.1 Background Subtraction

The scene flow derived using background subtraction is shown in Figure E.6 (a). This can be visually inspected and it appears to be very accurate when compared with the two frames of the scene. This method does require a framework which calculates the background subtraction. While a lot of human motion tracking algorithms use some form of background subtraction, it is not always possible to have a robust background subtraction. So while the result is very good it has its limitations in terms of the required prior knowledge of the background of the scene. Figure E.7 shows the difference between the original optical flow and the projection of the new scene flow in camera four. Most of the difference is very small, but there are bands and patches of large difference. It is unknown if this difference is due to error in the scene flow or in the original optical flow which has been corrected or a combination of the two.

E.4.2 Thresholding Optical Flow

The results of the thresholding of the optical flow can be seen in Figure E.5. When compared with the background subtraction a good detection of objects is observed but the edges are not as clean and some parts of the body disappear. The scene flow derived from this method can be seen in Figure E.6 (b). Due to the error in segmentation the resulting scene flow is not as good as when using background subtraction but the general outline of the body is visible. The motion of the head in camera 3 is in-line with the camera, therefore, the head disappears in the thresholded optical flow. This therefore causes the head to disappear in the scene flow. The advantage to this method is that no prior information about the scene is required. In theory using some better adaptive thresholding techniques and building information up over time could produce better results.

E.4.3 Thresholding Scene Flow

The scene flow results obtained by thresholding the scene flow can be seen in Figure E.6 (c). Again this is not as good as the scene flow derived from the background subtraction, but the outline of the person is visible. This method solves the problem of the previous method of when the motion is in-line with a camera, but it still suffers where the body is not moving. Again over time this could be filled in as knowledge of the scene builds up.

E.4.4 Histogram Data

The scene flow results obtained by using a histogram of all possible scene flows can be seen in Figure E.6 (d). This results is also not as good as the scene flow derived

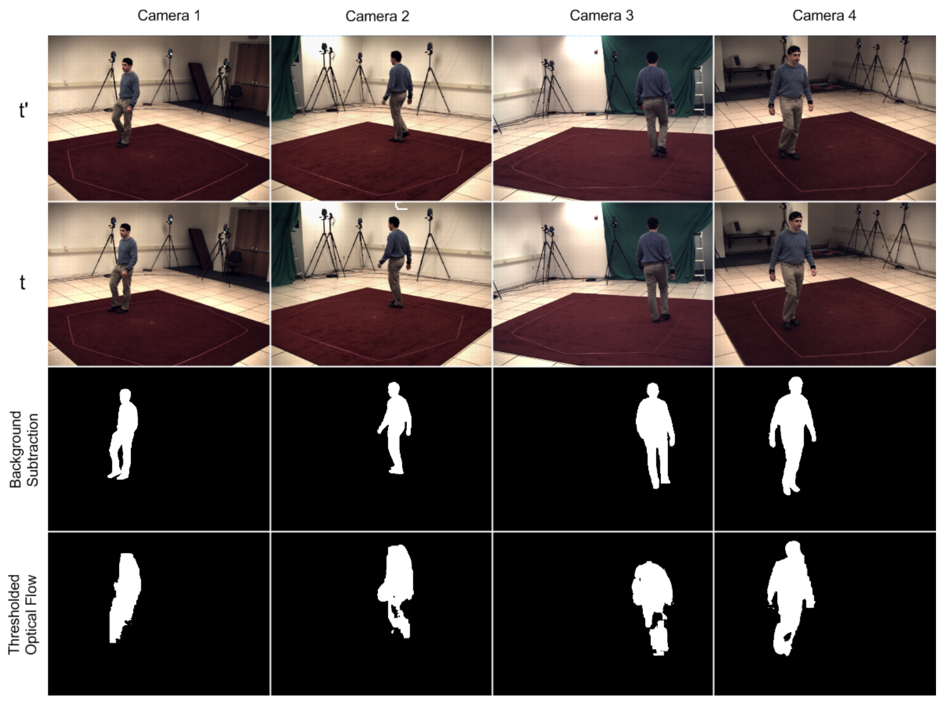


Figure E.5: Two consecutive frames of the Human Eva II dataset consisting of 4 cameras also the result of the background subtraction and the thresholding of the optical flow.

from the background subtraction, but again the outline of the person can be seen. This method assumes no prior knowledge of the scene. This method can be computationally intensive due to the 6D histogram, and the large number of scene flows required. To get good results requires fine tuning to pick the correct number and sizes of each of the dimensions of the histogram. Also the resulting scene flow is quantised and limited to the size of the dimensions of the histogram.

One overall disadvantage of using the Human Eva II dataset was the position of the cameras; four cameras each at a corner of a room. This arrangement means that a surface point is only ever visible from two cameras which only allows one estimation of the scene flow. With more cameras it might be possible to average the scene flow over multiple estimations and achieve a better result.

E.5 Conclusions and Future Work

In this appendix, four methods for determining the scene flow given multiple optical flows have been presented. From visual inspection of the results it can be seen that the background subtraction method performed the best, but the general motion can be seen using the other methods.

Further work is required to quantify the accuracy of the scene flow. This can be done using the motion capture data supplied with the Human Eva II dataset. Further work is required in experimenting with more cameras to have a surface point visible to more then just two cameras. Further work is also required to see how both colour information and temporal information can improve the result.

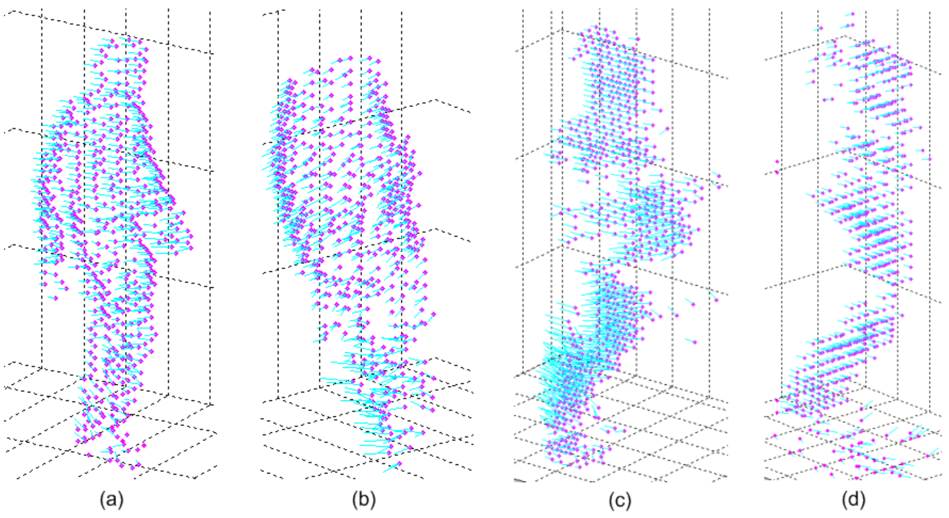


Figure E.6: Results of the 4 scene flow algorithms. (a) using the background subtraction, (b) thresholding the optical flow, (c) thresholding the scene flow and (d) exploring the histogram data. The computation time for each scenario is (a) 1.8 seconds (b) 1.4 seconds (c) 7.9 seconds and (d) 117 seconds. This does not include the time taken to calculate the optical flow or background subtraction.

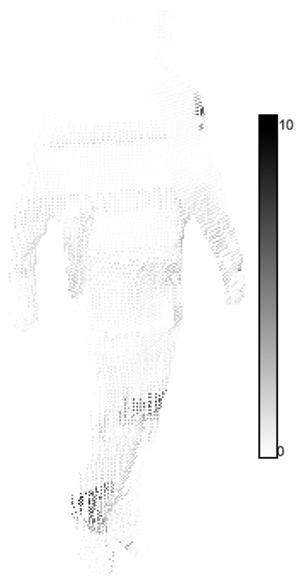


Figure E.7: The scene flow result obtained using the background subtraction is projected back into camera 4, and the magnitude of the difference between the original optical flow and the new optical flow is plotted.

Appendix F

Multi-Sensor Room

This appendix describes a multi-sensor room that was designed and built. This work addresses the challenge of synchronising multiple sources of video streams from a variety of devices, while capturing human motion in realtime and an initial sketch was published [87]. While other such augmented motion capture databases exist [6, 40, 106], the goal of this work is to build on these previous works. Critical areas of improvement are in the synchronisation between cameras and synchronisation between devices. Positioning of the cameras will be varied to give greater flexibility. An example layout of the room can be seen in Figure F.1. The system is designed to facilitate the testing and validation of human pose estimation and motion tracking techniques, among other applications. This appendix briefly describes some of the interesting challenges faced in setting up the pipeline for capturing the synchronised data and the novel approaches proposed to solve them.

F.1 System Overview

Video Capture: Six cameras (Point Grey Research Flea 2 FL2–03S2C with Fujinon Lens–15F3–60C), each 24-bit colour, capable of running at 80 frames per second (fps) with a 640 × 480 resolution, were used. An important consideration with the cameras is the volume of data being produced. Six cameras running at 60 fps produce 2.67 Gbps of data which has to be saved to hard disk. Each camera requires calibration to determine its intrinsic and extrinsic parameters. This is done using the standard

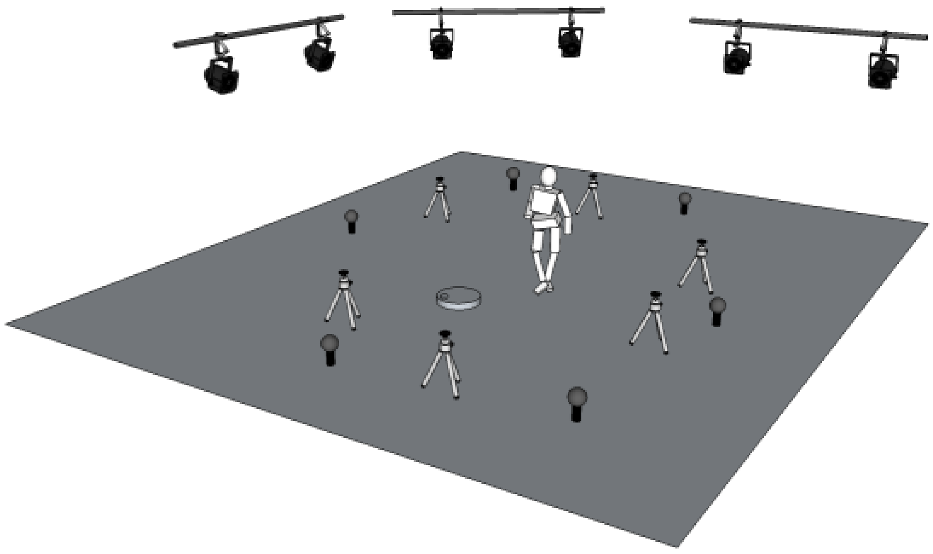


Figure F.1: Room layout with cameras, motion capture cameras and calibration tools.

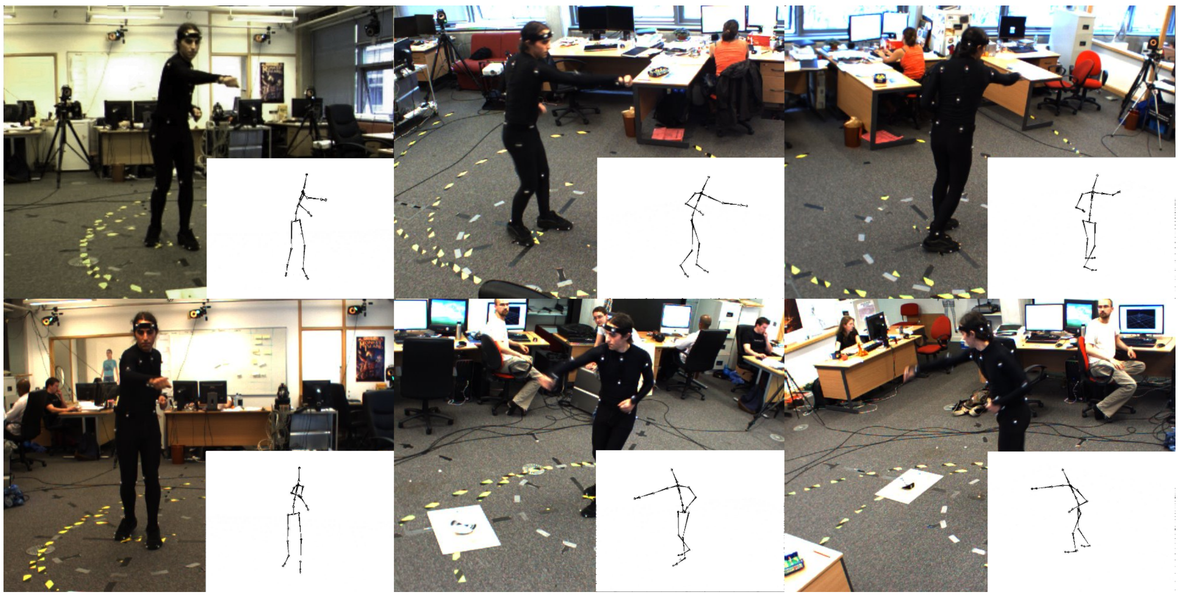


Figure F.2: An example capture showing the six images taken simultaneously and also including the motion capture results as would be seen from each camera view.

checker-board calibration grid and the Camera Calibration Toolbox for MATLAB. A unique feature of this acquisition system is that the positions of the cameras can be varied, from equally spaced 360 degrees around the subject, to one sided 180 degrees around the subject, to stereoscopic pairs of cameras. This allows a greater range of techniques to be tested on the repository. An example capture can be seen in Figure F.2.

Motion-Capture: A marker-based motion-capture system (Vicon iQ 2.5) is included which has the capacity of capturing data from 13 infrared cameras running at up to 200 fps. The motion-capture system requires the user to wear reflective markers at specific points tight on the body. This is best achieved by wearing a skin tight black Velcro suit, but ordinary clothes can be worn if they are skin tight and the markers are attachable. The Vicon's proprietary software is able to give a good estimation of the subject's skeletal motion from the relative position of the markers. However, when baggy clothes are worn the relative position of the marker to the bone is lost and the software cannot estimate the skeleton.

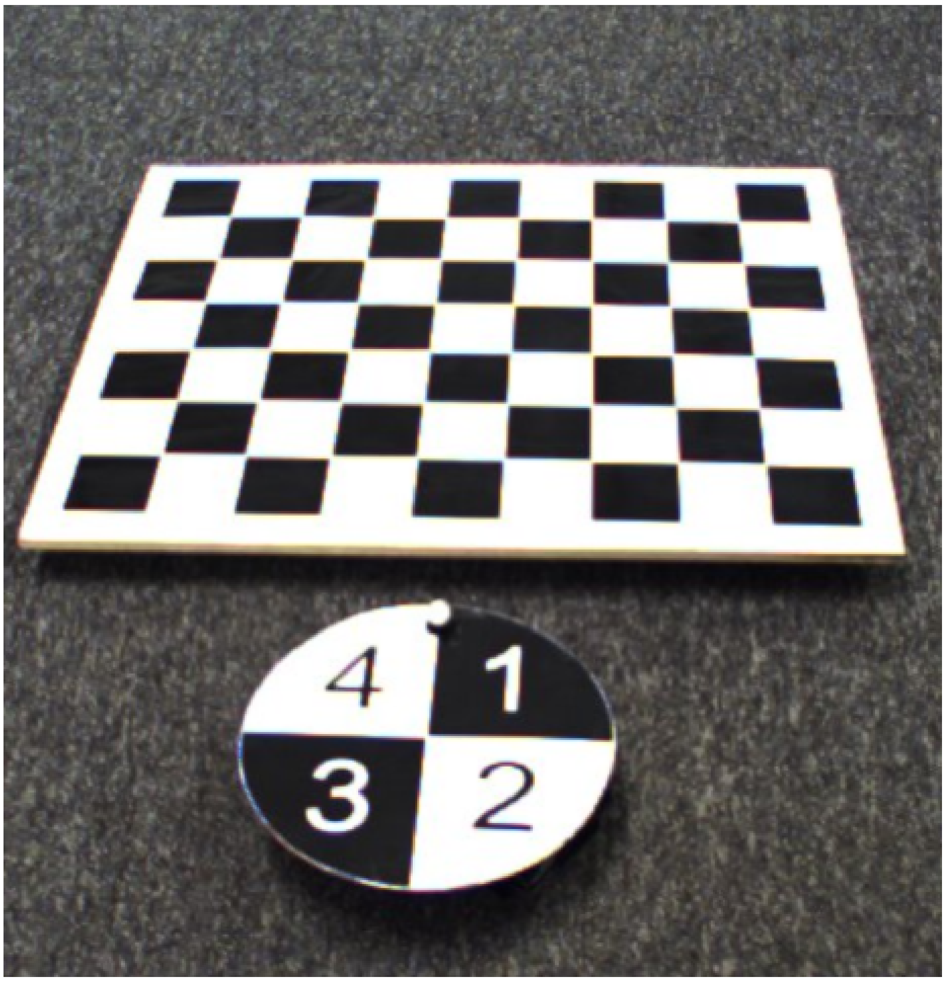
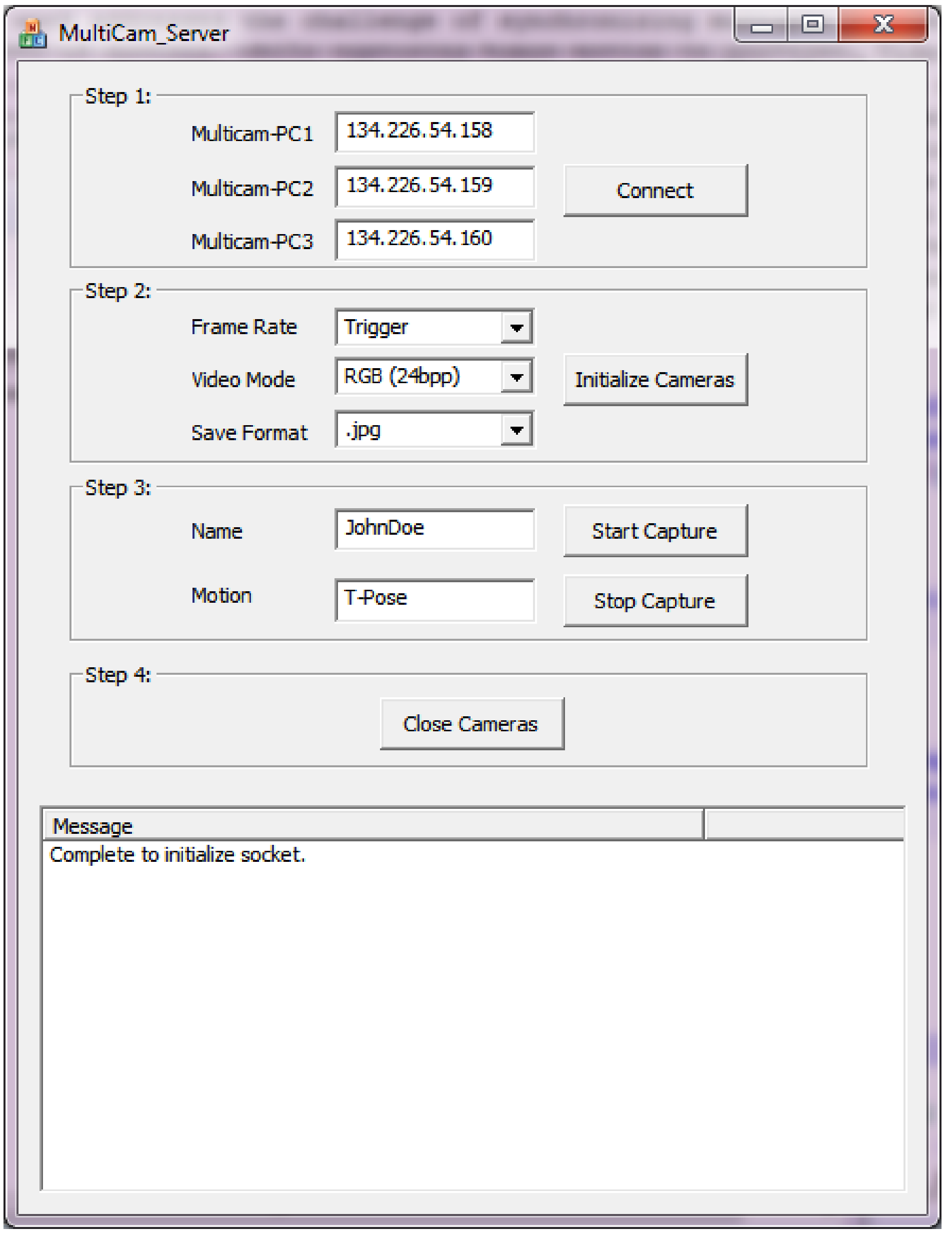


Figure F.3: Calibration tools

Synchronisation: A number of experiments were carried out to determine the best method to synchronise all the cameras and different devices. All the cameras can be synchronised via hardware to achieve best results. To synchronise the cameras to the audio and motion-capture system a device was built, as seen in Figure F.3, consisting of a rotating disk which can be seen by at least one of the video cameras. The disk contains a reflective marker which can be seen by the motion-capture system. This allows accurate synchronisation of each device by aligning the two sources together in a post-process.

Distributed Capture: As noted above the volume of video data captured by the six cameras running at a high frame-rate is considerable. It was not possible with current hardware to save all the data from all the cameras to one hard drive on the one computer. A solution was found by using three computers with two cameras feeding into each computer. A fourth computer or laptop can be used by the user to control the capture of the system. This software was custom developed to control the initialisation of the cameras as well as the individual captures. A screen-shot of the user interface developed can be seen in Figure F.4.


MultiCam_Server

Step 1:

Multicam-PC1: 134.226.54.158
Multicam-PC2: 134.226.54.159
Multicam-PC3: 134.226.54.160

Step 2:

Frame Rate: Trigger
Video Mode: RGB (24bpp)
Save Format: .jpg

Step 3:

Name: JohnDoe
Motion: T-Pose

Step 4:

Message

Complete to initialize socket.

Figure F.4: User interface to control the multi-camera system.

Bibliography

[1] Zeeshan Anwar and Frank Ferrie. Towards robust voxel-coloring: Handling camera calibration errors and partial emptiness of surface voxels. *International Conference on Pattern Recognition*, pages 98–102, 2006.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[3] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computer Surveys*, 27(3):433–466, 1995.

[4] R. Bhotika, D. J. Fleet, and K. N. Kutulakos. A probabilistic theory of occupancy and emptiness. *European Conference on Computer Vision*, pages 112–132, 2002.

[5] Michael J. Black and P. Anandan. Robust dynamic motion estimation over time. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 296–302, 1991.

[6] Michael J. Black and L. Sigal. HumanEva: Synchronized video and motion capture dataset for evaluation of articulated human motion, technical report CS–06–08. http://vision.cs.brown.edu/humaneva/index.html, 2006.

[7] J. S. De Bonet and P. Viola. Roxels: responsibility weighted 3D volume reconstruction. *IEEE International Conference on Computer Vision*, 1:418–425, 1999.

[8] Jean–Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm, 2002.

[9] Jean–Yves Bouguet. Camera calibration toolbox for matlab, 2010.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[11] Edmond Boyer and Jean-Sebastien Franco. A hybrid approach for computing visual hulls of complex objects. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:695–701, 2003.

[12] Derek Bradley, Tamy Boudekeur, and Wolfgang Heidrich. Accurate multi-view reconstruction using robust binocular stereo and surface meshing. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[13] A. Broadhurst, T.W. Drummond, and R. Cipolla. A probabilistic framework for space carving. *IEEE International Conference on Computer Vision*, 1:388–393, 2001.

[14] N. Campbell, G. Vogiatzis, C. Hernandez, and R. Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. *European Conference on Computer Vision*, pages 766–779, 2008.

[15] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:790–799, 1995.

[16] German Cheung, Simon Baker, and Takeo Kanade. Visual hull alignment and refinement across time: a 3D reconstruction algorithm combining shape-from-silhouette with stereo. *IEEE Conference on Computer Vision and Pattern Recognition*, 2:378–382, 2003.

[17] R. T. Collins. Mean-shift blob tracking through scale space. *IEEE Conference on Computer Vision and Pattern Recognition*, 2:234–240, 2003.

[18] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[19] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–575, 2003.

[20] Yan Cui, Sebastian Schuon, Chan Derek, Sebastian Thrun, and Christian Theobalt. 3D shape scanning with a time-of-flight camera. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1173–1180, 2010.

[21] Rozenn Dahyot. Statistical hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1502–1509, 2009.

[22] Rozenn Dahyot. Mean-shift for statistical hough transform. Technical report, Technical report department of Statistics, Trinity College Dublin, April 2009.

[23] Stanley Roderick Deans. Hough transform from the radon transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(2):185–188, 1981.

[24] Stanley Roderick Deans. *The Radon transform and some of its applications.* John Wiley and Sons, 1983.

[25] Koen Denecker, Jeroen Van Overloop, and Frank Sommen. The general quadratic radon transform. *Inverse Problems*, 14(3):615, 1998.

[26] M. Dewan and Gregory D. Hager. Toward Optimal Kernel-based Tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:618–625, 2006.

[27] Charles R. Dyer. Volumetric scene reconstruction from multiple views. *Foundations of Image Understanding*, pages 469–489, 2001.

[28] P. Eisert, E. Steinbach, and B. Girod. Multi-hypothesis, volumetric reconstruction of 3-d objects from multiple calibrated camera views. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3509–3512, 1999.

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

[30] Zhimin Fan, Ming Yang, and Ying Wu. Multiple collaborative kernel tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1268–1273, 2005.

[31] Zhimin Fan, Ming Yang, Ying Wu, Gang Hua, and Ting Yu. Efficient optimal kernel placement for reliable visual tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:658–665, 2006.

[32] J.S. Franco and E. Boyer. Efficient polyhedral modeling from silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(3):414–427, 2009.

[33] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density gradient, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21:32–40, 1975.

[34] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[35] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12:16–22, 2000.

[36] B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimensions: a texture classification example. *IEEE International Conference on Computer Vision*, 1:456–463, 2003.

[37] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The amsterdam library of object images. *International Journal on Computer Vision*, 61(3):103–112, 2005.

[38] N. Gkalelis, H. Kim, A. Hilton, N. Nikolaidis, and I. Pitas. The i3DPost multiview and 3D human action/interaction. *Conference for Visual Media Production*, pages 159–168, 2009.

[39] G. D. Hager, M. Dewan, and C. V. Stewart. Multiple kernel tracking with SSD. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:790–797, 2004.

[40] Miles Hansard, Radu Horaud, Michel Amat, and Seungkyu Lee. Projective alignment of range and parallax data. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3089–3096, 2011.

[41] Kim V. Hansen and Peter Toft. Fast curve estimation using preconditioned generalized radon transform. *IEEE Transactions on Image Processing*, 5:1651–1661, 1996.

[42] C. Harris and M. Stephens. A combined corner and edge detection. *Alvey Vision Conference*, pages 147–151, 1988.

[43] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2004.

[44] C. L. L. Hendricks, M. van Ginkel, P. W. Verbeek, and L. J. van Vliet. The generalized radon transform: Sampling, accuracy and memory considerations. *Pattern Recognition*, 38:2494–2505, 2005.

[45] Vu Hoang Hiep, Renaud Keriven, Patrick Labatut, and Jean-Philippe Pons. Towards high-resolution large-scale multi-view stereo. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1430–1437, 2009.

[46] Berthold K. P. Horn and Brian G. Schunck. *Determining optical flow*, pages 185–203. Artificial Intelligence, 1981.

[47] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard A. Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew J. Davison, and Andrew W. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. *ACM Symposium on User Interface Software and Technology*, 2011.

[48] Michal Jancosek and Tomas Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3121–3128, 2011.

[49] Avinash C. Kak and Malcolm Slaney. *Principles of Computerized Tomographic Imaging.* IEEE Pres, 1988.

[50] Donghoon Kim. *3D Object Reconstruction using Multiple Views.* PhD thesis, Trinity College Dublin, 2011.

[51] Donghoon Kim, Jonathan Ruttle, and Rozenn Dahyot. 3D shape estimation from silhouettes using mean-shift. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1430–1433, 2010.

[52] Edward Kim, Wei Wang, Hongsheng Li, and Xiaolei Huang. A parallel annealing method for automatic color cervigram image segmentation. In *In Medical Image Computing and Computer Assisted Intervention, MICCAI-GRID09 HPC Workshop*, 2009.

[53] Howon Kim and In So Kweon. Appearance-cloning: Photo-consistent scene recovery from multi-view images. *International Journal of Computer Vision*, 66(2):163–192, 2006.

[54] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. *European Conference on Computer Vision*, 3:82–96, 2002.

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

[56] A. Laurentini. How far 3D shapes can be understood from 2D silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):188–195, 1995.

[57] Svetlana Lazebnik, Edmond Boyer, and Jean Ponce. On computing exact visual hulls of solids bounded by smooth surfaces. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:156–161, 2001.

[58] Jianguo Li, Eric Li, Yurong Chen, Lin Xu, and Yimin Zhang. Bundled depth-map merging for multi-view stereo. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2769–2776, 2010.

[59] P. Li and L. Xiao. Mean shift parallel tracking on gpu. *Iberian Conference on Pattern Recognition and Image Analysis*, pages 120–127, 2009.

[60] S. Liu, K. Kang, J.-P. Tarel, and D. B. Cooper. Free-form object reconstruction from silhouettes, occluding edges, and texture edges: A unified and robust operator based on duality. *IEEE Transactions on Pattern Analysis and Machince Intelligence*, 30(1):131–146, 2008.

[61] Yang Liu, Lawrence K Cormack, and Alan Conrad Bovik. Statistical modeling of 3-d natural scenes with application to bayesian stereopsis. *IEEE Transcations on Image Processing*, 20(9):2515–2530, 2011.

[62] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

[63] D. G. Lowe. Object recognition from local scale-invariant features. *IEEE International Conference on Computer Vision*, 2:1150–1157, 1999.

[64] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.

[65] D. Ludwig. The radon transform on euclidean space. *Communications on Pure and Applied Mathematics*, 19:49–81, 1966.

[66] J. Manson, G. Petrova, and S. Schaefer. Straming surface reconstruction using wavelets. *Eurographics Symposium on Geometry Processing*, 27(5), 2008.

[67] W. N. Martin and J. K. Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1983.

[68] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximallyy stable extremal regions. *British Machine Vision Conference*, pages 384–396, 2002.

[69] Wojciech Matusik, Chris Buehler, and Leonard McMillan. Polyhedral visual hulls for real-time rendering. *Eurographics Workshop on Rendering Techniques*, pages 115–126, 2001.

[70] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. *European Conference on Computer Vision*, pages 128–142, 2002.

[71] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[72] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. *IEEE International Syposium on mixed and augmented reality*, 2011.

[73] David Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756 –770, june 2004.

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

[75] Emanuel Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[76] Pascal Paysan, Reinhard Knothe, and Brian Amberg. A 3D face model for pose and illumination invariant face recognition. *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 296–301, 2009.

[77] C. Pintavirooj and M. Sangworasil. 3D shape reconstruction based on radon transform with application in volume measurement. *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2002.

[78] J.P. Pons, R. Keriven, and O. Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *International Journal of Computer Vision*, 72:179–193, 2007.

[79] Michael Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics, and Image Processing*, 40(1):1–29, 1987.

[80] Johann Radon. Uber die bestimmung von funktionen durch ihre integralwerte lngs gewisser mannigfaltigkeiten. *Reports on the proceedings of the Saxony Academy of Science*, 69:262–277, 1917.

[81] G. N. Ramachandran and A. V. Lakshminarayanan. Three dimensional reconstructions from radiographs and electron micrographs: Application of convolution instead of fourier transform. *National Academy of Science*, 68:2236–2240, 1971.

[82] Malcolm Reynolds, Jozef Dobos, Leto Peel, Tim Weyrich, and Gabriel J Brostow. Capturing time-of-flight data with confidence. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 945–952, 2011.

[83] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer Verlag, 1999.

[84] Murray Rosenblatt. Remarks on some nonparametrix estimates of a density function. *Annals of Mathematical Statistics*, 27(3):832–837, 1956.

[85] Jonathan Ruttle, Michael Manzke, and Rozenn Dahyot. Estimating 3D scene flow from multiple 2D optical flows. *Irish Machine Vision and Image Processing conference*, pages 6–11, 2009.

[86] Jonathan Ruttle, Michael Manzke, and Rozenn Dahyot. Smooth kernel density estimate for multiple view reconstruction. *Conference on Visual Media Production*, pages 74–81, 2010.

[87] Jonathan Ruttle, Michael Manzke, Martin Prazak, and Rozenn Dahyot. Synchronized real-time multi-sensor motion capture system. *SIGGRAPH Asia, sketch Poster*, pages 50:1–50:1, 2009.

[88] Y. Sahillioglu and Y. Yemez. Coarse-to-fine surface reconstruction from silhouettes and range data using mesh deformation. *Computer Vision and Image Inderstanding*, 114:334–348, 2010.

[89] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002.

[90] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:519–528, 2006.

[91] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal on Computer Vision*, 35(2):151–173, 1999.

[92] C. Shen, M. Brooks, and A. van den Hengel. Fast global kernel density mode seeking: Applications to localization and tracking. *IEEE Transactions on Image Processing*, 16:1457–1469, 2007.

[93] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1297–1304, 2011.

[94] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall/CRC, 1986.

[95] S. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. *IEEE Conference International Conference on Computer Vision*, 1:349–356, 2005.

[96] 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.

[97] Dan Snow, Paul Viola, and Ramin Zabih. Exact voxel occupancy with graph cuts. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:345–352, 2000.

[98] Balaji Vasan Srinivasan, Qi Hu, and Ramani Duraiswami. GPUML: Graphical processors for speeding up kernel machines. *Workshop on High Performance Analytics - Algorithms, Implementations, and Applications, Siam Conference on Data Mining*, 2010.

[99] P. Srinivasan, P. Liang, and S. Hackwood. Computational geometric methods in volumetric intersection for 3D reconstruction. *IEEE International Conference on Robotics and Automation*, 1:190–195, 1989.

[100] R. Szeliski. Rapid octree construction from image sequences. *CVGIP:Image Understanding*, 58(1):23–32, 1993.

[101] C. Theobalt, J. Carranza, M. A. Magnor, and Hans-Peter Seidel. Combining 3D flow fields with silhouette-based human motion capture for immersive video. *Graphical Models*, 66(6):333–351, 2004.

[102] Tina Yu Tian and Mubarak Shah. Recovering 3D motion of multiple objects using adaptive hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1178–1183, 1997.

[103] Peter A. Toft. Using the generalized radon transform for detection of curves in noisy images. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 4:2219–2222, 1996.

[104] Engin Tola, Vincent Lepetit, and Pascal Fua. A fast local descriptor for dense matching. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[105] P. H. S. Torr and A. Zisserman. Feature based methods for structure and motion estimation. *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, pages 278–294, 2000.

[106] M. Valera and S. A. Velastin. Intelligent distributed surveillance systems: a review. *IEE Proceedings - Vision, Image, and Signal Processing*, 152(2):192–204, 2005.

[107] S. Vedula and S. Baker. Three-Dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):475–480, 2005.

[108] Yusuke Yoshiyasu and Nobutoshi Yamazaki. Topology-adaptive multi-view photometric stereo. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1001–1008, 2011.

[109] Andrei Zaharescu, Edmond Boyer, and Radu Horaud. Topology-adaptive mesh deformation for surface evolution, morphing and multiview reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):823–837, 2011.

[110] Ye Zhang and C. Kambhamettu. On 3-D scene flow and structure recovery from multiview image sequences. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 33(4):592–606, 2003.