2017

The Use of Three-Dimensional Documentation Technologies in Archaeological Applications

Jahraus, Adam

UNIVERSITY OF CALGARY

The Use of Three-Dimensional Documentation Technologies in Archaeological Applications

by

Adam Victor Jahraus

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN GEOMATICS ENGINEERING

CALGARY, ALBERTA

JANUARY, 2017

Abstract

In archaeology, it is useful to document the shape of features of interest. There are many three-dimensional measurement technologies available that can help accomplish this task. An error model for a handheld 3D scanner called the DPI-7 was created. This error model reduced the errors in the in-plane directions by up to 59%. The levels of precision in two technologies, terrestrial laser scanning and computer vision assisted photogrammetry, were determined through the simulation of observations in a virtual environment. It was found that terrestrial laser scanning point observations had a standard deviation (in the direction of least precision) of 6mm, while photogrammetry could achieve a value of 10mm. The point cloud data from the scans of an excavation in the Canadian arctic were used to create a detailed and coloured visual model of the site, and was subsequently used in a virtual reality visualization of the site in question.

Acknowledgements

First, I would like to extend gratitude to my supervisors, Dr. Derek Lichti and Dr. Peter Dawson for their generous help and guidance over the course of this degree. Your expansive knowledge in your respective fields were enormously useful resources, for which I am grateful.

I'd also like to thank Dr. Max Friesen for accommodating me during my time at Kuukpak, and to thank Colleen, Mike, Danii, Remi, Becky, Letitia, Rose, Lawrence, and everyone else for making my time there such a wonderful and memorable experience.

Thank you to my lab mates, including Jeremy, Fengman, Reza, Mostafa, Kaleel, Ivan, Sam, Ting, Herve, and all of my other colleges for their friendship, assistance and insight.

Thank you to all of my friends and family, both for their support, and for their patience whenever I attempt to explain what I do for a living.

Finally I'd like to thank the Canada Foundation for Innovation, and the Natural Science and Engineering Research Council of Canada for their financial support of this degree.

For my wife, Raquel

Thank you for your endless love and support

Table of Contents

List of Tables

List of Figures

List of Abbreviations

3D	Three-dimensional
AP	Additional Parameters
AR	Augmented Reality
ARCO	Augmented Representation of Cultural Objects
CCD	Charged-Coupled Device
CP	Check Point
DEM	Digital Elevation Map
DSM	Digital Surface Model
LiDAR	Light Detection and Ranging
GCP	Ground Control Point
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
MEMS	Micro-Electromechanical System
NIR	Near-Infra Red
PCA	Principal Components Analysis
RMSE	Root Mean Square Error
RTK GPS	Real Time Kinematic Global Positioning System
TLS	Terrestrial Laser Scanning (or Terrestrial Laser Scanner)
UAV	Unmanned Areal Vehicle
VR	Virtual Reality

Chapter 1 - Introduction

1.1 Background and Motivation

Archaeology is the study of ancient human beings through the recovery and analysis of their material culture, such as artifacts, architecture, biological remnants and landscapes. The goal of archaeology is to understand the lives of the people who would have lived within this culture. Archaeology in the arctic regions provides unique challenges, but also the opportunity to study many unique cultures found only in these places. Most of the archaeological sites of particular interest to arctic archaeologists are located on or near the edge of the large bodies of water, such as the Arctic ocean, or in the delta of the Mackenzie River. This is due to the relative abundance of marine life compared to terrestrial life, particularly in the winter. The largest settlement in the Mackenzie River basin was called Kuukpak by the local Inuvialuit, which translates to "A Place to Live," and is located on the coast of Richard's Island, in the Northwest Territories, Canada.

The Inuvialuit have lived in the Western Canadian arctic for many hundreds of years with archaeological evidence indicating a 500 to 600-year record, and were decedent from the earlier Thule people. Although there is no sharp distinction between the Thule period and the Mackenzie Inuit period, one major difference is that the Thule people primarily hunted bowhead whales, while the Inuvialuit's major food resources would have been the beluga whales, as well the fish, along the coast of the Mackenzie River. It is likely that the abundance of resources in the delta attracted them to settle in this region. Since this location is above the tree line, all of the wood that was used for construction in this area would have been driftwood, taken from the river as it was washed downstream. The Inuvialuit first made contact with European explorers relatively late, during the

late 1800s, due to the remoteness of the area. Today, there are around 5000 Inuvialuit, living in various towns in the western Canadian Arctic [47].

At its peak, Kuukpak would have been home to approximately five hundred family groups [10]. In order to shelter themselves from the cold, the Inuvialuit people built semi-subterranean homes constructed out of sod and driftwood, which were given additional insulation with packed snow during the winter. Many of these homes would have been large enough to shelter several family groups, and were of skillfully constructed to conserve heat during the harsh winters [7].

The preserved remains of these coastal houses are at risk of being destroyed. These sod and driftwood houses have been preserved for hundreds of years thanks to being buried deep enough for the permafrost to prevent the wood from decaying. However, due to rising global temperatures, the permafrost is melting, putting the wooden structures at risk of decaying. In addition, due to the increased strength and frequency of storms, the arctic coastline is eroding at an increasingly concerning pace. These factors, both attributable to global climate change, mean that the physical preservation of these sites is coming to an end, either through decay or through being washed into the ocean [17].

It is a tragedy to witness these significant features being lost, but due to the remoteness and inaccessibility of these sites, preservation through human intervention is not possible. Instead, a proposed solution is digital preservation. This is the idea that, rather than seeking to preserve the archaeological site itself through some form of intervention, one can preserve the site by creating a highly accurate digital version of it. This digital version can then be preserved, distributed, explored and studied in place of the physical site. While digital preservation has the notable drawback of not protecting the physical site, it has many other benefits, such as significantly reducing the cost associated with studying the site, providing opportunities for far reaching and

low cost education regarding the site, as well as bringing the opportunity for the Inuvialuit people alive today to connect more personally to their ancestry in a manner that contributes a sense of presence.

However, the process of creating a model to be digitally preserved is non-trivial. There are many technologies suited to the task of collecting the data of the site, which is only the first step of digital preservation. These technologies should be considered carefully, and the decision must be based on many different factors, such as the requirements of the final model, the cost and performance expectations of the 3D capture tools, the technical skill of the operator, as well as the subjects to be preserved. The decisions made in this step well affect every step of the process afterwards, so it should be done carefully.

One potential technology for collecting the data used in digital preservation is terrestrial lasers scanning (TLS), which uses light based range detection (LiDAR) combined with angular measurements to create complete panoramic range maps of the scene surrounding the scanner. Using tie points, multiple scans can be combined into a single coordinate system, enabling multiple viewpoints to be used at once, which assist in overcoming occlusion and changes in sampling density. Another potential technology is close-range digital photogrammetry, which involves taking many digital photos of the scene in question, identifying common tie points in multiple images, then relating those locations to the locations of the cameras, enabling the three-dimensional coordinates of both sets to be determined.

Both of these technologies, as well as others, have various advantages and disadvantages that should be considered. Influencing factors include the cost of equipment, the precision of the 3D measurements, the ease of performing measurements, the amount of time required to capture

data, as well as time to process the data. There are other types of 3D data collection tools, all of which have varying properties in all of these areas.

1.2 Research Objectives

It is the principal goal of this research to help determine the best practices of digital preservation via three-dimensional measurement techniques, in relation to the engineering principals involved. In addition to engineering principles, best practices should also take into consideration the logistical, financial and technical limitations inherent in archaeological applications. This research will provide a reference for archaeologists to consult to be able to make informed decisions about which 3D data collection method best suits their needs in regards to the engineering principles involved.

One of the most fundamental processes that engineers perform before collecting 3D spatial data is the instrument calibration. This step is so fundamental because it is through calibration that one establishes what level of trust in the measurements provided by an instrument is warranted. It is also through calibration that systematic errors (i.e. biases) can be detected and corrected through the addition of additional parameters (APs) in the observation equations. If no calibration were performed, then significant errors could go totally un-noticed, affecting the final models created using this data. It is the first objective of this research to test the performance and create an error model to calibrate the Dot Products' DPI-7 scanner. This scanner is a small, relatively inexpensive 3D scanning device that is simple to operate, making it an attractive option for archaeological documentation. This calibration procedure should be designed in a way that is tractable for none experts to follow, as an example of how such measurement devices can be evaluated.

To be able to make a decision about which 3D measurement technologies are right for a given task, one must recognize the difference in precision of various technologies. For instance,

there is an appreciable difference between the precision that can be expected from terrestrial laser scanning than what can be expected from computer vision assisted close-range photogrammetry. In order to determine the actual precision that can be obtained, a stochastic model can be constructed. This model, which uses observation equations and realistic error parameter values, can be used to show precisely the difference in metric performance between these two types of data collection. In principle, this type of error propagation modeling can be done for any type of 3D spatial measurement device, so the method of comparison can easily be extended to compare whatever technologies are under consideration. The second objective of this research is to quantify the precision of the two technologies and analyze their differences to make a recommendation for a given application.

Once the raw 3D spatial information has been collected, it must then be used to make a model which is useful for the researchers' purposes. Different models are appropriate for different purposes. For instance, in order to determine the change in shape over time of a landscape or feature, a deformation model must be constructed. However, if the goal is to create a visual representation of the site which accurately reflects the qualities of being there, then a different set of requirements is placed on the model. However, in both cases, the model must be as accurate as possible to the true site. Any deviations or artistic license taken while creating the model will affect the interpretations of the researcher, and potentially lead to faulty or unreliable conclusions. To this end, it is of critical importance that the process of virtual reconstruction be without bias, both in terms of systematic measurement errors, but also in personal bias on the part of the operator. One such operator bias might be to exaggerate the size of certain features, because they seem important, but in doing so the operator is misrepresenting the data. Therefore, when constructing the models from the raw data, it's important that the methods being used are as systematic as

possible. The third objective is to create high quality visual models that are both geometrically and visually accurate representations of the feature being modeled.

1.3 New Contributions

The first contribution of this thesis is a self-calibration method designed for use with a textured light based hand held scanner, the Dot Product's DPI-7. This scanner is similar in operating principle to the first generation Microsoft Kinect, except with the addition of a 7" tablet computer and proprietary data collection software. The software is designed to simplify the data collection process, making it approachable for non-experts such as archaeologists. However, the simplification conceals much of the raw data, making rigorous calibration difficult. In the process of performing the self-calibration of this device, it was found that there was a 2% scale factor error in the X and Y directions, relative to the imaging plane. This calibration is important for archaeologists to perform so that they can be confident that the spatial measurements, and the conclusions drawn from those measurements, can be trusted. However, it is also important that the calibration be understandable or tractable in terms of procedure by archaeologists as well, or else the benefit of the simple data collection is irrelevant.

The second contribution is the stochastic error model and simulation of observations with a virtual environment used to compare the performance of terrestrial laser scanners and computer vision assisted close-range photogrammetry in archaeological applications. This aspect is important for archaeologists so that they can understand the differing levels of precision that can be expected from typical spatial measurement systems, and what kinds of tasks can be expected from systems with these levels of precision. By assuming reasonable values for the variances of the observations (values taken from engineering literature) and mathematically propagating them to estimate the precision of the final point clouds, the expected precision of those clouds can be

determined. This process provides a quantifiable comparison between the two technologies that is easily understood by non-experts, such as an archaeologist.

The final contribution is the technique for applying images to the surface of mesh models created from terrestrial lasers scan data. This technique takes advantage of several different open source software products, which are used to create the mesh model from the point cloud data. Then, using the API provided by *Blender* [43], a free open source computer graphics software, the panoramic (i.e. equi-rectangular, or spherical) images taken by the scanner can be combined, using the registration parameters, with the mesh to create a high quality final product that is both geometrically and visually representative of the original site. By using only freely available software packages, anyone with access to a computer and the internet can replicate the process used in this paper to create their own models. This is an important product for archaeologists to have, since high quality 3D visualizations are the primary tool for them to be able to share the experience of being at an archaeological with those who would otherwise not have the luxury of physically being present.

1.4 Thesis outline

This thesis begins with a paper on the calibration of the DPI-7 hand is presented. This paper was first written for the SPIE Videometrics conference in June 2015 Munich, Germany.

Then, present the methods used to systematically compare the expected precision in a TLS model of the Kuukpak site, versus a photogrammetric model of the same site. This comparison was performed using error propagation in the observation equations and reasonable estimations for observation variances. This study found that the standard deviations in the point clouds from terrestrial lasers scanning were approximately half of those in the point cloud produced from computer vision assisted close-range photogrammetry.

The following chapter will present the process by which a visual model of the Kuukpak site was created. This includes the software pipeline, including a summary of the different algorithms used in the preparation of the data, as well as a custom algorithm for applying the colour of the scans to the mesh in a visually optimal way.

The final chapter will provide concluding remarks and recommendations for future work.

Chapter 2 – Background and Literature Review

2.1 History and State-of-the-Art of 3D documentation in archaeology

Conventionally, archaeological recordings are made using hand-drawn sketches, measured using level lines and metre sticks. This is commonly thought of as being an 'objective' documentation of the site. However, it's very easy for an individual to impart their own perception into their drawing, even while measurements are being made, since those measurements by necessity are only made to some key locations on the features themselves. Furthermore, since these drawings are in two dimensions, the depth information is always lost, which in turn can lead to ambiguity regarding the shape and orientation of objects being documented. It is no surprise, then that archaeologists are experimenting with various forms of three-dimensional (3D) documentation techniques.

Ever since the computer became a tool for researchers to use in their work, it has enabled previously tedious or impractical calculations to be done with ease. This in turn enables new forms of analysis and testing of data and hypotheses, which indeed did occur in archaeology, as detailed by Whallon [2]. In this paper, Whallon explores then recent impact that the computer had on archaeological thinking. He emphasizes that although the computer permits more complex manipulations of data and statistical calculations, "theory and method in archaeology are often explicitly derived from an anthropological frame of reference." This is to say that although computers enable improved tools of analysis, these modes of thought must be coordinated with those derived from anthropology. Whallon also provides examples of areas in which computers have made significant contributions, such as seriation (relative dating of artefacts), classification,

data banks, and simulations. What Whallon could not have anticipated in 1972 was the incredible increase of both computing power and number of potential applications that computers would offer in the coming decades.

Remondino [27] provides an excellent overview of many different technologies that are available to archaeologists should they wish to use 3D modeling in their documentation. The paper considers many different potential applications within archaeology, including "…historical documentation, digital preservation, cross-comparisons, monitoring of shape and colours, simulation of aging and deterioration, virtual reality/computer graphics applications, 3D repositories and catalogues, web-based geographic systems, computer-aided restoration, multimedia museum exhibitions, visualization and so on." He also goes over many different types of scanners and other devices and provides comparisons between different makes and models of similar types. It is also noted that a unified set of standards should be agreed upon by the industry, so that a common language can be used and comparisons between devices made more easily, particularly to benefit lay-persons. Finally, it is noted that there are several different main areas of research that remain for geomatics engineers in this area, including "New sensors and platforms; integration of sensors and data; automated processing; on-line and real-time processing; feature extraction; improvement of geospatial data and content; developments of new tool for non-expert users; and adoption of standards." It is with these recommendations in mind that this thesis seeks to contribute to the state of the art of 3D documentation as applied to archaeology, particularly in the areas of automated processing and development of tools for non-expert users.

2.1.1 Photogrammetry in archaeology

While references to photogrammetry in archaeology date back as far as 1875, the popularity of photogrammetry has significantly risen in recent years. This can largely be attributed

to the increased availability and reduced cost of digital cameras, as well as the ever-decreasing costs of computers. Lately, software solutions such as the *Python Photogrammetry Toolbox* [23], *Agisoft Photoscan*, and *Autodesk 123D Catch*, are able to simplify and improve the photogrammetric process significantly.

These tools have made the process of using photogrammetry for 3D documentation significantly simpler, faster and more reliable than it would be if archaeologists would have taken it upon themselves to program the solution. However, this also means that without a deep understanding of how photogrammetry is performed, an archaeologist might attempt to use the software incorrectly or without considering best practices, which may lead to poor results that could go unnoticed by the archaeologist and the general public. It is therefore in the interest of archaeologists who wish to use these software packages to understand, at least in part, how they work. For instance, knowing what a tie point is and how they are automatically identified can help an archaeologist decide if a surface material is appropriate for photogrammetric documentation, while an understanding of network design will assist in collecting images that will give better and more reliable 3D measurements.

Barazzetti et al. [22] studied the reliability and precision that can be expected from the 3D data collected from photogrammetric solution when assisted by highly automated computer vision algorithms. These algorithms were used to extract tie points from images, determine the correspondence of tie points between images, determine relative orientation of the images, detect outliers in tie point matches, and to measure the shape of the surface after the bundle adjustment had been completed. It was found that if a good imaging network was used and the subject had sufficient detail for tie point extraction, then the results of the automated procedure were as good as, or in some cases better than the quality achievable by an expert user. This study demonstrates

that computer vision algorithms can be used in photogrammetry application to drastically improve the speed of the process without necessarily negatively impacting the quality of the 3D information being extracted.

De Rue et al. [36] used photogrammetry to document the excavation of a 12th century Cistercian abbey. In this study, they decided to forego the traditional excavation documentation process and rely entirely on the model produced using *AgiSoft Photoscan*. The models produced from this data, which included ortho-photos, digital surface models (DSMs) and complete high resolution meshes were of high quality, both visually and geometrically. Other positive outcomes were the integration of the GIS and the semantic value of being able to see and explore the entire site. There were drawbacks, however, such as how, due to the processing time to produce the models, it was not possible to have immediate feedback on the quality of the recording, so the process of recording the features had to be very careful to collect good quality data. Another drawback was that contextual labels, such as descriptions and interpretations, had to be recorded separately and applied to the models after the fact. It was also necessary to change the workflow of the excavation, which in turn can negatively impact how effectively excavators are able to work. Regardless, the authors are confident that 3D image-based modeling provides enormous benefits to the process of archaeological excavation and recording. An important aspect that the author noted that was missing from the process was user-friendly tools for accessing and using the orthophotos and 3D models in the post-excavation processing.

Nocerino et al. [35] have investigated the impact that the photogrammetric network configuration has on the accuracy of the 3D model in cultural heritage documentation. They compared the effects of convergent vs. stereo-photographs in both close-range and aerial photogrammetric networks. They found that the inclusion of convergent images considerably

improves the strength of the network, and assists in avoiding global deformation of the resulting 3D model. It is therefore important for archaeologists to understand that the quality of the images and their configuration being used in a photogrammetric solution is as important as the software itself.

It's worth noting that different applications of photogrammetry offer different levels of precision, simply based on the network of photographs used. A large baseline between photographs will amplify point errors, while a small baseline will minimize them. This is to say that photogrammetry can be appropriately used in a wide variety of scales, if the camera being used can focus on the subject correctly. Therefore, the 3D point precision in photogrammetry is highly dependent on the precision of the calibration parameters, tie point extraction and matching, and network quality. The levels of precision present in photogrammetry naturally grow in proportion to the size of the subject, assuming that the network quality, tie point matching and calibration all remain strong. It is no surprise, then, that the 3D point precision of a large subject is less precise than the point precision available for a small subject.

Additionally, different levels of point precision are required for different applications. For instance, a visual model of an excavated house might only require precision around the centimetre level, but a model of a small artefact might require sub-millimetre accuracy to faithfully represent the subject. Fortunately, in the case of photogrammetry, the network of images used to estimate 3D points naturally trend towards appropriate levels of precision for the subject. One criteria that can be used to judge if a model is of high quality is to ask “are all relevant features of the subject accurately represented in the model?” The features of interest may vary from subject to subject, but they should be present in whatever form is best suited. In the case of a deformation analysis, then spatial changes on the mm level should be represented, where for visual models, one should

be able to see small details that are relevant to that subject (i.e. a structure might show the texture of its surfaces, or a small artifact might show fine details). However, in the case of visual models, the fine details may not need to be represented spatially if they can be well represented in the texture.

2.1.2 Aerial photogrammetry with UAVs in archaeology

Similar to close-range photogrammetry, aerial mapping techniques have also been used in archaeology, with increasing popularity in recent years. In past decades, the costs associated with hiring an air crew to perform aerial mapping would have been well beyond the budget of most archaeologists. However, the decreasing costs of GNSS sensors, digital cameras, MEMS accelerometers and gyroscopes, as well as rechargeable batteries have made small unmanned aerial vehicles a very cost effective tool in archaeological prospection, mapping and documentation.

Bendea et al. [19] used a small unmanned aerial vehicle (UAV) to collect photogrammetric areal images over the *Augusta Bagiennorum* test site. This site was selected because it has been used in the application of geomatics techniques in archaeology since the 1990s, including other aerial surveys. The whole system, including the camera, the aircraft and the navigation system were all 'low cost,' and the photogrammetric network of control points were surveyed using a total station and RTK GPS. Both artificial and natural control points were surveyed. The residuals of the aerial photogrammetric survey were found to be in the 10s of cm range for both the ground control points (GCPs) and the check points (CPs). This study shows that UAVs are a viable technology for archaeological mapping, and that they needn't be extraordinarily expensive to be useful or precise. However, as new regulations for UAVs come into place, this type of DIY approach may

A more recent study by Fernandez-Lezano and Gutierrez-Alonso [45] demonstrates how UAVs, accompanied by digital elevation maps (DEMs) from LiDAR and photogrammetric data can be used to improve the prospection of archaeological sites. The example presented in the paper was that of Roman gold mining remnants, including a hydraulic system left in the landscape and mining areas along the Eria River Valley Gold District. The use of different imaging mediums allowed for the comparison of the data at different levels of detail, as well as the comparison of different image processing tools for the purposes of detecting these features. It was found that the 1m LiDAR data provided by the UAV was sufficient for this application, and that the possibility for smaller detailed surveys using the UAV was a valuable asset. This study shows just one valuable 3D documentation application that is provided by UAVs.

2.1.3 Terrestrial laser scanning in archaeology

Another technology that has recently been creating an impact in archaeology is terrestrial laser scanning (TLS). Although the equipment required for this method is more specialized and expensive, the added benefits of increased precision, more robustness to environmental effects and automatic 3D measurements make TLS an attractive alternative to photogrammetry.

One of the earliest uses of laser scanning was “The Digital Michelangelo Project” [12], which sought to use a triangulation laser scanner, time of flights scanners as well as digital images to create detailed 3D models of 10 sculptures by Michelangelo, including the famous statue of David. Given that the technology used and the computational limitations at the time, the results are impressive, producing very high quality visual models, to the level of the chisel marks in the marble, which the social history of these objects. The process was both laborious and logistically difficult, since it required such large amounts of equipment to be used in delicate locations such

as museums and churches. While it took a large specialized team and a great deal of custom-made machinery and software, this study shows that high quality 3D documentation is in fact possible.

Dawson et al. [20] were among the first archaeologists to attempt to use laser based 3D documentation techniques for in situ archaeological applications, using a Minolta Vivid 910 to scan the structure of an excavated sod and driftwood house at Kuukpak, on the coast of the Mackenzie River. While the quality of the individual scans was quite good, the particular type of scanner (i.e. a triangulation beam laser scanner) was not ideal for collecting data across the entire site at once, nor for collecting surrounding contextual information. In spite of this, they found that 3D documentation techniques were a promising area of investigation.

In a subsequent study, Dawson et al. [30] used a Z+F Imager 5600i phase-based scanner to document the buildings located at Fort Conger on Northern Ellesmere Island. The quality of the data in this case were very good, collecting both high levels of detail and broad contextual information. A Minolta Vivid 910 scanner was also brought so that smaller artifacts and objects could be scanned in higher detail and greater precision than is possible with the Z+F scanner.

2.1.4 Range cameras

There are other 3D documentation technologies that are not necessarily ideally suited for archaeological applications, but that should be considered nevertheless. One such technology is range cameras, which come in two broad varieties, textured light (e.g. the Microsoft Kinect) or time-of-flight (e.g. Mesa SR-4000). In both cases, these cameras offer range data from a single perspective, usually with video frame rates. They tend to be most useful in real time applications, of which there are few in archaeology. However, there are technologies such as the Dot Product's DPI-7 scanner which is based on the principles of range cameras, but incorporate other measurement technologies, such as low cost accelerometers and gyroscopes. This is done in order

to create a real-time scanning device which is not limited to a single perspective, but rather can be used to scan a scene from a variety of perspectives by tracking the motion of the camera relative to the scene being scanned. The benefit of this is that it takes advantage of the simple operation and small size of a range camera while offering flexible and low cost 3D data. The limitations are that the precision and visual quality of the data are often significantly worse than if other measurement mediums were used.

2.2 Sensor Calibration

Sensors calibration is important in an archaeological context because it is the process by which one determines the precision and accuracy of the measurement device. This device is used to collect 3D data and to construct 3D models on which hypotheses are made and theories tested. If the data is incorrect then the conclusions made based on those measurements are called into question. The type of calibration that needs to be performed is necessarily dependent on the measurement medium being used. This is because each different medium will have different types of observations, and therefore will have unique sources of error. The calibration of several different devices will be considered here.

Kenefick et al. [3] first established the model which can be used to self-calibrate a photogrammetric camera, i.e. it would not be necessary to pre-calibrate a photogrammetric test-range. This technique uses the collinearity equations with additional parameters to enable highly accurate photogrammetric measurements to be made, and the interior and exterior orientation parameters to be reliably determined. Also noted was the importance of highly convergent photographs for the purposes of improving the network geometry and decorrelation of estimated parameters. It is this calibration method that lays the groundwork for the methods that are used today.

The calibration of digital cameras for use in photogrammetry is outlined in detail by Fraser [11]. In this paper, Fraser details how the self-calibration of metric cameras, mainly intended for aerial triangulation, can be applied to digital cameras, with particular emphasis on close-range imaging. Fraser outlines the need and methods for measuring the various additional parameters (APs) used in the self-calibrating bundle adjustment, considering the differences and similarities that exist between them. This method of calibration is still the preferred method of calibration for digital cameras today.

The calibration of terrestrial laser scanners (TLSs) was well determined by Lichti [18], who took advantage of the similarity of observation between TLSs and total stations, which have a calibration model that is well studied. The additional parameters used in this calibration model included terms for accounting for errors in the range, the horizontal, the vertical angle, as well in the ways that those measurements interact and interfere with each other. The inclusion of these APs improved the RMSE of target points in the calibration by between 22-45%. While there are many different models of laser scanners, this calibration can be used for any that have the same types of observations.

The measurement method used in the DPI-7 handheld scanner is similar to the one found in the Microsoft Kinect. Therefore, the calibration of the Kinect scanner can be used to improve the quality of the data collected using the DPI-7 scanner in archaeological applications. In order to calibrate the Kinect, a textured light based range camera, Chow and Lichti [28] performed an integrated self-calibrating bundle adjustment of the near infra-red (NIR) cameras, the visible light camera (RGB) and the NIR projector. This approach includes several different constraints such the rigid relative orientation parameters between the NIR camera and the RGB camera, and between the NIR camera and the NIR projector, as well as a planarity constraint, such that all

observations are understood to lie on a plane, and deviations from that planarity are therefore errors. This method of calibration was found to be extremely beneficial for improving the quality of Kinect data.

Currently, the state-of-the-art process for the calibration of 3D sensors emphasizes the increase in the accuracy and precision that can be gained in the device, but there is little regard for the how the calibration procedure can be applied by non-expert users in less than ideal environments or circumstances. For instance, it's good to be able to perform a calibration in controlled laboratory settings, but for an archaeologist collecting data in the field, they may not have the luxury of using a lab, or even an indoor room. It may be enough to perform a calibration on the device prior to leaving for the field, but it would still be advantageous to perform a calibration to verify the results when it's time to collect data of the feature in question. Furthermore, many calibration methods models are far too complicated and intensive for most non-experts to be able to understand or perform without training. What's needed in these circumstances is a method that can be used to calibrate measurement devices in a simple manner, and in non-ideal environments or circumstances. This is doubly beneficial in that this simpler version of the calibration can be used to verify the validity of the data in between more complex and systematic calibrations, which are typically only performed every couple of years. This is especially of concern for devices which are frequently shipped to remote locations, increasing the likelihood that they could be knocked out of alignment.

The calibration process for different technologies could be made easier for non-experts by standardizing procedures, in particular the targets fields and software, since a large portion of the work that goes into designing a calibration field is defining the network of control points, as well as extracting the target locations from the data. Large portions of the calibration procedure can be

automated if the style and distribution of targets is known before hand, especially extracting and matching targets. This, however, might only shift the difficulty from designing the target field, to having to reproduce the target field that the software expects. It may be possible to reach a middle ground, where the target field can be any distribution of points, if they follow some guidelines specified by the software. It may not be possible to build a fully automated system that works for all types of devices and all applications, even within a narrow category such as DSLR cameras, so some effort is almost certainly required of the user, even if large portions of the calibration procedure have been automated.

2.3 Stochastic error models

A stochastic error model is a tool that can be used to quantify the precision of a measurement system using simulated observations, observation equations, and estimates for the precision of those observations. The result is an estimate for the precision of all the final measurements, taking into consideration all relevant error sources. The precision of the 3D measurements in archaeology are important in any situation where the shape and size of the subject are important, since that information is contingent on the precision of the 3D measurements.

Lichti et al. [15] performed an error propagation for a directly Georeferenced terrestrial laser scanner network. Error sources such as registration errors and laser beamwidth are considered and included in the error propagation. Other errors, such as mixed pixels (i.e. pixels which overlap the boundary of a foreground and background object, reporting the range incorrectly as the mean of the two), detector saturation and multipath are artefacts of the materials and surfaces being scanned. These types of errors and cannot be accounted for in a standard error propagation, since they are unique to each scan and the environment in which it exists. They found that the resulting propagated errors for the point cloud were larger than might be expected from the relatively smaller

range observation precision, and they found that this would have largely been due to the beamwidth uncertainty and vertical angle precision. While different scanners will have different expected observation standard deviations, the model being used here can still be used to understand the nature of the errors and their impact on the final point clouds being produced. Earlier examples of quantifying the accuracy and precision of TLS are Hebert & Krotkov [8], Boehler & Marbs [13], and Gielsdorf et al. [14].

Garcia-Morena et al. [37] also performed an error propagation, considering a combination of both LiDAR and a spherical camera, in this case as part of a mobile mapping system mounted on a vehicle. Their propagation considers the convergence of the image observations in the digital camera from different perspectives, and the coincidence of the image observation with the LiDAR data using custom targets, which they identified in both the LiDAR and images with automated methods that gave highly accurate results. They found that the distance that an object is from the scanner will affect the quality of the estimated precision, with a reconstruction error of around 2cm if an object is around 4m, increasing at a rate of 0.031 mm/m. They also point out that these results were found in controlled laboratory conditions, and that an urban environment may have objects and materials that introduce unpredicted data artefacts which would decrease the expected precision.

These techniques are excellent for determining the expected level of precision that can be achieved with these measurement devices, but they are also rather abstract to a non-expert user. While ‘centimeter level precision’ may have meaning for someone familiar with the measurement medium, this is not necessarily so for a non-expert. In this case, an archaeologist may not particularly care about the specific level of accuracy that can be achieved with either TLS or photogrammetry, but rather they may want to consider how much better one is over the other

considering the same subject. These considerations further depend on the intended use of the data, since a visual model requires less precision than a deformation analysis, for instance. This thesis performs an error estimation comparing the performance of a TLS scanner and digital photogrammetry of an archaeological site as an example of the expected differences in an actual application.

2.4 Applications of 3D models

There are many different uses of 3D data after it has been collected. Many times, an archaeologist would like to be able to create a visual model from that data. A visual model is a representation of an object, scene or environment that emphasizes the visual quality as the highest priority. These are used to show the feature to a wide audience, emphasize the beauty of it, or to create an impression of what life might have been like in the past. In the case of archaeology, artifacts, excavations, landscapes, and other features might be subjects of a visual model.

When creating a visual model it is important that it is an accurate representation of how the feature appears in reality. There are many circumstances when it is acceptable to take creative license with virtual representations, since that is what many digital artists do on a daily basis. However, in the case of archaeology, speculative changes to the visual model without evidence to justify such a change may be a misrepresentation of the data, and in some circumstances a breach of ethics. This is why it is important to use techniques that work independently of the person creating the visual model. This is a significant departure from how many digital artists create their work, since most of the time it is up to the artist to shape and modify their creation until it ‘looks right.’ On the other hand, for a visual model of an existing object, the correct choice would be whatever leads towards accuracy, not what the model ‘should’ look like according to the operator.

In addition, virtual reality (VR) is becoming a valuable asset when viewing visual models. The perception of 3D and complete visual immersion of the individual creates a sense of presence that is difficult to achieve without the benefit of VR. In addition to VR, advances in 3D printing have also enabled researchers to create replicas of artifacts, enabling them to be handled and viewed as naturally as possible without the risk of having the precious originals become damaged or misplaced. Both of these technologies have a great deal of potential to engage very diverse peoples, and also rely on accurate data collected from physical objects.

Sylaiou et al. [21] performed a study into the impact that augmented realty (AR) has on an individual's enjoyment of cultural objects. They work within the framework that telling stories about cultural objects can establish an historical narrative, and allows visitors to construct semantic meaning about the objects. They present a technical infrastructure called the Augmented Representation of Cultural Objects (ARCO) system, which enables a technologically-augmented story-centric view of museum exhibitions. This involves not only 3D reconstructions, but also a database and 3D interaction interfaces, such as AR. It also enables media objects such as images, videos, texts and 3D models to be associated together. They found that a high degree of a 'sense-of-presence' was positively correlated with the amount of enjoyment and engagement that was experience with the virtual exhibit, that VR presence and an AR objects' presence were positively correlated, and that previous computer experience was not significantly correlated with the level of presence or enjoyment.

Dawson et al. [26] studied the impact that virtual environments of ancient dwellings had on the decedents of those who would have built such dwellings. In this case, the virtual environment was created using a 3D theater called a CAVE, and the models of the dwellings (one of a Thule whalebone house, and another of a Siglit-Inuvialuit sod house) were created from

archaeological data using digital replicas of various tool and materials. Groups of Elders from the Canadian arctic were given the opportunity to view these virtual worlds, and were asked about their experience. The reports were positive, many of them reporting that the experience made them recall stories or events of their youth, or that they felt it easier to imagine how their ancestors would have lived given a view of the interior of these homes. Due to logistical issues of translation and time constraints, the standard questionnaire to determine the level of perceived presence was not conducted, but their self-reports indicated that they found the experience to convey a great deal of presence. While this study is important and valuable, the methods involved are limiting. In particular, the process of creating the models from archaeological data was highly manually intensive, making it so that only a few such examples could be made. This is a limitation because different cultures created different dwellings from different material and tools, so the sense of presence or of personal significance may be reduced for those viewers. Also, because the virtual system being used is a specialized room and cannot be easily taken down or moved, the viewers must come to that location, which severely impacts the ability to share such an experience with a wide audience.

Grosman [42] provides an excellent literature survey of the many different applications of 3D analysis, digital archaeology, and visualization. The different sections outlined are the visualization and documentation of archaeological sites, art, and artefacts; the analysis of spatial data such as symmetry and roughness, centre of mass, geometric morphometric analysis, and complete 3D data analysis. The author also explores the potential future challenges that these new documentation and measurement technologies create. One such consideration is how this growing volume of data might inspire new modes of thought within archaeology, such as how global theoretical models can now be reliably tested using the abundance of data made possible by

computerization. The author also advocates that these 3D documentation techniques should, rather than be simply applied to archaeology, be integrated into it by experts who are familiar with both fields of study.

Lymer [40] used TLS to document rock art on a pair of panels at Loups's Hill, County Durham in England. Loups's Hill contains noteworthy concentrations of rock art, and is similar to many other locations which are spread across northern Britain and Ireland. The rock art is estimated to date from 4000-1500 BCE, and feature imagery usually referred to as cup and ring marks. The goal of this study is to explore the potential of ArcGIS for rock art research by focusing on 3D data collected using an Leica HDS 3000 laser scanner. Prior to modern 3D documentation techniques, these exaples of rock art were recorded using different analog methods, such as wax rubbings copied to graph paper, the same rubbing updated with digital photographs stitched together on a computer, and a free-hard drawing. These different recording techniques were super imposed, and compared. The results confirm previous findings that different people will produce different results in recording the same panel. To overcome this ambiguity, and to use precise, contactless documentation techniques, the TLS data is used. Several different analysis techniques are used to help interpret the data. These included Triangular Irregular Network interpolation, a Normalized DSM, Virtual Reflectance Transformation Imaging, and Raster data Principal Components Analysis. Since this was a pilot study, not all of these modeling tools proved to be useful in extracting the rock art from the data, but they did offer the researchers different methods of visualizing the rock art that were not available without the TLS data.

Plisson and Zotkina [41] also address 3D documentation of rock art, but in this case they use extremely close range photogrammetry to document structures at millimeter and submillimeter scales, using macroscopic and microscopic photography. These types of imaging allow the

archaeologists to investigate questions not previously addressable, such as the chronology of the marks, types of tools used, composition of pigments, etc. Since the depth of field for microscopic images is so small, it wasn't initially clear that photogrammetry could be used at this scale of imaging, so the authors performed tests on experimental glyphs using various SLR cameras and *Agisoft Photoscan.* They demonstrated that for soft rocks, metallic and lithic tool imprints are clearly registered, as they are for deep engravings. However, in order to produce images at microscopic levels which are entirely in focus, the researcher used a technique called 'focus stacking,' in which many images of the same subject with different depths of field are combined. This technique, however, was not well suited for photogrammetric application, due to the changing levels of distortion that the different depths of field introduce. However, the 3D surface of subject can be extrapolated from a set of focus stacked images, using software called *Helicon Focus* [48]. It is noted by the authors that photogrammetry and focus stacking are complimentary, since photogrammetry requires a large depth of field, while focus stacking requires a small depth of field. The results are impressive levels of detail in the resulting 3D models of the markings.

Bustillo et al. [39] studied the applicability of virtual visualization of 3D models of historical objects in an educational context. They evaluated the benefits of this technology by presenting the 3D visualization to a group of students during a teaching session on the topic of Cultural Heritage and Computer Graphics. The authors are especially interested in the real-time representation of 3D models, and so are interested in reducing the complexity of the models so that they can be viewed without large amounts of computing power. The 3D modeling is performed by students learning computer graphics using software such as *Blender*, and the visualization is performed in an adjacent room which consists of several projectors, a motion capture system, sound system, 3D positioning system, and a computer used to generate 3D models in real time.

The modeling was performed in five steps, which were 3D modeling, texturing, lighting, rendering and post-production. Although these techniques are clearly very labour intensive, their primary purpose were to be used as a teaching tool, which enables students to learn these methods, as well as learn about the subjects that they are modeling, at the same that they are producing beautiful 3D models.

Haukaas and Hodgetts [46] explore the use of close-range photogrammetry in documenting and modeling archaeological sites and artifacts in the Canadian arctic. They suggest that by engaging the Indigenous population with these technologies, it may 'democratize' the process of archaeological documentation and modeling, while simultaneously increasing interest with those populations. They also present their models online, further engaging with the local populations using Facebook, though they recognize that the issue of intellectual property may be a concern, since taking a collaborative approach with the Indigenous population may mean that that population should have part or even complete control over who has access to the data of their artifacts or archaeological sites.

Besides the potential applications, it's also important to consider techniques that can be used to take data collected using 3D sensors and efficiently turn them into useful models, such as visual models. For example, when a point cloud representing an object is created, the points that comprise the cloud, ideally, are located on the surface of the object. However, reconstructing the surface from which those points were collected is non-trivial, since there are an infinite number of surfaces which may exist that could have given these samples. If the normal direction of the points are also measured, then this additional information enables more advanced surface reconstruction, as detailed in Kazhdan et al. [16]. Conceptually, if an object is represented in space by a binary function, where it is zero value for outside the object and one for inside, then the gradient of that

function would be equal to the inward normal direction of the surface. Because the point cloud (with normals) is a set of oriented points that describe a surface, then the underlying surface can be estimated from those points by using a Possion formulation. This formulation considers all points simultaneously, and is therefore robust to noise, and yet can achieve very high levels of detail. This technique is highly valuable for creating detailed surface representations of objects that have been sampled as a point cloud.

Currently, there is a gap between the data that is collected through various imaging metrology technologies, and the creation of various digital model that represent the subject. There are very few streamlined tools that are available to create useful digital model directly from the data collected by TLS scanners, for example, and those that do exist may not take full advantage of all of the data available. The technique developed in this thesis for creating visual models from TLS data is designed to be both rigorous and precise in it treatment of the spatial data, and to fully take advantage of the visual data available from the scanner.

2.5 Logistical issues and considerations

There are many reasons why one may want to use one 3D documentation technology over another, since they all have different levels of precision, associated costs, computational requirements and levels of skill required for their operation.

The financial costs associated with any particular 3D documentation technology can have several different factors. The instrument itself has a price associated with it, requiring up-front capital investment, but there are other costs associated as well. There may be periodic maintenance required, software costs or other ongoing expenses that are required. There is also the cost associated with learning to use the device and associated software, which more often takes the form of time investment rather than capital costs. This may lead archaeologists to want to

collaborate with Geomatics engineers, whom likely already have the equipment and experience necessary to collect and process the data.

In many cases, an archaeologist may want to perform 3D documentation in areas that are difficult to get to due to either their remoteness or general inaccessibility. In these cases, the costs associated with shipping the equipment necessary may prove to be expensive and challenging. The environment in which the documentation takes place will also potentially impact the quality of data being collected, and may even make it impossible to collect data using a particular medium at all. This remoteness also makes it very difficult to transfer skilled equipment operators to the location to perform data collection, indicating that it is likely wise for archaeologists to learn to collect data themselves.

The following chapter will present the DPI-7 handheld scanner in detail, and explain the error models and calibration procedures used to improve the quality of the data collected by it.

Chapter 3 - Error Modeling and Calibration of DPI-7 Handheld Scanner

3.1 Introduction

The DPI-7 handheld scanner, as seen is Figure 3.1, has many features that make it an attractive tool for 3D documentation of archaeological sites or artifacts, particularly in situ applications or in remote environments. One of the first considerations, however, is the metric performance of the device. Many user reviews of these types of 3D measurement technologies [38] consider the cost, ease of use, and data collection speed, but it is significantly more difficult to verify the metric performance. The purpose of this chapter is to propose a geometric model for modeling the errors present in this device, and to calibrate the DPI-7 scanner can using that model.

Figure 3. 1 The DPI-7 scanner. Left: view of the imaging sensors; Right: view of the screen.

As outlined in Chapter 2, the data collection process of the DPI-7 is similar to that of the first generation Microsoft Kinect, since they both use the PrimeSense technology. However, it was noticed early in the data processing that there was a severe complication in the way that the DPI-7 stores its data. In order to make data collection easier for the user, the DPI-7 software automatically orients the point clouds being collected by the PrimeSense device to align with the internal accelerometer's measured direction of gravity. This ensures that the Z direction of the

resulting point cloud always points ‘upwards’ relative to the scene being collected, making the data collection more intuitive. However, there is no way to access the raw data that was used to create that point cloud (i.e. the un-rotated point cloud, the range image and the rotation information are all concealed from the user). In addition, it is not possible to estimate the internal orientation parameters, nor the image measurements from the point cloud data. The error model and calibration method proposed in this chapter can be used to circumvent these issues and correct the measurements directly in the point cloud rather than in the raw observations.

3.2 Definition of Models

3.2.1 Registration and Error Model

The model being used in this calibration system is

$$\bar{\mathbf{r}}^{\mathrm{obj}} = \mathbf{R}_{\mathrm{DPI}}^{\mathrm{obj}}\mathbf{r}^{\mathrm{DPI}} + \mathbf{t}_{\mathrm{DPI}}^{\mathrm{obj}} + \Delta(\mathbf{r}^{\mathrm{DPI}}) \tag{3.1}$$

where $\bar{\mathbf{r}}^{\mathrm{obj}} = [\bar{\mathrm{X}}^{\mathrm{obj}} \quad \bar{\mathrm{Y}}^{\mathrm{obj}} \quad \bar{\mathrm{Z}}^{\mathrm{obj}}]^{\mathrm{T}}$ are the reference coordinates in their own object space coordinate system; $\mathbf{R}_{\mathrm{DPI}}^{\mathrm{obj}}$ is the 3×3 rotation matrix from the DPI-7 coordinate system to the object space system; $\mathbf{r}^{\mathrm{DPI}} = [\mathrm{X}^{\mathrm{DPI}} \quad \mathrm{Y}^{\mathrm{DPI}} \quad \mathrm{Z}^{\mathrm{DPI}}]^{\mathrm{T}}$ are the target coordinates in the DPI coordinate system; $\mathbf{t}_{\mathrm{DPI}}^{\mathrm{obj}} = \left[\mathrm{X}_{\mathrm{DPI}}^{\mathrm{obj}} \quad \mathrm{Y}_{\mathrm{DPI}}^{\mathrm{obj}} \quad \mathrm{Z}_{\mathrm{DPI}}^{\mathrm{obj}}\right]^{\mathrm{T}}$ is the translation vector between the DPI-7 coordinate system and the object space system; and $\Delta(\mathrm{r}^{\mathrm{DPI}})$ indicates that the systematic errors present in the data are a function of the data itself, assuming that these can be expressed as an additive correction.

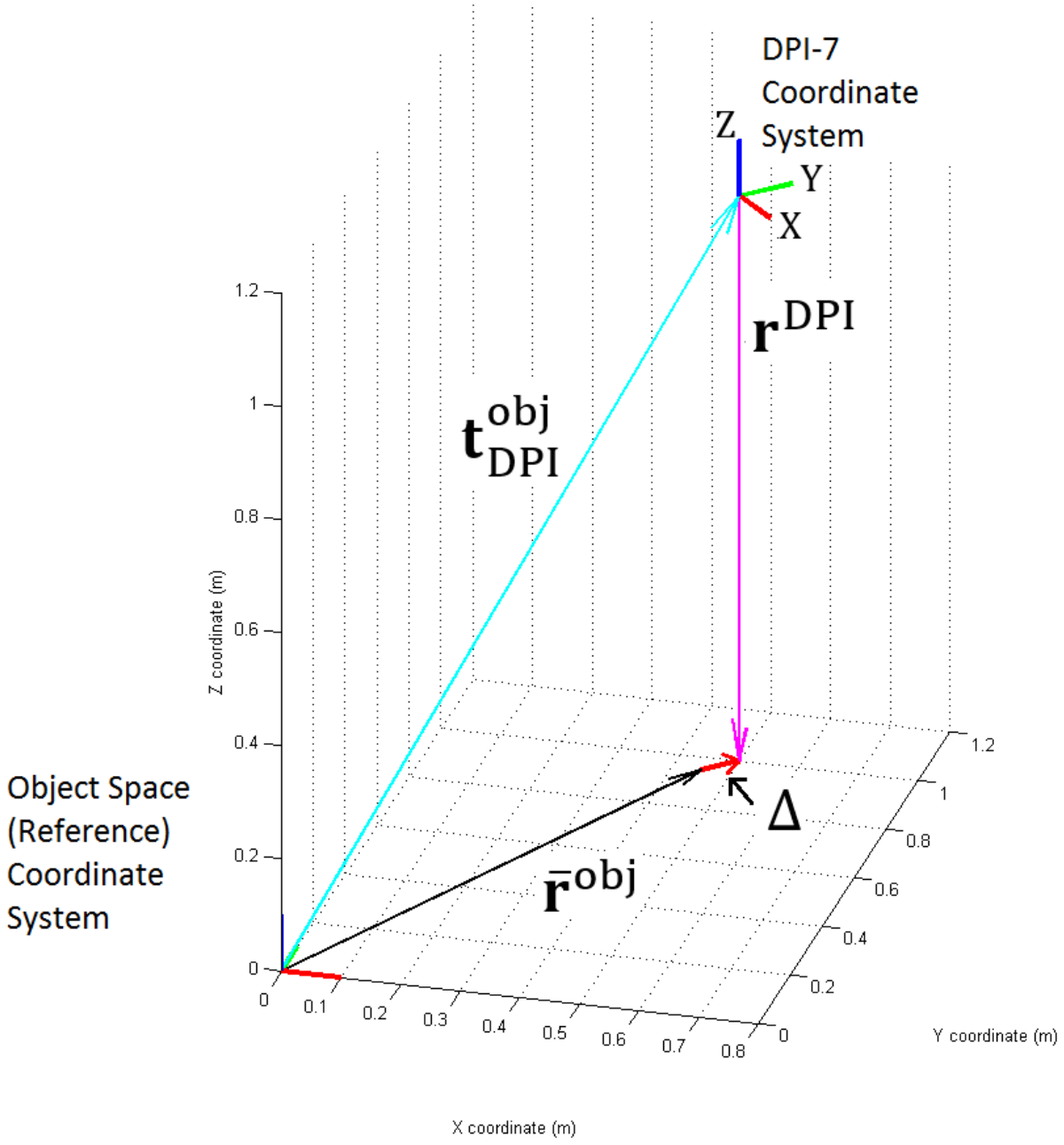

Figure 3. 2 The relationship between the different coordinate systems in the error model

Section 3.5.2 will show that this is the case, for the error model being developed here. Figure 3.2 shows the relationship between the different coordinate systems and error model terms.

In this model, the registration parameters, $\mathbf{R}^{obj}_{DPI}$ and $\mathbf{t}^{obj}_{DPI}$, are determined using Horn's closed-form solution for rigid body registration [6]. A more thorough calibration method would

estimate the error parameters and registration parameter simultaneously. However, one of the goals of this procedure is to be a simple enough model for non-experts, such as archaeologists, to be able to follow. With that in mind, the registration parameters and the error model parameters are estimated in separate steps.

Once the registration parameters have been estimated using Horn's method, the reference coordinates can be converted into the DPI-7 coordinate system, which will simplify the error model significantly. This conversion can be formulated as:

$$\bar{\mathbf{r}}^{\mathrm{DPI}} = \mathbf{R}_{\mathrm{obj}}^{\mathrm{DPI}} \left(\bar{\mathbf{r}}^{\mathrm{obj}} - \mathbf{t}_{\mathrm{DPI}}^{\mathrm{obj}} \right) \tag{3.2}$$

where $\bar{\mathbf{r}}^{\mathrm{DPI}}$ are the reference coordinates translated to the DPI-7 coordinate system, $\mathbf{R}_{\mathrm{obj}}^{\mathrm{DPI}}$ is the rotation matrix from the reference object space coordinate system to the DPI-7 coordinate system, and is equal to $\mathbf{R}_{\mathrm{DPI}}^{\mathrm{obj}\ \mathrm{T}}$. Substituting this into equation (3.1) simplifies the error model such that:

$$\bar{\mathbf{r}}^{\mathrm{DPI}} = \mathbf{r}^{\mathrm{DPI}} + \Delta(\mathbf{r}^{\mathrm{DPI}}) \tag{3.3}$$

with $\Delta(\mathbf{r}^{\mathrm{DPI}})$ expressed in the DPI-7 coordinate system.

Through investigation of the data, as outlined in section 3.5.2, the error model used to represent $\Delta(\mathbf{r}^{\mathrm{DPI}})$ is as follows:

$$\Delta(\mathbf{r}^{\mathrm{DPI}}) = \begin{bmatrix} S & 0 & 0 \\ 0 & S & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{r}^{\mathrm{DPI}} + \begin{bmatrix} \mathrm{x}^{\mathrm{c}} \\ \mathrm{y}^{\mathrm{c}} \\ 0 \end{bmatrix} \tag{3.4}$$

where S is the scale factor applied to the X and Y coordinates, and x^{c} and y^{c} are the additive offsets between the origin in the DPI coordinate system and the measured center of scale. These differences are expressed in the DPI-7 coordinate system. Substituting equation (3.4) for $\Delta(\mathbf{r}^{\mathrm{DPI}})$ gives the following formulation:

$$\bar{\mathbf{r}}^{\mathrm{DPI}} = \mathbf{r}^{\mathrm{DPI}} + \begin{bmatrix} S & 0 & 0 \\ 0 & S & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{r}^{\mathrm{DPI}} + \begin{bmatrix} \mathrm{x}^{\mathrm{c}} \\ \mathrm{y}^{\mathrm{c}} \\ 0 \end{bmatrix} \tag{3.5}$$

which in turn gives:

$$\Delta(\mathbf{r}^{\mathrm{DPI}}) = \bar{\mathbf{r}}^{\mathrm{DPI}} - \mathbf{r}^{\mathrm{DPI}} = \begin{bmatrix} \delta X \\ \delta Y \\ 0 \end{bmatrix} \tag{3.6}$$

where $\Delta(\mathbf{r}^{\mathrm{DPI}}) = [\delta \mathrm{X}\ \delta \mathrm{Y}\ 0]^{\mathrm{T}}$ is the difference between the reference target coordinates and the measured target coordinates in the DPI-7 coordinate system. This is the model used to formulate the least squares model in the form of $\boldsymbol{\ell} = \mathbf{A}\mathbf{x}$,

$$\begin{bmatrix} \delta \mathrm{X}^{1} \\ \delta \mathrm{Y}^{1} \\ \vdots \\ \delta \mathrm{X}^{\mathrm{n}} \\ \delta \mathrm{Y}^{\mathrm{n}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \mathrm{X}^{\mathrm{DPI}^{1}} \\ 0 & 1 & \mathrm{Y}^{\mathrm{DPI}^{1}} \\ \vdots & \vdots & \vdots \\ 1 & 0 & \mathrm{X}^{\mathrm{DPI}^{\mathrm{n}}} \\ 0 & 1 & \mathrm{Y}^{\mathrm{DPI}^{\mathrm{n}}} \end{bmatrix} \begin{bmatrix} \mathrm{x}^{\mathrm{c}} \\ \mathrm{y}^{\mathrm{c}} \\ \mathrm{S} \end{bmatrix} \tag{3.7}$$

where $\delta \mathrm{X}^{\mathrm{n}}$ and $\delta \mathrm{Y}^{\mathrm{n}}$ are the differences between the reference and the measured target locations in the X and Y directions for the n^{th} point, respectively, and $\mathrm{X}^{\mathrm{DPI}^{\mathrm{n}}}$ and $\mathrm{Y}^{\mathrm{DPI}^{\mathrm{n}}}$ are the measured DPI-7 coordinates of the n^{th} point.

The misclosure vector can be calculated as

$$\mathbf{w} = \mathbf{A}\mathbf{x} - \boldsymbol{\ell} \tag{3.8}$$

from which the normal equations (unweighted least-squares) are formulated

$$\mathbf{N} = \mathbf{A}^{\mathbf{T}}\mathbf{A} \tag{3.9}$$

$$\mathbf{u} = \mathbf{A}^{\mathbf{T}}\mathbf{w} \tag{3.10}$$

which gives the estimated parameter vector

$$\hat{\mathbf{x}} = -\mathbf{N}^{-\mathbf{1}}\mathbf{u} \tag{3.11}$$

which contains the optimal estimates of the calibration parameters.

3.2.2 Sphere Fitting Model

In order to accurately measure the location of the spherical targets in the DPI-7 coordinate system (i.e. $\mathbf{r}^{\mathrm{DPI}}$ in Figure 3.1), it is necessary to determine the sphere center based on point measurements on its surface. The centre estimation may be numerically weak and therefore biased due to limited sampling of the sphere, i.e. at most half of the surface can be observed in each scan. The centre estimation accuracy can be improved by introducing a known radius constraint. In this case, because the targets being used are table tennis balls, their diameter is precisely controlled.

The equation of a sphere can be expressed as:

$$(\mathrm{x_n} - \mathrm{x_c})^2 + (\mathrm{y_n} - \mathrm{y_c})^2 + (\mathrm{z_n} - \mathrm{z_c})^2 - \mathrm{r}^2 = 0 \tag{3.12}$$

where $[\mathrm{x_n}\ \mathrm{y_n}\ \mathrm{z_n}]^{\mathrm{T}}$ are the $\mathrm{n^{th}}$ point on the surface of the sphere; $[\mathrm{x_c}\ \mathrm{y_c}\ \mathrm{z_c}]^{\mathrm{T}}$is the location of the sphere's center; and r is the radius. This equation is actually a constraint, since the parameters are constrained so that the measured points lie on the surface of the sphere. This would mean that the 'observation' in this least squares model is the value zero, on the right hand side of equation (3.12). By treating the value zero as an observation, a parametric equation can be formed, which has the benefit of enabling the estimated value of the radius to be specified [4]. Necessary for this formulation is an initial estimate of the sphere's center, which is taken as the centroid of all points belonging to the sphere. Therefore, the observation equation in the form of $\boldsymbol{\ell} = \mathbf{Ax}$ can be formulated as:

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} -2(\mathrm{x_1} - \mathrm{x_c}) & -2(\mathrm{y_1} - \mathrm{y_c}) & -2(\mathrm{z_1} - \mathrm{z_c}) & -2\mathrm{r} \\ & \vdots & & \\ -2(\mathrm{x_n} - \mathrm{x_c}) & -2(\mathrm{y_n} - \mathrm{y_c}) & -2(\mathrm{z_n} - \mathrm{z_c}) & -2\mathrm{r} \end{bmatrix} \begin{bmatrix} \mathrm{x_c} \\ \mathrm{y_c} \\ \mathrm{z_c} \\ \mathrm{r} \end{bmatrix} + \mathbf{w} \tag{3.13}$$

where the misclosure vector is

$$\mathbf{w} = \begin{bmatrix} (\mathrm{x_1} - \mathrm{x_c})^2 + (\mathrm{y_1} - \mathrm{y_c})^2 + (\mathrm{z_1} - \mathrm{z_c})^2 - \mathrm{r}^2 \\ \vdots \\ (\mathrm{x_n} - \mathrm{x_c})^2 + (\mathrm{y_n} - \mathrm{y_c})^2 + (\mathrm{z_n} - \mathrm{z_c})^2 - \mathrm{r}^2 \end{bmatrix} \tag{3.14}$$

The normal equations (unweighted case) are expressed as:

$$\mathbf{N} = \mathbf{A}^{\mathbf{T}}\mathbf{A} \tag{3.15}$$

$$\mathbf{u} = \mathbf{A}^{\mathbf{T}}\mathbf{w} \tag{3.16}$$

The known sphere radius condition can be enforced with a parameter constraint, with the parameter weight, $\mathbf{P_x}$, defined as:

$$\mathbf{P_x} = \text{diag}\left(0\ 0\ 0\ \frac{1}{\sigma_{r_p}^2}\right) \tag{3.17}$$

and the parameter misclosure vector, $\mathbf{w_x}$, defined as

$$\mathbf{w_x} = \left[0\ 0\ 0\ \left(r - r_p\right)\right]^T \tag{3.18}$$

where r is the estimated radius; and r_p is the a priori radius estimate and σ_{r_p} is the standard deviation of the a priori radius. The parameter constraint is added in the following manner, according to [4]:

$$\mathbf{N} = \mathbf{A}^{\mathbf{T}}\mathbf{A} + \mathbf{P_x} \tag{3.19}$$

$$\mathbf{u} = \mathbf{A}^{\mathbf{T}}\mathbf{w} + \mathbf{P_x}\mathbf{w_x} \tag{3.20}$$

The parameter correction vector for the initial estimates of the sphere parameters is calculated as:

$$\delta\mathbf{x} = -\mathbf{N}^{-1}\mathbf{u} = \begin{bmatrix} \delta x_c \\ \delta y_c \\ \delta z_c \\ \delta r \end{bmatrix} \tag{3.21}$$

This correction vector is added to the initial estimates of the sphere parameters, and the process is iteratively repeated until no further corrections are needed (i.e. the largest value of $\delta\mathbf{x}$ is smaller than a tolerance).

3.2.3 Principal components analysis / Plane fitting model

The principal components analysis (PCA) technique is a numerical analysis tool which, given any number of input variables, returns the linear combination of those values which are uncorrelated, and mutually orthogonal. That is to say that the first principal component (PC) of a set of variables will be the linear combination that has the most variance, while the second PC will be orthogonal to the first and contain the second largest variance, and so on until the final PC will contain the least variance while remaining orthogonal to all previous PCs.

In the case of a set of points with three spatial dimensions, the PCA will return the same cloud, aligned such that the axis with the largest variance is aligned to the X axis of a new coordinate system, the second largest axis (orthogonal to the first) aligned to the Y axis, while the axis of least variance will be aligned to the Z axis. The direction of the third PC in this case is also the normal direction of (unweighted, least squares) best fit plane which passes through the centroid of the cloud.

If the set of points in the point cloud are represented by the vector:

$$\mathbf{r}_i' = [x_i \quad y_i \quad z_i]^T \tag{3.22}$$

where $[x_n\ y_n\ z_n]$ represent the x y and z coordinates of the nth point in the cloud, then the centroid of the cloud can be expressed as

$$\bar{\mathbf{r}} = \frac{\sum_{i=1}^{n} \mathbf{r}'_i}{n} = [\bar{x} \quad \bar{y} \quad \bar{z}]^T \tag{3.23}$$

And the point cloud, reduced to the centroid is

$$\mathbf{r}_i = \mathbf{r}_i' - \bar{\mathbf{r}} = [x_i - \bar{x} \quad y_i - \bar{y} \quad z_i - \bar{z}]^T \tag{3.24}$$

which changes the data to be zero-mean. The covariance matrix of the reduced point cloud is calculated as

$$\mathbf{C} = \frac{1}{\mathrm{n}-1}\sum_{i=1}^{n} \mathbf{r}_\mathrm{i}\,\mathbf{r}_\mathrm{i}^\mathbf{T} \tag{3.25}$$

While the eigenvalues can be calculated using the equation

$$|\mathbf{C} - \lambda\mathbf{I}| = 0 \tag{3.26}$$

And eigenvectors calculated as

$$(\mathbf{C} - \lambda_\mathrm{i}\mathbf{I})\mathbf{e_i} = 0 \tag{3.27}$$

If the eigenvectors are normalized to unit vectors, the directions of the principal components in the cloud are $[\mathbf{e_1}\ \mathbf{e_2}\ \mathbf{e_3}]$ such that eigenvector $\mathbf{e_i}$ corresponds to eigenvalue λ_i, and $\lambda_1 \geq \lambda_2 \geq \lambda_3$. The magnitude of the eigenvalues correspond to the variance of the point cloud in the direction of the respective eigenvector. Knowing that the plane of best fit has the least variation in its normal direction, then the plane of best fit can be found using

$$\mathbf{e_3} = [\mathrm{a\ b\ c}]^\mathrm{T} = \vec{\mathbf{n}} \tag{3.28}$$

And

$$\mathrm{d} = \vec{\mathbf{n}}^\mathrm{T}\bar{\mathbf{r}} \tag{3.29}$$

Such that the equation of a plane is:

$$\mathrm{ax} + \mathrm{by} + \mathrm{cz} - \mathrm{d} = 0 \tag{3.30}$$

Since $[\mathbf{e_1}\ \mathbf{e_2}\ \mathbf{e_3}]$ are mutually orthogonal, and they are all normalized to unit vectors, then they can be used as a rotation matrix.

$$\mathbf{r}_\mathrm{PCA} = [\mathbf{e_1}\ \mathbf{e_2}\ \mathbf{e_3}]\mathbf{r} \tag{3.31}$$

where the first row of $\mathbf{r}_\mathrm{PCA}$ corresponds to the 'along plane' coordinate, the second row the 'across plane' coordinate, and the third corresponds to the 'out-of-plane' coordinate or orthogonal direction. It is this last value, the orthogonal distance, that is used to help classify points in the next section.

However, the cloud itself may not actually have a planar shape. In these cases, it may be helpful to test the shape of a neighbourhood of points. PCA is able to assist with this task as well. For a 3D point cloud, the eigenvalues correspond to the variance in the direction of the eigenvectors, as shown in [24]. This gives the standard deviation in those directions as

$$[\sigma_1, \sigma_2, \sigma_3] = \left[\sqrt{\lambda_1}, \sqrt{\lambda_2}, \sqrt{\lambda_3}\ \right] \tag{3.32}$$

These standard deviations can now be used to calculate the approximate shape of the neighbourhood of points, such

$$a_{1D} = \frac{\sigma_1 - \sigma_2}{\sigma_1} \tag{3.33}$$

$$a_{2D} = \frac{\sigma_2 - \sigma_3}{\sigma_1} \tag{3.34}$$

$$a_{3D} = \frac{\sigma_3}{\sigma_1} \tag{3.35}$$

where the a_{1D} indicates the relative linearity of the cloud, a_{2D} indicates the relative planarity, and a_{3D} the relative volumetric shape of the cloud, as defined in [24]. Taking the maximum of these values gives the estimated shape of the neighbourhood. In section 3.4, the 'shape' parameter, used to help classify point as target/background is equation 3.35.

In order to determine what the best neighbourhood radius is, the entropy of these shape parameters (Equations 3.33-35) can be used,

$$e = -\sum_{n=1}^{3} a_{nD}\ \log a_{nD} \tag{3.36}$$

Conventionally, entropy is the measure of the disorder of a system. A low entropy indicates that the neighbourhood shape is in a high state of order, meaning it is a close match to the largest shape parameter. Conversely, a large entropy indicates that there is a low state of order, or that there is a poor match (i.e. the neighbourhood of points is not clearly linear, planar or volumetric).

While this can be used to dynamically estimate the neighbourhood shape for each point in the cloud, in this case it was used to select the most appropriate neighbourhood size from a range of possible values for detecting the spheres.

3.3 Experimental Procedure

3.3.1 Calibration Field

Figure 3. 3 Calibration Field

In order to estimate the systematic errors of the DPI-7 sensor, a calibration field, seen in Figure 3.3, made of a rigid backboard (0.91m by 1.22m, 6mm thick medium-density fibreboard) and 42 spherical targets was constructed. Spherical targets were selected rather than planar targets (i.e. black and white checker board targets) because spherical targets are omnidirectional, making their measured position well determined from any viewing angle. Table tennis balls were used as the spherical targets because they have a very reliable size and shape (40mm diameter, ± 0.5mm [33]), are sufficiently inexpensive to purchase in bulk and are light enough to be permanently fixed to a back board with hot glue while maintaining their shape and location without risk of becoming dislodged under their own weight. The locations of the targets were selected such that the pattern could be uniquely orientable, but still fill the field of view of the DPI-7 without leaving significant gaps in the center of the image. The mean separation between targets is 11cm.

3.3.2 DPI-7 Data Acquisition

The experiment was performed in a low-traffic, windowless room, where the air temperature was constant (20°C) and lighting conditions (overhead fluorescent bulbs) were constant. This environment was selected in order to ensure repeatable conditions, and to limit the number of un-controlled factors. In order to limit the sagging effects of gravity on the flatness of the target field, it was placed flat on a level floor during data capture. The DPI-7 scanner was allowed to warm-up for 10 minutes before any scans were collected, as recommended in Chow et al [32]. A single frame was collected at each of forty different positions and orientations relative to the target field. These positions varied in their distance from the target field, as well as in the angles between the sensor and the targets. The 57.5° horizontal field of view of the DPI-7 meant that data could be acquired from a minimum range of 60 cm (the minimum range quoted by the manufacturer) to a maximum of 2 m. The latter is lower than the manufacturer-specified maximum range (as seen in Table 3 density of the targets was too low. Once each frame had been captured, a "Global Optimization" was performed, a proprietary software feature that eliminates mixed pixels and performs a fine registration on the collected clouds.

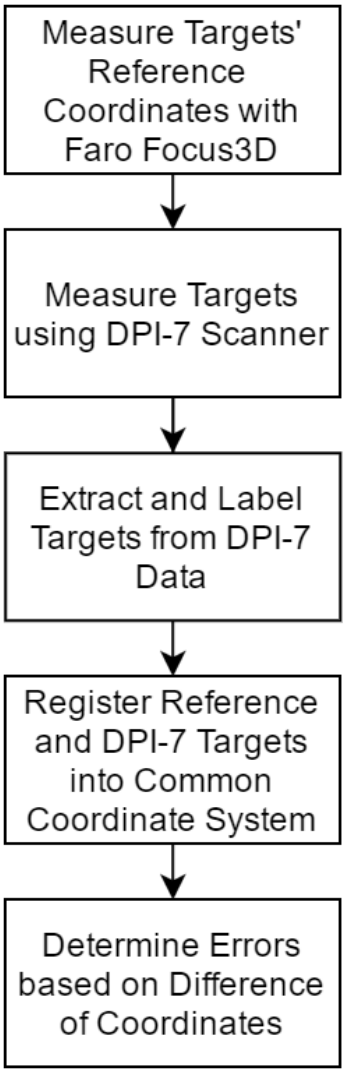

Figure 3. 4 Flowchart of methodologies

Table 3. 1 Reported measurement accuracy of the DPI-7 scanner, [34]

Range (m)	Typical Accuracy (RMS)	Minimum Accuracy (RMS)
< 1m	0.2%	0.4%
1m to 2m	0.5%	0.8%
2m to 3m	0.8%	1.2%
>3m	Not Specified	Not Specified

The images collected of the calibration field were well distributed across the entire sensor range, in many different convergent orientations. Some measurements were oblique to the calibration field, with varying rotations about the depth axis and were varying distances from the calibration field. Others were convergent, imaging the calibration field at low angles, also at varying in rotation about the depth axis and in their distance from the scanner. In all, 40 scans were collected, with the calibration field in many different positions and orientations (relative to the DPI-7 sensor) as seen in Figure 3.5, which shows (A) the scan locations relative to the target points in the XY plane, and (B) relative to the XZ plane. Notice that in Figure 3.5B, the Z axis of all images are parallel, in spite of the convergent imaging. This is due to the internal rotation of the point cloud, such that they are aligned with gravity.

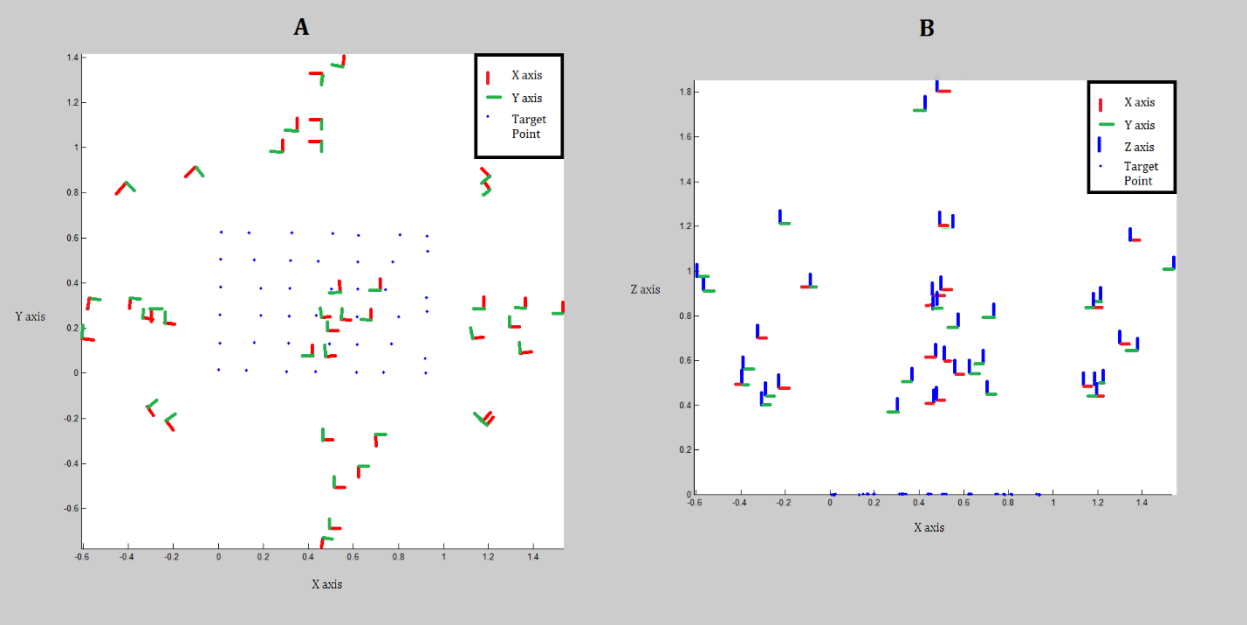

Figure 3. 5 A) Image locations relative to Target points (X Y Plane) B) Image locations relative to Target Points (X Z Plane)

The purpose of the convergent imaging is to de-correlate the exterior orientation parameters of the sensor from the estimated calibration parameters, and to help determine the impact that the range has on the estimated point precision. However, due to the modifications that the device makes to the measurements (i.e. rotation of the Z axis to align with gravity, as seen in Figure 3.5B), most of the collected data was not usable in this form of calibration. Ultimately, only the 8 normal scans of varying distances from the calibration field (between 0.6 and 2 meters) were able to be used in estimating the calibration parameters.

In order to have a reliable, independent estimate of the relative position of all of the spherical targets to which the DPI-7 estimates can be compared, the calibration field was scanned with a more accurate scanner, the Faro Focus3D terrestrial laser scanner. This scanner was selected due it its availability, and because the range measurements to the surface of the spheres did not exhibit errors due to multipath or subsurface scattering, unlike other available scanners. A comparison of the quoted specifications for both the Faro Focus3D and the DPI-7 is given in Table 3.3. The entire calibration procedure is summarized in Figure 3.4.

Table 3. 2 Comparison of manufacturer specifications of the Faro Focus3D terrestrial laser scanner and the DotProduct's DPI-7 scanner [34] [31]

Scanner	Point Density	Range noise	Max Range
DPI-7	≤ 1.7mm at 1m distance	5mm at 1m	~3.3m
Faro Focus 3D	≤0.16mm at 1m distance	0.6mm at 10m	~120m

3.4 Automated target extraction

Automated target extraction can greatly increase the speed at which large amounts of calibration data can be processed by removing the slow manual selection part of data processing. Using manual techniques, individual targets must be selected from each point cloud. Having many targets (which is advantageous for the quality of the calibration) can lead to a very tedious process of target identification, which may tempt the user to use fewer targets or images than they may otherwise. Using automated target extraction, the user is free to use as many targets that they can and still take have a quick processing time.

In order to develop a consistent method of automatically extracting the sphere from the calibration data, the .ptx files exported from the DPI-7 device were first cropped to just the calibration field. This was done to limit the search space, and to ensure consistency in the data so that extraction can be done more easily. This process was done in *Leica Cyclone*.

Distinguishing between the background of the target field and the spherical targets themselves took several steps. One reason for this was that the DPI-7 measurements were rather noisy, which resulted in the background to appear to be bumpy and partly curved rather than smooth and flat. This meant that simple methods of extracting the spheres from the background data would not work, so the process needed to be more rigorous and therefore more complex.

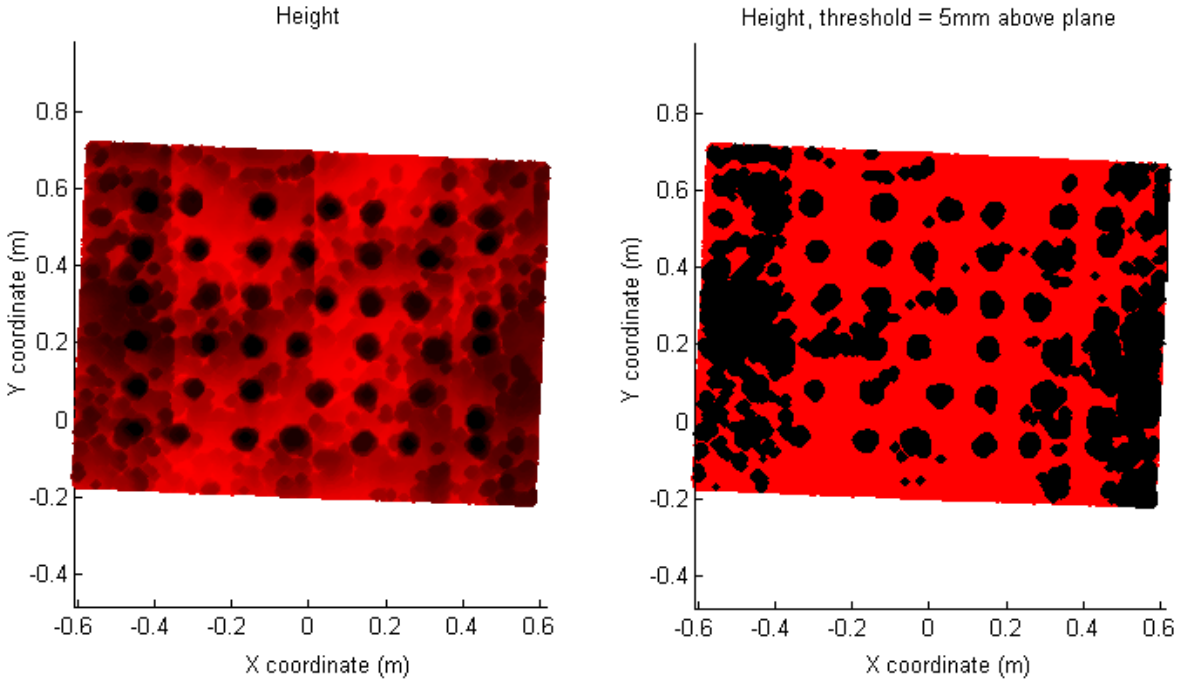

Figure 3. 6 Left: Height map of point cloud Right: Classification of points with threshold = 5mm

Several different metrics were used to determine if a point was part of the background or if it belonged to one of the spheres. First, the height was used, as can be seen in Figure 3.6. This was determined by fitting a planar section to the point cloud, and determining the orthogonal distance that each point was from that surface. Points with the largest orthogonal distance were more likely to be targets, since the targets stood above the background. This was not a faultless method, however, since distortions in the DPI-7 caused the periphery of the data to distort towards

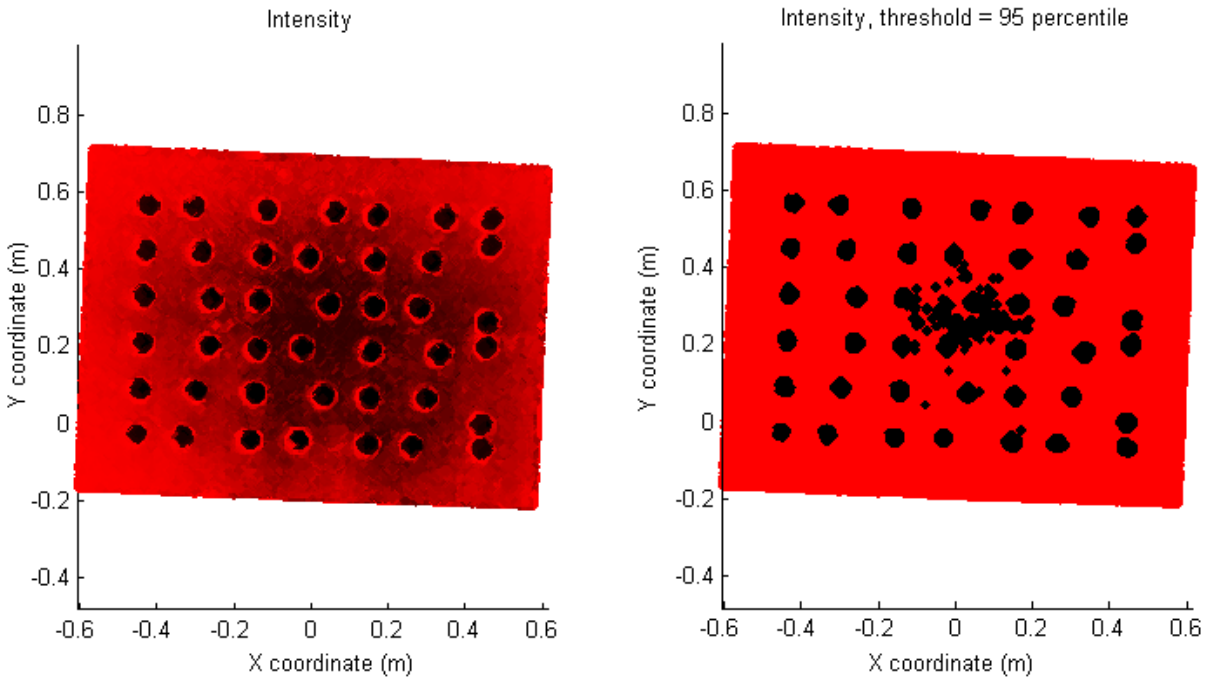

Figure 3. 7 Left: Intensity map of point cloud Right: Classification of points with threshold = 95 percentile

the camera, causing the flat board to appear to be curved. This made the edges of the board also appear to lay above the best fit plane. Additionally, discontinuities in the height are visible in Figure 3.6. This is a known phenomenon in data collected from PrimeSense sensors, as noted in [32].

Another metric that was used was the intensity, or the amount of infra-red light that was reflected back to the infra-red sensor, as can be seen in Figure 3.7. The white spherical targets reflected a large amount of light, and the background generally reflected less, so this was also a good indication of which points belonged to targets. This was again not sufficient however, since the oblique nature of the collected point clouds put the infra-red light source directly facing the board, which would reflect large amounts of light in a specular manner in small areas.

The final metric that was used to classify points into targets and background was the 'shape' of their neighbourhood of points. For each point in the sample, the points within a

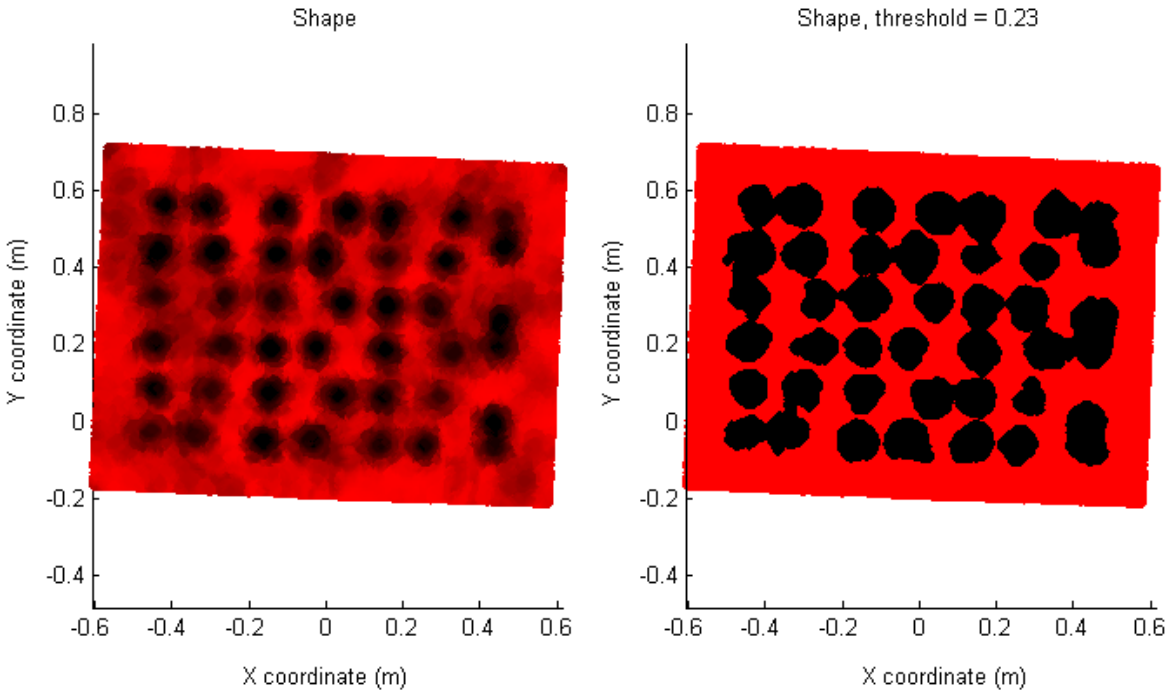

Figure 3. 8 Left: Shape map of point cloud Right: Classification with threshold = 0.23

neighbourhood radius of 5 cm were collected, and the value a_{3D} was collected using equations (3.22) through (3.34). This neighbourhood radius was determined by testing several neighbourhood radii, and using the one which had the minimum entropy, calculated using equation (3.36). In this case, the background was expected to be primarily planar, while the targets were expected to be primarily volumetric. This can be seen in Figure 3.8, where the background is primarily planar (i.e. light in colour) while the spheres are primarily volumetric (i.e. appear black

in color). However, since this process was very computationally intensive, only points which passed the previous two tests (i.e. height and intensity) were given a shape parameter. This reduced the number of calculations necessary since calculating the neighbourhood shape of a point which did not pass one of the two previous tests would have been redundant.

Thresholds of height > 5mm, intensity of the 95 percentile and a_{3D}>0.23 were set.

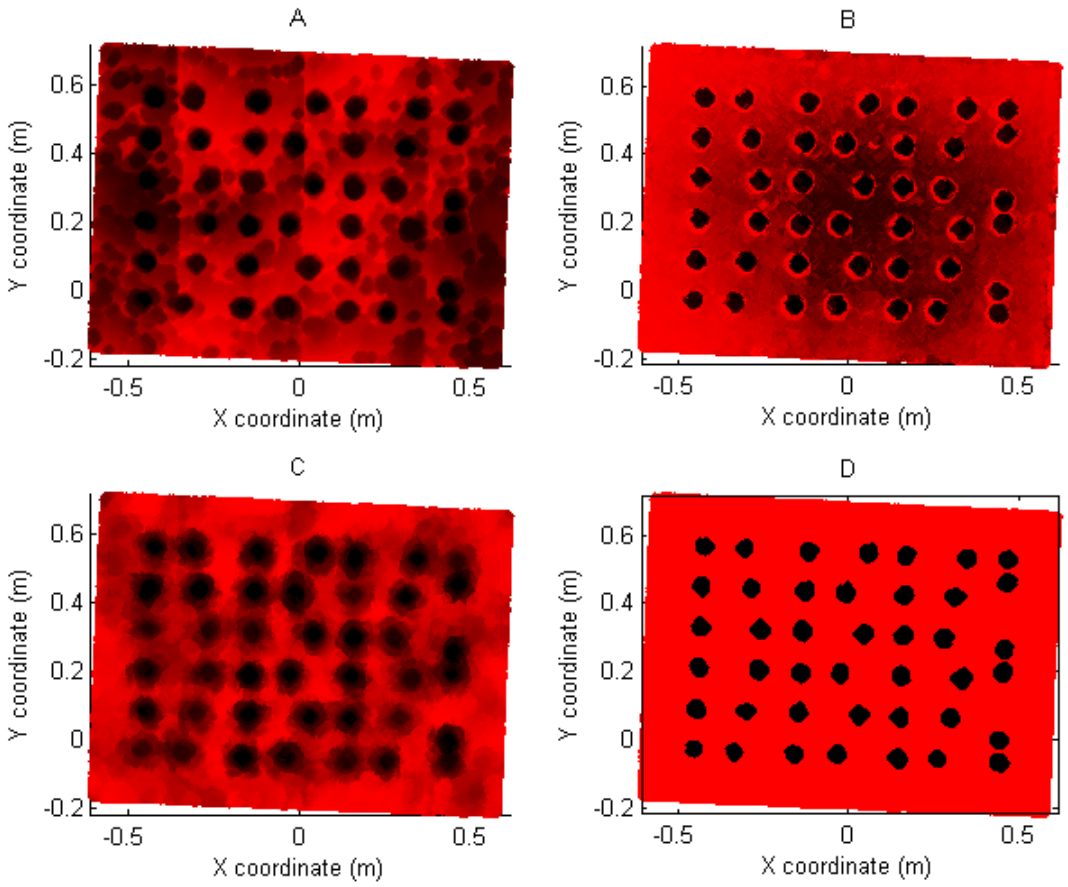

Figure 3. 9 Target/backboard differentiation. A) Height map of calibration field. B) Intensity map. C) Shape map. D) Final results

strong classification. An example of the classification can be seen in Figure 3.9, where the height, intensity, shape, and the final extraction can be seen. In all cases, dark colours represent that the points are more likely to be considered as belonging to a sphere than the background.

In order to determine the reliability of the thresholds used here, a sensitivity analysis was performed. This was done by varying the value of the threshold for each of the different tests by a factor of $\pm 20\%$, to determine if this variation has a significant impact on the classification of points.

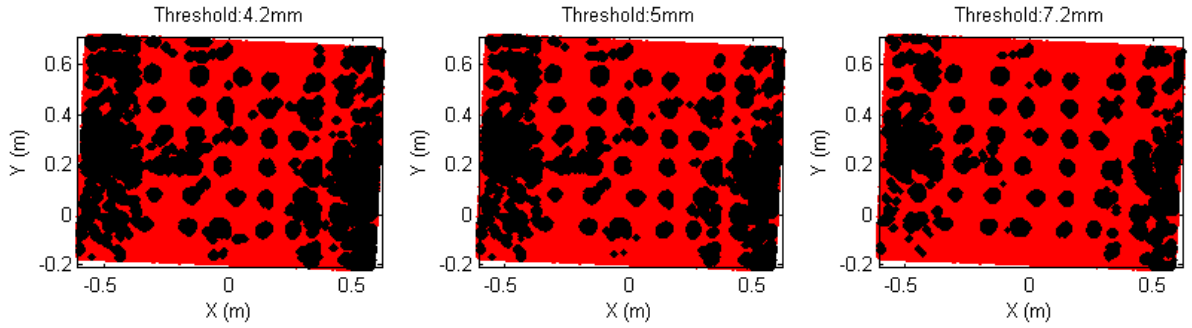

Figure 3. 10 Height Threshold Sensitivity Analysis.

For the height, the original threshold was 5mm above the plane of best fit, so the test thresholds were set to 4.2mm, and 7.2mm above. Figure 3.10 demonstrates the change is the classification that these different thresholds make. The difference between these thresholds is not very large, indicating that the height parameter is fairly stable.

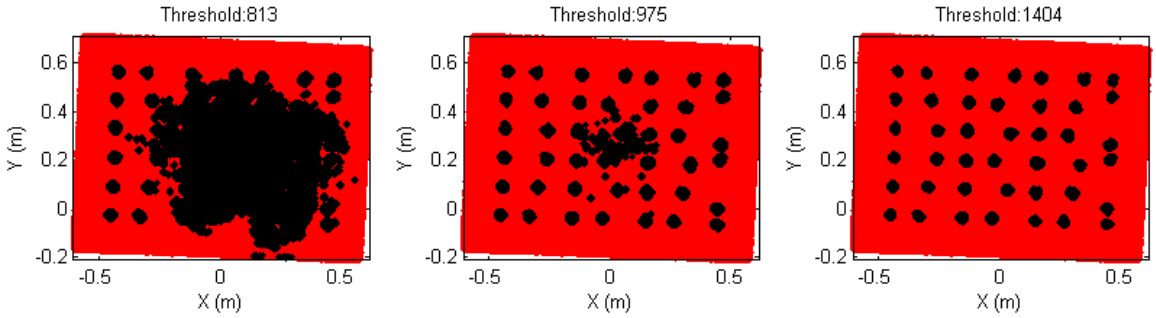

Figure 3. 11 Intensity Threshold Sensitivity Analysis

For the intensity, the original threshold was set to be the brightness at which only the top 5% of brightest points were included, which was the value 975 (unitless). Therefore the two thresholds used for testing were 813, and 1404. Figure 3.11 demonstrates the impact that this

variation has. The threshold of 813 is clearly an inappropriate choice, since an enormous portion of point belonging to the background are classified as targets. On the other hand, the threshold of 1404 appears to be a better selection than even 975, since it contains no false-positive points. However, it was determined that a higher threshold would not always a better choice, since in images which were oblique to the IR camera, many targets were dimmer than the threshold value would permit. A more permissive threshold in these cases would allow for other parameters to filter out the false-positives.

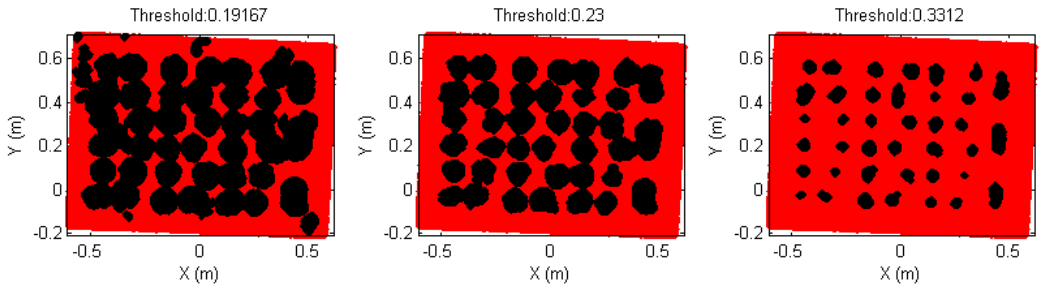

Figure 3. 12 Shape Threshold Sensitivity Analysis

Finally, the threshold for the shape parameter was tested. The original threshold was 0.23, and so the test thresholds were 0.192, and 0.331. It can be seen in Figure 3.12 that the threshold is fairly sensitive to variation, producing a large number of extra points for the lower threshold, and excluding point around the edges of targets in the higher threshold. Similar to the intensity threshold, a more permissive threshold was chosen.

After the points were classified as belonging to a target, further processing was used on these points, isolated from the background, to refine the results. Most of the false positives were stray points, meaning they had few or no immediate neighbours. In this case, a minimum neighbourhood density was used to eliminate these points. A radius of 2cm around each point was searched, and if there were fewer than 15 other points within that volume, then the point in question

was removed from the cloud of target points. These points were removed only after the entire cloud was searched, so the number of point was stable during the search.

The next task was separating each target from the others. This was done by selecting a random seed point, and collecting points in its neighbourhood of a radius of 44mm, which is equal to twice the radius of the targets, plus an additional 10%. This allows a point landing anywhere on the sphere to consider all points elsewhere on the sphere to be in its neighbourhood, plus an additional margin for errors. These points were given the target ID of 1, which was an arbitrary label. Then, a new random point was selected from the set of points that have not yet been given an ID. The neighbourhood of points for that point in now also selected, using the same radius as before. If any points that have been newly selected already had an ID, then the point in question is assigned to the group that has a closer centroid. This process is repeated until all points in the target points set have an ID.

Occasionally, some points may not match the IDs of the other points in their immediate neighbourhood. This only ever occurs to points that belong on the edge of a target, and usually only a couple of points at a time. The principal cause of this is if the erroneously labeled point is assigned to the wrong neighbourhood early on, and it lies outside the search radius of the point selected to identify its correct neighbourhood. In these cases, if a target ID has fewer than 16 points, then that neighbourhood of points is simply removed from the cloud. It may be more optimal to reassign these points to their nearest neighbourhood, but this was not implemented.

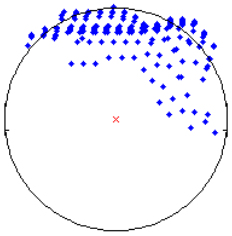

Figure 3. 13 Example of a sphere fit to points

Once each surface point has been grouped into their neighbourhoods, a sphere was fit to each one, as can be seen in Figure 3.10. This was done using the constrained least squares model

that can be seen in equations (3.12) to (3.21). There is then a process of interactive correspondence determination, which asks the user to determine the correspondence of three targets, and uses back projection to determine the correspondence of the rest of the targets.

It is likely possible to fully automate this process, for instance by finding the largest span between targets in the point cloud, orienting the other targets according to that vector, then checking the different permutations of possible orientations of the point cloud to determine which is the correct labeling of the targets. This system works well if all targets are present in the cloud, since it's likely that only a few orientations would be even close to the correct orientation. However, this method works much worse if only a small selection of points are available, since so many more combinations and orientations of points are possible matches. Due to this complication, it was decided that a semi-manual target labeling approach was an appropriate choice for this application.

After this step, all of the targets in each point cloud have been extracted, modeled and correctly labeled according to the labels in the prior target location file. These target locations are then used in the next steps, including the calibration.

3.5 Experimental Results

3.5.1 Model Identification

Once the targets were correctly labeled and registered to the prior target locations, the difference between the prior target location and the measured target center were calculated. One of the most telling figures is a comparison of the error vectors compared to the target locations in the XY plane for all 8 stations, as seen in Figure 3.11. There's a clear systematic effect that the

position of the point has on the magnitude and direction of the error, specifically that a point further from the center of the scan has a larger error vector, directed towards the center.

Another view that demonstrates the relationship between object points and their errors is a graph of X coordinates on the X axis, with errors in the X coordinate on the Y axis, seen in Figure

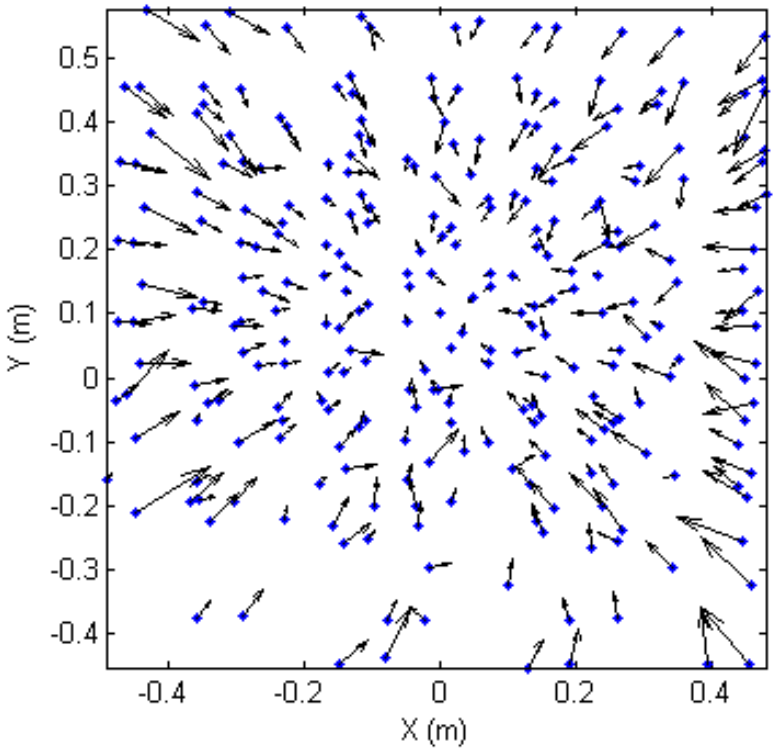

Figure 3. 14 Vector plot of errors. Vectors Scaled by a factor of 5 for visibility

3.12. This produces a scatter plot indicating a high degree of correlation between the two values, indicating a linear relationship between the two. Plotting the Y coordinate versus the error in the Y direction produces a nearly identical plot, indicating that the slope of the correlation is the same. This reveals that there is a single scale factor error that impacts the X and Y coordinate errors in equal measure. The model used in this instance is outlined in equations 3.1 to 3.11

The improvements gained by applying the calibration can be seen in seen in Figure 3.12. Figure 3.12A shows the original errors in X and Y, plotted against their respective X and Y coordinates. There is a clear linear correlation between these values. Figure 3.12B shows the errors

in X and Y, after the corrections have been applied. The recalculated differences in X and Y (Figure 3.12B) clearly indicate that the systematic errors in X and Y were successfully removed with the inclusion of the scale factor in the model.

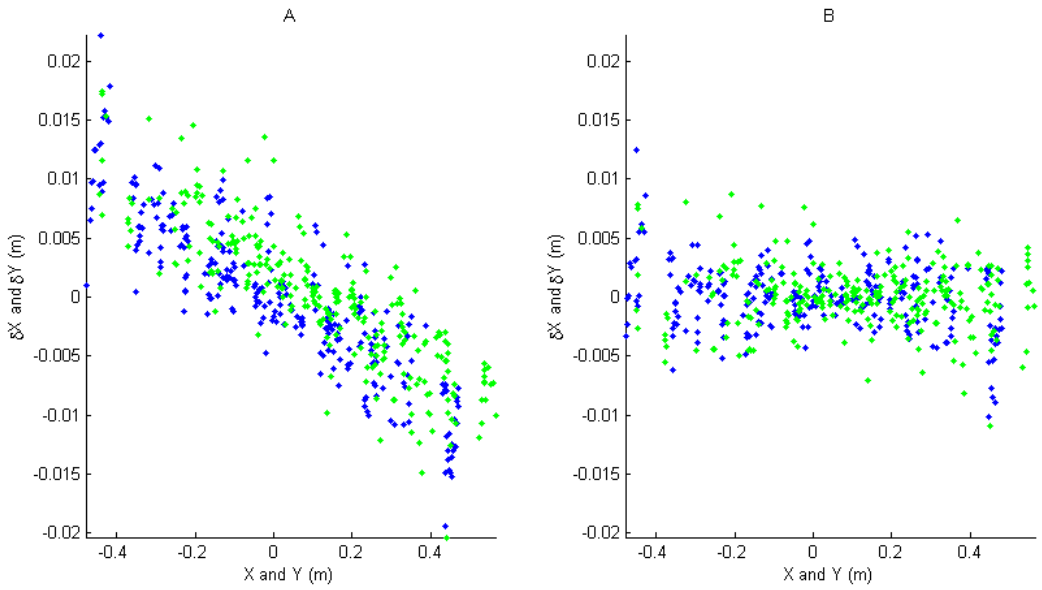

Figure 3. 15 A) Un-modeled X (blue) and Y (green) errors. B) Calibrated X and Y errors

The estimated values of the model parameters are presented in Table 3.3. The values of x^c and y^cwere expectedly small in magnitude, while the scale factor shows that the X and Y coordinates measured by the DPI-7 are smaller than reality by a factor of 2% (i.e. the DPI-7 measurements are 98% the size that they should be). This difference in scale can contribute up to 4 centimeters of error within the 2 metre operational range of the device.

Table 3. 3 Estimated values of calibration parameters

X^c	Y^c	S
$0.2\ \text{mm} \pm 0.22\text{mm}$	$2.4\ \text{mm} \pm 0.23\text{mm}$	-0.0218 ± 0.00061

The correlation coefficients of the parameters can be seen in Table 3.4. These coefficients are all quite low, the largest in magnitude -0.29 between Y^c and S. This gives the indication that the parameters are generally functionally independent and that the solution is numerically stable.

Table 3. 4 Correlation coefficients between calibration parameters

ρ_{X^c,Y^c}	$\rho_{X^c,S}$	$\rho_{Y^c,S}$
0.01	-0.03	-0.29

Figure 3.13 show the direction and magnitude of errors (scaled by a factor of 5, for visibility) in the X and Y directions before the additional parameters were applied. Figure 3.13B shows the same information, after the calibration has been applied. It is clear that prior to the calibration there is a systematic bias in the coordinates, directed towards the center of the target field and increasing in magnitude with distance from the center. After the scale factor is applied, the errors are significantly reduced, and appear to be random in direction. This is another good indication that the systematic errors in the X and Y directions have been identified and modeled.

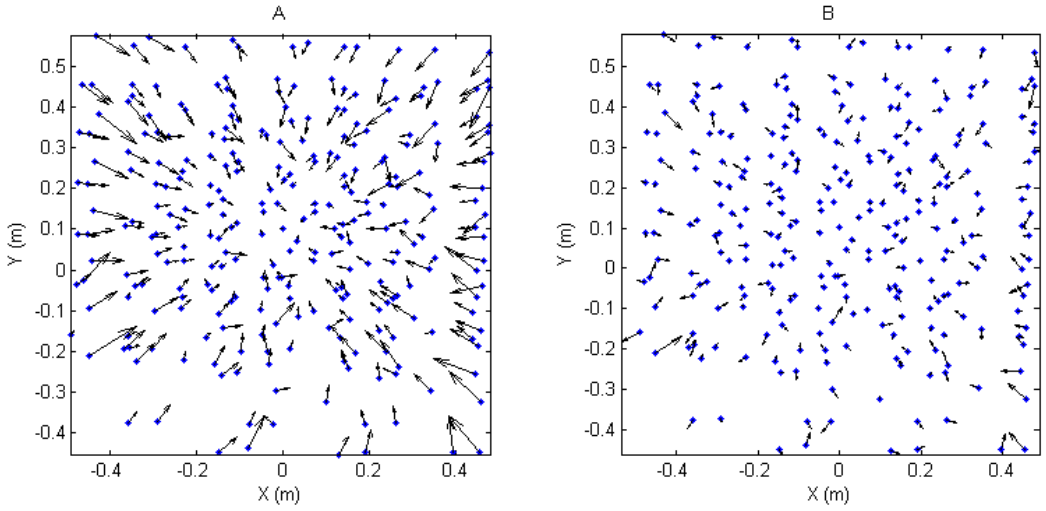

Figure 3. 16 A) Vector plot of errors, before calibration. B) Vector plot of errors, after calibration. Vectors Scaled by a factor of 5 for visibility

3.5.2 Discussion of Results

The scale factor in this calibration procedure is analogous to the principal distance of the camera in a photogrammetric calibration. This can be seen in Figure 3.13A, which resembles the distortions commonly seen in camera calibrations. This is due to the fact that, in photogrammetry, X and Y coordinates in object space are calculated using the geometry between the principal point and the image coordinates of that point, projected onto the image plane. Therefore, an error in the principal distance causes an error in the X and Y coordinates in object space, linearly proportional to the X or Y coordinate. In this case, however, the interior orientation parameters are not accessible, so corrections to measured coordinates must be made using this scale factor, rather than by adjusting the value of the principal distance. Similarly, X^c and Y^c are analogous to the principal point in photogrammetry, but are similarly inaccessible and must be applied to the point clouds directly. In this case, X and Y are analogous to the image space coordinates, and Z to the depth coordinate. In this case, Z is calculated based on the projector/NIR camera geometry, rather than parallax shift between two cameras.

Analysis of the errors in the Z direction revealed that a modelling procedure similar to that used in the X and Y directions was not required. No systematic effects in the error in the Z direction can be seen in relation to the Z coordinate, as can be been in Figure 3.14. This maybe because errors in the Z direction are entirely compensated by the displacement of the perspective center, since the lack of depth in the target field and the inability to collect convergent imagery eliminated the usual methods that can be used to de-correlate these two variables.

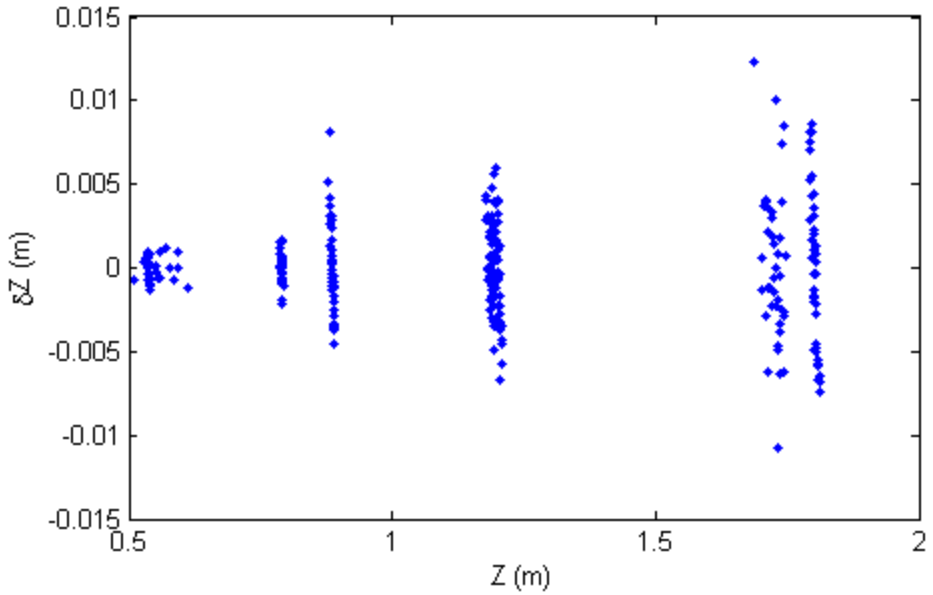

Figure 3. 17 Error in Z direction vs. Z coordinate

The RMS errors of the target coordinates before and after the scale factor was applied are summarized in Table 3.5. The RSME of the X and Y directions have been reduced by approximately 50%, using only a single scale factor parameter applied in the X and Y directions.

Table 3. 5 Comparison of RMSE of target coordinates, before and after application of scale factor

	RMSE X (mm)	**RMSE Y (mm)**	**RMSE Z (mm)**
Prior to scale factor	6.6	6.0	3.1
After scale factor	2.7	2.7	3.1
Percent improvement	59%	54%	0%

3.6 Chapter Summary

In this chapter, the metric performance of the DPI-7 handheld scanner was measured, and the device was calibrated using a custom error model. Due to the inaccessibility of the raw image measurements, the calibration was applied to the point cloud data after some internal processing was performed by the device. A scale factor error of 2% was discovered in the X and Y directions

of the device, while no error was discovered in the Z direction due to the perfect correlation between those errors and the devices' perspective center. In order to assist with the calibration, a highly automated target extraction and labeling method was also developed.

Chapter 4 - Error Propagation in TLS and Photogrammetry in Archaeological Applications

4.1 Introduction

It is popularly believed that TLS data are more precise than photogrammetric data. This is particularly true when photogrammetric data are collected using highly automated computer vision techniques. However, how much more precise is a difficult question to answer. It depends on the network geometry, the specific measurement devices and their design, the quality of automated techniques, as well as the scene itself. In order to quantify these values, all of these factors need to be considered together, since they interact and interfere with each other. For instance, the quality of the automated target extraction used in photogrammetry very much depends on the visual distinctiveness of the surface being modeled. Similarly, the reflective characteristics of the surface will affect the LiDAR range measurements, particularly if the surface absorbs and re-emits the light, or has highly specular reflections.

The procedure of quantifying the expected precision of the different measurement techniques is important because archaeologists need to be able to trust the measurements that they are making. In many cases there aren't opportunities to retake or verify these measurements after the fact, such as when the location of the feature being measured is too remote or inaccessible to make the journey again, or when the feature in question was destroyed in the process of excavation. Is these cases, the archaeologist needs to know that the measurements taken were in fact correct and reliable the first time they were collected. Additionally, it could be the case that the more precise method is excessive in its precision when compared to the less precise method, which suits the desired application very well.

In this chapter, stochastic error modeling and testing will determine what level of precision can be expected from the different measurement technologies, and therefore can help archaeologists select the right technology given a particular task.

4.2 Observational Principles

In order to determine the three-dimensional location of a point using a given instrument, it is essential to determine the behavior of the raw observations of that instrument, and the behavior of those observations as they are converted into other coordinate systems. In the case of terrestrial laser scanners, the observation space is that of spherical geometry, while in photogrammetry the observations are in a projective geometry. Both of these different geometries have implications and effects on the final object space coordinates.

When many sets of data are collected of the same scene, but from different vantage points or perspectives it is necessary to bring them into a common coordinate system so that all of the data can be considered at once, describing a common subject. In the case of TLS, this is accomplished by rotating and translating each cloud into a single common system by estimating the best fit rigid body transformation parameters. For photogrammetry on the other hand, the process is more complicated, due to the loss of information introduced in the projective geometry, which must be recovered using the collinearity equations and convergent imaging. It is therefore necessary that the image point observations be used to estimate the tie point coordinates and the camera orientation parameters simultaneously using the bundle adjustment.

4.2.1 Principles of TLS

The purpose of TLS data collection is to capture the shape and size of whatever feature that one wishes to document. This process has several steps, however, in order to faithfully represent

the feature in question. The first step is to collect data all over the feature, using scans from many different locations. This is so that no areas of interest on the feature's surface are left blank, and thus are unable to be modeled. Next, the scans from different locations must be registered together, which is to say they must be brought into the same coordinate system. This is done by placing targets around the feature during scanning which serve as points of reference between scans. These targets' positions can be reliably estimated, and their relative locations used to bring the targets into a common coordinate system.

The nature of the observations of the TLS being examined in this thesis are a range measurement from a LiDAR sensor, and one horizontal and one vertical angle. The angular measurements are made in increments, and a single range measurement is made at each one. These observations are used to model the environment around the scanner. Additionally, there are two extra observations per scan that can be used to improve the overall quality of the registration. These observations are called level constraints, and can be included due to the fact that the scanner being used has an internal level device, which allows the vertical direction of the scanner to be align with the direction of gravity.

The reliability of the estimation of the registration parameters depends heavily on the precision of the targets being used as tie points between coordinate systems. In this thesis, spherical targets are used. Spherical targets are preferable to planar targets for use with TLS because the coordinates of the target is determined using the geometry of the point cloud, rather than the intensity of the reflected laser. They are also preferable because they can be scanned from any direction, whereas for planar targets the precision is significantly decreased for high angles of incidence. The angle of incidence effect can be mitigated if the planar target is mounted on a

vertical axis and rotated to face the scanner, but this assumes that the axis of rotation passes directly through the center of the planar target, which may lead to errors if this is not the case.

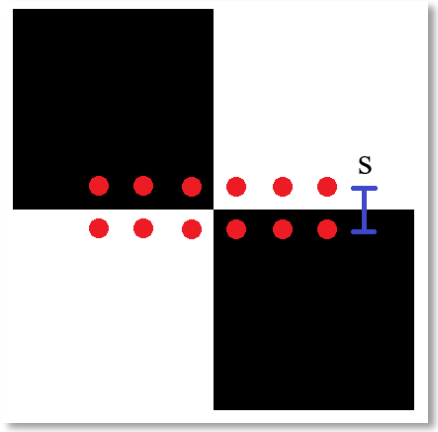

Figure 4. 1 The true location of this black and white target is unknown in the interval S

The estimation of the coordinates of a spherical target are also commonly highly redundant and reliable, particularly if the radius of the target is known. On the other hand, a planar target's coordinates are more difficult to estimate with high precision, since its location is determined (most commonly) at the intersection of the black and white sections, as seen in Figure 4.1. This relies on the intersection of black and white sections being scanned at very high point density. If they are not, then there can be a very large margin of error for the target, the interval S in Figure 4.1, as the actual intersection of black and white sections may lie anywhere between scan points. The advantage of black and white planar targets, however, is that they are much less expensive than spherical targets, and are also much more compact. Another advantage is that they can be independently observed using a total station, which can make georeferencing of the network easier.

4.2.1.1 TLS Observation Equations

The observations of a TLS device are made in spherical coordinates, but can be converted into a Cartesian coordinates using the formulation:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \rho \cos\theta \cos\alpha \\ \rho \sin\theta \cos\alpha \\ \rho \sin\alpha \end{bmatrix} \tag{4.1}$$

where ρ is the range measurement, θ is the horizontal direction and α is the vertical angle, measured from the horizon, as can be seen in Figure 4.2. This formulation assumes that the scanner's perspective center is at the center of the Cartesian coordinate system.

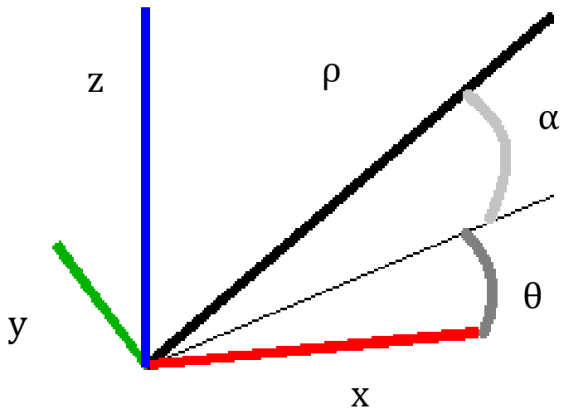

Figure 4. 2 Cartesian to spherical coordinate transform, the local coordinate system of a TLS scanner.

The precision of the point cloud, however, is estimated initially in spherical coordinates, as seen in equation 4.2.

$$\mathbf{C}_{\ell_{\rho\theta\alpha}} = \begin{bmatrix} \sigma_\rho^2 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & \sigma_\alpha^2 \end{bmatrix} \tag{4.2}$$

The errors can be propagated from the spherical observations to the Cartesian coordinates using the design matrix, **A**,

$$\mathbf{A} = \begin{bmatrix} \cos\alpha\cos\theta & -\rho\cos\theta\sin\alpha & -\rho\cos\theta\sin\alpha \\ \cos\alpha\sin\theta & \rho\cos\alpha\cos\theta & -\rho\sin\theta\sin\alpha \\ \sin\alpha & 0 & \rho\cos\alpha \end{bmatrix} \tag{4.3}$$

using the equation

$$\mathbf{C_x} = \mathbf{A}\,\mathbf{C}_{\boldsymbol{\ell}_{\rho\theta\alpha}}\mathbf{A^T} \tag{4.4}$$

where $\mathbf{C_x}$ is the variance-covariance of the $[\mathrm{x\ y\ z}]^{\mathrm{T}}$ coordinates. It is assumed that there is no correlation between observations $[\rho\ \theta\ \alpha]^{\mathrm{T}}$. The standard deviation of the range observation (σ_ρ) is impacted by several different factors. One of these factors is the angle of incidence of the laser on the surface being observed. A larger angle of incidence spreads the incident laser light across a larger set of possible ranges, increasing the standard deviation of the range measurement according to the equation

$$\sigma_\rho = \sigma_\rho^0 \sec\eta \tag{4.5}$$

where η is the angle of incidence, σ_ρ^0 is the nominal range observation precision, and σ_ρ is the range precision, accounting for the angle of incidence, as stated in [25]. The impact that $\sec\eta$ has on range precision can be seen in Figure 4.3. It can be seen that the angle of incidence has relatively little impact on the range precision for low angles of incidence, doubling the standard deviation at an incidence angle of 60°. Beyond that angle, however, the precision degrades very quickly, amplifying the standard deviation by a factor of nearly 6 at an angle of 80°.

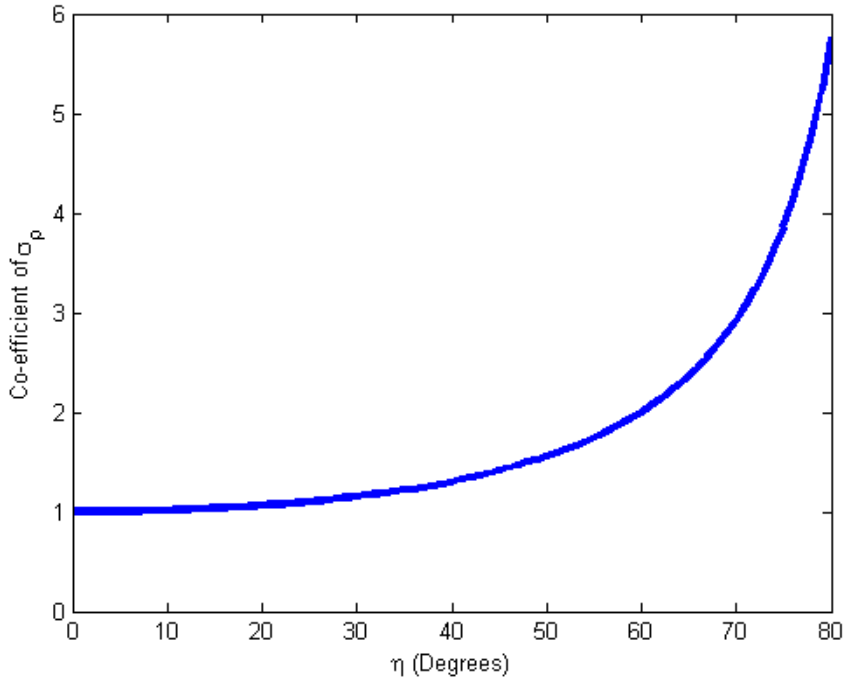

Figure 4. 3 Relation between range precision and angle of incidence

Another factor that impacts the precision of the TLS observations is the beamwidth of the laser. Details which are smaller than the laser beamwidth cannot be precisely measured, since they necessarily become lost within the beam's footprint. A large beamwidth can also contribute to increased occurrences of 'mixed' range values, which occur when the beam partly measures a close surface and a further surface, resulting in a range measurement partway between both surfaces. Mixed range measurements and beamwidth errors are not simulated in this chapter.

4.2.1.2 Modeling of spherical targets

The center of a spherical target, as determined from the scan data and visualized in Figure 4.4, is calculated using the same basic sphere modeling procedure that was used in Chapter 3, equations (3.12) to (3.15).

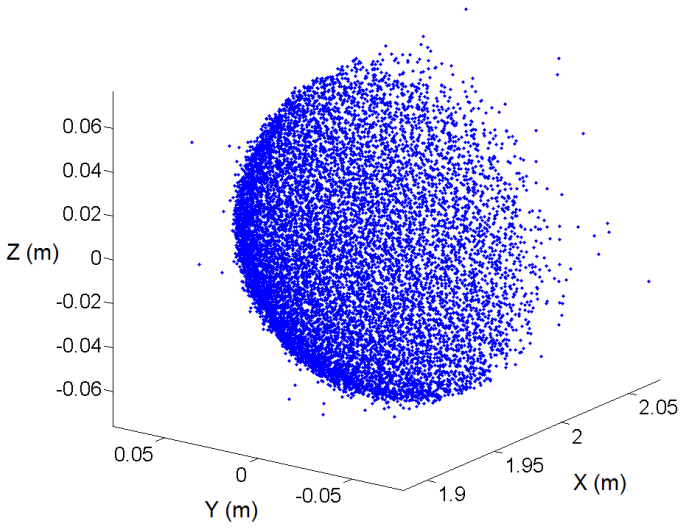

Figure 4. 4 Example of point measurements on surface of a sphere

However, in this case we are also interested in the precision of the estimated sphere center. To calculate this, we must formulate the sphere parameters as a function of the points on its surface. However, the sphere parameters cannot be modeled directly as a function of $[\mathrm{x\,y\,z}]^{\mathrm{T}}$ so the observation has to be converted to the formulation used in equation (3.12), which will have the standard deviation calculated as

$$\sigma_{\ell_0}^2 = \mathbf{A_0}\,\mathbf{C_x}\mathbf{A_0^T} \tag{4.6}$$

where

$$\mathbf{A_0} = \begin{bmatrix} \frac{\partial \ell_0}{\partial x} & \frac{\partial \ell_0}{\partial y} & \frac{\partial \ell_0}{\partial z} \end{bmatrix} = [2\,(x - x_c) \quad 2\,(y - y_c) \quad 2\,(z - z_c)] \tag{4.7}$$

and $\mathbf{C_x}$ is calculated earlier using equation (4.4). This allows the variance-covariance matrix of the sphere observations to be calculated as

$$\mathbf{C_{\ell_0}} = \begin{bmatrix} \sigma^2_{\ell_0^1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma^2_{\ell_0^n} \end{bmatrix} \tag{4.8}$$

which in turn allows the normal equations, $\mathbf{N}$ and $\mathbf{u}$, to be calculated as

$$\mathbf{N} = \mathbf{A_0^T C_\ell^{-1} A_0} + \mathbf{P_x} \tag{4.9}$$

$$\mathbf{u} = \mathbf{A_0^T C_\ell^{-1} w} + \mathbf{P_x w_x} \tag{4.10}$$

with $\mathbf{P_x}$ and $\mathbf{w_x}$ defined in equations (3.17) and (3.18). This differs only slightly from equations (3.15) to (3.21) in that the observation weight matrix was not included in Chapter 3.

Once the solution has converged and the center of the target estimated, the variance-covariance matrix of the location of that sphere can be calculated as

$$\mathbf{C_{x_s}} = \hat{\sigma}_0^2 \mathbf{N^{-1}} \tag{4.11}$$

where the variance of the $[x\ y\ z]^T$ positions are the first three terms in diagonal of the $\mathbf{C_{x_s}}$ matrix, and the a posteriori variance factor, $\hat{\sigma}_0^2$, is calculated as

$$\hat{\sigma}_0^2 = \frac{\mathbf{v^T C_{\ell_0}^{-1} v}}{n - u} \tag{4.12}$$

where n are the number of observations (i.e. points on the surface of the sphere), u is the number of parameters, $\mathbf{v}$ are the residuals of the observations, in this case because $\boldsymbol{\ell} = \mathbf{0}$,

$$\mathbf{v} = \mathbf{w} \tag{4.13}$$

4.2.1.3 Network Design in TLS Modeling

In order to be registered, each scan requires 6 parameters: $[t_x\ t_y\ t_z\ \omega\ \phi\ \kappa]$. Additionally, each three-dimensional target provides three coordinates. It may then seem that only two targets are necessary to register two scans together. However, due to the ambiguity of the rotation about the line formed by the two targets, either six coordinates among three non-collinear targets are necessary, or a seventh coordinate is necessary to resolve the registration. If every target appears in every scan, then only three targets are strictly necessary. If the scans are known to be level within a certain tolerance, then only 4 parameters need to be estimated, and 2 targets will suffice, although the registration will have a low redundancy and potentially weak solution.

As is usually the case in establishing control networks, a larger number of targets, spread throughout the volume being measured, with high redundancy of observations to those targets is preferable. Having large numbers of targets increases the redundancy of the registration parameters, which contributes to reducing errors in their estimated values. Spreading the targets out allows the coordinates of the scan centers relative to the targets to be estimated more precisely. This is analogous to interpolation versus extrapolation, the location of the scans is more reliable if they are between the targets rather than outside of them. The choice of target also impacts the precision of the registration parameters, as is discussed in Section 4.2.1.

There is a complex relationship between the TLS observations and the final, registered point clouds. The estimated errors for each point in the cloud are dependent on the errors inherent in the observations, and on the precision of the registration parameters. However, the precision of the registration parameters are dependent on the precision of the targets, which in turn are dependent on the original scan point precision. This implies that an increase in the precision of the

observations of a TLS instrument will not only impact the point precision directly, but also indirectly through increased precision of registration parameters.

4.2.1.4 Datum Definition for TLS

In order to define the coordinate system that the TLS data will be registered to, several options are available. The first option to define the network using one of the scans as the 'home', and all other scans are registered to that scan. This is a very simple datum definition, because it does not require any additional parameters be calculated or constraints enforced. The drawback to this datum definition is that the errors in points far from the home scan are amplified. This may cause a decrease in the overall precision of the registration, since the errors in the home scan are propagated into the other scans.

Another datum definition is the inner constraints method, as is defined in equations (4.50) to (4.54). This definition uses the average position and orientation of all object points to define the network, allowing individual points to shift. This method enables the home scan to absorb some of the errors in the network, resulting in overall improved precision.

In this thesis, the simpler datum definition using the home scan is used. This technique is also used in Leica Cyclone, a popular TLS processing software, so using this technique is a reflection of the precision that can be expected in data when that particular software is in use.

4.2.1.5 Registration of TLS scans

Once the spherical targets in each scan have been identified and modeled, the scans can be registered together. The initial alignment of the registration is performed using Horn's method [6]. This has the advantage of being non-iterative and a least-squares solution, but has the disadvantage

of not using the knowledge of the precision of the sphere's locations to improve the quality of the adjustment. It also does not take advantage of the level constraints.

The equation that governs the registration is expressed as

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{4.14}$$

where $[x\ y\ z]^T$ are the original, un-registered coordinates of the point in question, $[X\ Y\ Z]^T$ are the updated, registered coordinates, $[t_x\ t_y\ t_z]^T$ are the translation parameters from the original system's origin to the registered system's origin, and $r_{n,m}$ are the terms of the rotation matrix, in this case defined using the Euler angles. This registration is visible in Figure 4.5.

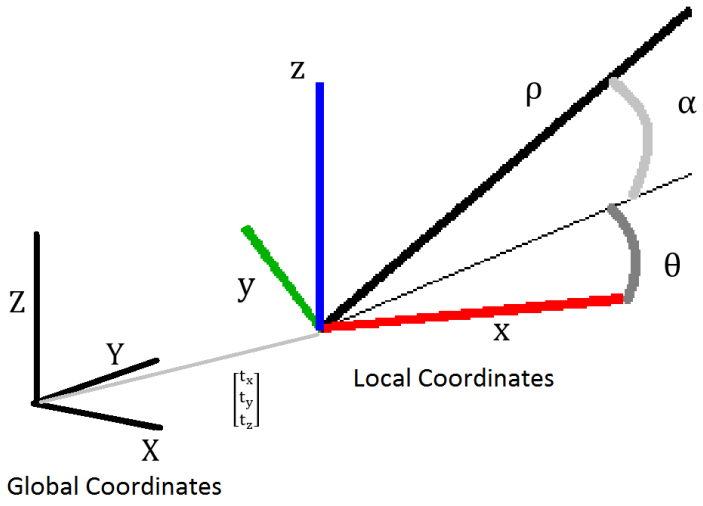

Figure 4. 5 Relationships between TLS observations, Local and Global Coordinates

Horn's method parameterizes the rotation in terms of a quaternion, so to obtain the Euler angles, the rotation matrix can be parameterized as

$$\mathbf{R}(\omega, \phi, \kappa) = \mathbf{R_3}(\kappa)\mathbf{R_2}(\phi)\mathbf{R_1}(\omega) \tag{4.15}$$

$$\mathbf{R_1}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{4.16}$$

$$\mathbf{R_2}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{4.17}$$

$$\mathbf{R_3}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.18}$$

$$\mathbf{R} = \begin{bmatrix} \cos\phi\cos\kappa & \cos\omega\sin\kappa + \sin\omega\sin\phi\cos\kappa & \sin\omega\sin\kappa - \cos\omega\sin\phi\cos\kappa \\ -\cos\phi\sin\kappa & \cos\omega\cos\kappa + \sin\omega\sin\phi\sin\kappa & \sin\omega\cos\kappa + \cos\omega\sin\phi\sin\kappa \\ \sin\phi & -\sin\omega\cos\phi & \cos\omega\cos\phi \end{bmatrix} \tag{4.19}$$

where $[\omega, \phi, \kappa]^T$ are the Euler angles, also known as the rotation angels about the X, Y and Z axes, respectively. Therefore, in order to calculate the Euler angles, $[\omega\ \phi\ \kappa]^T$, the following formulation can be used

$$\begin{bmatrix} \omega \\ \phi \\ \kappa \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{-r_{3,2}}{r_{3,3}}\right) \\ \arcsin(r_{3,1}) \\ \arctan\left(\frac{-r_{2,1}}{r_{1,1}}\right) \end{bmatrix} \tag{4.20}$$

where $r_{n,m}$ is the value in the n^{th} row and m^{th} column of the rotation matrix, **R**, given by Horn's method. There are 6 registration parameters for each scan, with the exception of the 'Home' scan. This is the scan to which the others are aligned, as it is used to define the datum of the registration. Additionally, a level constraint of $\phi = 0$, and $\omega = 0$ were applied to each scan.

If the registration model is expressed in terms of the $\boldsymbol{\ell} = \mathbf{Ax}$ linear form, then the observation vector, $\boldsymbol{\ell}$, takes the form of

$$\boldsymbol{\ell_i} = \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{4.21}$$

for the i^{th} target. The parameter vector, **x**, takes the form of

$$\mathbf{x}_j = \begin{bmatrix} t_{x_j} \\ t_{y_j} \\ t_{z_j} \\ \omega_j \\ \phi_j \\ \kappa_j \end{bmatrix} \tag{4.22}$$

for the j^{th} scan, which finally gives the Design matrix, $\mathbf{A}$,

$$\mathbf{A_{ij}} = \begin{bmatrix} 1 & 0 & 0 & a_{1,4} & a_{1,5} & a_{1,6} \\ 0 & 1 & 0 & a_{2,4} & a_{2,5} & a_{2,6} \\ 0 & 0 & 1 & a_{3,4} & a_{3,5} & 0 \end{bmatrix} \tag{4.23}$$

for the i^{th} target and the j^{th} scan, and the elements of $\mathbf{A_{ij}}$ defined in Appendix A.

Due to the internal level of the device, the vertical direction of the scanner is aligned with gravity. This means that the rotations about the X and Y axes are expected to be zero. These observations, $\omega = 0$ and $\phi = 0$, can be included in the least squares registration by calculating the normal equations as

$$\mathbf{N} = \mathbf{A^T C_{x_s}^{-1} A^T} + \mathbf{P_x} \tag{4.24}$$

$$\mathbf{u} = \mathbf{A^T C_{x_s}^{-1} w} + \mathbf{P_x w_x} \tag{4.25}$$

where $\mathbf{P_x}$ and $\mathbf{w_x}$ are the parameter weight matrix and parameter misclosure vector, which take the form

$$\mathbf{P_{x_j}} = \text{diag}\left[0\ 0\ 0\ \frac{1}{\sigma_{\omega_j}^2}\ \frac{1}{\sigma_{\phi_j}^2}\ 0\right] \tag{4.26}$$

$$\mathbf{w_{x_j}} = \left[0\ 0\ 0\ \omega_j\ \phi_j\ 0\right]^T \tag{4.27}$$

for the j^{th} scan, and $\mathbf{C_{x_s}}$ is the variance-covariance matrix of the location of the spherical targets, as determined for each in equations (4.11) and (4.12).

4.2.1.6 Propagation of Registration Uncertainty to Point Cloud Data

Once the registration parameters have been calculated, with their associated precision, it is now possible to determine the precision of each point which make up the point cloud in their new, post-registration coordinates. Conceptually, the new coordinates' precision is a combination of the precision of the original point observation, rotated into the global coordinate system; the precision of the translation parameters; and the precision of the rotation parameters, as applied to the point in question. Therefore, if the post registration coordinates are expressed in equation (4.14) then the observation equation, in the form of $\boldsymbol{\ell} = \mathbf{Ax}$ is

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [\mathbf{A_1} \quad \mathbf{A_2} \quad \mathbf{A_3}]\begin{bmatrix} t_x & t_y & t_z & \omega & \phi & \kappa & x & y & z \end{bmatrix}^T \tag{4.28}$$

where

$$\mathbf{A_1} = \mathbf{I}_{3\times 3} \tag{4.29}$$

$$\mathbf{A_2} = \begin{bmatrix} a_{1,4} & a_{1,5} & a_{1,6} \\ a_{2,4} & a_{2,5} & a_{2,6} \\ a_{3,4} & a_{3,5} & 0 \end{bmatrix} \tag{4.30}$$

with $a_{n,m}$ defined in Appendix A,

$$\mathbf{A_3} = \mathbf{R} \tag{4.31}$$

as defined in equation (4.19).

Therefore, the variance-covariance matrix of the final, registered points in the point cloud can be estimated using the equation

$$\mathbf{C}_{\mathbf{x}_{\mathbf{reg}}} = \mathbf{A}\mathbf{C}_{\boldsymbol{\ell}_{\mathbf{reg}}}\mathbf{A}^{\mathbf{T}} \tag{4.32}$$

The precision of the post-registration point cloud in the direction of the different axes can be found on the diagonal of the $\mathbf{C}_{\mathbf{x}_{\mathbf{reg}}}$ matrix. However, in order to determine the maximum expected variance of any point, the variance-covariance matrix of that point can be converted into

its equivalent ellipsoid parameters, using eigenvalue decomposition. The square root of the largest eigenvalue gives the length of the largest axis of the ellipsoid, which will be used to represent that point's precision in subsequent analysis.

4.2.2 Principles of Photogrammetry

The basic principle of photogrammetry is to make 3D measurements from 2D images. This is accomplished by using the pin-hole camera model, in which it is assumed that rays of light from the subject being measured pass through a single point, called the perspective center, and land upon a planar surface, where measurements can be made to features within the image. This is

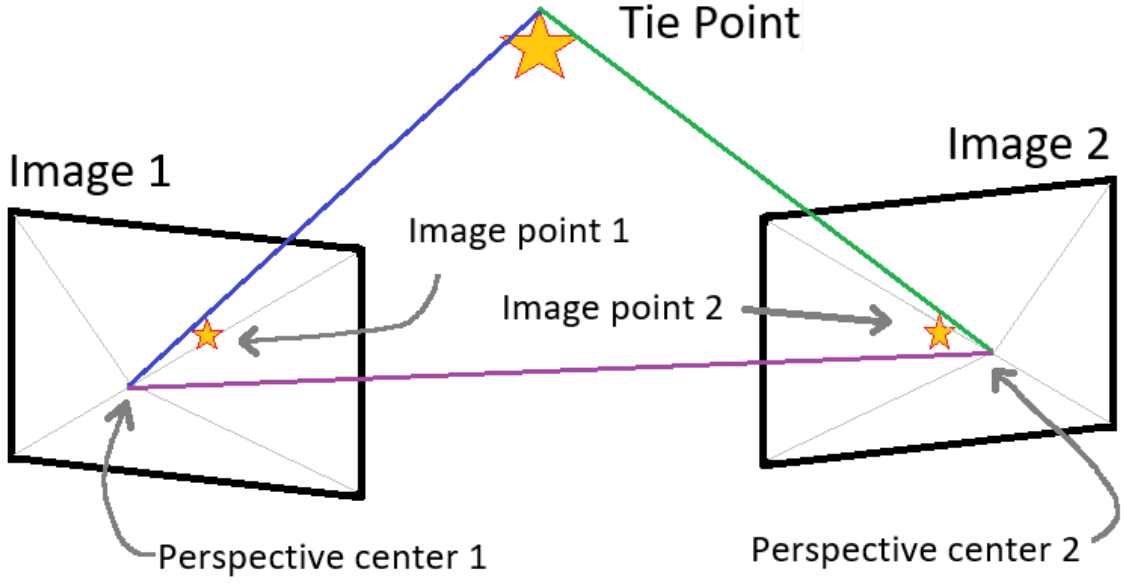

Figure 4. 6 Demonstration of basic photogrammetric intersection

demonstrated in Figure 4.6. While it is possible to infer relative 3D coordinates from a single image based on contextual information, it is not possible to reliability measure them. Therefore, a second image from a different perspective can be used to triangulate the 3D location of points that can be seen in both images.

The process of collecting photogrammetric data involves capturing many photographs of the subject being documented from many different positions and orientations, preferably with large amounts of overlap in the images, and highly convergent image geometry (i.e. the photographs lines of sight intersect at close to 90°). Large amounts of overlap between images ensures that identified 'tie points' (i.e. observable points without coordinates that are known a priori) found in pairs of images will be able to appear across the entirety of both images. While this helps map the subject in high detail, these tie points can also be used to calculate the relative orientations of the images. Having many tie points across the entire image will help determine the relative orientations more precisely.

Unless additional information is available, only relative measurements can be made using this technique, and the scale of the resulting point cloud will always be arbitrary. There are several solutions to this problem, the simplest being to include a feature of known size in the images, such as a metre stick. Knowing the size of this feature allows the other tie points be to scaled relative to it, and the images' locations can be determined relatively to the tie points. Alternatively, if a network of known coordinates, called control points, are within view of the images, then the tie points can be scaled and positioned relative to that network, strengthening the precision of their estimated positions.

In many applications of photogrammetry, the tie points can be identified on the surface of the feature being modeled. This can be done either manually, with a user finding matching points by hand, or automatically by extracting identifiable points in each image, then matching them between images according to a metric of similarity. These points, due to their visual distinctiveness, can be useful in determining the alignment of the cameras, but are less useful for providing details of the feature due to their relative sparseness. Therefore, after the cameras have

been aligned using the tie points, the point cloud is made more dense by performing simple intersections to point with less visual distinctiveness. This produces points which are much more numerous, but are also of significantly reduced precision.

Due to the computational complexity of matching automatically identified tie points between all possible images in a photogrammetric network, most automated photogrammetry software packages do not attempt to match tie points between all images, instead only making matches between points with very high similarity metrics. This will also impact the precision of the estimated tie point locations, due to the poor redundancy. Some software packages use a more sophisticated technique of point tracking and back projection to match tie points, but this thesis does not simulate this technique.

Tie points are useful for determining the relative positions of the images, but are not useful in defining the datum of the network. If only tie points and no control points are available, as is the case in many automated photogrammetric systems, then the bundle adjustment will have a poorly defined datum, which will decrease the precision of the final model.

4.2.2.1 Collinearity Equations

The photogrammetric positioning equations can be derived using the following formulation, and their relationships can be visualized in Figure 4.6.

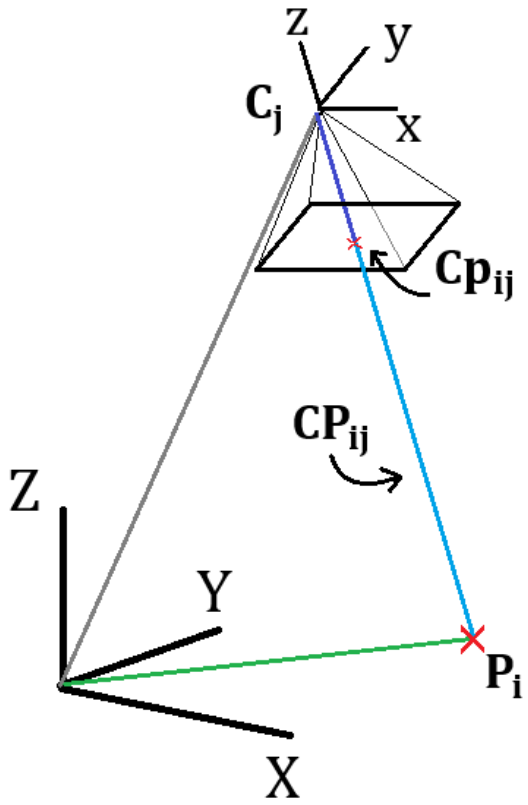

Figure 4. 7 Vector diagram showing the relations of vectors the collinearity equation

The coordinates of a point of interest in the object space coordinate system,

$$\mathbf{P_i} = \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{4.33}$$

can be expressed as the sum of the location of a camera,

$$\mathbf{C_j} = \begin{bmatrix} X_j^c \\ Y_j^c \\ Z_j^c \end{bmatrix} \tag{4.34}$$

and the vector between the camera and the point of interest, $\mathbf{CP_{ij}}$, relative to an object space coordinate system.

$$\mathbf{P_i} = \mathbf{C_j} + \mathbf{CP_{ij}} \tag{4.35}$$

Assuming that the point $\mathbf{P_i}$ is within the field of view of the camera $\mathbf{C_j}$, then the vector $\mathbf{CP_{ij}}$ passes through the imaging plane of $\mathbf{C_j}$. The intersection of $\mathbf{CP_{ij}}$ and the imaging plane is the point $\mathbf{Cp_{ij}}$ which can be measured, and expressed in the camera's internal coordinate system as

$$\mathbf{Cp_{ij}} = \begin{bmatrix} x_{ij} - x_{p_j} \\ y_{ij} - y_{p_j} \\ -c_j \end{bmatrix} = \begin{bmatrix} \bar{x}_{ij} \\ \bar{y}_{ij} \\ -c_j \end{bmatrix} \tag{4.36}$$

where x_{ij} and y_{ij} are the coordinates of the image measurement, x_{p_i} and y_{p_i} are the coordinates of the principal point, c_j is the principal distance. $\bar{x}_{ij}$ and $\bar{y}_{ij}$ are the image point coordinates reduced to the principal point.

The vectors $\mathbf{CP_{ij}}$ and $\mathbf{Cp_{ij}}$ are related, differing only by a rotation from image space to object space, and by a unique scale factor. However, $\mathbf{Cp_{ij}}$ is observable, while $\mathbf{CP_{ij}}$ is not. Therefore, equation (4.35) can be rewritten to take advantage of the observable nature of $\mathbf{Cp_{ij}}$,

$$\mathbf{P_i} = \mathbf{C_j} + \lambda_{ij}\mathbf{R_j^T Cp_{ij}} \tag{4.37}$$

such that λ_{ij} is the unique scale factor from camera j to point i, and $\mathbf{R_j^T}$ is the rotation matrix from image space j to object space ($\mathbf{R_j}$ being the rotation from object space to image space j, and is defined in equation (4.19)).

Rearranging these values to solve for the observable value $\mathbf{Cp_{ij}}$ gives the equation

$$\mathbf{Cp_{ij}} = \frac{1}{\lambda_{ij}}\mathbf{R_j}\left(\mathbf{P_i} - \mathbf{C_j}\right) \tag{4.38}$$

or in terms of the vector components

$$\begin{bmatrix} x_{ij} - x_{p_j} \\ y_{ij} - y_{p_j} \\ -c_j \end{bmatrix} = \frac{1}{\lambda_{ij}}\mathbf{R_j}\begin{bmatrix} X_i - X_j^c \\ Y_i - Y_j^c \\ Z_i - Z_j^c \end{bmatrix} = \frac{1}{\lambda_{ij}}\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\begin{bmatrix} X_i - X_j^c \\ Y_i - Y_j^c \\ Z_i - Z_j^c \end{bmatrix} = \frac{1}{\lambda_{ij}}\begin{bmatrix} U_{ij} \\ V_{ij} \\ W_{ij} \end{bmatrix} \tag{4.39}$$

Rather than estimating for λ_{ij} individually for each point in each image, we can eliminate it from our equation by dividing the first two terms by the third, which gives

$$\frac{x_{ij} - x_{p_i}}{-c_j} = \frac{r_{11}(X_i - X_j^c) + r_{12}(Y_i - Y_j^c) + r_{13}(Z_i - Z_j^c)}{r_{31}(X_i - X_j^c) + r_{32}(Y_i - Y_j^c) + r_{33}(Z_i - Z_j^c)} \tag{4.40}$$

and

$$\frac{y_{ij} - y_{p_i}}{-c_j} = \frac{r_{21}(X_i - X_j^c) + r_{22}(Y_i - Y_j^c) + r_{23}(Z_i - Z_j^c)}{r_{31}(X_i - X_j^c) + r_{32}(Y_i - Y_j^c) + r_{33}(Z_i - Z_j^c)} \tag{4.41}$$

which can be used to isolate the image observations

$$x_{ij} = x_{pj} - c_j \frac{r_{11}(X_i - X_j^c) + r_{12}(Y_i - Y_j^c) + r_{13}(Z_i - Z_j^c)}{r_{31}(X_i - X_j^c) + r_{32}(Y_i - Y_j^c) + r_{33}(Z_i - Z_j^c)} = x_{p_j} - c_j \frac{U_{ij}}{W_{ij}} \tag{4.42}$$

and

$$y_{ij} = y_{pj} - c_j \frac{r_{21}(X_i - X_j^c) + r_{22}(Y_i - Y_j^c) + r_{23}(Z_i - Z_j^c)}{r_{31}(X_i - X_j^c) + r_{32}(Y_i - Y_j^c) + r_{33}(Z_i - Z_j^c)} = y_{p_j} - c_j \frac{V_{ij}}{W_{ij}} \tag{4.43}$$

Equations (4.42) and (4.43) are the collinearity equations, which are the basis for calculating three-dimensional coordinates from sets of two dimensional images. It can be seen that 9 parameters need to be calculated from just 2 observations. In general, in order to determine the object space coordinates of point i, it must appear in at least 2 images, and in order to determine the EOPs of a camera, it must image at least 3 non-collinear points. Additionally, without further information, this formulation by itself is not uniquely solvable. Therefore, the datum of the bundle adjustment must be defined.

The design matrix for the collinear equations is defined in Appendix B.

4.2.2.2 Datum Definition for Bundle Adjustment

As mentioned previously, the datum of the bundle adjustment must be defined in order for the bundle adjustment to be uniquely solvable. This means that the basis of comparison for the object space coordinate system must have a specific scale, orientation and location, which requires 7 parameters. One method is to fix the position and orientation of a particular camera in place (providing 6 datum constraints), while all other cameras are free to adjust in order to match the data. This has the advantage of being a simple network definition, in that it does not require control points (i.e. can be used if only tie points are present in the adjustment), but has the disadvantage of amplifying the uncertainty of points that are far from the fixed camera, or of points that are not imaged directly by the fixed camera.

This datum definition can be accomplished by modifying the normal equations of the bundle adjustment,

$$\mathbf{N} = \mathbf{A^T P A} + \mathbf{P_x} \tag{4.44}$$

$$\mathbf{u} = \mathbf{A^T P w} + \mathbf{P_x w_x} \tag{4.45}$$

where $\mathbf{P_x}$ is the parameter weight matrix, and $\mathbf{w_x}$ is the parameter observation misclosure vector. $\mathbf{P_x}$ is the same size as $\mathbf{N}$, and $\mathbf{w_x}$ as $\mathbf{u}$, but have zeros in every element with the exception of those that correspond to the EOPs of the fixed camera, camera j. Those value take the form of

$$\mathbf{P_{x_j}} = \text{diag}\left[\frac{1}{\sigma^2_{X^c_0}} \frac{1}{\sigma^2_{Y^c_0}} \frac{1}{\sigma^2_{Z^c_0}} \frac{1}{\sigma^2_{\omega^c_0}} \frac{1}{\sigma^2_{\phi^c_0}} \frac{1}{\sigma^2_{\kappa^c_0}}\right] \tag{4.46}$$

$$\mathbf{w_{x_j}} = \begin{bmatrix} X^c - X^c_0 \\ Y^c - Y^c_0 \\ Z^c - Z^c_0 \\ \omega - \omega_0 \\ \phi - \phi_0 \\ \kappa - \kappa_0 \end{bmatrix} \tag{4.47}$$

where parameters with the subscript of 0 indicate the initial value that is to be fixed, while those without indicate the estimated value for that iteration of the least squares adjustment.

Fixing one camera's initial position will define 6 of the necessary 7 datum constraints, the remaining one being the scale. For this, an additional measurement must be made, usually as a distance between two points within the field of view of the cameras. This measurement can be formulated as an observation, and can be expressed as

$$d_{ij} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 - (Z_i - Z_j)^2} \tag{4.48}$$

which, in linearized form gives

$$d_{ij} = \begin{bmatrix} \frac{(X_i - X_j)}{d_{ij}} & \frac{(Y_i - Y_j)}{d_{ij}} & \frac{(Z_i - Z_j)}{d_{ij}} & -\frac{(X_i - X_j)}{d_{ij}} & -\frac{(Y_i - Y_j)}{d_{ij}} & -\frac{(Z_i - Z_j)}{d_{ij}} \end{bmatrix} \begin{bmatrix} \delta X_i \\ \delta Y_i \\ \delta Z_i \\ \delta X_j \\ \delta Y_j \\ \delta Z_j \end{bmatrix} \tag{4.49}$$

where d_{ij} is the distance between point i and j. This linear form can be inserted into the design matrix, **A**, of the bundle adjustment. With this observation and the parameter constraints in equations (4.48) and (4.49), the solution becomes unique and the normal matrix, invertible.

Alternatively, the datum can be minimally defined using the inner constraints method, also called a free network. This method minimally constrains the network using the initial estimated values of the control points and camera parameters as a basis. As each parameter is adjusted, the others are likewise adjusted to counteract any net changes being made to the network geometry (i.e. relative to the centroid, location, relative orientation and scale are all unchanged). This can be done by padding the normal matrix with the matrix G, such that

$$\begin{bmatrix} \delta_{X_i} \\ \delta_{Y_i} \\ \delta_{Z_i} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & -Z_i^0 & Y_i^0 \\ 0 & 1 & 0 & Z_i^0 & 0 & -X_i^0 \\ 0 & 0 & 1 & -Y_i^0 & X_i^0 & 0 \end{bmatrix} \begin{bmatrix} \delta_{t_x} \\ \delta_{t_y} \\ \delta_{t_z} \\ \delta_{\omega} \\ \delta_{\phi} \\ \delta_{\kappa} \end{bmatrix} \tag{4.50}$$

or,

$$\hat{\boldsymbol{\delta}} = \mathbf{G}\boldsymbol{\delta}_{\mathbf{h}} \tag{4.51}$$

for each object point i. It is noteworthy that this formulation of $\mathbf{G}$ does not include a constraint for scale, since that is set using the scale measurements in equations (4.48) and (4.49).

By setting $\boldsymbol{\delta}_{\mathbf{h}} = \mathbf{0}$, it ensures that there is no overall change in the network's position or orientation. Thus, equation (4.51) introduces the constraint

$$\mathbf{G}^{\mathbf{T}}\hat{\boldsymbol{\delta}} = \mathbf{0} \tag{4.52}$$

which is used to extend the normal matrix,

$$\begin{bmatrix} \mathbf{A}^{\mathbf{T}}\mathbf{P}\mathbf{A} & \mathbf{G} \\ \mathbf{G}^{\mathbf{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\delta}} \\ \hat{\mathbf{k}} \end{bmatrix} + \begin{bmatrix} \mathbf{A}^{\mathbf{T}}\mathbf{P}\mathbf{A} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{4.53}$$

which in turn ensures that the normal matrix is invertible, which allows $\hat{\boldsymbol{\delta}}$ (and a vector of Lagrange multipliers) to be solved like so,

$$\begin{bmatrix} \hat{\boldsymbol{\delta}} \\ \hat{\mathbf{k}} \end{bmatrix} = -\begin{bmatrix} \mathbf{A}^{\mathbf{T}}\mathbf{P}\mathbf{A} & \mathbf{G} \\ \mathbf{G}^{\mathbf{T}} & \mathbf{0} \end{bmatrix}^{-\mathbf{1}} \begin{bmatrix} \mathbf{A}^{\mathbf{T}}\mathbf{P}\mathbf{w} \\ \mathbf{0} \end{bmatrix} \tag{4.54}$$

It is also possible to constrain only a sub-set of parameters, which is commonly done in photogrammetry to define the datum using the coordinates of control points, while still enabling the camera EOPs and tie points to freely adjust without impacting the datum definition.

4.3 Simulated Observation Methods and Models

In order to estimate the precision of the points after they have been registered, a stochastic model and physical simulation of a realistic environment and the observation devices has been constructed. These models simulate an environment for the observation models to collect data from, and use realistic precision estimates to determine the errors in the final point clouds. These simulations have access to all relevant information to know with confidence the expected errors present in a real data set. For instance, the angle of incidence affects the precision of the range estimate in TLS data. In a simulation, it is possible to know precisely the actual angle of incidence, while in a real data set, the angle of incidence has to be estimated from the data itself. This creates a problem: the estimated angle of incidence may not be reliable, due to the error created by the angle of incidence. Using simulated observations and reasonable values for their precision, as well as a simulated environment that resembles a real environment, the precision of both the TLS and the photogrammetry models can be estimated reliably.

The purpose of this simulation is to compare the levels of precision that can be expected using different measurement technologies in an archaeological application.

4.3.1 TLS Registration Modeling Equations

The TLS data will be simulated by creating a nearly complete sphere of angular measurements, and one range measurement for each. If there is no object in the direction of an angular observation from the perspective of the scanner, then this angular observation will also be removed, so that only true observations are included in the simulated data. Otherwise, the range observation will reflect the actual range to the first object it hits. Furthermore, the estimated

variance of each observation will be simulated as well, including the angle of incidence' impact on the range precision.

4.3.1.1 Simulating data collection from a sphere

In order to accurately simulate the data that would exist on the surface of a sphere, the location of the sphere has to be converted into the coordinate system of the scanner. This can be done using the two step process of, first, converting the global $[X\ Y\ Z]^T$ coordinates of the sphere into the local $[x\ y\ z]^T$ coordinates that of the scanner in question through a rigid body transformation, the inverse of the one performed in equation (4.14), specifically

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R^T}(\omega, \phi, \kappa)\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}\right) \tag{4.55}$$

Second, the local $[x\ y\ z]^T$ coordinates of the sphere center have to be converted into the spherical coordinates that the scanner makes observations in,

$$\begin{bmatrix} \rho \\ \theta \\ \alpha \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \arctan\left(\frac{y}{x}\right) \\ \arctan\left(\frac{z}{\sqrt{x^2 + y^2}}\right) \end{bmatrix} \tag{4.56}$$

Once the center of the sphere is known, it is now possible to calculate the surface points on the sphere. The scanner varies the vertical and horizontal angles in discrete increments, so it is first necessary to determine which angular coordinates lay upon the surface of the sphere. There exists a fairly simple trigonometric identity that can be used to determine the angle from the center of the sphere to the edge of that sphere from the perspective of an observer. Because the edge of

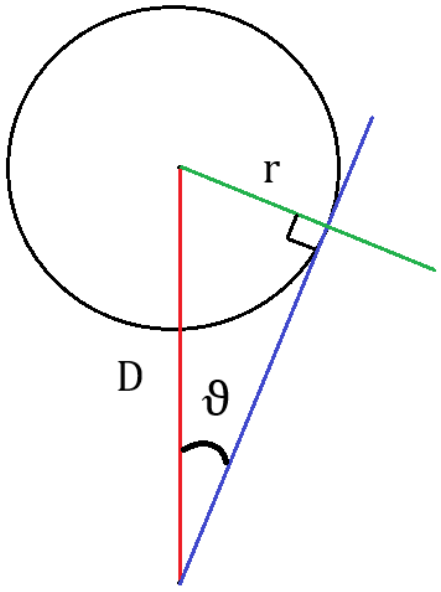

Figure 4. 8 Demonstration of a sphere's angular size

the sphere from the perspective of the scanner is necessarily tangent to the surface of the sphere, the distance from the center of the sphere to that point tangent to the sphere's surface is always the radius of the sphere, as can be seen in Figure 4.8. So therefore, half the angular foot print of the sphere is always

$$\vartheta = \arcsin\frac{r}{D} \tag{4.57}$$

where ϑ is the angle from the center of the sphere to the edge from the perspective of the scanner, r is the radius of the sphere, and D is the distance from the scanner to the center of the sphere.

Therefore, the location of any observation that belongs on the surface of the sphere can have an angular distance from the center of the sphere no greater than ϑ from the perspective of the scanner. This information can be used to test which angular observation points belong on the surface of the sphere, since only those need be considered as valid observations for this surface. Using the Spherical Law of Cosines, visualized in Figure 4.9,

$$\cos c = \cos a \cos b + \sin a \sin b \cos C \tag{4.58}$$

it is possible to calculate which set of angular values are within the angular distance of the center of the sphere.

Equation (4.58) simplifies further, since the value C is always 90° due to the orthogonal nature of

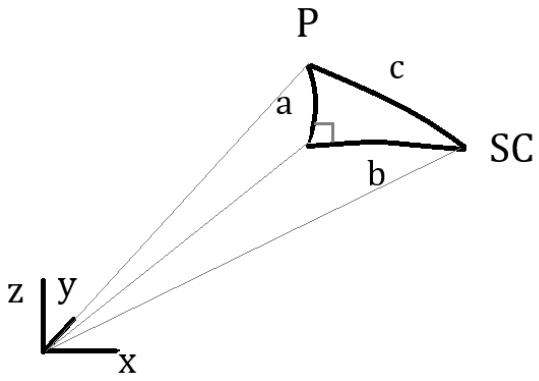

Figure 4. 9 Spherical Law of Cosines

the horizontal and vertical angles, giving

$$\cos c = \cos a \cos b \tag{4.59}$$

where c is the angular distance between the observation, P, and the center of the sphere, SC from the perspective of the scanner; a is the vertical angle between the observation P to the sphere center SC from the perspective of the scanner; and b is the horizontal angle between the observation P and the sphere center, SC. The values a and b are distinct from θ and α, the spherical angle coordinates, because they are measured relative to the center of the sphere in question, rather than the reference axis of the scanner.

At this point, all potential observations to the sphere's surface have been determined as either belonging on the surface of the sphere or not. It is now possible to determine the range measurement for all valid angular values.

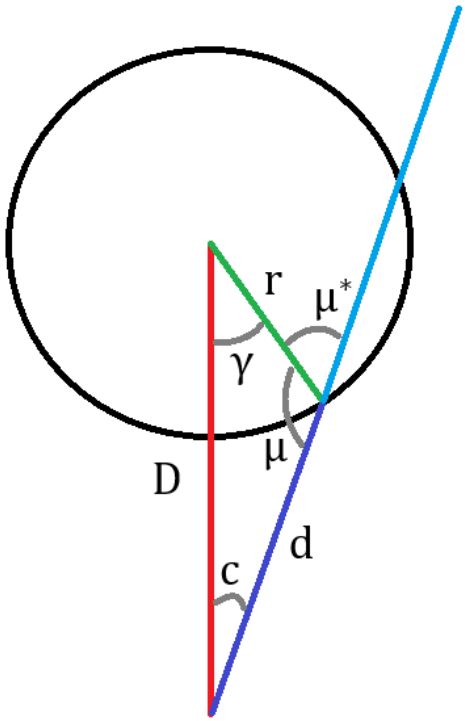

Figure 4. 10 Angular relations to a point on a sphere

The range measurement to the surface of the sphere can be calculated using the formulas

$$\mu = \arcsin\left(D\frac{\sin c}{r}\right) \tag{4.60}$$

which represents the angle at the observation point on the surface of the sphere between the scanner and the center of the sphere. This is visualized in Figure 4.10. The internal angle of the observation can be calculated like so:

$$\gamma = 180° - \mu - c \tag{4.61}$$

and the range to the surface measurement can be calculated using the equation

$$d = r\frac{\sin \gamma}{\sin c} \tag{4.62}$$

In cases where the internal angle of the observation, γ, is less than a threshold, the range value is simply taken as

$$d = D - r \tag{4.63}$$

which is the distance to the center of the sphere minus the sphere's radius. This is done in cases where γ is too small to accurately calculate using the arcsin function and floating point precision.

In some circumstances, equation (4.60) returns the value μ^* rather than μ, which represents the intersection of the observation angles at the back of the sphere, rather than the front. This value is equal to

$$\mu^* = 180° - \mu \tag{4.64}$$

Therefore, both values μ and μ^* are calculated and used in equations (4.61) and (4.62) to determine two values for the distance d, and only the minimum of these two values is kept as the range observation.

4.3.1.2 Simulating data collection from a plane

Similar to collecting range measurements to the surface of spheres, part of the model being simulated here is made up of planar sections, and so range measurements to the surface of planes must be made as well. In this case, the point-on-plane method was used to determine if a given observation fell within the limits of the given plane.

First, the plane is transformed into the local coordinates of the scanner, using the same procedure as equations (4.55). The plane is defined by the vertexes of its corners, and the planes are always triangular, as demonstrated in Figure 4.11. The normal direction of the plane is determined using the equation

$$\mathbf{n} = \frac{\mathbf{P_{1,2}} \times \mathbf{P_{1,3}}}{\left\| \mathbf{P_{1,2}} \times \mathbf{P_{1,3}} \right\|} \tag{4.65}$$

where $\mathbf{P_{1,2}}$ is the vector from point one to point two, and $\mathbf{P_{1,3}}$ is the vector from point one to point three.

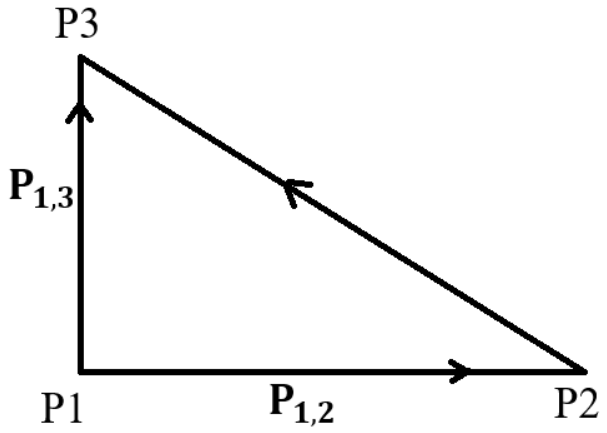

Figure 4. 11 Definition of points and vectors of a planar section

Not every plane will be observable from every scan. The first check that's performed in order to determine if a plane is observable from a scanner is if **n** is towards or away from the scanner. If the plane's normal direction is away from the scanner (i.e. if the dot product of the normal direction and a vector from the scanner to the plane is positive), then no observations are made to this plane. Another test is to see if the minimum distance to the scanner is less than the maximum range of the scanner. Otherwise, no points on the plane can be observed. This value is equivalent to the value d in the plane equation

$$ax + by + cz - d = 0 \tag{4.66}$$

and the values [a b c] correspond to the elements of the normal vector, and

$$d = \|\mathbf{n} \cdot \mathbf{P}\| \tag{4.67}$$

where **P** is any point on the plane, such as any of the three vertices P1, P2 or P3.

Next, the maximum limits of the horizontal and vertical angles of the plane are estimated, in order to limit the number of potential observations that have to be made. This is done by creating 1000 samples of the vectors between the three corners of the plane, and checking the extreme values of the horizontal and vertical coordinates of each one.

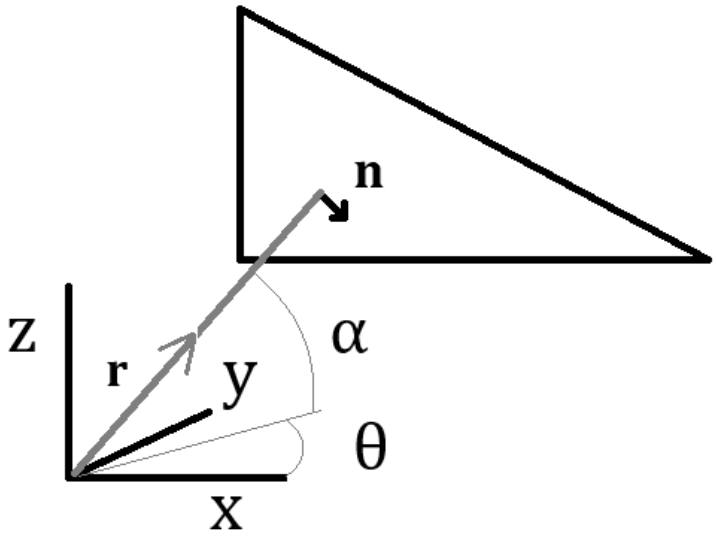

Figure 4. 12 Identifying the intersection of a unit vector and a plane

Once the vertical and horizontal extents of the plane are determined, it is then possible to start testing the angular observations as potential planar points. A unit vector is created from the scanner in the direction of the potential observation.

$$\mathbf{r} = \begin{bmatrix} \cos\theta \cos\alpha \\ \sin\theta \cos\alpha \\ \sin\alpha \end{bmatrix} \tag{4.68}$$

and the distance from this vector to its intersection with the plane is

$$s = -\frac{d}{\mathbf{n} \cdot \mathbf{r}} \tag{4.69}$$

where d is defined in equation (4.67), **n** is the normal direction of the plane, and is **r** the unit vector from equation (4.68). This relationship is visible in Figure 4.12. Therefore, the location of the intersection of this angular observation and the plane is given as

$$\mathbf{P} = s * \mathbf{r} \tag{4.70}$$

Once this intersection is calculated, it is necessary to test if that point falls within the triangular section that defines the plane. So far, the plane has been treated as infinite in size, but in reality it is not. For this purpose, the barycentric technique, visualized in Figure 4.13, is used.

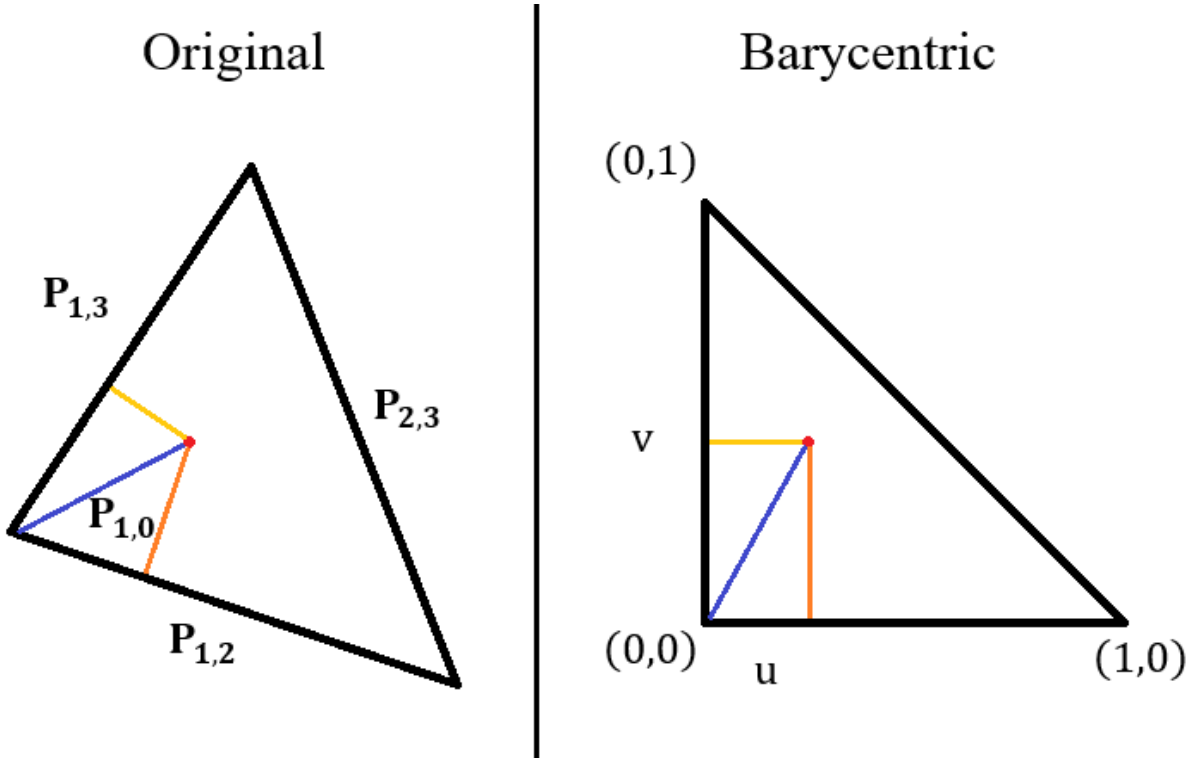

Figure 4. 13 Left: A point on a plane in object space. Right: In barycentric space

This technique re-parameterizes the coordinates of the point in question in terms of the vectors defining the extents of the triangular planar section. These parameters are calculated as

$$d_{00} = \mathbf{P_{1,3}} \cdot \mathbf{P_{1,3}} \tag{4.71}$$

$$d_{01} = \mathbf{P_{1,3}} \cdot \mathbf{P_{1,2}} \tag{4.72}$$

$$d_{11} = \mathbf{P_{1,2}} \cdot \mathbf{P_{1,2}} \tag{4.73}$$

$$d_{02} = \mathbf{P_{1,3}} \cdot \mathbf{P_{1,0}} \tag{4.74}$$

$$d_{12} = \mathbf{P_{1,2}} \cdot \mathbf{P_{1,0}} \tag{4.75}$$

$$u = \frac{(d_{11} * d_{02}) - (d_{01} * d_{12})}{(d_{00} * d_{11}) - (d_{01} * d_{01})} \tag{4.76}$$

$$v = \frac{(d_{00} * d_{12}) - (d_{01} * d_{02})}{(d_{00} * d_{11}) - (d_{01} * d_{01})} \tag{4.77}$$

where $\mathbf{P_{n,m}}$ is the vector from the n^{th} to the m^{th} point defining the boundary of the planar section, $\mathbf{P_{1,0}}$ is the vector from the first vertex of the plane to the potential observation point in question, **P**, and u and v are the Barycentric parameters. The point **P** is only within the bounds of the planar section if

$$0 < u < 1 \tag{4.78}$$

$$0 < v < 1 \tag{4.79}$$

and

$$u + v < 1 \tag{4.80}$$

If this is the case, then the range observation for this set of angular observations is set to be equal to the value s in equation (4.69). Otherwise, no observation exists for this angular increment, for this plane.

4.3.1.3 Calculating Angle of Incidence for Planes and Spheres

In order to accurately estimate the precision of the range measurement, it is necessary to determine the angle of incidence of a TLS observation to the surface being measured. The angle of incidence for a planar observation is calculated as

$$\eta = \arccos(\mathbf{n} \cdot -\mathbf{r}) \tag{4.81}$$

where **n** is the normal direction of the plane, and **r** is the unit vector giving the direction from the scanner to the observation point on the surface of the plane, defined in equation (4.68).

However, to calculate the angle of incidence of a spherical surface, the calculation is quite different. In this case, the angle of incidence, η, is equal to the sum of the angles c and γ, which are visualized in Figure 4.14. These can be calculated using the formulation

$$c = \arccos\frac{\|\mathbf{P}\|}{\|\mathbf{SC}\|} \tag{4.82}$$

where **P** is the point of interest, **SC** is the sphere center. The angle γ is calculated similarly,

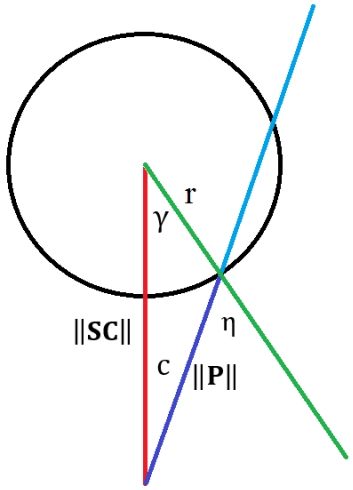

Figure 4. 14 Angle of incidence to a point on a sphere

$$\gamma = \arccos\frac{r}{\|\mathbf{SC}\|} \tag{4.83}$$

where r is the radius of the sphere, which gives

$$\eta = c + \gamma \tag{4.84}$$

4.3.2 Simulating Automated Photogrammetric Observations

4.3.2.1 Simulating Tie Point Locations from the Planar Models

In order to create the points on the plane that will stand in as our identified tie point locations, each plane is sampled at regular intervals according to the desired total number of

samples that the user would like to have. To do so, the desired number of points is divided by the total area of the planar sections.

The area of each planar section is determined using the equation

$$A = \frac{1}{2}\left\|\mathbf{P_{1,2}} \times \mathbf{P_{1,3}}\right\| \tag{4.85}$$

where $\mathbf{P_{1,2}}$ is the vector from point 1 to point 2, etc. The approximate number of point per plane is then determined

$$n_i = \frac{NA_i}{\Sigma A} \tag{4.86}$$

where n_i is the number of points intended for the i^{th} plane, N is the desired number of points in the whole model, A_i is the area of the i^{th} plane, and ΣA is the total area of all the planes in the model. Once the number of points for a given plane is known, that plane can then be sampled.

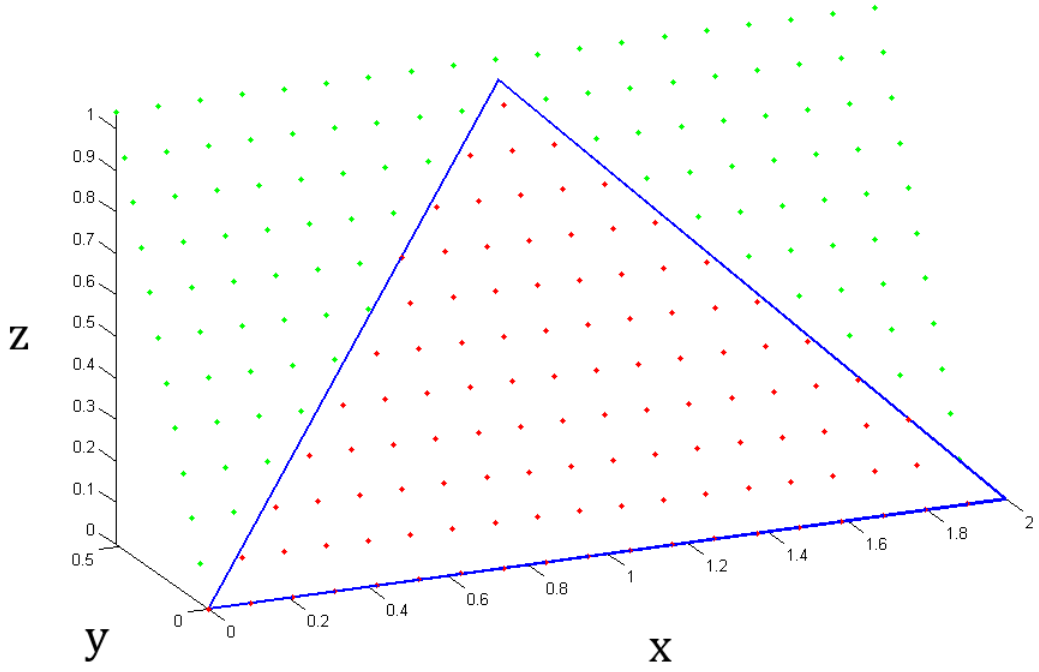

Figure 4. 15 A regularly sampled plane
Green points are calculated but not included as observable

The point spacing for a given plane is calculated as

$$s_i = \sqrt{\frac{A_i}{n_i}} \tag{4.87}$$

where s_i is the distance between points, A_i is the area of the plane, and n_i is the number of points designated for this plane. The plane is then sampled by finding the longest edge of the plane, and creating sampled points along that vector with a spacing equal to s_i. Then the next row is sampled, at a distance of s_i, mutually orthogonal to the longest edge and the normal direction. Additional rows are added until the entire triangular planar section is sampled in a regular grid of points. Since many of the points fall outside the bounds of the triangular section, the barycentric method seen in equations (4.71) to (4.77) is used to remove points that do not fall within the bounds of the planar section. This process is demonstrated in Figure 4.15.

This method of plane sampling works well to produce a regular grid of approximately equally spaced samples across the entire model, but produces an inexact number of point samples.

4.3.2.2 Simulating Image Data

The simulated image observations are calculated by first calculating the vector to the plane point in the coordinate system of the camera,

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \mathbf{R} \begin{bmatrix} X - X^c \\ Y - Y^c \\ Z - Z^c \end{bmatrix} \tag{4.88}$$

where $[X\ Y\ Z]^T$ are the plane sample point, $[X^c\ Y^c\ Z^c]^T$ and $\mathbf{R}$ are the EOPs of the camera.

Then, the image point coordinates can be estimated using the formulation

$$x = -c\frac{U}{W} \tag{4.89}$$

and

$$y = -c\frac{V}{W} \tag{4.90}$$

where c is the principal distance of the camera, assuming that the principal point $[x_p\ y_p]$ is equal to zeros. However, since image measurements are made in discrete units of pixels, the values of x and y must be rounded to the nearest whole multiple of pixel distances, which can be done

$$x = \text{round}\left(\frac{x}{ps}\right) * ps \tag{4.91}$$

where ps is pixel size. The same equation can be used for the y image coordinate rather than x.

Of course, not every point in the model is within all images' field of view. Therefore, it is necessary to filter out observations to points, such that the image observations x and y are within the size of the CCD array, and that the point is in front of the camera (not behind).

$$|x| < \frac{N}{2} \tag{4.92}$$

$$|y| < \frac{M}{2} \tag{4.93}$$

and

$$W < 0 \tag{4.94}$$

where N and M represent the size of the CCD array in pixels.

As part of simulation process, it is necessary to consider occlusions, since not only are point observation limited by the field of view of the camera, but they are also limited by the geometry of the scene being modeled. In this simulation, if two points within an image have the equal image point measurements x and y after rounding to the nearest pixel, then the further point is considered occluded, and the image point observation is removed from list of observations.

After the occluded observations are removed, each point is reduced to only appear in two images, those images being selected by the geometry of the intersection of their lines of sight to that point.

4.3.2.3 Reducing Imaging Redundancy to Pairs

As mentioned previously, due to the computational complexity of matching automated image point measurements between all possible images in a photogrammetric network, most automated photogrammetry software only matches image points between a limited set of cameras. To reflect this property, the number of observations that are made to any given image point will be reduced to exactly two, and these two images observing each point are selected by the quality of their intersection angle. This solution will have an optimal intersection but low redundancy, so will reflect reasonable estimates of precision that can be expected from automated tie point extraction software. Some software uses a combination of point tracking and back projection to match points in neighbouring images, but this is not simulated in this thesis.

In general, the intersection geometry is improved if the intersection of the two image vectors meet at close to a 90° angle, and the point is approximately halfway between the two image locations. In order to find the best image pair for a given point, the quality of the intersection needs to be quantified, in this case, as a cost function. This cost function takes into consideration both qualities of strong image geometry, and is visualized in Figure 4.16. The image pair with the lowest cost function is selected as the only two images which will maintain their observations to the point in question. The observation to the current point will be deleted from the other images.

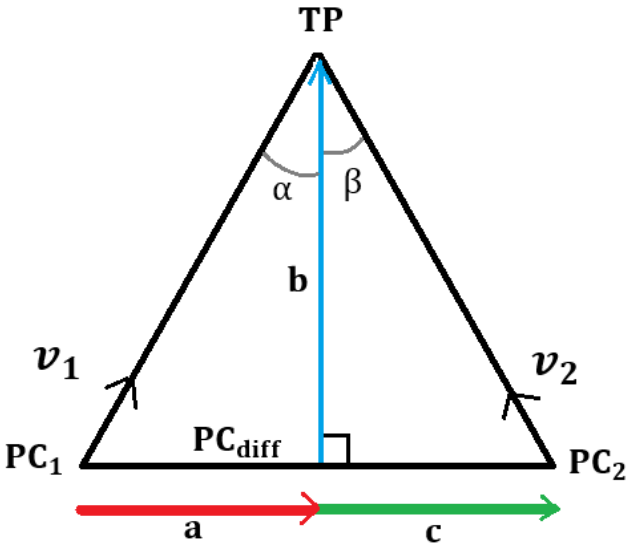

Figure 4. 16 Geometry of a photogrammetric intersection

The portion of the cost function that represents the intersection angle can be represented as the cosine of the intersection angle, which has the property of being zero at 90° and a maximum of 1 at 0° or 180°. However, the magnitude of the dot product of the unit vectors, describing the direction of the vectors from the camera location to the sample point, already correspond to the cosine of the angle of intersection. Therefore, this quality can be represented with the simple equation

$$ct_1 = \boldsymbol{v_1} \cdot \boldsymbol{v_2} \tag{4.95}$$

where $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ are the unit vectors from the camera locations to the sample point as seen in Figure 4.15, and $ct_1$is the cost associated with the intersection angle.

For the sample point to be approximately halfway between the two cameras, it's location can be projected onto the vector between the two images. The vector from the tie point to its projection on the image separation line, ideally, would bisect the angle of intersection. If it does not, then the geometry is not optimal. The two portions of the angle of intersection, here labeled α

and β, should be as close to equal as possible. Any deviation contributes to the cost function increasing. This can be expressed using the formulation

$$\mathbf{PC_{diff}} = \mathbf{PC_2} - \mathbf{PC_1} \tag{4.96}$$

$$\mathbf{a} = \mathbf{PC_{diff}} \cdot \left(\mathbf{PC_{diff}} \cdot (\mathbf{TP} - \mathbf{PC_1})\right) \tag{4.97}$$

$$\mathbf{b} = (\mathbf{TP} - \mathbf{PC_1}) - \mathbf{a} \tag{4.98}$$

$$\mathbf{c} = \mathbf{a} - \mathbf{PC_{diff}} \tag{4.99}$$

$$\alpha = \arctan\frac{\|\mathbf{a}\|}{\|\mathbf{b}\|} \tag{4.100}$$

$$\beta = \arctan\frac{\|\mathbf{c}\|}{\|\mathbf{b}\|} \tag{4.101}$$

where $\mathbf{PC_1}$ and $\mathbf{PC_2}$ are the first and second perspective centers, $\mathbf{PC_{diff}}$ is the vector from the first to the second perspective center, $\mathbf{a}$ is the vector from the $\mathbf{PC_1}$ to the projection of the tie point along the vector $\mathbf{PC_{diff}}$, $\mathbf{b}$ is the vector orthogonal to $\mathbf{a}$, from vector $\mathbf{PC_{diff}}$ to the tie point, and $\mathbf{c}$ is the vector from $\mathbf{a}$ to $\mathbf{PC_2}$.

Since the cost is low when the difference between α and β is also low, the sine of the difference of the angles can be used to calculate the cost

$$ct_2 = \sin(\alpha - \beta) \tag{4.102}$$

The total cost of the intersection geometry is calculated as

$$ct = ct_1^2 + ct_2^2 \tag{4.103}$$

where ct is the final cost. By taking the square of ct_1 and ct_2, the cost rises fairly slowly as the intersection angle deviates from 90° and as α and β diverge in value from each other. It also prevents a negative value in ct_1 or ct_2 from lowering the overall cost (i.e. the cost should only increase as values diverge from their expected value, whatever the direction).

This cost is used to calculate the best pair of images that any point can be viewed in. The pair of images that have the lowest cost preserve their observations of the point, while all others delete their observations of that point.

4.4 Experimental Design

4.4.1 Virtual Environment

In order to ensure that both the TLS and photogrammetry simulations are being compared on equal terms, observations within the same virtual environment have been simulated. This environment is designed to mimic the approximate dimensions and shape of the actual data collected at Kuukpak in July of 2014. This site, as can be seen in Figure 4.17, is a large cruciform semi-subterranean house, approximately 12m by 8m, cut into a hillside on the coast of the MacKennzie delta. The structure of the house is primarily of wooden planks cut from driftwood, while much of the scans is of the surrounding hillside and the dirt walls leftover from excavation. The floor of the house is relatively flat, and the surrounding hillside is sloped towards the river. At the highest point, the hill is approximately 1m above the wooden floor of the structure.

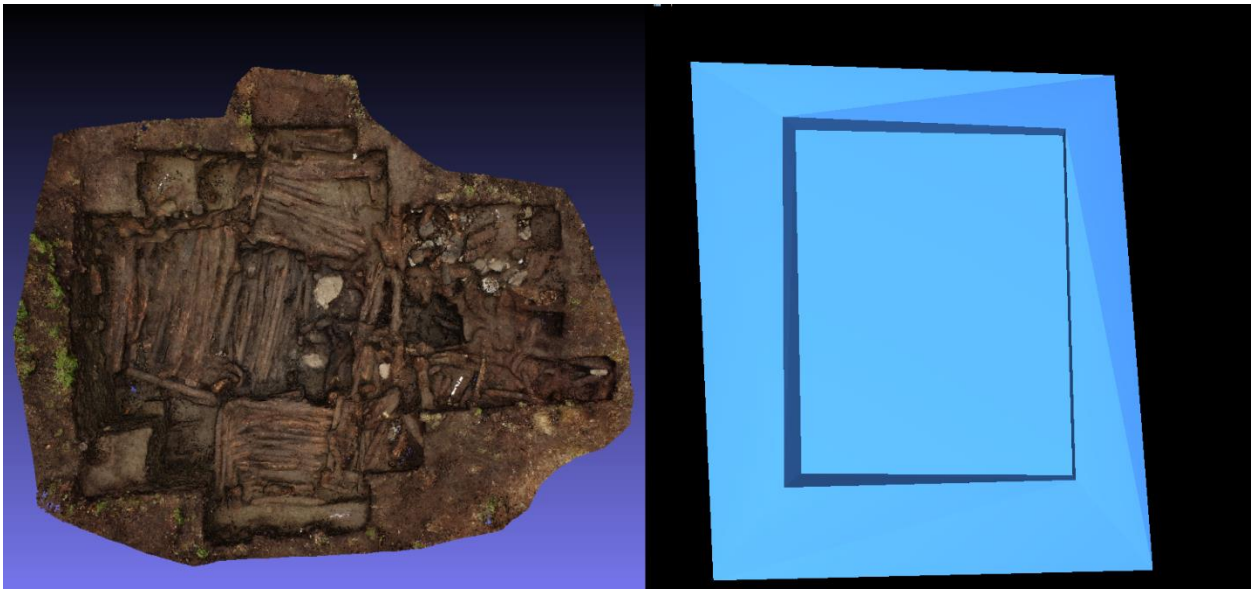

Figure 4. 17 Comparison of virtual environments
Left: High resolution point cloud of the house excavated at Kuukpak. Right: Simplified planar model used in this chapter.

The model used in this simulation is simplified significantly, so that only 18 triangular planar sections are used to represent it, as can be seen in Figure 4.17. This has the benefit of improving the computational speed that the simulated observations can be made at. The principal features of the level floor plan, the near vertical excavated walls and the sloped hillside are all represented in the model, although details such as the structure of the wooden planks or the partially excavated entry tunnel have been excluded.

4.4.2 TLS Error Modeling Parameters

In this simulation, there are several different representations of the same point, in different coordinate systems. First, the point in defined in the spherical coordinate system centered on that particular scan. That same point can also be expressed in the local XYZ coordinate system centered on that scan, as well as the global XYZ coordinate system that the point is transformed into after registration. All of these coordinates are related, but distinct, and are shown in Figure 4.5. Furthermore, each one has its own unique representation of the errors associated with the point, which propagate from one to the other as more uncertainty is introduced.

The observation standard deviations used for the TLS model is this experiment are summarized in Table 4.1.

Table 4. 1 Observation standard deviations for TLS

Observable Standard deviation	**Magnitude**
σ_ρ^0	1.7mm
σ_θ	66”
σ_α	45”
σ_ω	37”
σ_ϕ	37”

These values are collected from the calibration of the Faro Focus 3D scanner performed by Chow et al. [28]. Furthermore, the Faro Focus 3D has an internal digital level, which provides the condition that the ω and ϕ angles (i.e. rotations about the X and Y axes) are equal to zero, within a standard deviation also reported in Table 4.1.

This level constraint was important because it helped to decorrelate registration parameters, which results in a stronger solution, overall. Many TLS error models include an additive error and a linear error term for the range measurement. This has the effect that the range always has a minimum standard deviation, but the standard deviation increases linearly with the range of the measurement. This error behaviour is commonly seen in real TLS data sets, but over the interval of range measurements being used in this simulation, the addition of a linear range error term was not considered necessary, since it would have added complexity without adding an appreciable difference to the point precision.

Five different scan positions were simulated in this example. This number was selected because it provided good coverage of all the planar sections, without creating an unnecessary amount of extra data. In the actual data of Kuukpak, 10 TLS scans were used to ensure excellent coverage and an abundance of data, as well as high levels of detail. This was not necessary in the simulation, due to the simplified geometry. Additionally, 6 spherical targets were placed throughout the scan. These were used to reference each of the scans together, and all targets were visible by all scans. The network arrangement is visible in Figure 4.18.

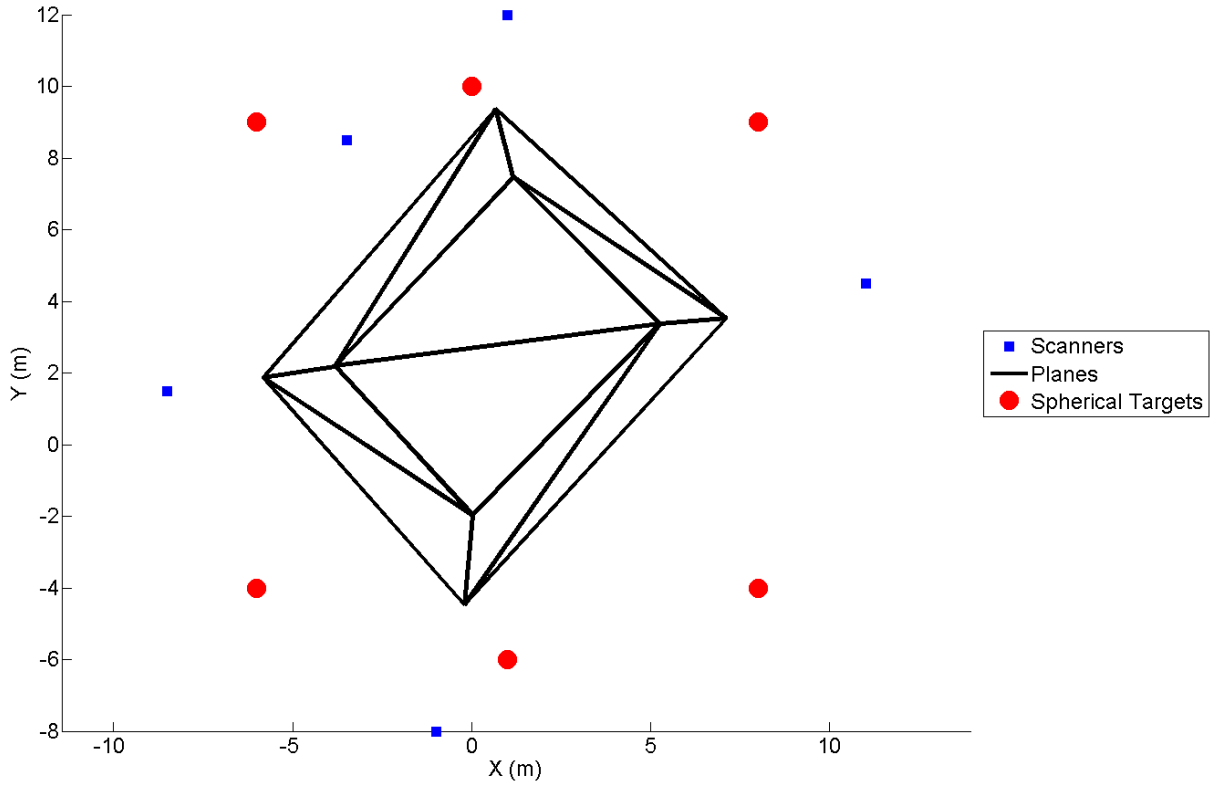

Figure 4. 18 Arrangement of scan locations and targets relative to planar sections

Since the true location of the spherical target were known, the extraction of their surface observations was a simple task of collecting all the points that are approximately as far away from

the center of the sphere as the value of the radius. Once these points are collected, it is possible to determine the location of that target using the equations in section 4.2.1.2.

4.4.3 Photogrammetric Simulation Experimental Design

The photogrammetric system was simulated by taking a collection of image measurement locations and tie point locations, along with 4 scale measurements made between arbitrary points in the cloud, to produce the photogrammetric bundle adjustment, using inner constraints and a fixed camera datum in turn. These scale measurements were used because it mimics the methods used by Rémi Méreuze, who independently collected photogrammetric data of the same site on the same day that the TLS data was collected. In this simulation, the specified number of planar sample points was 2000, but the actual number was 2324.

4.4.3.1 Generating Camera Locations

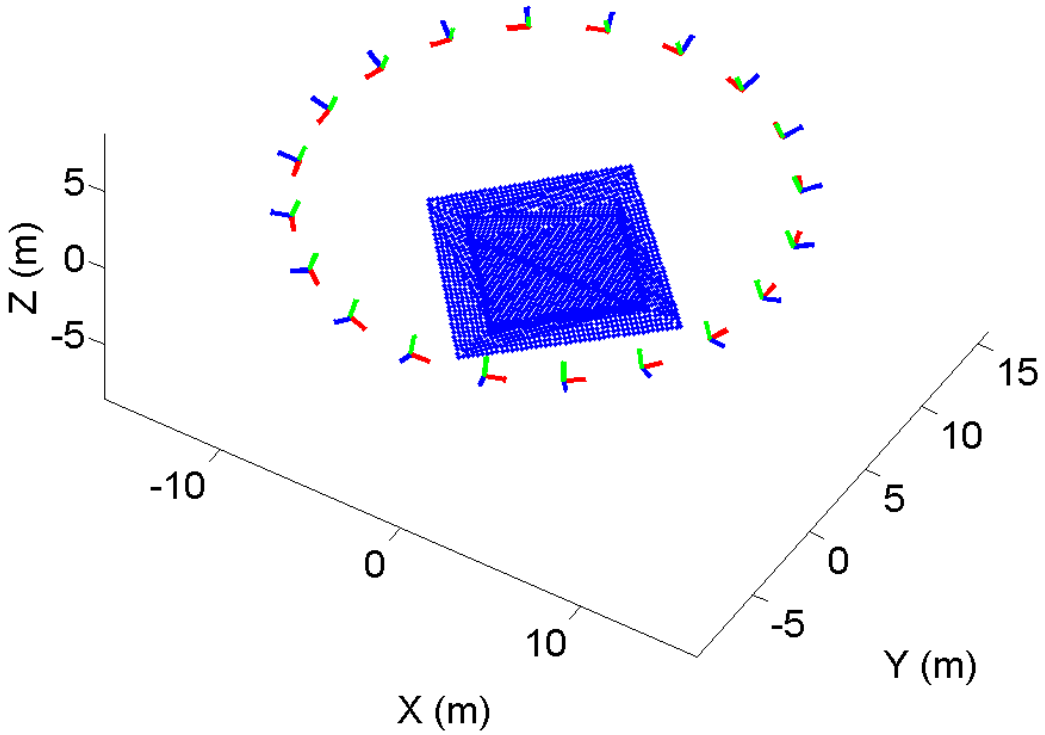

Figure 4. 19 Demonstration of camera locations relative to sampled model.
Scale measurements were made between two randomly selected points of appropriate separation.

The camera locations in this simulation were formed in a circle, centered approximately 2m above the planar models with a 12m radius. This can be seen in Figure 4.19. There were 20 camera locations, equally spaced around the circle. Each camera faced the approximate center of the model, angled downward to have a good vantage point on the whole model. These image locations were selected in order to strike a balance between being accurate to the actual photogrammetric data collected at Kuukpak, and being simple to recreate within the simulation.

4.4.3.2 Camera Parameters

The images used in this simulation are closely modeled after the camera used to collect the field data, which was the Pentax k-30. The specifications of the camera are in the Table 4.2.

Table 4. 2 Camera Specifications

Specification	Value
Principal distance	24mm
Number of pixel columns	4928 pixels
Number of pixel rows	3264 pixels
Pixel Size	0.00481 mm
Principal point	0, 0

In this simulation, no lens distortions were modeled, assuming instead that the camera was well calibrated.

4.4.3.3 Image Point Precision

There are many different factors that can affect the precision that image points can be measured at. The main factor is the visual distinctiveness across the surface of the subject being documented, since a homogenous visual texture does not provide features to serve as tie points. Another factor is the lighting conditions of the environment, since a low light environment will necessitate a wider aperture or a larger ISO, which will either put parts of the images out of focus, or increases the levels of noise in the images, respectively. Additionally, changing lighting conditions between images (i.e. if subsequent images are darker or lighter than previous images) then the tie point extraction might be individually reliable for each image, but tie point matching will be significantly impaired, to the point of making it a futile process. None of these factors are considered in this simulation, instead assuming that the matching is performed without error, and that all points are extracted with maximum precision.

Barazzetti et al. [22] showed in that, in favourable conditions, automatically extracted tie points can be as precise as can be expected from tie points extracted from 'expert' users, and are in almost all cases more numerous, with the added benefit of being significantly faster and less tedious to extract. The precision that was reported for automatically extracted image measurements of tie points was as low as 0.29 pixels in the x and y directions. The standard deviation used in this simulation differs from this value slightly, simply due to numerical convenience.

Table 4. 3 Observation Standard Deviations for Photogrammetry

Observable Standard deviation	**Magnitude**
σ_x, σ_y	0.33 pixels
σ_d	$\pm$1 mm

4.4.3.4 Performing the Bundle Adjustment using FEMBUN

Once all of the images' observation have been reduced to pairs, it is then possible to create the files that will be used with the software FEMBUN [9] to perform the bundle adjustment. These files contains the camera information, the initial camera exterior and interior orientations, a list of control points (or tie points), as well as a set of scale observation.

The scale observation is designed to mimic how the actual photogrammetric model was scaled in the field. At Kuukpak, a set of rulers of varying lengths (50cm, 40cm, 30cm and 20cm) were placed in and around the excavation site. These each provided one scale measurement, of precision $\pm$ 1mm, as seen in Table 4.3 These rulers were then identified in the point cloud, and the point cloud scaled so that each one reflected their true size, as accurately as possible. In this simulation, then, 4 scale measurements were used as well. First, a random point was selected, and a second point was selected according to its distance from the first, that distance corresponding to

the length of the scale observation being replicated. If no other point in the cloud was within a threshold of the first point, then a new random point was selected.

There were two datum definitions that were possible for the bundle adjustment, the fixed camera datum (i.e. a single camera was fixed at its initial position and orientation to define the datum), and the inner constraints datum (or free network). This simulation has the opportunity to test both definitions to compare how they perform relative to each other, as well as how they both compare to the TLS data. While most automated photogrammetry software likely uses the fixed network, the free network generally gives better performance in terms of point precision, and so represents the best case scenario for data in this simulation.

4.5 Experimental Results

The precision of the TLS data is compared to the precision of the photogrammetric model using both inner constraints and a fixed camera datum in Figures 4.20 to 4.24.

The precision of the points can be summarized by the standard deviation in the direction of least precision for that point, which can also be described as the length of the largest axis of the standard error ellipsoid describing the precision of each point. This is the best method for comparing the point precisions because it expresses the precision of each point in a single value rather than in a set of values. It also does not overstate the precision of the points, since this value corresponds to the direction of least precision. The clouds' mean point precisions are compared in Table 4.4, and the point precisions is visualized in Figure 4.20, demonstrating the impact that choice of datum has on the precision of points across the model. The histograms of the least point precisions for all three measurement methods can be seen in Figure 4.21. These figures also demonstrate that the inner constraints datum for photogrammetry produces results comparable those attainable through TLS.

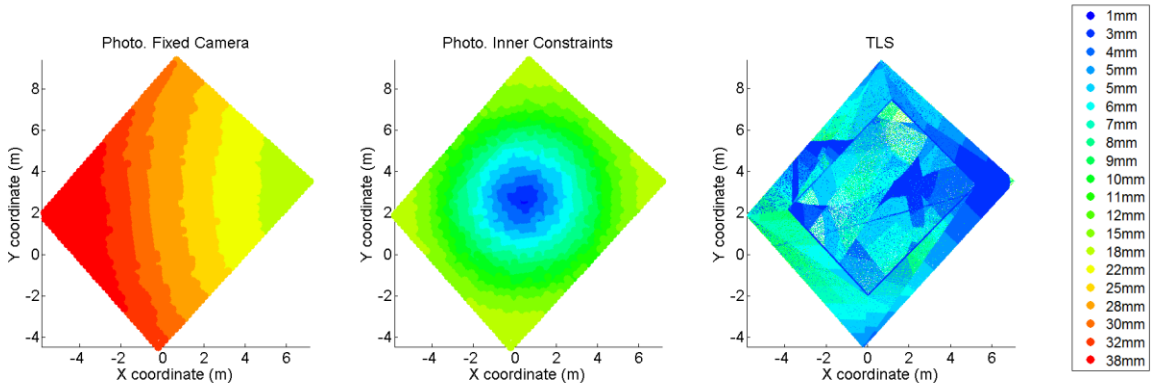

Figure 4. 20 Visualization of precisions, in direction of least precision.
Dark indicates a low precision, light indicates a high precision

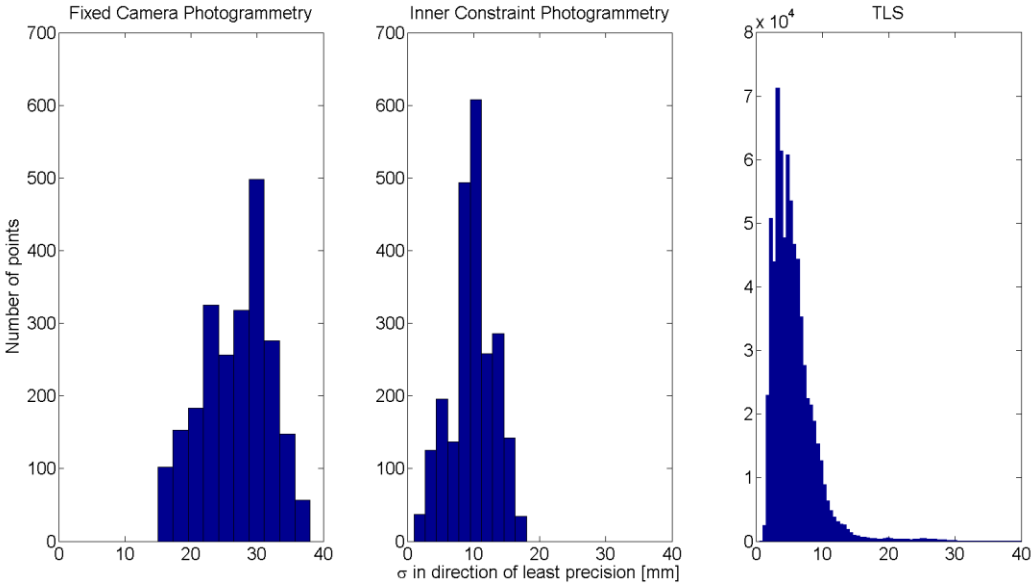

Figure 4. 21 Histograms of point precision in direction of least precision for fixed camera datum (left), IC datum (centre) and TLS (right)

The precision of the TLS in the X Y and Z directions is presented in Figure 4.22. It can be seen that although the precision in each direction follows the same positively-skewed behaviour, the precision in the Z direction is better than in the X or Y directions. This is due to the primarily horizontal planar sections, which will generally produce range observations that are oblique in the

horizontal directions, increasing the range observation standard deviation in either X or Y, but usually not in Z.

Table 4. 4 Comparisons of mean point precision of different methods

	TLS	Inner Constraints	Fixed Camera
Mean point precision	6 mm	10 mm	27 mm

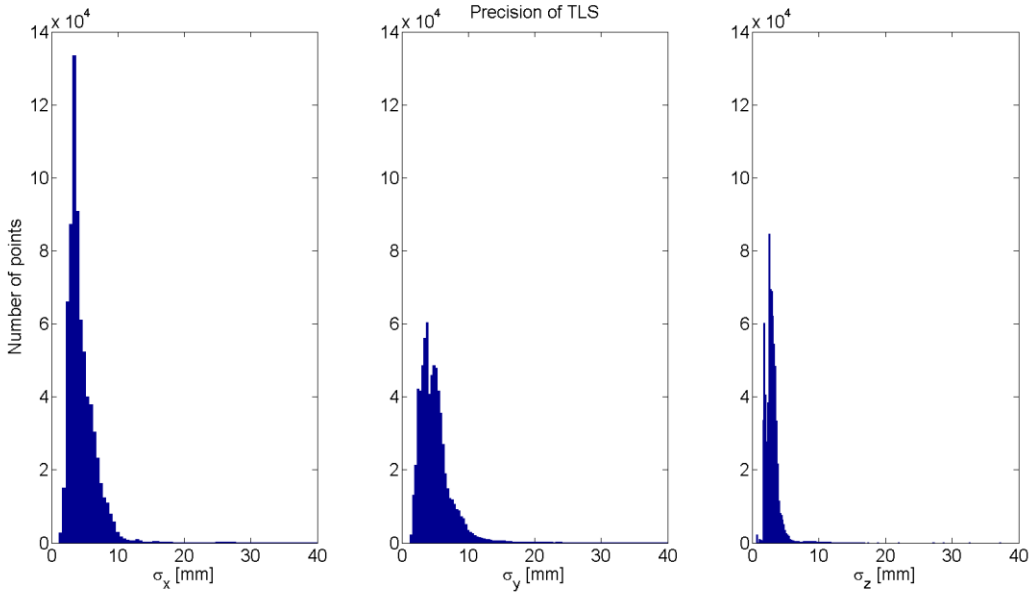

Figure 4. 22 Histograms of Point Precisions for TLS in the X, Y and Z directions

Figure 4.23 shows the point precision in the X, Y and Z directions for photogrammetry with the fixed camera datum. It can be seen that the X direction is significantly less precise than the Y or Z directions. This is primarily due to the fact that the camera that was fixed in this example was in the positive X direction relative to the model, and that the precision of the point cloud decreased relative to the distance from the fixed camera. The histogram of Z precision are grouped the way they are due to being recorded to the nearest millimeter.

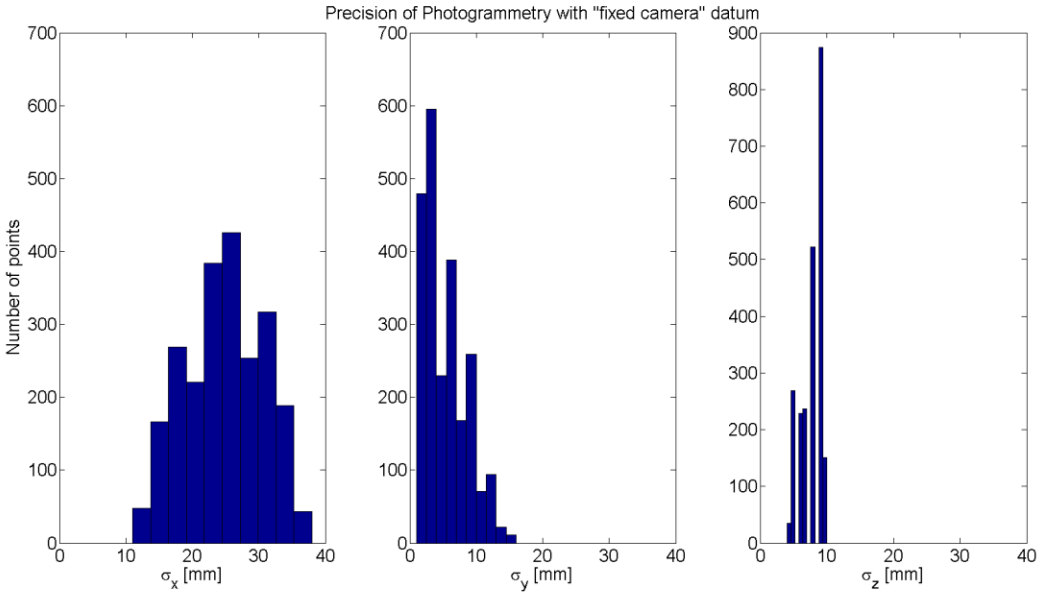

Figure 4. 23 Histograms of Point Precisions for photogrammetry using a fixed camera in the X, Y and Z directions

The precision of photogrammetry using inner constraints is shown in Figure 4.24. The precision in all three axes is improved relative to the fixed camera datum, but is especially improved in the X direction.

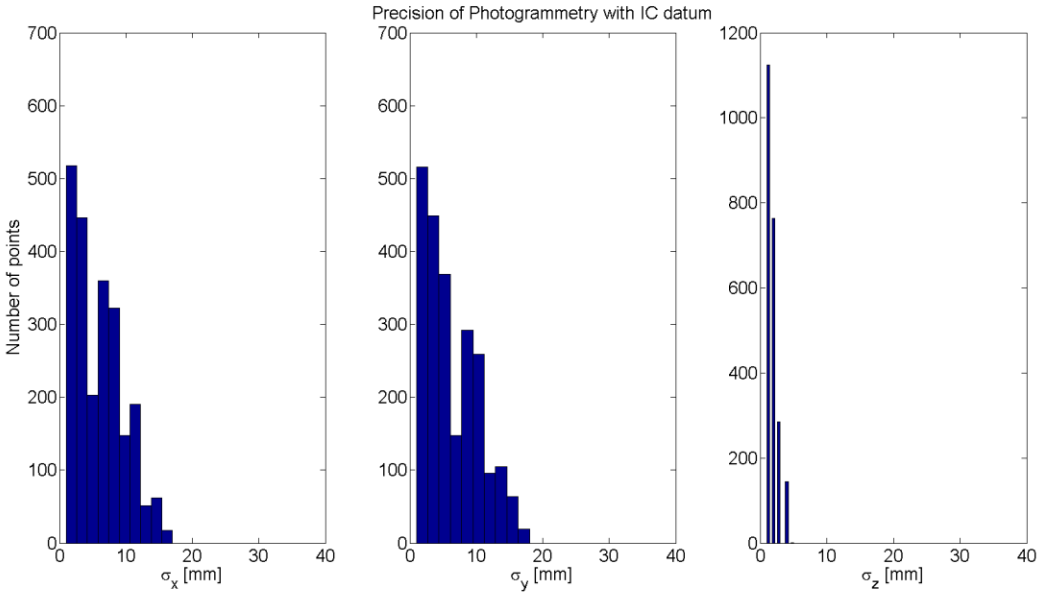

Figure 4. 24 Histograms of Point Precisions for photogrammetry using inner constraints in the X, Y and Z directions

Between the two photogrammetric models, it is clear that the fixed camera network has significantly worse average precision, by a factor of about 3. It can be seen in Figure 4.20 that the precision of the points in the fixed camera network get systematically worse the further they are from the location of the fixed camera, which is located to the right of the cloud. This is due in part to the fact that the non-fixed cameras determine their position relative to the fixed camera via their shared tie points. Any camera that does not share tie points with the fixed camera will have to determine their relative position to those that do, through their tie points. This process amplifies the uncertainty of the EOPs each time, which in turn propagates to the final point cloud. This demonstrates the importance of the choice of datum definition in the precision of the final model. It is, however, nearly impossible to automate the process necessary to use the inner constraints datum, since it relies on the initial estimates of the control points, information that must be acquired independently of the photogrammetric process.

It can be seen that the average point in the TLS data set has approximately half the standard deviation as the average point in the photogrammetric model using inner constraints. Additionally, the fixed network photogrammetric model produced a mean standard deviation approximately four times that of the TLS model. This confirms the popular belief that TLS data is generally more precise than data collected via photogrammetry. However, there are additional factors to consider.

In the case of photogrammetry, there were only 2324 points that were produced, while in TLS there were approximately 710,000 points produced. While these numbers appear to be significantly different, they are actually difficult to compare. This is because the photogrammetric method might have had significantly more points if it weren't for the bundle adjustment software running out of memory. On the other hand, the automated photogrammetric software tends to take create their point clouds in several steps. One step identifies well defined points in the images that

can be used to orient the cameras in the bundle adjustment (or a similar procedure), and another step referred to as dense matching, which performs a simple intersection to less well defined points after the cameras have been oriented. This second batch of points is of significantly lower precision, due to the fact that any errors in the cameras positions are directly propagated into these points, and because the tie point measurements are necessarily of lower quality. The simulation presented here only accounts for the first step in the process, indicating that a more dense but less precise point cloud could be produced with the addition of the second step.

It is possible for the photogrammetry datum to be improved, specifically if the network scale were to be defined using ground control point which were independently measured, for instance, with a total station and prism. If this were the case, then the network could be defined exclusively with these control points, assuming that their locations could be accurately measured within the digital images. This would not only define the position and orientation of the network, but also the scale, and with a precision not achievable using metre sticks. This would be especially important for larger scale photogrammetric surveys. However, the skills necessary to establish a control network are likely beyond those of most archaeologists.

Additionally, the tie points in the photogrammetric process were optimally spread out across the entire surface of the model. In actual photogrammetric images, the tie points are unlikely to follow this behaviour. They are much more likely to be grouped together in areas of high contrast, while areas of low contrast are likely to have very sparse points. This varying distribution of points would affect the imaging geometry significantly, most likely decreasing the precision of the cameras' EOPs, and therefore the precision of the entire point cloud. Whether or not the tie point distribution is better or worse than the distribution used by the simulation depends on the

visual characteristics of the subject being captured and of the light in the environment the subject is being imaged within.

Finally, for both imaging technologies, it was assumed throughout this thesis that both instruments being using in the simulation are well calibrated, with no un-modeled systematic errors present. While it is certainly possible, and desirable, to use well calibrated instruments when performing data collection, it is often difficult for a non-expert to calibrate their equipment properly [44]. In the case of TLS, these instruments are expensive, precision machines and are commonly calibrated by the manufacture (although the error model used by manufacturers are not always equivalent to the recommended error models determined in academia). However, in the case of photogrammetry, non-experts often calibrate their cameras themselves using software and procedures written by other people, with potentially dubious results. If a poorly calibrated camera is used when collecting photogrammetric data, then the precision of the data will suffer because of it.

4.6 Chapter Summary

In this chapter, TLS and photogrammetric data collection methods were simulated, and their measurement precision was modeled. The results of the error modeling process were compared systematically between the different measurement techniques, and it was found that the TLS data was more precise, on average, by a factor of 2 when compared to the more precise inner constraint network, and by a factor of about 4 when compared to the fixed camera network.

Chapter 5 – Modeling and Visualizing 3D Spatial Data

5.1 Introduction

Data, without a model to describe it in context, is just a set of numbers. For an archaeologist to find use in the 3D data that can be collected, it must be modeled in a way that is useful. In some instances, the model is concerned with the spatial information directly, such as in a deformation analysis. However, in many cases, archaeologists are less concerned about the precise shape and size of the spatial data, and more about the visual representation of the features themselves. In these situations, a very different type of modeling is necessary. Specifically, visual modeling. For a visual model, not only does the spatial data need to be faithfully represented, but the visual properties (i.e. colour, reflectance, lighting, etc.) need to be represented as accurately as possible as well.

Visual models are useful for many reasons. An archaeologist may want to create a visual model of a feature so that it can be shared with colleges who study similar features so that they may compare them. They may be shared with the indigenous peoples whose ancestors created those features, in a form of repatriation of cultural or traditional knowledge. Or they may be shared with the public at large to achieve public education and outreach. These goals can be achieved in different capacities through different methods of representation of the visual models, such as in panoramic photography, fully intractable 3D environments, or even in virtual reality.

Once the visual model has been created, there remains the question of "how does one best view this model?" There's a colloquialism that if a program is easy to use, then it was hard to make. This phrase applies in the case of creating tools for individuals to view and interact with models. Interacting with models in graphics software takes skill and practice but little work once

the model has been created. On the other hand, creating an application in which the user requires little to no instruction or practice requires significant amounts of work after the model itself has been created.

There are many different methods that can be used to create a visual model, many of which require a certain amount of artistic license, to create a model that 'feels like' the scene rather than being precisely accurate according to the data. This chapter will describe the process that can be taken to create a visual model that is both accurate to the spatial data, but also visually pleasing. The uses and applications of different visualization tools are also explored in this chapter.

5.2 Creating Geometric Models

Although there is a very large amount of geometric information that is available in a point cloud, it is not necessarily useful until it is used to create a model. In the context of this chapter, a geometric model will refer to a triangulated mesh which, as closely as possible, represents the geometric information present in a point cloud and therefore the subject being documented. A visual model on the other hand is any representation of the subject that primarily emphasizes its visual characteristics. These properties are not mutually exclusive, and in fact visual geometric models can offer information not necessarily available in either a geometric or visual model alone.

5.2.1 Triangulation of a single point cloud with known observation adjacency

In some situations, a point cloud can be directly converted into a simple mesh model if the adjacency of point observations relative to the measurement device are known. For instance, TLS measurements can be converted into a simple mesh by connecting point measurements according to their adjacency in the spherical coordinates they were originally measured in. Put another way,

a point observation can serve as a vertex for a triangular planar section (also known as a face) if there exists a point observation for one angular increment either above or below, and either left or right of the point in question. These three points define a face, but this face may not faithfully represent the geometry of the scene being scanned, especially if one of the points has a range observation significantly different than the others. This can create faces that demonstrate false connectivity between point observations. However, these faces have an angle of incidence (from the perspective of the scanner location) that are very large, so a threshold can be set to exclude these faces from the model. Figure 5.1 shows the results of this, with the coloured point cloud on the left, a triangulation with an angular threshold of 60° on the right. The point cloud demonstrates the abundance of information present in the scan, while the triangulated mesh on the right demonstrates the connectivity present in the scan. The triangulated mesh on the right also demonstrates the relative loss of information when excluding faces with high angles of incidence.

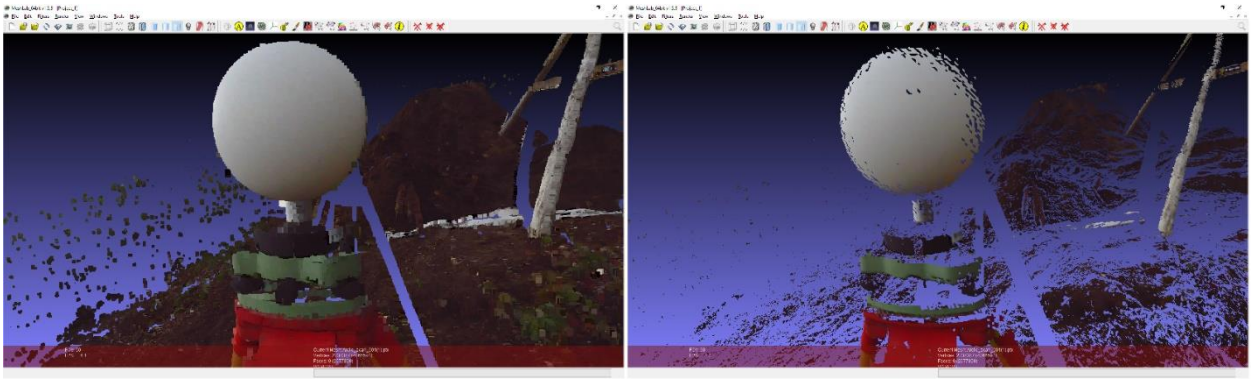

Figure 5. 1 Left: A point cloud of a spherical target. Right: Triangulated mesh, excluding faces with an angle of incidence >60°

The drawback of this method of triangulation is that, although it is simple and can create highly detailed models in areas of high point density, it tends to create ‘over sampled’ regions, where many triangular faces are used to represent areas in which only a few larger faces would represent the same information. It can also create models with blank sections, where the angle of incidence was too high or range observations were sparse, since faces can only be added when sets

of three adjacent points coincide. This method also does not take into consideration the potential variance of range data, since even small errors in range will cause the mesh to appear uneven or bumpy.

Furthermore, when several models from different scans are combined, sections of overlap between the scans will have potentially conflicting geometric information. Combining these models in a way that faithfully represents the data but results in a 'manifold mesh' (i.e. that the model is a single, non-self intersecting surface) is a non-trivial problem.

5.2.2 Poisson Surface Reconstruction

If the adjacency of points in a point cloud is not known (i.e. if the point cloud exists only as a set of XYZ coordinates) then the simple method of triangulation of the point cloud mentioned previously will not work, since connectivity cannot be determined with confidence. Instead, areas of connectivity need to be inferred from the point cloud itself, which is a very challenging prospect. Without any additional information, many common arrangements of points become ambiguous. For example, it may not be clear if two planar sets of points which are parallel yet offset by a small amount belong to two separate surfaces, perhaps either side of a thin feature, or belong to the same side of a features and the offset is the result of a registration error. This is visualized in Figure 5.2, where two different scenes produce the same arrangement of point measurements.

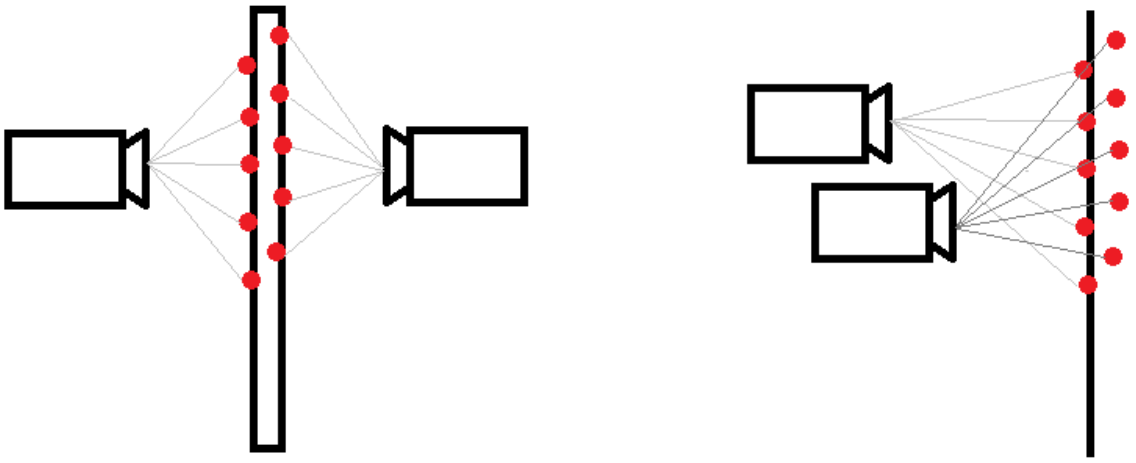

Figure 5. 2 Left: Point measurements on two opposing sides of a thin feature. Right: Point measurements on the same side of a features, offset due to registration errors. Both arrangements of points are identical, but represent different surfaces.

Fortunately, in the case of point clouds collected from TLS or photogrammetry, there is usually additional information that can be used to significantly improve the process of creating models from unorganized point clouds. The information that is most useful is the estimated normal direction of each point, which can be calculated based on the neighbourhood of points for each point in the point cloud, ensuring that the normal direction is in the direction towards the device that measured the point. Conceptually, this information can help disambiguate under-lying shapes of point clouds, such as in the example above, in that two close but separate layers of a thin feature will have normal directions opposite each other, whereas if they are of the same layer but demonstrate an error in registration, then they will have normal vectors in approximately the same direction. This is visualized in Figure 5.3, where the red dots represent surface samples, and the black arrows represent their estimated surface normal vectors.

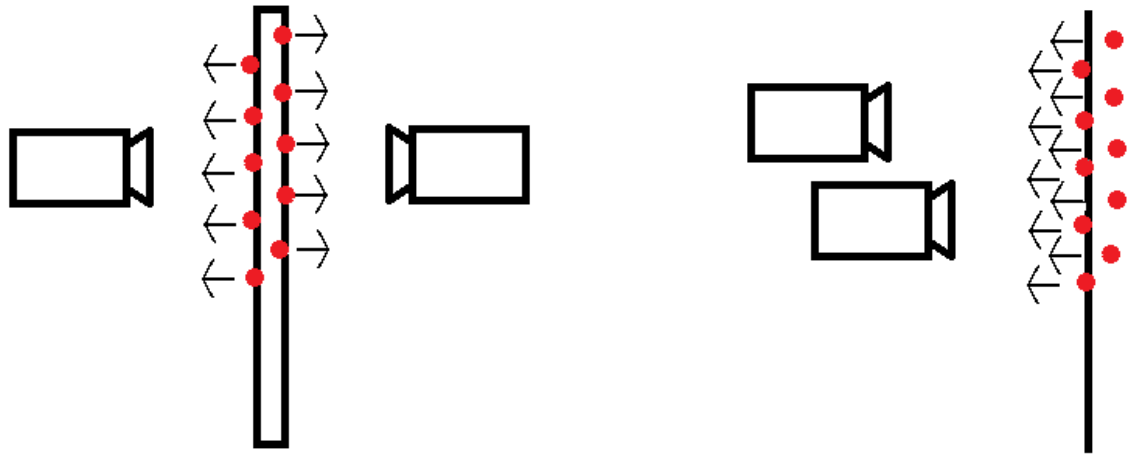

Figure 5. 3 Left: Point clouds with differing normal vectors, indicating separate surfaces. Right: Point clouds with aligned normal vectors, indicating same surface.

However, knowledge of the normal of points can do far more than disambiguate, they allow for a highly accurate method of surface modeling, called Poisson Surface Reconstruction [16]. Conceptually, the process can be understood as follows. Take an object whose surface, ∂M, one wishes to model. If the model can be described as having a value of 1 internal to the object, and a value of 0 external to the object (i.e. an indicator function, χ), then the gradient of that function will have a value of zero everywhere except at the surface of the object, where it will be equal to the normal vector directed inwards, producing the vector field $\vec{V}$. This can be expressed as

$$\nabla\chi = \vec{V} \tag{5.1}$$

Therefore, a collection of samples, $s \in S$, which contain a location at the surface of the object, $s.p$, and contain an estimate for the normal direction at that point, $s.\vec{N}$, can be thought of as samples of the vector field produced by the gradient of the indicator function at that location, $\vec{V}$. However, for χ to exist the vector field $\vec{V}$ must be conservative (i.e. the curl must be exactly

equal to zero). Therefore, instead of calculating χ directly, the divergence can be found for equation (5.1),

$$\nabla^2\chi = \nabla \cdot \vec{V} \tag{5.2}$$

and χ be found according to a least-squares approximate solution. This is a Poisson equation, which has known solutions.

A Poisson equation is a partial differential equation, formulated as

$$\nabla^2\varphi = f \tag{5.3}$$

where ∇^2 is the Laplace operator (divergence of the gradient), φ is an indicator function and f a scalar field, both on a manifold, and visualized in Figure 5.4. In this case, the value f corresponds to $\nabla \cdot \vec{V}$, the divergence of the normal direction of the surface points, and φ corresponds to χ, indicator function of surface ∂M.

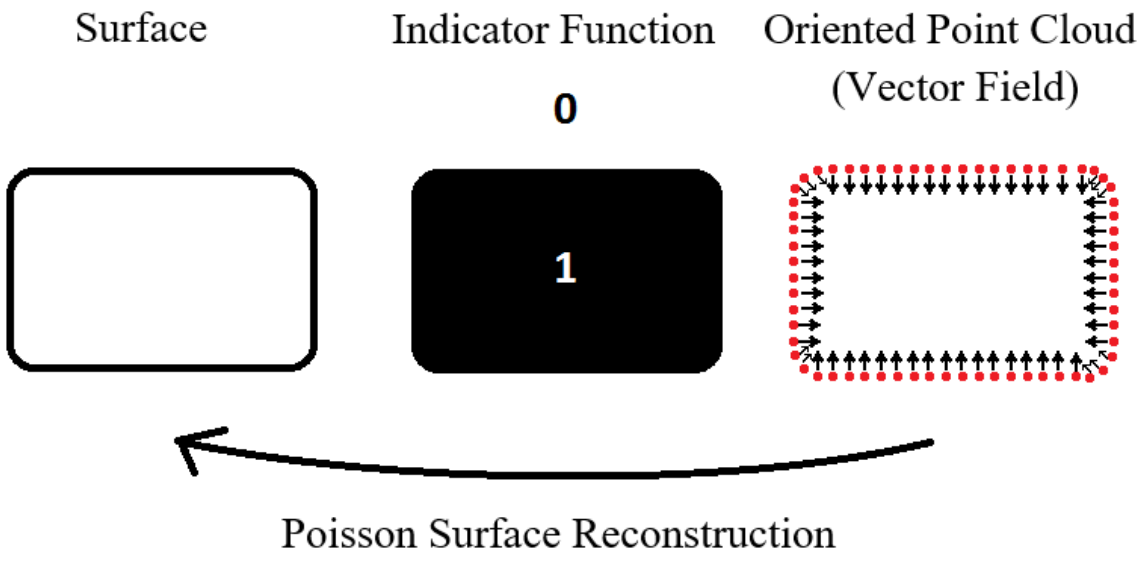

Figure 5. 4 Poisson Surface Reconstruction

The first step involved in performing the Poisson Surface Reconstruction is to approximate the gradient field of the indicator function, $\vec{V}$, from the set of surface samples S. A necessary step

in this process is to smooth the indicator function, χ, with a smoothing kernel, F, so that the gradient is bounded. This means that the integral is approximated with a summation over discrete patches, P_s, which make up the surface ∂M, where each patch is centered on sample s. This is written as

$$\nabla(\chi * F)(q) \approx \sum_{s \in S} |P_s| \, F_{s.p}(q) \, s.\vec{N} \equiv \vec{V}(q) \tag{5.4}$$

where F is a Gaussian with variance on the order of the sample spacing, q is a location in the vector field, $|P_s|$ is the area of patch P for sample s. This formulation implies that the sample spacing is constant, which may not be the case. This is addressed at a later step.

Now that the vector field $\vec{V}$ is estimated, a solution to equation (5.2) can be found. First, the space must be discretized. For this problem, an adaptive octree $\mathcal{O}$ is used, since an accurate representation of the implicit function is only necessary near the reconstructed surface. The position of the sample points are used to define the octree and associate a function F_o to each node $o \in \mathcal{O}$, choosing the tree and functions so that the following conditions are satisfied:

1. The vector field V can be represented as the as the linear sum of the F_o
2. The matrix representation of the Poisson equation, expressed in terms of the F_o can be solved efficiently
3. A representation of the indicator function as the sum of the F_o can be evaluated near the surface of the model.

Given a set of point samples S and a maximum tree depth of D, the octree $\mathcal{O}$ is defined as the minimal octree with the property that every point sample falls into a leaf node at depth D. For every node $o \in \mathcal{O}$, F_o is set to be the unit-integral "node function" centered about the node o and stretched by the size of o,

$$F_o(q) = F\left(\frac{(q - o.c)}{o.w}\right)\frac{1}{o.w^3} \tag{5.5}$$

where o.c and o.w are the center and width of node o, respectively.

If each sample were to be replaced with the center of the node containing it, then $\vec{V}$ could be expressed as the linear sum of F_o by setting

$$F(q) = \tilde{F}\left(\frac{q}{2^D}\right) \tag{5.6}$$

since each sample would contribute a single term (the normal vector) to the coefficient corresponding to its leaf's node function. Since the sample width is 2^{-D} and the samples all fall into leaf nodes of depth D, the errors arising from the clamping can never exceed half the sampling width, and is further improved using trilinear interpolation. The approximation of the gradient field of the indicator function can be written as

$$\vec{V}(q) = \sum_{s \in S} \sum_{o \in Ngbr_D(s)} \alpha_{o,s} F_o(q)\, s.\vec{N} \tag{5.7}$$

where $Ngbr_D(s)$ are the eight depth-D nodes closest to s.p, and $a_{o,s}$ are the trilinear interpolation weights.

Now that the vector field has been defined, it is possible to solve for the function χ by minimizing the function

$$\sum_{o \in \mathcal{O}} \left\| \langle \nabla^2 \chi - \nabla \cdot \vec{V}, F_o \rangle \right\|^2 = \sum_{o \in \mathcal{O}} \left\| \langle \nabla^2 \chi, F_o \rangle - \langle \nabla \cdot \vec{V}, F_o \rangle \right\|^2 \tag{5.8}$$

Where $\langle \nabla^2 \chi, F_o \rangle$ indicate the inner product of $\nabla^2 \chi$ and F_o. Given then $|\mathcal{O}|$-dimentional vector v whose o-th coordinate in $v_o = \langle \nabla \cdot \vec{V}, F_o \rangle$, the goal is to solve for the function χ such that the vector obtainted by projecting the Laplacian of χ onto each of the F_o is as close to v as possible.

To express this is matrix form, let $\chi = \sum_o x_o F_o$ so that the vector x is to be solved for. Then, let the $|\mathcal{O}| \times |\mathcal{O}|$ matrix L be defined such that Lx returns the dot product of the Laplacian with each of element of F_o. Specifically, for all $o, o' \in \mathcal{O}$, the (o, o')-th entry of L is set to:

$$L_{o,o'} \equiv \langle \frac{\partial^2 F_o}{\partial x^2}, F_{o'} \rangle + \langle \frac{\partial^2 F_o}{\partial y^2}, F_{o'} \rangle + \langle \frac{\partial^2 F_o}{\partial z^2}, F_{o'} \rangle \tag{5.9}$$

This formulation means that solving for χ amounts to

$$\min \|Lx - v\|^2$$

which is a basic least-squares formulation. Furthermore, because F_o are compactly supported, L is sparse, and because $\int f'' g = -\int f' g'$, L is symmetric.

Once the indicator function χ has been found, the reconstructed isosurface ∂M can be extracted by selecting an isovalue. The isovalue is chosen so that the extracted isosurface closely approximates the position of the input samples. This is done by evaluating χ at the sample points and using the average of the values for isosurface extraction

$$\partial M \equiv \{q \in \mathbb{R}^3 | \chi(q) = \gamma\} \tag{5.10}$$

with

$$\gamma = \frac{1}{|S|} \sum_{s \in S} \chi(s.p) \tag{5.11}$$

This value of isovalue has the property that scaling χ does not affect the isosurface, meaning that the isosurface can be determined with only knowledge of $\vec{V}$.

Finally, the case of non-uniform samples is considered. The first step is to estimate the local sampling density, and to scale the contribution of each point accordingly. In addition, the kernel width is adapted, so that fine features in densely sample regions can be preserved while sparsely sampled region are smoothly fit. Given a depth $\hat{D} \leq D$ the density estimator is set to be the sum of node function at depth $\hat{D}$

$$W_{\hat{D}}(q) = \sum_{s \in S} \sum_{o \in Ngbr_{\hat{D}}(s)} \alpha_{o,s} F_o(q) \tag{5.12}$$

Using the density estimator, equation (5.7) can be modified to

$$\vec{V}(q) = \sum_{s \in S} \frac{1}{W_{\widehat{D}}(s.p)} \sum_{o \in Ngbr_{Depth(s.p)}(s)} \alpha_{o,s} F_o(q) \tag{5.13}$$

where

$$Depth(s.p) \equiv \min\left(D, D + \log_4 \frac{W_{\widehat{D}}(s.p)}{W}\right) \tag{5.14}$$

This formulation is used because each sample's contribution is proportional to its associated area on the surface. However, better noise filtering is possible by adapting the width of the smoothing filter $\tilde{F}$ to the local sampling density.

Finally, the isovalue is determined using the formulation

$$\gamma = \frac{\sum_{s \in S} \frac{1}{W_{\widehat{D}}(s.p)} \chi(s.p)}{\sum_{s \in S} \frac{1}{W_{\widehat{D}}(s.p)}} \tag{5.15}$$

which produces a weighted average of χ at the sample positions.

5.2.3 Sampling spatial data to produce oriented point clouds – Poisson-disk Sampling

In some situations, mesh models and points clouds can have much more information than can be practically handled by a user. In the case of points clouds, this over-abundance of data may be due to over sampling in certain regions, for example, in regions very close to a TLS device. In the case of mesh models, the excess of data is likely due to their creation using high quality modeling parameters. In both cases, having more data than is necessary will cause programs to run slowly, or to crash unexpectedly as computers run out of usable memory.

To overcome this, the data must be sampled, but this is not necessarily a simple task. In the case of a point cloud, taking a random subset of points as a sample will not overcome a disparity

of point cloud density. In the case of a mesh model, creating a sample in even less intuitive. For instance, projecting a regular grid of sample points onto the surface will not necessarily produce samples that are regularly spaced on the surface. Instead, the method used to sample the clouds and meshes in this thesis in the "Poisson-disk Sampling [29]." This method generates a uniformly random distribution where the minimum distance between two samples is 2r, meaning that a disk of radius r centered on each sample cannot overlap with any other disk. This method ensures that the data will be uniformly sampled, while enabling a certain minimum number of samples be specified by controlling the radius of the disks.

Generating the location of the samples can be a complex process when not dealing with uniform surfaces. For instance, the connectivity of a 3D mesh, or the number or shape of the faces might impact the sampling algorithm. Fortunately, the Poisson-disk Sampling algorithm in [29] is independent of all these factors, producing well distributed samples across the entire surface.

5.3 Creating Visual Models from Geometric Models

By default, meshes produced through the Poisson Surface Reconstruction have no visual information associated with them, apart from the shape of the surface. However, in the process of performing data collection with TLS, many scanners can be used to collect full colour panoramic photos from the perspective of the scanner. These are usually used to colour the point clouds, but can also be used to colour the mesh produced by point clouds as well. This is advantageous since a coloured point cloud can only have one colour value associated per point, whereas a mesh can have a texture with as high a resolution that one wishes to use. In cases where the surface was created through photogrammetry rather than TLS, this data is even more well suited for adding visual information to the surface, due to the necessarily visual data used to perform photogrammetry. In fact, it is even possible to add visual information to a model created from TLS

data using photographs taken of the same scene completely independently of the scan, as long as those photos can be oriented relative to the model.

5.3.1 Principles of UV mapping

The first task that is necessary for a texture (i.e. digital image) to be applied to a 3D surface is for the surface to be mapped to a 2D coordinate system. This process is very similar to how the surface of the earth is mapped via mapping projections. This 2D coordinate system relates a texture to each planar section of the 3D model, so that the texture appears on the surface of the model. The process of mapping the vertices of a 3D model to a 2D coordinate system is called "UV mapping", since the 2D coordinates system is referenced to as UV space. The UV coordinates are mapped to values between 0 and 1, with the coordinate [0,0] being the lower left-hand corner of the texture, while the coordinate [1,1] represents the upper right-hand corner of the texture. This is the case no matter the dimensions of the texture being used, so the scale of UV space relative to pixel size may be different between the U and V directions. This also means that the same UV map can be used for different textures, no matter the shape or size of those textures. Figure 5.5 demonstrates an example of a UV map for a complex model, in this case, of a money head.

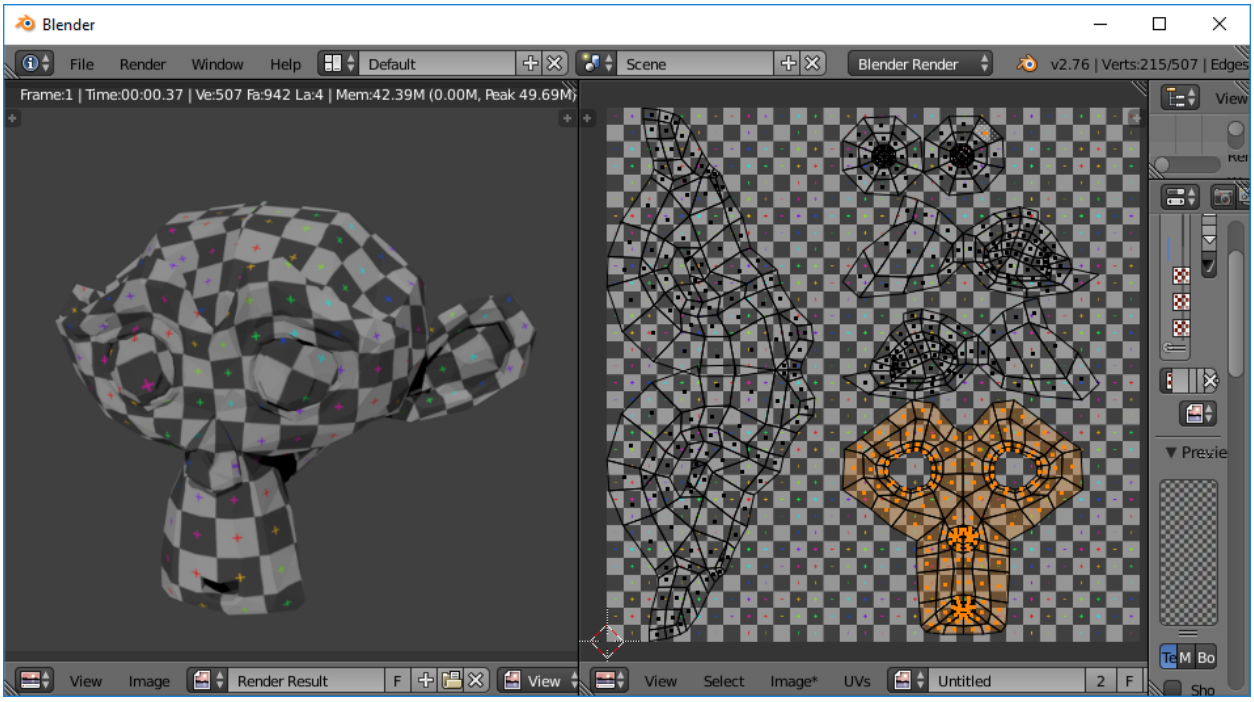

Figure 5. 5 UV map example. The 3D faces of the model (left) have been mapped to the 2D UV map (right). Highlighted in orange is the UV map corresponding to the face of the monkey.

A desirable property for a UV map is for the relative size of faces in 3D space to match their projections in UV space. This will ensure that each face will have proportionate numbers of pixels, so that details aren't concentrated in some faces and are sparse in others. Another beneficial property is that the shape of the face's UV projection is approximately the same as the 3D face. Figure 5.6 demonstrates the effects that a bad quality UV map can have on the model. On the right, there are enormous disparities in the size of mapped UV faces, some being too small to see clearly, and others very large. This has the effect on the model that different faces will have different have amount of information from the texture. For instance, the textures around the monkey's right eye is very compact, indicating an abundance of pixels dedicated to those faces. On the other hand, the texture around the monkey's left eye is very spread out, indicating very few pixels are used for those areas.

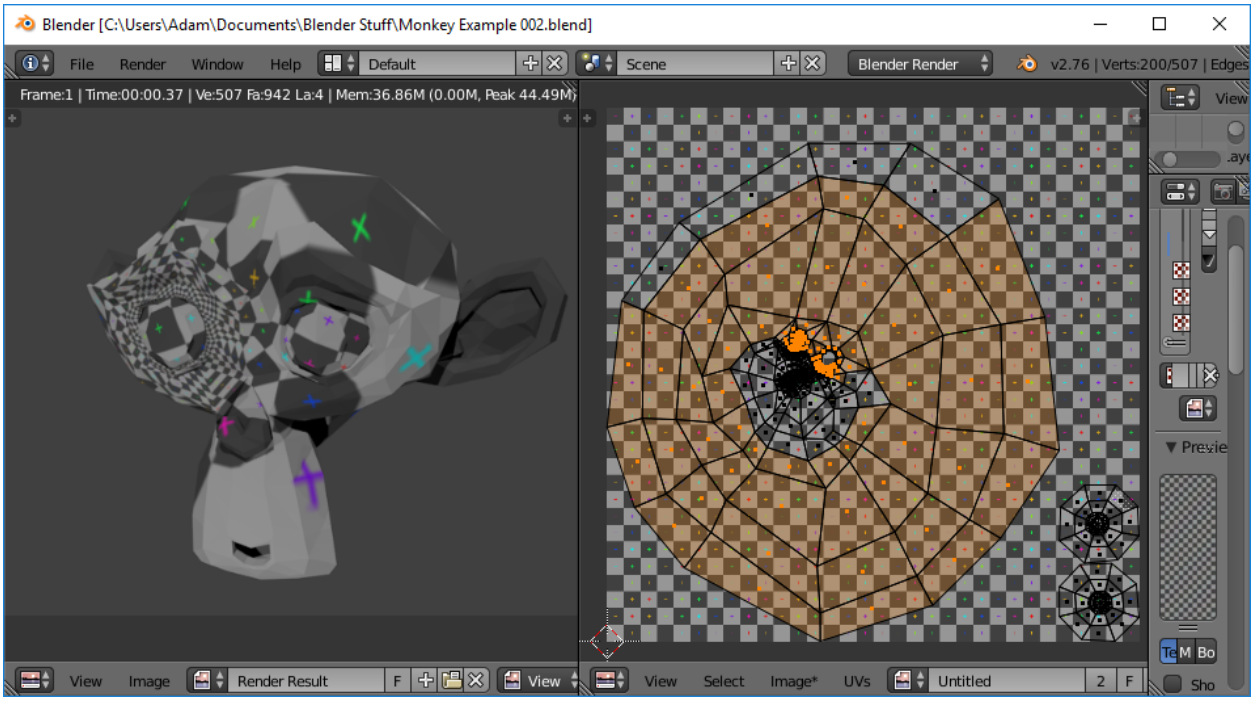

Figure 5. 6 A bad quality UV map of the monkey model.

UV maps should be made in a way that try to make the most possible use out of the available texture, since the pixels in the texture are what ultimately define the visual quality of the model. If large portions of a texture are left unused by the UV map, then the memory used to define those pixels is wasted. Therefore, a good UV map will not leave significant portions of the texture unused, instead closely packing the different sections of the UV map together to maximize the amount of visual information the texture can store for the model.

Finally, another beneficial property is for the UV map to be as continuous as possible, such that faces which share edges in the 3D model also share edges in UV space. Unless the model is a flat 2D shape, it's unrealistic to expect the UV map to be entirely continuous, so discontinuities should be managed strategically. This helps to reduce the amount of 'seam tearing,' a phenomenon where discontinuities in the texture are visible on the model, and visualized in Figure 5.7. This property is at odds with the first two properties, however, since for complex models, the shape and size of UV mapped faces can only match their 3D counterparts if the model is liberally cut apart,

and the UV map can only be continuous if the shape and size of the faces are changed dramatically. Usually, strategic edges on the model are specified by the user as being UV seams, meaning that discontinuities of the texture in these areas are less important, and can therefore be used to improve the quality of the shape and size of the faces, while not creating texture seams in important areas. The results of this method are visible in Figure 5.5 right, in that the different sections of the monkey's head (the face, both ears, eyes and the back of the head) are separate from each other but otherwise continuous.

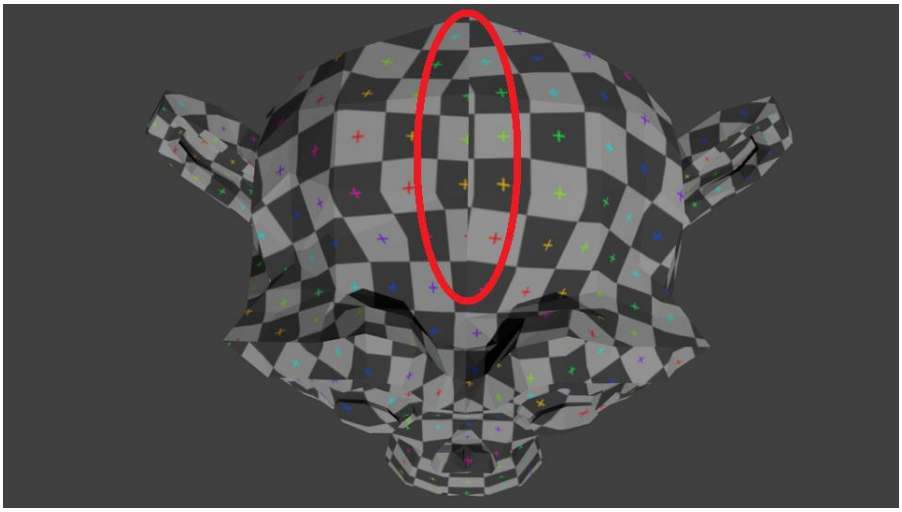

Figure 5. 7 The UV is discontinuous along the back of the monkey's head, causing the texture to not line up.

Creating a good UV map for a given model is a complex task that is more of an art than a science in many cases. However, a tool that will always be a part of that process is a UV mapping algorithm, which are discussed next.

5.3.2 UV mapping algorithms

A UV mapping algorithm is any algorithm that can be used to relate the 3D coordinates of a face within a model to a 2D equivalent. Commonly, the desirable properties of UV maps will be prioritized in one form or another, though different algorithms prioritize different properties.

Additionally, many algorithms will ensure that the coordinates of each face will be unique, but others will not ensure this, instead preserving other properties.

An example of a UV mapping algorithm is the "Unwrap" method, included in *Blender* (version 2.76 and earlier). This method tries to unwrap the model continuously, while preserving the shape and size of the faces if possible. This method can produce varying results. The UV map in Figure 5.6 was produced by this algorithm, and demonstrates the extreme distortions that preserving continuity can have on a UV map. On the other hand, Figure 5.5 was also produced using this algorithm, but by including seams in specific locations. Seams are used to tell the UV mapping algorithm to treat certain edges as disconnected, which can therefore be used to more accurately preserve the shape and size of the faces in the model. This method can produce very good results, but usually requires a large amount of user interaction to do so.

Another mapping algorithm available from *Blender* is the "UV Smart Project" algorithm. This algorithm minimizes distortions of size of shape of the model's faces, liberally cutting the model into sections of contiguous faces where necessary, as visualized in Figure 5.8. The Smart Project algorithm can be seen to produce a UV map which is tightly packed, using nearly all of the available pixels in the texture, and that each face is appropriately shaped and scaled. However, unlike the "Unwrap" algorithm, the model is cut into many small parts without concern for persevering continuity. This algorithm is popular because it makes useful UV maps while requiring very little user input.

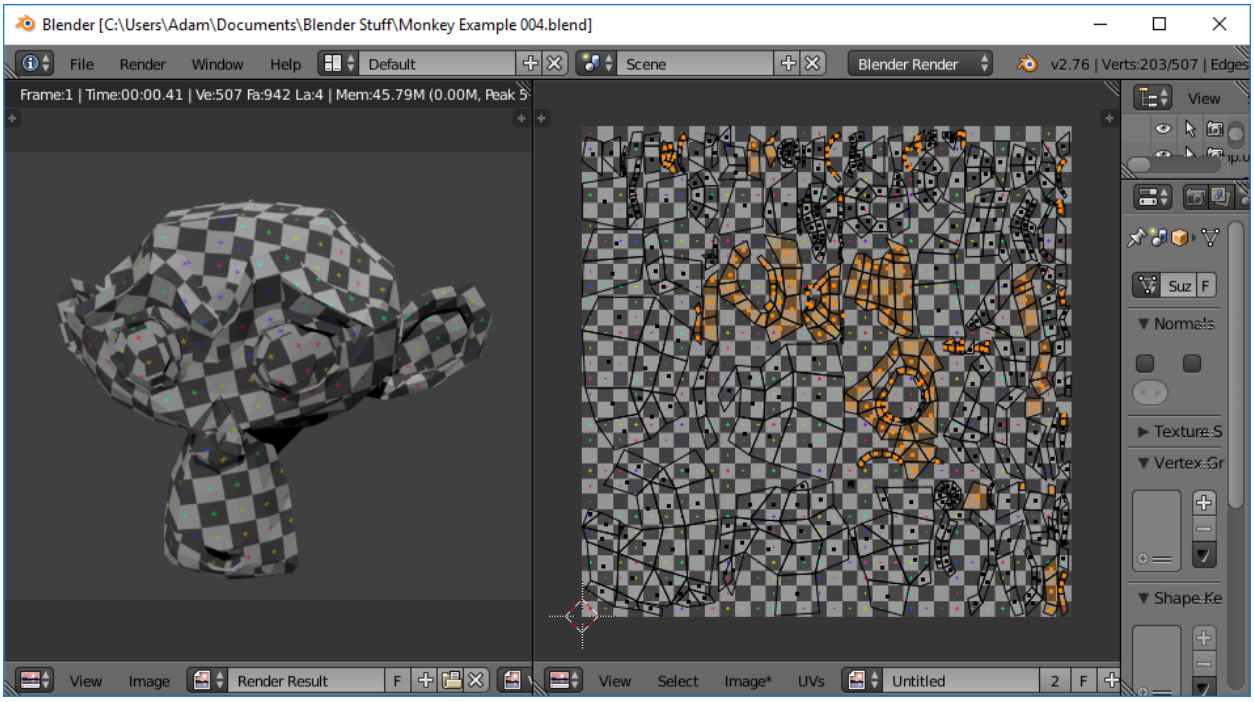

Figure 5. 8 UV map produced by "UV Smart Project"

Another UV mapping algorithm available in *Blender* is “Lightmap Pack,” which simply packs all faces as orthogonal shapes (right angled triangles, squares or rectangle), such that all faces lay adjacent, and use as much of the texture as possible, as seen in Figure 5.9. This algorithm prioritizes making efficient use of the texture over preserving the shape and size of the faces, and does not consider connectivity at all. This method is especially useful for maximizing the use of memory allocated for a texture, which is useful if memory resources are limited.

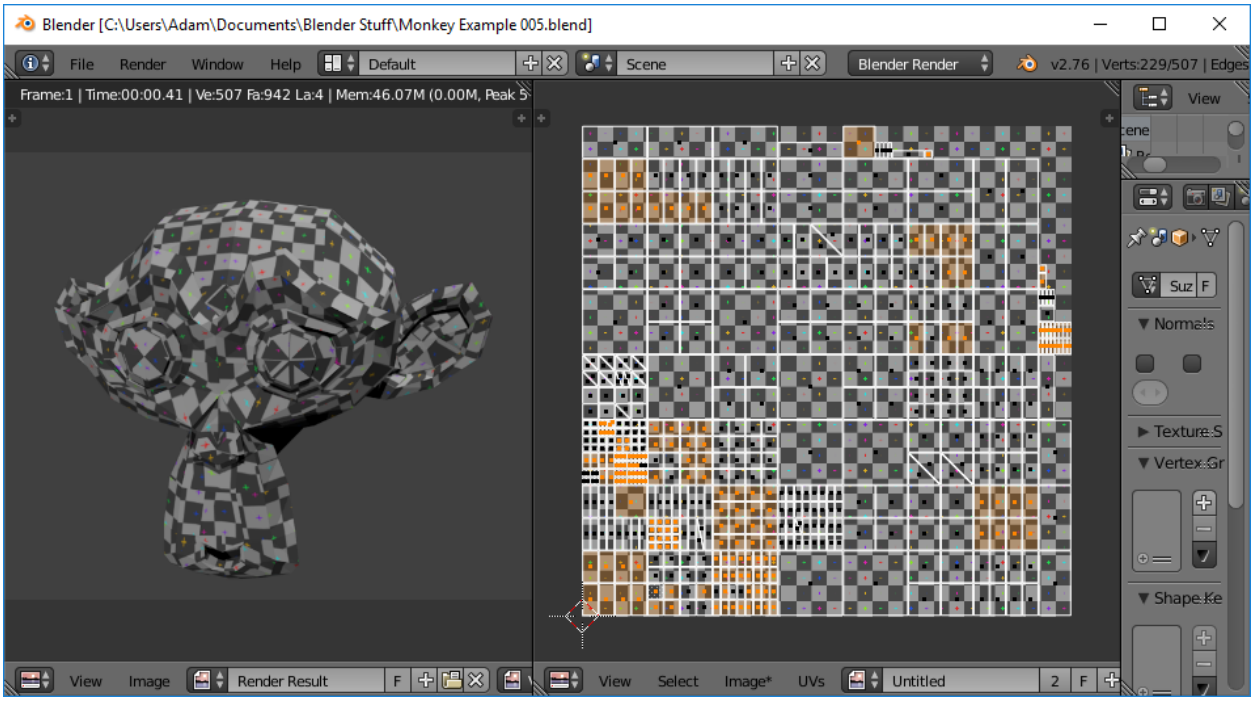

Figure 5. 9 UV map produced by "Lightmap Pack"

All of the UV mapping algorithms discussed so far are designed to ensure that each face of the model is uniquely mapped within UV space, which is to say that faces do not overlap in UV space. This is good if each face is to have its own unique portion of the texture, but in some cases overlapping UV faces is acceptable. For instance, it is possible to take advantage of the symmetry in a model by mapping only half of the model, and repeating the texture for the appropriate symmetric portions. Another instance when UV faces may overlap is if the texture already exists, and the UV map is being made to match the texture. This is in contrast to the UV mapping algorithms already discussed, since in those cases the UV map is created first, and the texture would be created after the fact, according to the UV map. If the UV map is being made to match a texture that already exists, (such as a digital photograph), then there may not be any actual control over how the faces are laid out in UV space.

One such UV mapping technique is to use projective geometry from a specific perspective to calculate the UV coordinates. This method will not usually be useful for mapping all faces of a

model, but it can be very useful for using photographs as textures if the location of the photo is known relative to the model being textured. This technique is particularly useful when texturing a model created through digital photogrammetry, since the digital photographs used to create the model can be used to texture the model directly, since their coordinates relative to the model would be known though the bundle adjustment. An example of how the build-in “Project from View” UV mapping algorithm might appear can be seen in Figure 5.10. From this perspective, the model has the regular checker-board pattern displayed in the texture, but from a different perspective, the pattern can be seen to distort, as in Figure 5.11.

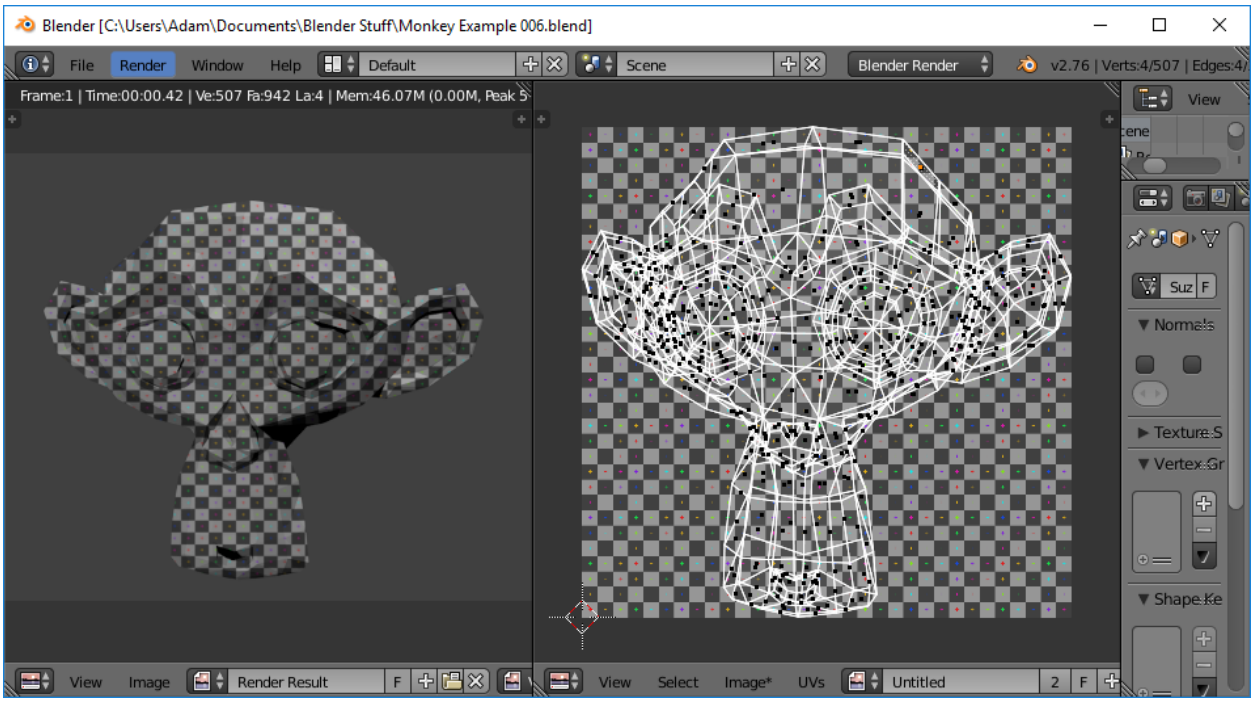

Figure 5. 10 UV map produced by "Project from View”

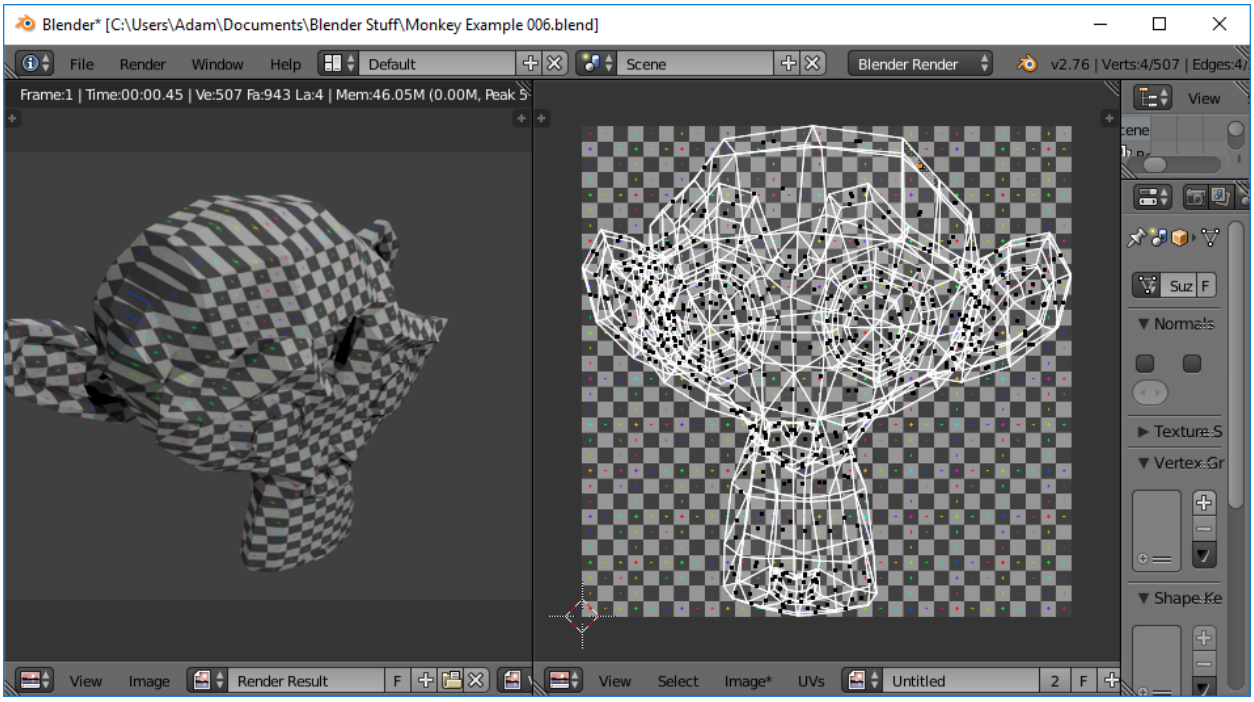

Figure 5. 11 Alternate view of the model with UV map created from "Project from View"

Another useful UV mapping technique is to express the UV coordinates of the model's faces in terms of angular spherical coordinate values. This is useful if the model is approximately spherical, although can also lead to scale issues near the poles of the sphere. This UV mapping technique is particularly useful for using panoramic photographs (also called equirectangular photographs) as textures. If the panoramic photo takes up the full 360° by 180° field of view, then angular spherical coordinates map directly onto the image, if the perspective centre of the photosphere is known relative to the model being textured. This can be seen in Figure 5.12, where the left side is a portion of the panoramic photograph, and the right side is the UV map that corresponds to that portion of the image. The shape of the model is clearly visible in the structure of the UV map as it follows the curves and contours.

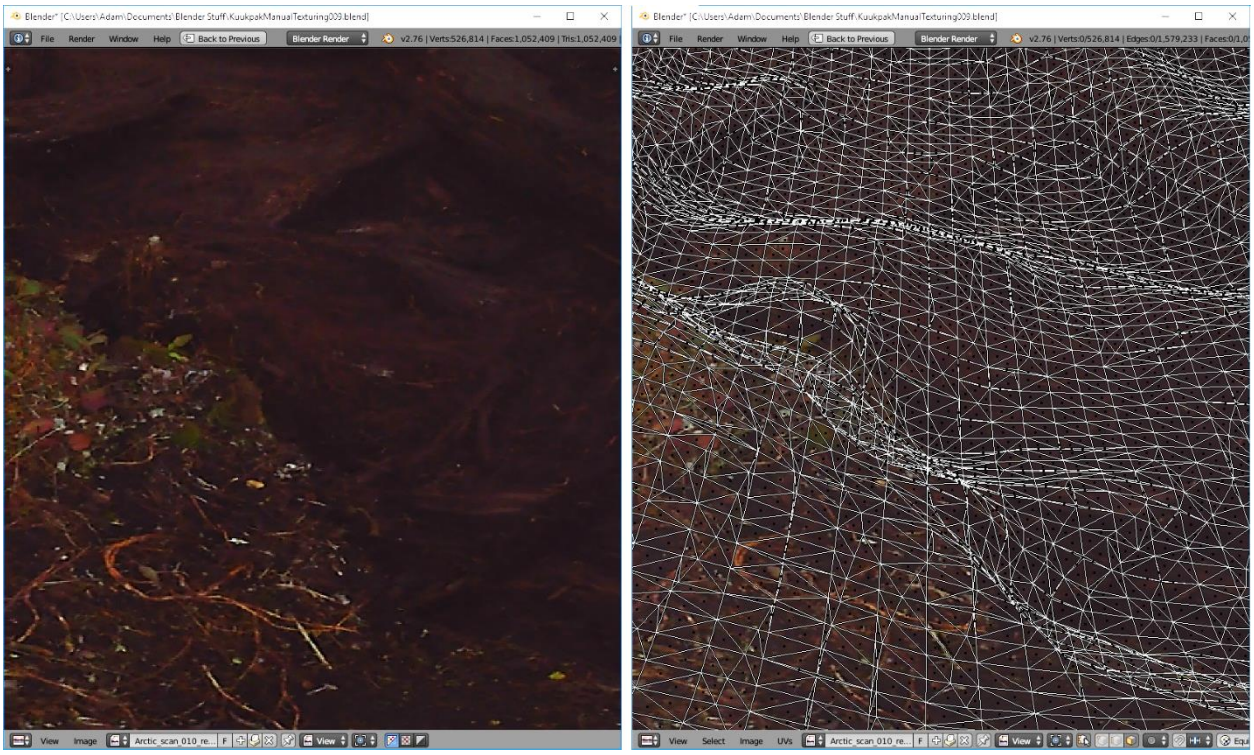

Figure 5. 12 Example of panoramic UV mapping

5.3.3 Calculating UV Maps for Photographic Textures

When modeling real world objects, it is commonly desirable to use digital photographs of these objects as the texture, because under normal lighting conditions these digital photographs can contain more visual detail than is often practical to produce artificially. However, using digital photographs introduces complications which must be overcome to produce a good quality visual model.

It is noteworthy that in many 3D modelling applications, a model will be mapped to UV space, and a texture will be created for that model according to the UV map and the visual properties that the user wants the model to have. In other situations, such as the case where one wishes to use digital photographs as textures, the texture already exists and the UV map must be made to match that texture. In the first case, the UV map is made according to the desirable

properties of a UV map, where as in the second case the UV map is made to precisely match the shape of the subject within the digital photograph.

In most 3D modeling software, only a limited number of UV maps are available for each model. This is because using more than one UV map simultaneously is computationally expensive, and tends to produce visually poor results. On the other hand, having more than one UV map for a model enables some powerful texturing techniques to be used, even if only a single UV map is used when the model is being rendered.

One such useful texturing technique that takes advantage of multiple UV maps is to create a single 'Master' UV map that represents the entire model and has all the desirable properties of a UV map, while each individual digital photograph gets its own UV map representing that image's perspective of the subject. Each digital photo can then be copied to the appropriate portion of the Master texture, creating a composite texture representing the entire surface. There are many steps necessary to perform this procedure, which will be outlined here.

For a digital photograph to be used as a texture for a given model, the UV coordinates of the model must be calculated. In the case of normal projective photographs, the coordinates of the camera must be known relative to the model. This is can be done through either a single photo resection, or through the bundle adjustment, depending on how the model was produced and what features are available to orient the camera.

Once the camera's position is known relative to the model, then the UV coordinates of the vertices of the model can be calculated using the collinearity equations, modified slightly so that the minimum observable image point coordinates in x and y are equal to 0, and the maximum observable image point coordinates are equal to 1,

$$U_{uv_{ij}} = \frac{x_{ij}}{x_{size}} + \frac{1}{2} \tag{5.16}$$

and

$$V_{uv_{ij}} = \frac{y_{ij}}{y_{size}} + \frac{1}{2} \tag{5.17}$$

where $U_{uv_{ij}}$ and $V_{uv_{ij}}$ are the UV coordinates of the model's i^{th} vertex in the j^{th} image, x_{ij} and y_{ij} are the image point coordinates of the model's i^{th} vertex in the j^{th} image as calculated using the collinearity equations (4.42) and (4.43), and x_{size} and y_{size} are the dimensions of the CCD array of the camera in use. Note that because not every vertex in the model will fall within the image plane of the camera, some UV coordinates will fall outside of the 0 to 1 range. This indicates that there is no data for those faces from this image, so the alpha channel (i.e. the level of transparency) for those sections can be set to 0 (i.e. fully transparent) so that they do not impact the texturing process.

On the other hand, if the photograph being used is a photosphere (i.e. a composite photograph representing a full 360° by 180° view of the surrounding scene), then a different UV mapping procedure is necessary. In this case, the relative location of the camera to the model is still necessary, but rather than an adapted collinearity equation being used to calculate the vertex UV coordinates, adapted spherical coordinates are used instead,

$$U_{uv_{ij}} = \frac{\left(\operatorname{atan}\left(\frac{\left(Y_i - Y_j^c\right)}{-\left(X_i - X_j^c\right)}\right) - \kappa_j\right)}{2\pi} + \frac{1}{2} \tag{5.18}$$

and

$$V_{uv_{ij}} = \frac{\operatorname{atan}\left(\frac{\sqrt{\left(Y_i - Y_j^c\right)^2 + \left(X_i - X_j^c\right)^2}}{-\left(Z_i - Z_j^c\right)}\right)}{\pi} \tag{5.19}$$

where $[X_i\ Y_i\ Z_i]$ are the coordinates of the i^{th} vertex of the model, $[X_j^c\ Y_j^c\ Z_j^c]$ are the coordinates of the center of the j^{th} panoramic photo relative to the model, and κ_j is the rotation about the vertical axis of the j^{th} panoramic photo relative to the model, assuming that the panoramic photos are aligned with the vertical axis of the model. These different formulations are necessary because the panoramic photograph already combined projective photographs into the spherical coordinates, so the UV map simply needs to follow suit.

The formulation in equations 5.18 and 5.19 results in a UV map which creates an accurate correspondence between the panoramic photograph and the model, but which has nearly none of the desirable properties of a UV map, as seen in Figure 5.13. The UV map (right) is overlaid on a colour grid, so that the distortions and connectivity of the UV map can be seen clearly on the model (left). It's clear that in the model, the scale of the texture varies wildly across the image, with an abundance of pixels dedicated near the bottom of the panoramic image, and extremely few pixels dedicated at the model's extents, stretching the texture significantly. Furthermore, since faces in the UV map can easily overlap, occlusions are not accounted for. These issues are the motivation for creating a single, Master texture which combines all the visual information of each different panoramic image, rather than using a different, suboptimal UV map for each panoramic image. In this thesis, the Master UV map is created using *Blender*'s "Smart UV project" algorithm, and can be seen in Figure 5.14. This UV map has many more of the desirable properties, especially considering the relative scale of faces in the texture.

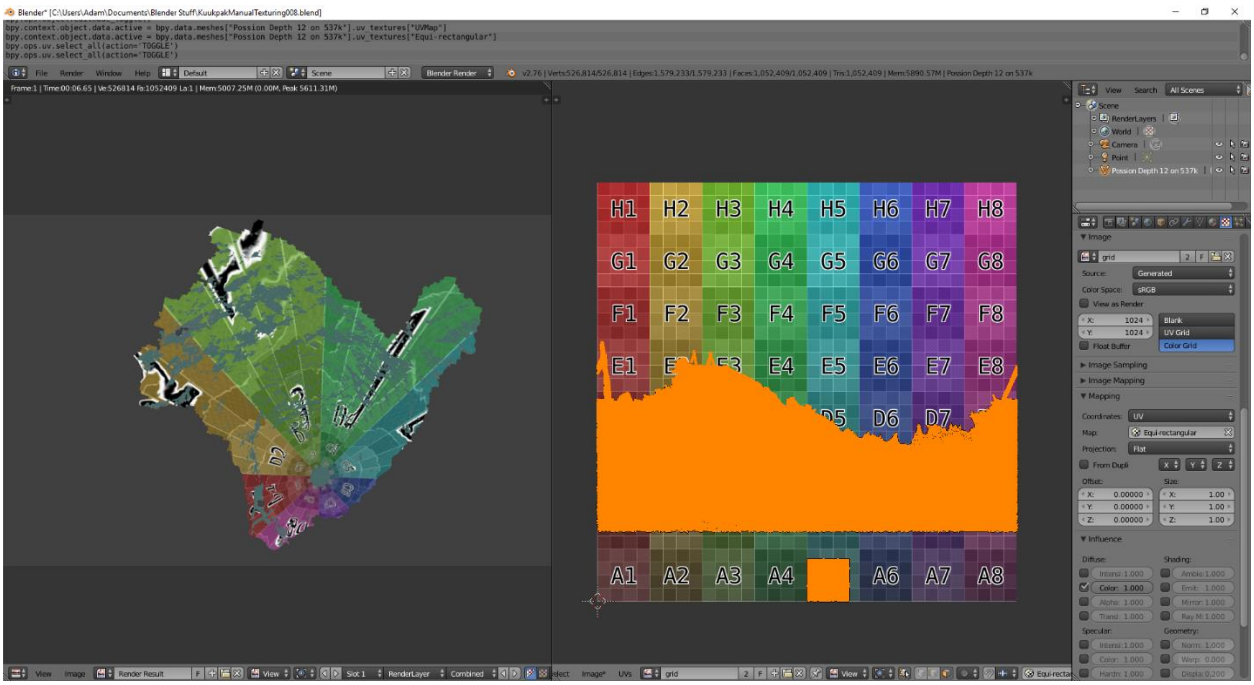

Figure 5. 13 Demonstration of the distortions present in a panoramic UV map

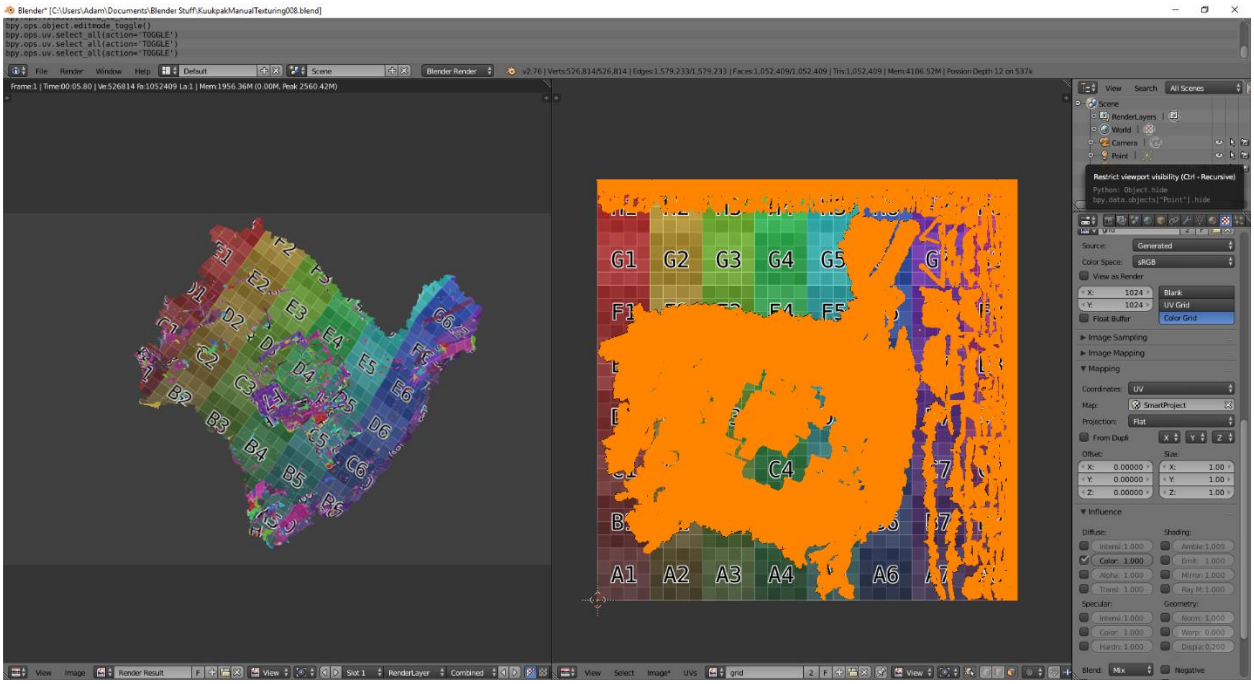

Figure 5. 14 Smart UV project on the model of the excavation at Kuukpak

In formulation in equations 5.18 and 5.19, all vertices will have a UV coordinate that falls between 0 and 1, but this can cause a problem for faces which overlap the left and right edges of the panoramic photo. In these cases, one vertex in the three defining a face will have a U coordinate

close to 1 or 0, while the other two vertices have a U coordinate the opposite. This causes the face to be come incorrectly spread across the entire texture, creating errors like in Figure 5.15 Top. The desired property of the face should be to take some of its texture from one edge of the photo, and the rest from the other side, as in Figure 5.15 Bottom.

Figure 5. 15 Top: Results of incorrect UV calculation around the edge of panoramic photos. Bottom: The corrected results of the texture.

The solution to this issue is to set the coordinates of the vertices of each face to be consistent with the coordinates of the centroid, allowing the vertices to take values greater than 1

or less than 0, preserving the correct shape of the face, as viewed in Figure 5.16. In this case, however, rather than setting the texture of faces outside the 0 to 1 range to be transparent, the texture can be repeated, treating a value of 1.01 as 0.01, correctly wrapping the texture for faces overlapping the edge of the panoramic photograph.

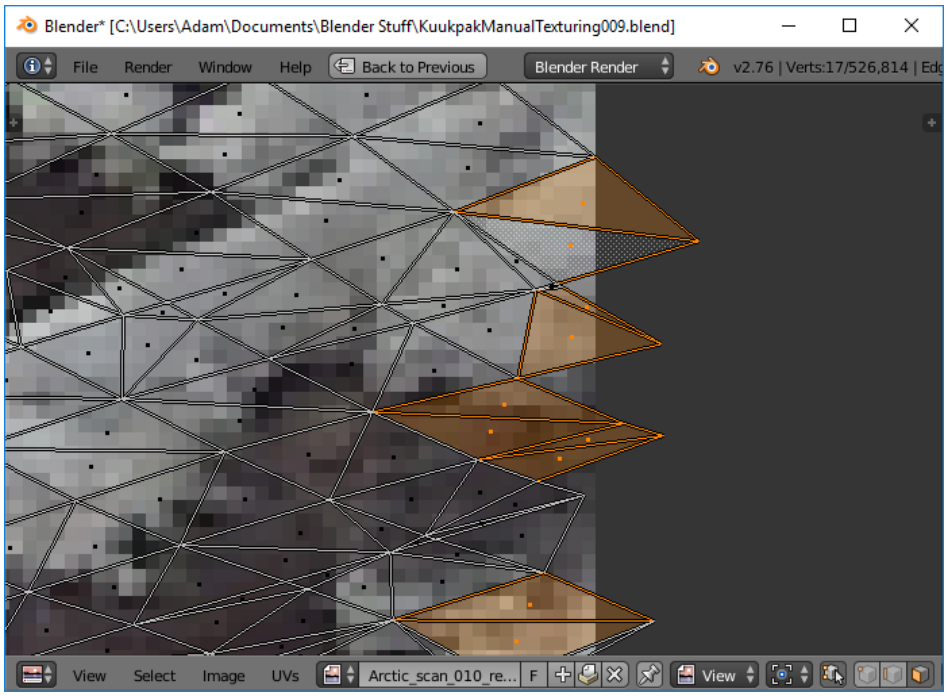

Figure 5. 16 UV faces, highlighted in orange, indicating overlap with the edge of the texture

Whenever a digital photograph is to be used as a texture for a model representing a real-world subject, it is important that lens distortions, such as radial or decentering distortions, be considered. These distortions are caused by departures of the physical camera's behaviour from the pin-hole camera model used in photogrammetry. If the model of the feature is accurate, but the photographs have some unmodeled lens distortions, then the model will not be projected onto the correct portions of the photographs, creating errors in texturing. These errors will be most prominent around the edges of the photographs, and in areas of high contrast. These lens distortions can be overcome by either including the Additional Parameters from the bundle adjustment when calculating the model's UV map, or by 'undistorting' the photos as per those same parameters, and

using the basic collinearity equations when calculating the UV map. Because of these potential distortions, if one has access to the raw photographs that were used to produce a panoramic photograph, better results are likely possible by using the raw photographs rather than the panoramic photograph. This allows the user to calculate an individual UV map for each photo, including the distortion modeling, rather than a single UV map for the panoramic photograph, which may or may not have accounted for photographic distortions.

Once the UV map for each individual image has been created, the textures can be copied to a new texture in the Master UV map format, so that they can be combined into a single texture at a later step. This process is called ‘texture baking,’ and simply copies the data from the UV coordinates in one texture to the appropriate UV coordinates of another texture. The two different textures are visualized in Figure 5.17, with the panoramic image on the left and the baked texture on the right. In this case, the texture bake copies the digital photograph’s information to the appropriate part of another texture which uses the Master UV map. This creates new textures, one for each digital photograph, containing all their visual information but in a common coordinate system, allowing for them to be compared and combined more easily.

Figure 5. 17 The panoramic image for scan 1 (left) is used to create the image texture (right) through texture baking to the Master UV map

5.3.4 Handling Self-Occlusions in UV maps

For any complex model, there will almost inevitably be a perspective from which the model blocks part of itself from view. In a digital photograph, this simply means that the foreground object occludes the background object. In a projective UV map, however, this means that the UV coordinates of the foreground face and the background face overlap, and the UV map has no way of knowing which face occludes the other. This produces errors where the texture from the foreground is duplicated onto the background incorrectly, as seen in Figure 5.18.

Figure 5. 18 Circled is an example of duplicated texturing due to a lack of occlusion modeling

The distance between a face and the scanner is an indication of which face occludes the other, but making use of that information when creating a UV map proves to be difficult. If one were to add faces to the UV map one at a time according to their range from the scanner, it would be possible to know that the faces added first are not occluded, and so any subsequently added overlapping faces are occluded. This algorithm, however, is imperfect. It fails to handle situations where faces are partially occluded, and its complexity grows geometrically with the number of faces making up the model, making it a slow and error prone method.

Fortunately, there are other solutions. 3D modeling software is largely concerned with the behaviour of light within the environment being represented in the scene, which means that there exist highly optimized tools used to track the path of light across a model. We can take advantage of these tools to create an occlusion map of our model given a perspective center. If a point light source were to be placed at the perspective center of the camera, then any surface that the light hits directly is within view of the camera, and any surface not directly hit by the light is occluded. The

test used to determine which faces are hit by the light is ray casting [1, 5]. Given a unit vector from a perspective, ray casting will determine the first surface struck. Therefore, given a pixel in a texture corresponding to a face, ray casting can be performed in the direction of the 3D equivalent position of that pixel. If the first surface struck does not correspond to the same face, then this pixel is occluded and it is set to black. If it does correspond, then it is not occluded, and it is set to white. This process can be done for every pixel in a texture that correspond to a face in the model. Furthermore, the many ray cast operations can be performed in parallel, meaning that it can be performed on a GPU to accelerate rendering times.

This method can be used to create a black-and-white texture of the model in the Master UV map representing the occluded and un-occluded regions of the model for a given camera, and is visualized in Figure 5.19 right. This texture and the texture baked from the digital photograph in the Master UV map (Figure 5.19 left) can be combined using a pixel-wise multiplication, the results of which can be seen in Figure 5.20. This allows for occlusions to be dealt with in a simple and efficient manner.

Figure 5. 19 Left: Texture created from a panoramic image. Right: Occlusion map for the same image. Some occluded sections are circled in red

Figure 5. 20 Generated texture with occlusion modeling

5.3.5 Creating a Weighted Average of Image Textures

Because each image contains differing levels of detail for various parts of the model, the process of combining these textures into a composite texture should consider how the visual information in an image changes based on the position and orientation of the camera relative to the model.

The weight used when calculating the influence each texture has on the composite texture depends on the geometry of the face relative to the camera. The amount of influence that a section of a digital photograph should have over the corresponding section of the composite texture corresponds to the number of pixels contained within the face's projection in that image. This is approximately proportional to the area of the face, to the angel of incidence from the camera to the plane, and inversely proportional to the square of the distance to the plane. For large faces, the distance and the angle of incidence from the camera can vary greatly across the surface. However, for small faces (such as the faces making up the model used in this thesis) the differences vary little, so each face' approximate projected area can be estimated by using the distance to, and angle of incidence at the centroid. Furthermore, because each face has a unique location in the Master UV map, the area of each face is a constant multiplicative value, which can then be excluded from the weight calculation. Therefore, the influence each face found in an image has over the corresponding section of the composite texture can be calculated as

$$w = \frac{\cos \eta}{d^2} \tag{5.20}$$

where w is the weight for a given face, η is the angle of incidence at the centroid of the face from the camera, and d is the distance from the camera to the centroid of the face, for each face in the model.

These weight values are used to create an entirely new texture in the Master UV map. These textures will be greyscale, with values corresponding to the weight that each pixel will contribute to the composite texture scaled between zero and one, and is visualized in Figure 5.21. In this case, a value of one corresponds to the weight of the closest face that is approximately normal to the camera, while zero corresponds to the furthest face that's nearly parallel to the camera. This scaling was performed so that the weight images could be saved and loaded as .png files which simplifies the data processing. In cases where the largest weight and lowest weight are very different between images, this scaling would cause miscalculations in the relative weights. In this case, however, the closest face was very similar between images due to the consistent camera height and spherical geometry of the panoramic images, and the minimum weight was negligibly small in all images, and therefore unlikely to contribute to the scaling in any significant manner.

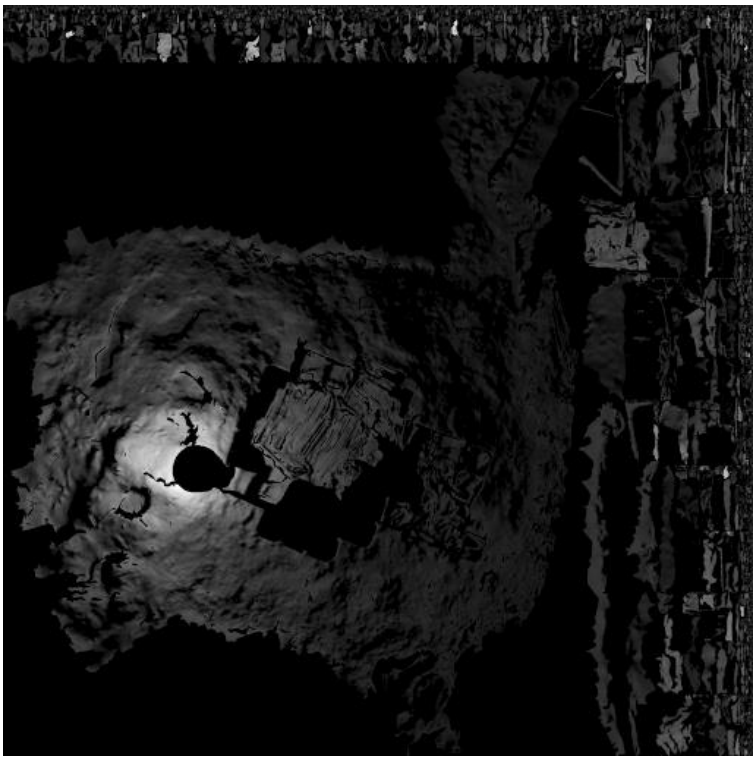

Figure 5. 21 Estimated weight image. This texture is used in addition to those in Figure 5.19 to create a Master texture.

The result of the preceding three sections are 3 textures for each photograph: A digital image, an occlusion map, and a weight map, each in the Master coordinate system. By combining these textures through an array-wise multiplication, summing all the images and normalizing over the sum of the weights, a weighted average texture is produced, combining the most detailed sections of each digital photograph into a single composite texture, which is visible in Figure 5.22.

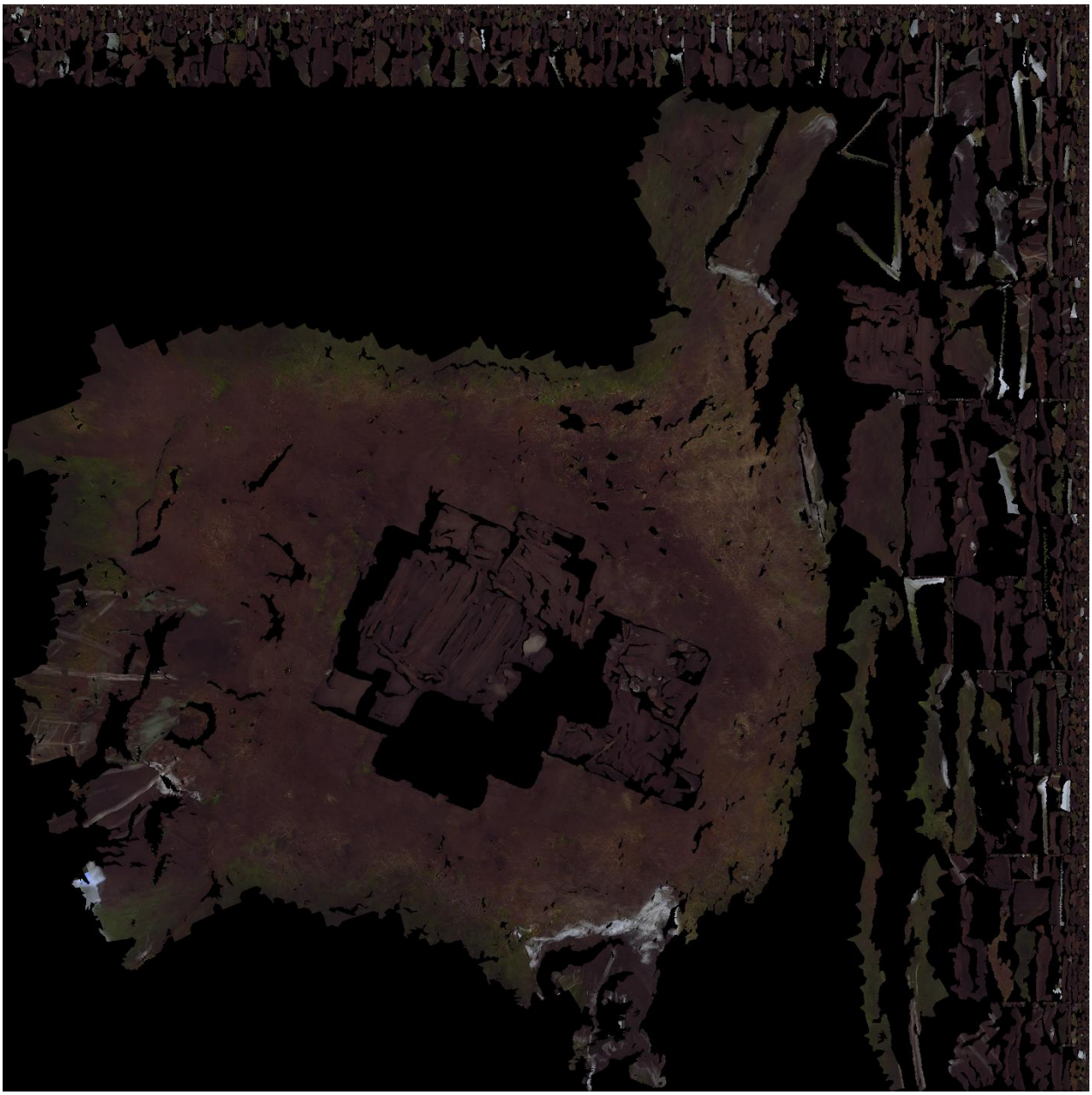

Figure 5. 22 Master texture, composed of a weighted average of all 10 original panoramic photographs of the feature.

5.3.6 Limitations of and Complications to Creating Composite Textures

The procedure outlined here is not flawless, of course. There are many small details that one must consider and potentially overcome to produce a high-quality composite texture of a model. One such consideration is that if there are any objects in the foreground of a digital photograph that one would like to use as a texture for a model, then the foreground object must be removed, as seen in Figure 5.23. Obviously, erasing an object from a digital image does not reveal what was behind it, so this process relies on there being redundant visual information of the subject, so that the missing information from a particular camera does not significantly impact the quality of the Master texture.

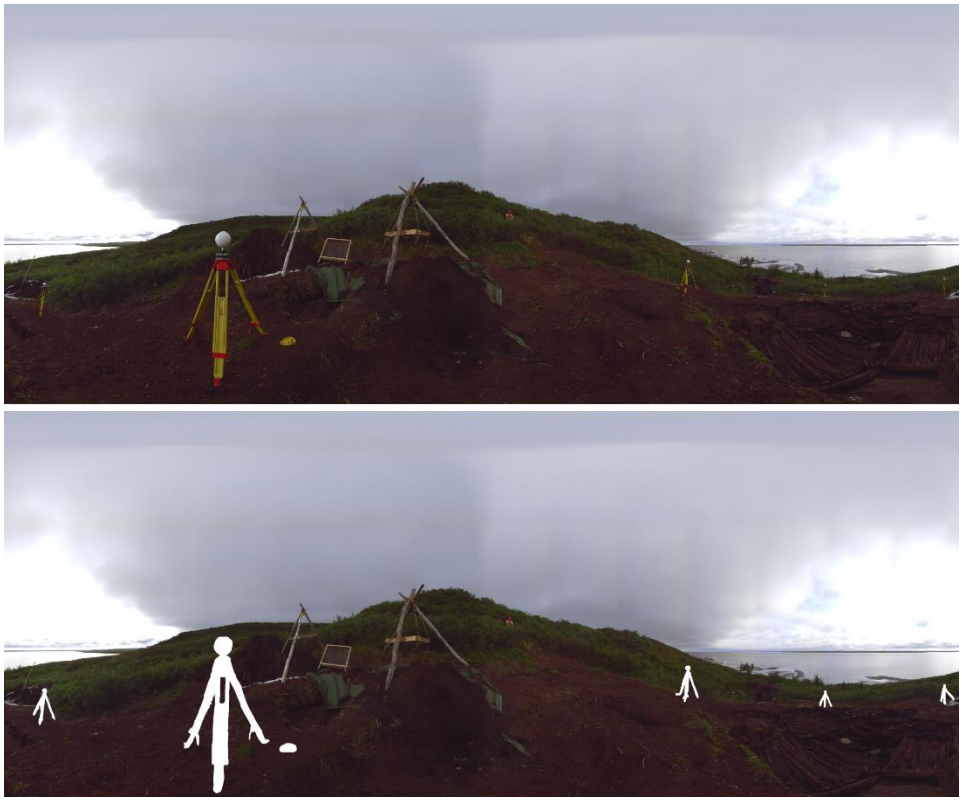

Figure 5. 23 Top: Original panoramic image. Bottom: Image with foreground objects manually removed (i.e. fully transparent)

Another complication is if the model of the subject is an imprecise or inaccurate representation of the actual subject. In these cases, the visual information present in the digital photographs will not correspond to the model correctly, creating distortions in the visual quality. Furthermore, since all the images present will be incorrectly projected onto the model, the textures are less likely to correspond correctly, resulting in a degradation of texture quality. If one image texture has a much higher weight than the other image textures, then this texture will be distorted as it is projected onto the model. On the other hand, if many textures contribute their information, the visual errors likely manifesting as an apparent blur in the details as the different sections of the image textures are blended together incoherently. This can be seen in Figure 5.24, where the images of a large flat stone fail to align, likely due to the boundary of the stone being poorly defined in the mesh model. This blurring is less likely to occur in representations of surfaces with flat surfaces and distinct edges that are well represented by the triangular faces which make up the model.

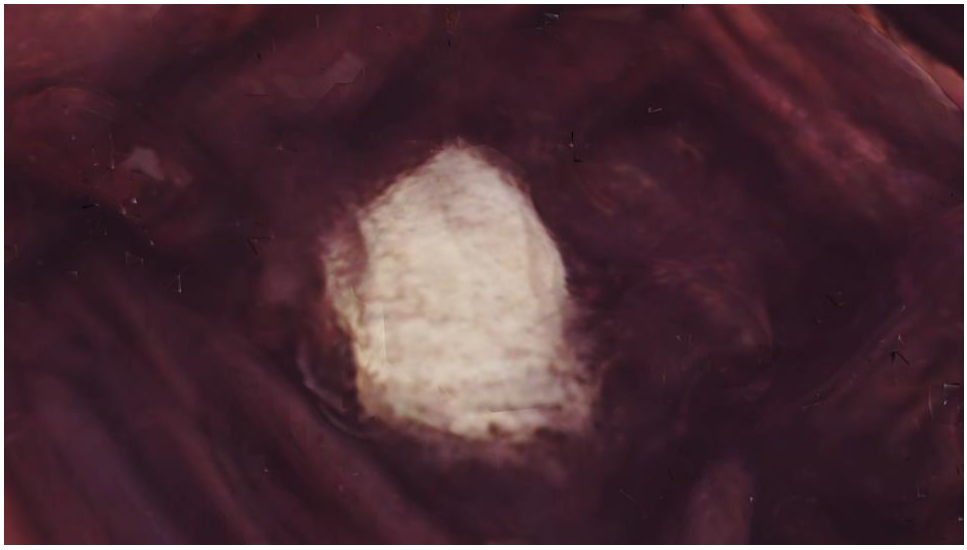

Figure 5. 24 Poor texture results, likely due to imprecise surface modelling

Because the textures being used in these models are digital photographs of actual environments, the textures inherently include light and shadow as part of their visual information.

This is in contrast to how model textures are usually created, which is independent of any lighting or shadow information, artists usually preferring to allow the rendering software and virtual lights to produce bright and dark areas on the model based on the scene being constructed. This can be overcome by setting the model to be 'shadeless,' meaning that it does not interact with the lighting information in the scene, but is instead always visible exactly as the texture defines it to be. This solution works well if the lighting information is intended to be static, but if dynamic lighting and shadows are desired, then the textures will need to be modified to be homogeneously lit, a very challenging task.

Finally, particularly small details, such as grass or small plants, that are not able to be represented in the model due to their fine nature will be projected onto the surface of the model, which can potentially create strange appearances in the texture. This is especially noticeable around edges, such as the one viewed in Figure 5.25, where leaves extend beyond the edge of the model, and are then projected onto the background. To limit this effect, a small Gaussian blur ($\sigma = 1$ pixel) was used on the occlusion map, decreasing the weight of sections of texture that are nearly occluded. This blurred texture was then array-wise multiplied with the original non-blurred occlusion map, so that the black (i.e. occluded areas) remained fully black, but the un-occluded areas close to occluded areas were blurred.

Figure 5. 25 Plant matter in the foreground incorrectly projected on the surface in the background

5.4 Creating Visualizations and Applications

One of the principal benefits of creating a model of the subject that has been fully textured is that it can be viewed from almost any angle or perspective and visual quality remains high. Furthermore, because the model is three-dimensional, it is well suited for visual mediums, or even Virtual Reality (VR) applications. Visual mediums may include pre-rendered video tours of archaeological sites or artifacts, interactive and explorable environments, or forensic style documentation tools for professional exploration and annotation of the models. VR applications are particularly useful as visualization tools because the third dimension is an inherent part of the visualization of the data. This means that scale of the visualization is not relative to the position of the 'camera' relative to the model, but rather an inherent part of how the medium visualizes information. This creates a sense of 'presence' in the user, in part because VR creates the illusion of a 3D space that the user inhabits, but also in that viewing an environment in VR is as simple and intuitive as possible, since motion tracking makes it possible to change perspectives as simply as turning your head.

5.4.1 Visualisation Mediums and Applications

There are many different potential applications of visual models of archaeological subjects. For example, once the model has been prepared, it is a simple task to create an animation showing off the model from many different perspectives, or to pair the animation with an expository voiceover to provide context for what is being displayed in the animation at any given time. Pre-rendered animations also have the benefit of being a well-established medium as an educational resource, and can be converted into many different formats (e.g. digital video files, DVDs, YouTube videos, etc.) for highly adaptable distribution.

If a more interactive medium is desired, it is also possible to create freely explorable environment which features the visual model. In this case, many different options are available for relaying information about the model to the user, whether it be textual or voice-over. This is a more involved process, since if the user is being allowed to explore the environment at their own pace, then the flow of information is uncertain. All information that is presented in the virtual environment must make sense regardless of the order that it is presented in, since the user, not the developer choose which parts of the environment to explore. On the other hand, some user autonomy can be limited, and the information only made available in the order the developer wishes, which could be described more along the lines of a 'virtual guided tour'.

Another potential application of the visual model may be that of a 'virtual survey,' in which the model can be explored, viewed, measured, annotated and augmented by a user. Specific details of interest to archaeologists could be visually highlighted and emphasized, and comments added. It would also be possible to bring several different features together into a single virtual survey. For example, it would be possible for models of small artifacts to be included in the correct relative position to the larger feature that they were discovered within. This juxtaposition could provide

insights not necessarily available when viewing either model on their own. This type of application could certainly be a very useful tool for an archaeologist, but it would be a large undertaking, and would need to be developed in a collaboration between a team of software developers and the archaeologists who wish to make use of the tool.

The tools outlined here are perfectly suited to the typical types of computer human interaction that many of people are already familiar with, specifically with a monitor, mouse and keyboard as interfaces. There is however a rapidly growing new paradigm of computer human interaction, specifically Virtual Reality (VR). With dropping costs of VR headsets and development tools, there is growing interest in the potential new applications that VR makes available.

5.4.2 Simple VR Visualizations

There exist very simple VR setups, such as Google VR which uses a headset fitted with powerful lenses and an Android smart phone, as seen in Figure 5.26. The screen of the smart phone is used to visualize both perspectives of the model simultaneously from two predetermined offset positions, replicating how the human visual system works. The lenses enable the user to focus on the screen of the smart phone even when the screen is only a few centimeters from the user's eyes, and the head tracking is performed via the smart phones internal accelerometers and gyroscopes. All these features come together to create a simple but effective VR setup.

Figure 5. 26 Example of a Google VR headset

Despite the simplicity, very impressive visualization applications are still possible with these devices. Real-time head tracking and reasonably high frame rates contribute to a sense of presence and freedom not available in previous generations of VR, and the technology is available at a fraction of the cost, particularly if the user already owns a compatible smart phone. The simplicity of this type of VR is undoubtedly its strength, but the simplicity also introduces constraints on the design of applications. For instance, without additional input devices, the user only has two options available to interact with the device, a gaze-locked cursor similar to a mouse, and a single input activated when the screen is tapped. (Most Google VR headsets include a button on their external surface so that the screen can be tapped without intruding into the users' field of view.) These limited inputs from the user mean that more work from the designer is necessary to create an application that is simple and intuitive to use. Because of the limited inputs available, it is difficult to allow the user to navigate an environment freely. It is much simpler to allow the user to explore the environment at discrete, predetermined locations, removing some autonomy in favour of simplicity.

This approach has the benefit of simple navigation, but also enables some creative use of panoramic photos. By setting the specified view locations to be equal to the location of the panoramic photos relative to the model, not only are unimaged areas of the model hidden from the user, but the panoramic photos from the perspective of the scanner can be projected onto a skybox (i.e. spherical surface indefinitely large in radius, centered on the observer), enabling the visual context of the model to be presented almost seamlessly with the model itself. This is, of course, not totally seamless, because 3D information only exists for the model in the foreground, creating something of a visual 'cliff' between the model and the skybox. This visual cliff could be concealed through the use of virtual plants placed to conceal the transition, and by extending the ground model arbitrarily far until no change between the model and the skybox were discernable.

5.4.3 Advanced VR Applications

More advanced VR headsets, such as the Oculus Rift or the HTC Vive, improve upon the visual resolution and the motion tracking techniques of the simpler Google VR, and additionally have head straps so that the headset can be worn and the hands kept free. By freeing the hands, they enable the user to use a mouse and keyboard, a video game controller, or even motion tracked handheld controllers. While a mouse and keyboard or game controller certainly offer many more input options, handheld motion tracked controllers can act as surrogate hands in digital environments, and are commonly reported as being surprisingly intuitive to master, further enhancing the level of immersion in the digital environment.

In addition, it is possible to create a physical space in which the user can walk around, and have that same motion represented in the virtual environment, if they remain within the view of the motion tracking sensors. This further increases the levels of intuition, immersion and presence

that the simulation can offer, meaning that there is essentially zero barrier for a user to fully enjoy and interact with these virtual environments, if the virtual environment is well designed.

This level of intuitiveness and immersion could be enormously beneficial to archaeologists who wish to create simulations of environments or events for the purposes of education and outreach. As an example, one can imagine creating a simulation of a beluga whale hunt, putting the user in the role of a hunter. The actions of rowing, throwing harpoons, bringing in a catch, butchering, and others can all be simulated in VR putting the user in the center of these events. Furthermore, all tools, environments, animals, clothing and structures can be replicated as closely as possible to fact, using traditional knowledge of indigenous communities, evidence and artifacts uncovered by archaeologists, and 3D scanning and modeling technologies. Additionally, the traditional language, music, customs and mythology of the indigenous peoples can be represented faithfully and in context, providing an emotional and intellectual framework for the user to understand these in a way not easily done otherwise. This type of large scale simulation would need to be created through a collaboration of archaeologists, VR simulation designers, engineers and, importantly, indigenous communities.

5.5 Results and Discussion

5.5.1 Methodology

A visual model of the excavation at Kuukpak was created from the 10 TLS scan captured using the Faro Focus 3D on August 2nd 2014. The scans included the point clouds, as well as the panoramic photographs created by the device for the purposes of adding colour to the point cloud. The scans were registered in Faro Scene, using 5 spherical targets each 6 inches in diameter. The scans were located around the excavated feature in approximately equal increments, to provide

good coverage of the feature from many angles. The scans were then exported in a PTX file format for use in other programs. Also exported was the registration information for each scan, specifically the location of the scan and the rotation angle about the axis of rotation. The scans were known to be level, and the axis of rotation for the registration was, in every case, very close to perfectly vertical, so to simplify subsequent calculations the axis of rotation was assumed to be the vertical axis.

The PTX files were then imported into *MeshLab* (version 1.3.3, 64 bit). At first, the full point clouds were used. Faces with an angle of incidence less than 60° relative to the scanner were kept for each scan, and all scan compressed into a single layer, discarding 'unreferenced points' (i.e. points in the scans which did not belong to a triangular face). This produced a model which contained 25 million faces and 18 million vertices, but with highly varying levels of detail across the model. This model was reduced to the areas of interest, and the Poisson Surface Reconstruction algorithm was performed, at an octree depth of 12. The results were a highly detailed geometric model of the excavation, with 16 million faces, and can be seen in Figure 5.27. While this model was of very high quality, the memory requirements to interact with and edit this model were beyond those available. While there exist techniques to maintain the graphical quality of the model while reducing the total number of triangular faces used to represent the model (i.e. normal mapping and surface decimation), they were beyond the scope of this thesis. This high quality geometric model was instead a representation of the type of high detail model that is potentially available from TLS or photogrammetric data, even though this model was not used to create a visual model.

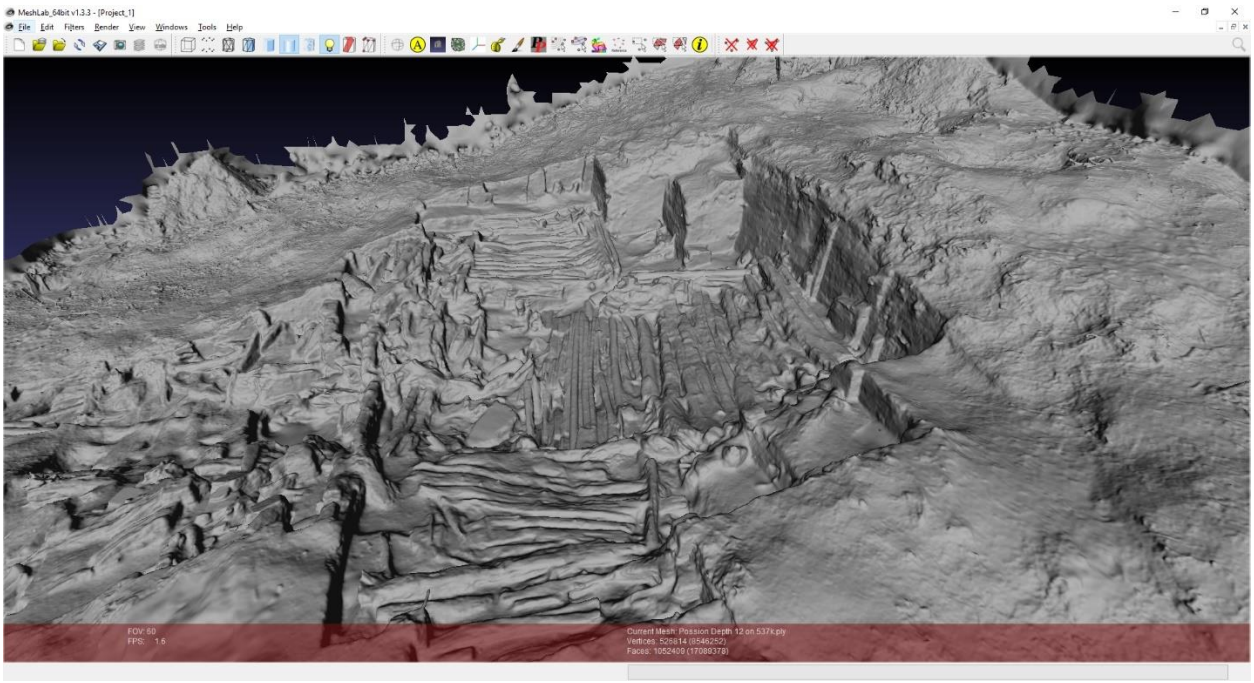

Figure 5. 27 View of a high-resolution model of Kuukpak (16 million faces)

Rather than attempting to create a visual model from the full point clouds, another model was created, using a sample of points from each cloud using the "Poisson-disk Sampling [25]" algorithm available in *Meshlab*. The number of desired samples was set to be equal to the number of points in each cloud, but the number of points output by the sampling procedure was always significantly less. Due to the properties of the Poisson Disk Sampling, the point density across the extents of each cloud was significantly more homogenous. Once every cloud was imported and sampled, they were collapsed into a single layer. Once collected into a single layer, the remaining point cloud had approximately 670,000 points, a reduction in the number of vertexes from the full resolution by a factor of approximately 25.

Alternatively, the Poisson-disk Sampling could have been performed on the high-resolution model directly, which would have produced an oriented point cloud, similar to the one produced by sampling and combining the PTX point clouds. This method allows more control over the number of points used to create the reduced density mesh, but instead requires the high-

resolution mesh to be produced first, which takes additional processing time. If similar numbers of samples are used, then the two methods produce nearly identical results.

Not all the point in this cloud were useful for the purposes of creating the visual model, however. Some of the points were sparsely distributed across vegetation, and much of the cloud was of complicated topology that did not serve to improve the quality of the visual model. These points were removed from the cloud, resulting in 537,700 points remaining.

From there, the Poisson Surface Reconstruction algorithm was performed, with an octree depth of 12, with otherwise default parameters. The result was a surface with 1.1 million faces, but many of which were superfluous, extending well beyond the extents of the point cloud use to create the surface. This was easily remedied, however, because the extra faces nearly universally had very long edge lengths, so they could easily be filtered out. In this case, any face with an edge length longer than 20 cm was deleted. Once the majority of the spurious faces had been deleted, the largest continuous set of faces was selected, and the selection inverted to select and remove any spurious faces with edge lengths smaller than 20 cm. The resulting mesh surface had 1.05 million faces and can be seen in Figure 5.28. It was then exported from *Meshlab*.

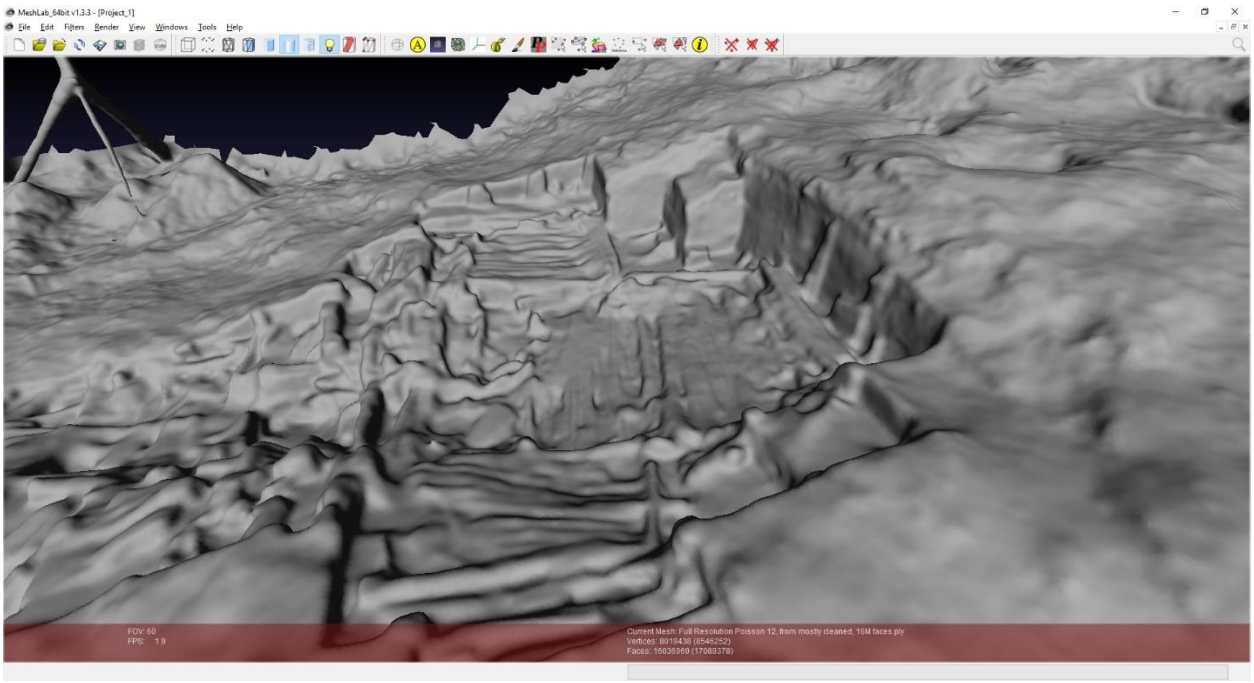

Figure 5. 28 View of a medium resolution model of Kuukpak (1 million faces)

By default, the images taken by the Faro were not a full panoramic image, since the lowest 30° were cut off. For the formulations used in this thesis, the panoramic images must be the full 360° by 180° extent, so the images were padded in size with transparent pixels. The original size of the images are 8168 by 3414 pixels, but for the vertical size to be exactly half of the horizontal size (as would be expected from a full panoramic image) the images were filled with transparent rows of pixels at the bottom until they reach the size of 8168 by 4084 pixels. Furthermore, any un-modeled foreground objects in the images, specifically the scan targets and tripods, were manually erased from the images as well. These sections were set to be fully transparent, so that they would not interfere with the process used to create the Master texture of the model.

Once the images had been edited to the appropriate size and content, they and the mesh model were loaded into *Blender* (version 2.76b). It is very important that none of the point clouds nor mesh models of the feature be modified in position or orientation between the registration procedure performed in Faro Scene and the texture creation performed in *Blender*. This is because

to correctly project the images on the model, the registered scan locations are used for the coordinates of the center of the images. If the model had been moved from its initial position, then the registered scan locations are no longer accurate, and the texture creation process would fail. Fortunately, this process uses the "mesh" coordinates (i.e. the location of the mesh relative to its parent object) for calculation, meaning that the "object" coordinates (i.e. the location of the parent object within the scene) can be edited freely. The custom program "createOptimalTexture.py" as seen in Appendix C, was run in *Blender*'s console, and a single optimal texture for the selected mesh was created using the loaded images and scan locations. The resulting visual model can be seen in Figure 5.29.

Figure 5. 29 Aerial view of final, textured model of Kuukpak (1 million faces)

Once the texture was finished, the mesh and texture together can be exported as OBJ files, and can be loaded into many other applications to make use of them, or can be further used within *Blender*. In this case, they were imported into Unity (version 5.4) for the purposes of creating a

3D VR visualization application, and can be viewed in Figure 5.30. The Google VR API is available for free online and is designed as a plugin for Unity, making development fast and simple.

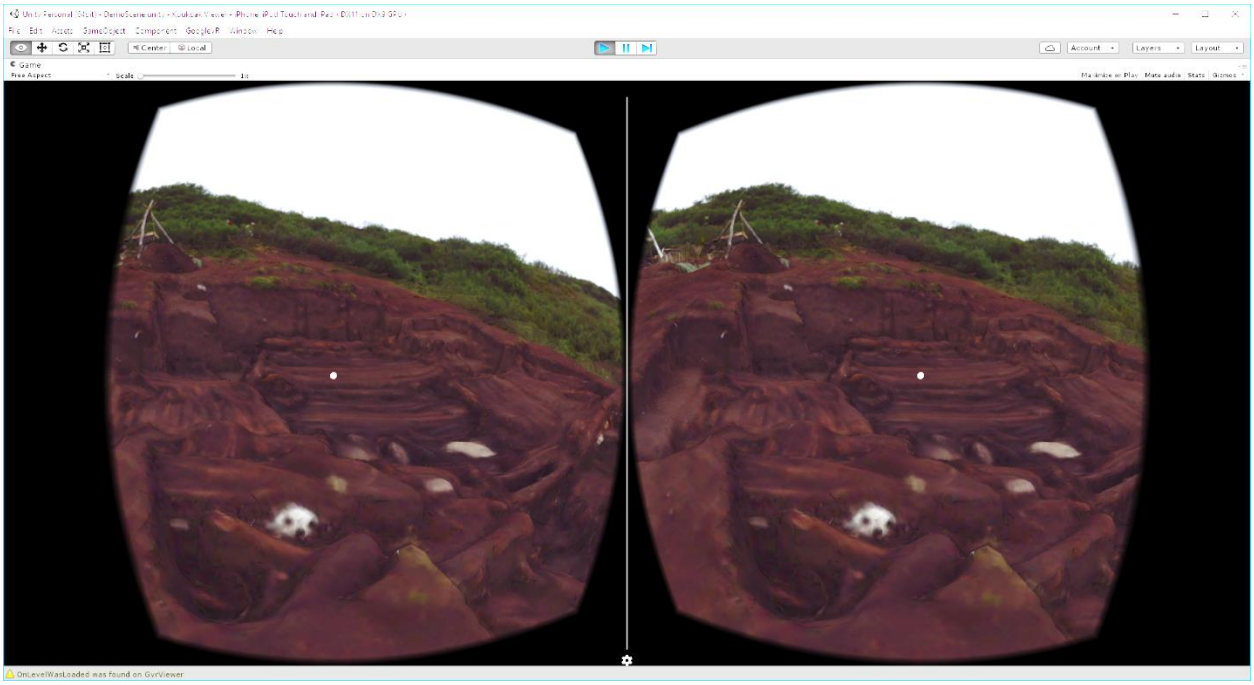

Figure 5. 30 Demo of the Kuukpak Viewer VR App in Unity 5.4

5.5.2 Discussion

In this application, the user's perspective is set to be equal to the scan locations, so that any areas on the mesh not directly observed by the scanner are hidden from view. These positions are also useful, in that if the panoramic images are applied to a skybox, the foreground lines up nicely with the background from these perspectives. Tapping the screen sets the user's perspective in the next position in a loop around the feature, eventually arriving back at the initial position.

This application, despite the simplicity, does a wonderful job of letting the user explore the virtual environment of Kuukpak. There are limitations, however. The model produced by this methodology has approximately 1 million triangular faces, which is significantly reduced from the original 16 million faces available in the high-resolution model. However, 1 million faces, compared the models used in many 3D graphic applications, is an extremely large number of faces

to describe any object. In most applications, the number of faces used to describe the features is reduced, and other techniques used to preserve and improve the visual quality instead. For example, one such technique is to use “normal maps,” a method which changes the way light reflects off the surface of a model, providing the appearance of surface details without adding faces to the model. This allows for the geometry of the model to be simplified (i.e. the number of triangular faces reduced), but for the detailed appearance of the shape to be preserved. However, this technique relies on the interaction of the model and the virtual lighting present in the scene. As mentioned in section 5.3.6, the model is set to be ‘shadeless’, meaning that the model itself does not interact with the virtual lighting. Therefore, the inclusion of a “normal map” would have no effect. In order for the model to interact with lighting and to maintain high quality visual detail, then the texture would need to be modified to reverse the effects that the natural lighting had on the panoramic photographs of the site. This is a very complex task, and is further complicated by the lack of a clear definition of the lighting sources in the original panoramic photographs, due to the overcast lighting.

Another issue is that the Master texture used to colour the model is 4096 by 4096 pixels. This is, by the standards of many graphics applications, very high definition. However, due to the size of the model and the potential visual information available in the panoramic photographs, significantly more visual information is available should a larger texture be used. This, however, puts significant strain on graphics resources, especially in the case of smart phones which have comparatively very weak GPUs. It also amplifies the processing times, since texture baking times are highly correlated with the texture size. This is especially an issue with the geometric weight texture, since it is not a highly optimized process.

There are some areas of the model which do not have direct colour information available from the panoramic photographs. Currently, these areas appear on the model to be black patches, which stand out considerably as can be seen in Figure 5.31. While these sections do not have accurate colour information, it may be possible to conceal the black patches in another way. When the Master texture is being created, it would be possible to keep track of the areas which are remain occluded after each occlusion map is created. After every occlusion map is created, a final map detailing which areas of the master texture remain occluded can be produced, and this map can be used to create the filler texture to conceal the occluded areas, but this was not implemented.

Figure 5. 31 Examples of totally occluded areas within the Master Texture

Finally, it is noticeable that not every texture has the same visual properties (specifically the brightness or colour saturation), likely due to small changes in the lighting between when panoramic photographs were taken. This is especially visible directly around the blank areas beneath the scan locations. The areas directly outside the circle are heavily weighted towards the panoramic photograph centered above the circle, while the areas inside that circle has precisely

zero weight from that panoramic photograph. This creates a noticeable change in the quality of the texture around these circles, as seen in Figure 5.32. This might be remedied in a couple of ways. The first is that a larger Gaussian blur can be applied to the occlusion maps to ensure a smoother transition between areas of data and no data from an image, but this also has the adverse property that it discounts some areas of high visual detail in each panoramic photo. Instead, the panoramic photos could be pre-processed to have similar levels of brightness and contrast, so that the transitions between photos are less noticeable, but this was not implemented.

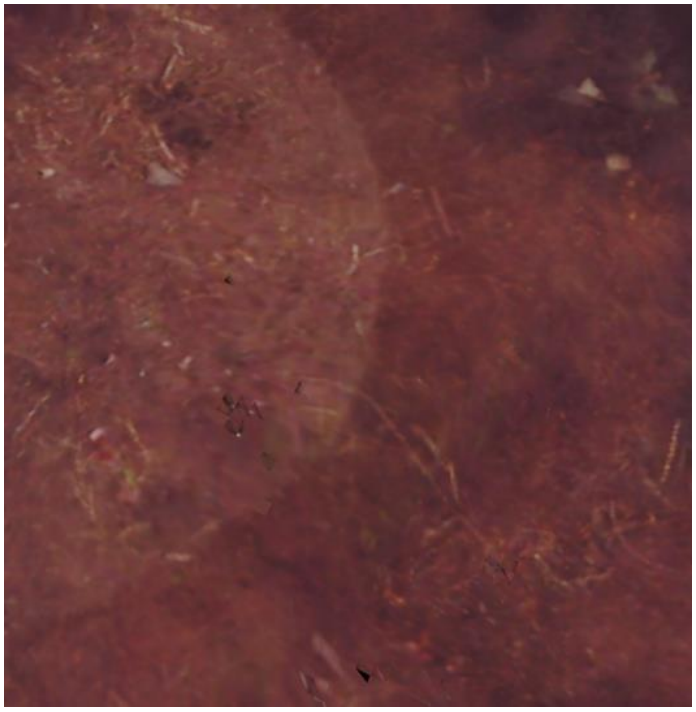

Figure 5. 32 Noticeable transition between different textures directly beneath panoramic image

Chapter 6 – Conclusions and Future Work

6.1 Conclusions

This thesis is centered around the technologies and techniques that an archaeologist may be concerned with when they are considering taking advantage of 3D documentation in their field. The factors to consider that are discussed here are selecting the right devices for a given application, determining the quality of them measurements of a device, comparing the precision between devices, as well as different modeling and visualization techniques available.

6.1.1 DPI-7 Error Models and Calibration

In regards to the calibration of the DPI-7 scanner, the device was tested and an error model was constructed, a calibration method was presented and tested, and an automated method for extracting the spherical targets from the scan data was tested as well. The error model used a single scale factor which is applied to the X and Y coordinates cloud (relative to the imaging array) of the DPI-7 point. This scale factor helped to reduce the RMSE in the X and Y directions by up to 59%. If the systematic errors in the device were unaccounted for, then the virtual representation of the feature being documented would be unreliable. This is particularly problematic if the scale factor in the point cloud is not homogenous in the depth direction. This would imply that the shape of the data from a measurement orthogonal to the surface would differ from the shape of the data oblique to the surface. This inconsistency makes cloud to cloud registrations imprecise, and models made from such measurements highly error prone. There is a lack of evidence, either for or against a scale factor error in the depth direction. Should one be discovered, and if it were different than the one discovered in the X and Y directions, then this would be a significant source of errors in

the device. It is possible that the issues present in the DPI-7's data has been addressed in a software update, or in a newer model of the device.

6.1.2 Simulation of the precision of Photogrammetric and TLS data

The metric performance of close-range photogrammetry and terrestrial laser scanning was compared in a simulation. It was found that the expected errors present in TLS were lower than the expected errors present in the computer vision assisted photogrammetry by a factor of around 40% in the case where the photogrammetry uses the inner constraints datum definition, and by 78% when using a fixed camera datum definition. While the relative differences of the errors are large, the absolute differences are rather small, (6mm, 10mm and 27mm, respectively) especially since all the different methods are of acceptable levels of accuracy for most applications within archaeology, such as visual modeling. Certain applications, such as deformation analysis, require very high levels of precision, and so the for those applications the more precise TLS should be used in addition to geo-referencing techniques.

6.1.3 Visual Modeling

The visual model produced from the TLS data collected at Kuukpak was used to create several different visual models. One visual model, constructed through the maximum density point clouds and a Poisson surface reconstruction at octree depth 12, had a great deal of visual information inherent in the shape of the mesh. It demonstrated precision in areas of large amount of curvature, as well as fine levels of detail, such as the texture of the soil surrounding the feature. A simpler model with fewer triangular faces was then used to produce a coloured visual model. The use of fewer faces significantly reduces the processing time and improves the performance when interacting with or editing the mesh. This visual model was added to a VR environment,

taking advantage of the Google Cardboard asset for Unity, so that it could be freely viewed and explored at the user's pace.

6.1.4 Guidelines for Selecting 3D Documentation Technologies

This thesis has explored many different factors that affect the quality of data that can be expected from both TLS and photogrammetry. These range from network definitions, error propagations, calibration procedures, amongst others. Taking these factors into consideration, some recommendations on which technology is an appropriate choice can be offered.

The first consideration when selecting a 3D documentation technology is "What are the goals for the data?" or equivalently, "What is the application?" This is the first question to answer because it significantly impacts all other considerations. The requirements of the data are different for the creation of a visual model than they are for a deformation analysis, and so the choice of technology should account for that difference.

Another very important factor in deciding which technology is the right choice for a given application is "What is the desired spatial resolution?" or "Over what area will the data be collected?" These two questions are related, but distinct. It is generally the case that data density and the scope of the data are inversely related, since the memory requirements to store the data at a constant point density increase with the volume of the space in which the data exists. Therefore, limiting the area over which data will be collected will significantly help to reduce the memory requirements. Additionally, situations where very large scope and very small point resolution are uncommon, meaning that scope and point density are usually closely related. Table 6.1 gives a quick reference for choosing a technology based on the scope and resolution of the data.

Table 6.1 Recommended Technologies based on scope and resolution

Approximate length of side	Approximate point spacing	Appropriate technology
1 Km	1 m	Aerial photogrammetry, aerial LiDAR, satellite altimetry
100 m	10 cm	Aerial photogrammetry, TLS
10 m	1 cm	TLS, close-range photogrammetry
1 m	>1 mm	Close-range photogrammetry, triangulation laser scanning

In the case where the choice is between photogrammetry or TLS, several factors must be considered. First, and foremost on may people's minds are the financial costs associated with using either technology. In this case, photogrammetry offers the most cost effective solutions, since many archaeologists already possess or can easily acquire a DSLR camera suitable for close-range photogrammetry. The only other requirements are software, much of which is available for free to the user, and computers, which are also ubiquitous. TLS, on the other hand, require specialized equipment and often expensive software licences. Photogrammetry is also preferable in that very little additional equipment is needed, which keeps shipping costs down.

There are, however, instances where photogrammetry cannot be used, or can only be used with great care. For instance, if the ambient light in the subject's environment is unstable, then automatic tie point matching procedures will not function correctly, resulting in a sparse or biased point cloud. Similarly, if the subject's surface has insufficient visual features, such as fresh clean snow or flat matte walls, then no or few tie points can be extracted, causing similar problems. On the other hand, TLS will work very well for these structures, since no tie points need to extracted, and LiDAR measurements are highly resistant to changes in ambient lighting. Finally, since photogrammetry relies on large separations between camera locations to extract 3D information,

if no large baselines can be used, then the 3D information is highly error prone. On the other hand, because TLS measures the 3D information directly, this is not such an issue.

Another consideration when selecting between TLS or photogrammetry is the contextual information that can be collected by both techniques. Photogrammetry can only measure 3D data in locations that have been imaged by two or more photographs. On the other hand, TLS captures data panoramically, meaning that the 3D data of the surrounding area is also collected, providing additional contextual information.

Finally, the difference in precision between these two technologies should be considered. While TLS has been shown in chapter 4 to be more precise than photogrammetry, the difference is not drastic. In good conditions with well defined networks and for suitable subjects, photogrammetry and TLS are very comparable in precision. Therefore, unless the application requires the absolute highest precision available, or if the subject or environment are unsuitable, then photogrammetry is the recommended solution for 3D documentation in archaeology.

6.2 Future Work

6.2.1 DPI-7 Error Models and Calibration

It may be possible to significantly improve upon the quality of the DPI-7 calibration if the original, unmodified observations are available from the device. However, after investigation, it was determined that although the original image point measurements could nearly be extracted from the point cloud data, there were distortions present in the precise locations of the rows and columns of the measurements. This indicates that the point clouds were modified from their original rows and column of pixels. It was noted that a calibration file was used during data collection of the DPI-7 scanner. Some data was collected without that calibration file active so

determine its effects on the data, and it was found that the exclusion increased the scale factor error. It may be the case that by excluding this calibration file, the original image point observations could have been extracted from the point cloud data, and a more rigorous calibration be created as a result. This hypothesis was not tested, because to do so the data collection would need to be repeated, which was not possible due to time constraints.

Additionally, the scale factor in the depth direction of the DPI-7 scanner could have been resolved if the target field used during the calibration was not planar, and had many different targets and varying heights above the flat surface the targets were mounted on. This was not done for two principal reasons. The first was that type of target field would have been more difficult to create and transport while maintaining its correct shape and the relative position of the targets. Secondly, the addition of varying depth to the target field was not considered necessary at the time of data collection because of the use of convergent images of the target field. This would have provided variation in the depth, if the device did not automatically align the point clouds with the devices' internal measurement of the direction of gravity. The fact that the device modifies the point cloud data in this way was not discovered until long after the data was collected, at which point time constraints prevented the collection of a new set of calibration data.

6.2.2 Simulation of the precision of Photogrammetric and TLS data

There is potential for improvement in the simulation of photogrammetric observation data in this thesis as well. Currently the location of photogrammetric tie points are spaced in a regular grid across the surfaces of the triangular planar sections which define the simplified model of the site. Additionally, the matching of the tie point observation between images is done in a simple way, which has low redundancy but good intersection geometry. Theses methods are appropriate given the constraints of the MATLAB development environment that the simulation was

programmed in, but can be significantly improved upon by using visual models within *Blender*, (or a similar 3D modeling software) instead. The strength that *Blender* has in this simulation is that the complex geometry and visual information present in the 3D model is already present, and that rendered images of the visual model can be used to create digital images suitable for photogrammetry, using virtual cameras with any properties and in any position one chooses. These rendered images can then be supplied as inputs to actual photogrammetry software, such that no assumptions or simplifications need to be made at all. In fact, because the precise shape of the visual model is already known, the estimated shape output from the photogrammetry software can be directly compared to it, giving not just the estimated precision, but the absolute accuracy of the method as well.

This type of software verification through the use of visual models has a large number of potential applications. For instance, the same rendered images can be input to two different photogrammetry software bundles, and both be compared to the original visual model in order to compare and contrast the quality of their outputs. Additionally, the impact that non-ideal material properties can also be measured using this method. For instance, within *Blender* there exist many different materials that can be applied to objects, such as sub-surface scattering effects (absorption and re-emitting of light, visible in materials such as warm butter or marble), semi-transparent effects (such as in glass or water) or strand effects (such as grass or hair). These effects can pose potential problems for photogrammetry software, particularly in the tie point extraction and matching steps. The use of 3D modeling and digital renders offer a simple, cost-effective, and accurate method of testing how these material properties impact the quality of photogrammetry algorithms.

Similar to photogrammetry, the simulation of TLS data can also be performed within *Blender* on a visual model to take advantage of the more detailed geometry, but the simulation of the observations would largely be the same. One potential improvement, however, would be to simulate the effects of mixed range measurements, which would be more prominent in the more complex environment of the visual model. This can be done by simply adding additional range measurements in several angular offsets from the center of the simulated laser, and setting the observed range to be equal to the average of those ranges. This also has the effect that the standard deviation of the range can be well estimated by the standard deviation of those range samples, since the equation is used to affect the singular range measurement produces results similar to the effect that the angle of incidence has on the angularly offset range measurements.

6.2.3 Visual Modeling

The visual models produced in this thesis are of good quality, but certainly can be improved upon. As mentioned previously, there are some locations on the visual model which do not have colour information, since they were not directly imaged by any camera during the laser scanning process, and it would be good to conceal these areas in some way. Additionally, the geometry of the visual model is very complex, so it would be good to reduce that complexity, and to maintain the visual quality of the model through advanced texturing techniques. Also regarding the geometry of the model, the digital images would more precisely project onto the surface of the model if the more precise and complex model were used, which would help to reduce the number of visual errors on the model, but doing this would also have a severe impact on the speed that the texturing process can be performed at, to the point that it may not be possible given memory constraints of personal computers. Ideally, an algorithm could be designed to combine the mesh simplification, the image texturing and the other advanced texturing processes, to create a high

quality mesh with a low polygon count and several different types of textures, such that the visual quality is maintained but data simplified.

The VR visualization of the visual models presented in this thesis is a rather simple demonstration of the type of applications possible using 3D spatial data. It would be further advantageous for the visual model to be used to create a more interactive and immersive experience, perhaps using more advanced VR head sets. This type of application would be significantly more work, and would have likely fallen outside of the scope of this thesis.

References

[1] A. Appel, “Some techniques for shading machine renderings of solids,” *Proc. April 30–May 2, 1968, spring Jt. Comput. Conf.*, pp. 37–45, 1968.

[2] R. Whallon, “The computer in archaeology: A critical survey,” *Comput. Hum.*, vol. 7, no. 1, pp. 29–45, 1972.

[3] J. F. Kenefick, M. S. Gyer, and B. F. Harp, “Analytical Self-Calibration,” *Photogramm. Eng. Remote Sensing*, vol. 38, no. 11, pp. 1117–1126, 1972.

[4] E. M. Mikhail and F. Ackermann, *Observations and Least Squares*. 1976.

[5] S. D. Roth, “Ray casting for modelling solids,” *Comput. Graph. Image Process.*, vol. 18, no. 2, pp. 109–144, 1982.

[6] B. K. P. Horn, “Closed-form solution of absolute orientation using unit quaternions,” *J. Opt. Soc. Am.*, vol. 4, no. 4, p. 629, Apr. 1987.

[7] C. D. Arnold and E. Hart, “The Mackenzie Inuit Winter House,” *Arctic*, vol. 45, no. 2, pp. 199–200, 1992.

[8] M. Hebert and E. Krotkov, “3D measurements from imaging laser radars: how good are they?,” *Image Vis. Comput.*, vol. 10, no. 3, pp. 170–178, 1992.

[9] D. D. Lichti and M. A. Chapman, “Constrained FEM Self-Calibration,” *Am. Soc. Photogramm. Remote Sens.*, vol. 63, no. 9, pp. 1111–1119, 1997.

[10] D. Morrison, “The Inuvialuit of the Western Actic: From Ancient Times to 1902,” *Canadian Museum of History*, 1997. [Online]. Available: http://www.historymuseum.ca/cmc/exhibitions/aborig/inuvial/villagee.shtml.

[11] C. S. Fraser, “Digital camera self-calibration,” *ISPRS J. Photogramm. Remote Sens.*, vol. 52, no. 4, pp. 149–159, Aug. 1997.

[12] M. Levoy *et al.*, “The digital Michelangelo project: 3D scanning of large statues,” *SIGGRAPH*, pp. 131–144, 2000.

[13] W. Boehler and A. Marbs, “Investigating Laser Scanner Accuracy,” *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 34, pp. 696–701, 2003.

[14] F. Gielsdorf, A. Rietdorf, and L. Gruendig, “A Concept for the Calibration of Terrestrial Laser Scanners,” in *Proceedings of the FIG Working Week. Athens, Greec*, 2004, pp. 1–10.

[15] D. D. Lichti, J. Gordon, and T. Tipdecho, “Error Models and Propagation in Directly Georeferenced Terrestrial Laser Scanner Networks,” *J. Surv. Eng.*, vol. 131, no. November, pp. 135–142, 2005.

[16] M. Kazhdan, M. Bolitho, and H. Hoppe, “Poisson Surface Reconstruction,” *Eurographics Symp. Geom. Process.*, 2006.

[17] J. C. Mars and D. W. Houseknecht, “Quantitative remote sensing study indicates doubling of coastal erosion rate in past 50 yr along a segment of the Arctic coast of Alaska,” *Geology*, vol. 35, no. 7, pp. 583–586, 2007.

[18] D. D. Lichti, “Error modelling, calibration and analysis of an AM–CW terrestrial laser scanner system,” *ISPRS J. Photogramm. Remote Sens.*, vol. 61, no. 5, pp. 307–324, Jan. 2007.

[19] H. Bendea, F. Chiabrando, F. G. Tonolo, and D. Marenchino, “Mapping of archaeological areas using a low-cost UAV. The A Ugusta Bagiennorum test site,” *XXI Int. CIPA Symp.*, vol. XXXVII-5/W, no. October, pp. 117–122, 2007.

[20] P. C. Dawson, R. M. Levy, and G. Mackay, “Documenting Mackenzie Inuit architecture using 3D laser scanning,” *Alaska J. Anthropol.*, vol. 7, no. 2, pp. 29–44, 2009.

[21] S. Sylaiou, K. Mania, A. Karoulis, and M. White, “Exploring the relationship between presence and enjoyment in a virtual museum,” *Int. J. Hum. Comput. Stud.*, vol. 68, no. 5, pp. 243–253, 2009.

[22] L. Barazzetti, F. Remondino, and M. Scaioni, “Combined use of Photogrammetric and Computer Vision techniques for fully automated and accurate 3D modeling of terrestrial objects,” in *Videometrics*, 2009.

[23] P. Moulon and A. Bezzi, “Python Photogrammetry Toolbox : A free solution for Three-Dimensional Documentation,” 2010.

[24] J. Demantke, C. Mallet, N. David, and B. Vallet, “DIMENSIONALITY BASED SCALE SELECTION IN 3D LIDAR POINT CLOUDS,” *ISPRS Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. Commission, no. WG III/2, 2011.

[25] S. Soudarissanane, R. Lindenbergh, M. Menenti, and P. Teunissen, “Scanning Geometry: Influencing factor on the quality of terrestrial laser scanning points,” *ISPRS J. Photogramm. Remote Sens.*, vol. 66, no. 4, pp. 389–399, 2011.

[26] P. C. Dawson, R. M. Levy, and N. Lyons, “‘Breaking the fourth wall’: 3D virtual worlds as tools for knowledge repatriation in archaeology,” *J. Soc. Archaeol.*, vol. 11, no. 3, pp. 387–402, 2011.

[27] F. Remondino, “Heritage Recording and 3D Modeling with Photogrammetry and 3D Scanning,” *Remote Sens.*, vol. 3, no. 12, pp. 1104–1138, 2011.

[28] J. C. K. Chow, D. D. Lichti, W. F. Teskey, and C. Key, “Accuracy assessment of the FARO Focus 3D and Leica HDS6100 panoramic- type terrestrial laser scanners through point-based and plane-based user self-calibration,” *FIG Work. Week*, no. May, pp. 6–10, 2012.

[29] M. Corsini, P. Cignoni, and R. Scopigno, “Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes,” *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 6, pp. 914–924, 2012.

[30] P. C. Dawson, M. M. Bertulli, R. M. Levy, C. Tucker, L. Dick, and P. Cousins, “Application of 3D Laser Scanning to the Preservation of Fort Conger , a Historic Polar Research Base on Northern Ellesmere Island , Arctic Canada,” *Arctic*, vol. 66, no. 2, pp. 1–12, 2013.

[31] FARO Technologies Inc, “FARO Focus3D – Features, Benefits & Technical Specifications,” *FARO Technologies Inc*, 2013. [Online]. Available: http://www.faro.com/en-us/products/3d-surveying/faro-focus3d/downloads-us#Download. [Accessed: 01-Aug-2016].

[32] J. C. K. Chow and D. D. Lichti, “Photogrammetric Bundle Adjustment With Self-Calibration of the PrimeSense 3D Camera Technology: Microsoft Kinect,” *IEEE Access*, vol. 1, pp. 465–474, 2013.

[33] T. Küneth, “The New Plastic Balls,” *Iternational Table Tenis Feredartion*, 2013. [Online]. Available: http://www.fftt.com/doc/. [Accessed: 20-Jun-2008].

[34] DotProducts LLC, “DPI-7 Kit,” *DotProducts LLC*, 2014. [Online]. Available: www.dotproduct3d.com/assets/pdf/dotproduct-brochure_WEB.pdf. [Accessed: 20-Jun-2008].

[35] E. Nocerino, F. Menna, and F. Remondino, “Accuracy of typical photogrammetric networks in cultural heritage 3D modeling projects,” *ISPRS Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. XL-5, no. June, pp. 465–472, Jun. 2014.

[36] J. De Reu, P. De Smedt, D. Herremans, M. Van Meirvenne, P. Laloo, and W. De Clercq, “On introducing an image-based 3D reconstruction method in archaeological excavation practice,” *J. Archaeol. Sci.*, vol. 41, pp. 251–262, 2014.

[37] A.-I. García-Moreno, D.-E. Hernandez-García, J.-J. Gonzalez-Barbosa, A. Ramírez-Pedraza, J. B. Hurtado-Ramos, and F.-J. Ornelas-Rodriguez, “Error propagation and uncertainty analysis between 3D laser scanner and camera,” *Rob. Auton. Syst.*, vol. 62, no. 6, pp. 782–793, Jun. 2014.

[38] J. Fameli, “Tested In-Depth: Structure Sensor 3D Scanner,” *Tested.com*, 2014. [Online]. Available: http://www.tested.com/tech/3d-printing/472702-tested-depth-structure-sensor-3d-scanner/. [Accessed: 01-Aug-2016].

[39] A. Bustillo, M. Alaguero, I. Miguel, J. M. Saiz, and L. S. Iglesias, “A flexible platform for the creation of 3D semi-immersive environments to teach Cultural Heritage,” *Digit. Appl. Archaeol. Cult. Herit.*, vol. 2, no. 4, pp. 248–259, 2015.

[40] K. Lymer, “Image processing and visualisation of rock art laser scans from Loups’s Hill, County Durham,” *Digit. Appl. Archaeol. Cult. Herit.*, vol. 2, no. 2–3, pp. 155–165, 2015.

[41] H. Plisson and L. V. Zotkina, “From 2D to 3D at macro- and microscopic scale in rock art studies,” *Digit. Appl. Archaeol. Cult. Herit.*, vol. 2, no. 2–3, pp. 102–119, 2015.

[42] L. Grosman, “Reaching the Point of No Return: The Computational Revolution in Archaeology,” *Annu. Rev. Anthropol.*, vol. 45, no. 1, p. annurev-anthro-102215-095946, 2016.

[43] Blender Foundation, “Blender Documentation,” *Blender Foundation*, 2016. [Online]. Available: https://www.blender.org/api/blender_python_api_2_77_release/. [Accessed: 20-Jun-2008].

[44] T. Luhmann, C. Fraser, and H. G. Maas, “Sensor modelling and camera calibration for close-range photogrammetry,” *ISPRS J. Photogramm. Remote Sens.*, vol. 115, pp. 37–46, 2016.

[45] J. Fernandez-Lozano and G. Gutierrez-Alonso, “Improving archaeological prospection using localized UAVs assisted photogrammetry: An example from the Roman Gold District of the Eria River Valley (NW Spain),” *J. Archaeol. Sci.*, vol. 5, pp. 509–520, 2016.

[46] C. Haukaas and L. M. Hodgetts, “The Untapped Potential of Low-Cost Photogrammetry in Community-Based Archaeology: A Case Study From Banks Island, Arctic Canada,” *J. Community Archaeol. Herit.*, vol. 3, no. 1, pp. 40–56, 2016.

[47] C. Arnold, “Development of Mackenzie Inuit Culture,” in *The Oxford Handbook of the Prehistoric Arctic*, M. Friessen and O. Mason, Eds. 2016.

[48] Helicon Soft Ltd., “Helicon Soft,” 2017. [Online]. Available: http://www.heliconsoft.com/heliconsoft-products/helicon-focus/. [Accessed: 24-Jan-2017].

Appendix A: Partial Derivatives for Design Matrix of TLS Registration

The Design matrix of rigid body registration, **A**, is defined as

$$\mathbf{A} = \begin{bmatrix} \frac{\partial X}{\partial t_x} & \frac{\partial X}{\partial t_y} & \frac{\partial X}{\partial t_z} & \frac{\partial X}{\partial \omega} & \frac{\partial X}{\partial \phi} & \frac{\partial X}{\partial \kappa} \\ \frac{\partial Y}{\partial t_x} & \frac{\partial Y}{\partial t_y} & \frac{\partial Y}{\partial t_z} & \frac{\partial Y}{\partial \omega} & \frac{\partial Y}{\partial \phi} & \frac{\partial Y}{\partial \kappa} \\ \frac{\partial Z}{\partial t_x} & \frac{\partial Z}{\partial t_y} & \frac{\partial Z}{\partial t_z} & \frac{\partial Z}{\partial \omega} & \frac{\partial Z}{\partial \phi} & \frac{\partial Z}{\partial \kappa} \end{bmatrix} \quad (A.1)$$

$$\mathbf{A_{ij}} = \begin{bmatrix} 1 & 0 & 0 & a_{1,4} & a_{1,5} & a_{1,6} \\ 0 & 1 & 0 & a_{2,4} & a_{2,5} & a_{2,6} \\ 0 & 0 & 1 & a_{3,4} & a_{3,5} & 0 \end{bmatrix} \quad (A.2)$$

$$a_{1,4} = z_i(\cos\omega_j \sin\kappa_j + \cos\kappa_j \sin\omega_j \sin\phi_j) - y_i(\sin\kappa_j \sin\omega_j - \cos\kappa_j \cos\omega_j \sin\phi_j) \quad (A.3)$$

$$a_{1,5} = y_i(\cos\kappa_j \cos\phi_j \sin\omega_j) - z_i(\cos\kappa_j \cos\omega_j \cos\phi_j) - x_i(\cos\kappa_j \sin\phi_j) \quad (A.4)$$

$$\begin{aligned} a_{1,6} = {} & y_i(\cos\kappa_j \cos\omega_j - \sin\kappa_j \sin\omega_j \sin\phi_j) \\ & + z_i(\cos\kappa_j \sin\omega_j + \cos\omega_j \sin\kappa_j \sin\phi_j) \\ & - x_i(\cos\phi_j \sin\kappa_j) \end{aligned} \quad (A.5)$$

$$a_{2,4} = z_i(\cos\kappa_j \cos\omega_j - \sin\kappa_j \sin\omega_j \sin\phi_j) - y_i(\cos\kappa_j \sin\omega_j + \cos\omega_j \sin\kappa_j \sin\phi_j) \quad (A.6)$$

$$a_{2,5} = x_i(\sin\kappa_j \sin\phi_j) + z_i(\cos\omega_j \cos\phi_j \sin\kappa_j) + y_i(\cos\phi_j \sin\kappa_j \sin\omega_j) \quad (A.7)$$

$$\begin{aligned} a_{2,6} = {} & -y_i(\cos\omega_j \sin\kappa_j + \cos\kappa_j \sin\omega_j \sin\phi_j) \\ & - z_i(\sin\kappa_j \sin\omega_j - \cos\kappa_j \cos\omega_j \sin\phi_j) - x_i(\cos\kappa_j \cos\phi_j) \end{aligned} \quad (A.8)$$

$$a_{3,4} = -y_i(\cos\omega_j \cos\phi_j) - z_i(\cos\phi_j \sin\omega_j) \quad (A.9)$$

$$a_{3,5} = x_i(\cos\phi_j) - z_i(\cos\omega_j \sin\phi_j) + y_i(\sin\omega_j \sin\phi_j) \quad (A.10)$$

for the i^{th} target in the j^{th} scan.

Appendix B: Partial Differentials for Design Matrix of Bundle Adjustment

The Design matrix of the collinear equation is as follows

$$A_{ij} = \begin{bmatrix} \frac{\partial x}{\partial X} & \frac{\partial x}{\partial Y} & \frac{\partial x}{\partial Z} & \frac{\partial x}{\partial X^c} & \frac{\partial x}{\partial Y^c} & \frac{\partial x}{\partial Z^c} & \frac{\partial x}{\partial \omega} & \frac{\partial x}{\partial \phi} & \frac{\partial x}{\partial \kappa} \\ \frac{\partial y}{\partial X} & \frac{\partial y}{\partial Y} & \frac{\partial y}{\partial Z} & \frac{\partial y}{\partial X^c} & \frac{\partial y}{\partial Y^c} & \frac{\partial y}{\partial Z^c} & \frac{\partial y}{\partial \omega} & \frac{\partial y}{\partial \phi} & \frac{\partial y}{\partial \kappa} \end{bmatrix} \quad (B.1)$$

where

$$\frac{\partial x}{\partial X} = -\frac{\partial x}{\partial X^c} = \frac{c}{W^2}(r_{31}U - r_{11}W) \quad (B.2)$$

$$\frac{\partial x}{\partial Y} = -\frac{\partial x}{\partial Y^c} = \frac{c}{W^2}(r_{32}U - r_{12}W) \quad (B.3)$$

$$\frac{\partial x}{\partial Z} = -\frac{\partial x}{\partial Z^c} = \frac{c}{W^2}(r_{33}U - r_{13}W) \quad (B.4)$$

$$\frac{\partial y}{\partial X} = -\frac{\partial y}{\partial X^c} = \frac{c}{W^2}(r_{31}V - r_{21}W) \quad (B.5)$$

$$\frac{\partial y}{\partial Y} = -\frac{\partial y}{\partial Y^c} = \frac{c}{W^2}(r_{32}V - r_{22}W) \quad (B.6)$$

$$\frac{\partial y}{\partial Z} = -\frac{\partial y}{\partial Z^c} = \frac{c}{W^2}(r_{33}V - r_{23}W) \quad (B.7)$$

$$\frac{\partial x}{\partial \omega} = -\frac{c}{W^2}\left((Y - Y^c)(Ur_{33} - Wr_{13}) - (Z - Z^c)(Ur_{32} - Wr_{12})\right) \quad (B.8)$$

$$\frac{\partial x}{\partial \phi} = -\frac{c}{W^2}\begin{pmatrix} (X - X^c)(-W\sin\phi\cos\kappa - U\cos\phi) \\ +(Y - Y^c)(W\sin\omega\cos\phi\cos\kappa - U\sin\omega\sin\phi) \\ +(Z - Z^c)(-W\cos\omega\cos\phi\cos\kappa + U\cos\omega\sin\phi) \end{pmatrix} \quad (B.9)$$

$$\frac{\partial x}{\partial \kappa} = -\frac{cV}{W} \quad (B.10)$$

$$\frac{\partial y}{\partial \omega} = -\frac{c}{W^2}\left((Y - Y^c)(Ur_{33} - Wr_{23}) - (Z - Z^c)(Vr_{32} - Wr_{22})\right) \quad (B.11)$$

$$\frac{\partial x}{\partial \phi} = -\frac{c}{W^2}\begin{pmatrix} (X - X^c)(-W\sin\phi\cos\kappa - V\cos\phi) \\ +(Y - Y^c)(W\sin\omega\cos\phi\cos\kappa - V\sin\omega\sin\phi) \\ +(Z - Z^c)(-W\cos\omega\cos\phi\cos\kappa + V\cos\omega\sin\phi) \end{pmatrix} \quad (B.12)$$

$$\frac{\partial x}{\partial \kappa} = -\frac{cU}{W} \quad (B.13)$$

For point i and image j.

Appendix C: “CreateOptimalTexture.py”

```
# Import the proper libraries
import bpy
import bmesh
import numpy as np
from scipy import misc, ndimage
from math import *
from random import random
from collections import namedtuple
import matplotlib.pyplot as plt
import time


# Function definitions that can probably be moved to another file
def equirectangularuvmap( ob, scan, UVlayerName ):
    """map mesh faces to a equirectangular texture"""
    bm  = bmesh.from_edit_mesh(ob.data)
    uv_layer = bm.loops.layers.uv[UVlayerName]
    #ob.data.uv_textures[uv_layer].active = True

    bm.faces.layers.tex.verify()  # currently blender needs both layers.

    # Assume vertical alignment for now
    currName = scan.name
    scanX = scan.x
    scanY = scan.y
    scanZ = scan.z
    scanAngle = radians(scan.ang)

    # adjust UVs
    for f in bm.faces:

        f_c = f.calc_center_median()
        planeX = f_c.x - scanX
        planeY = f_c.y - scanY
        planeZ = f_c.z - scanZ
        planeVec = [planeX, planeY, planeZ]

        dotProd = (f.normal.x * planeX) + (f.normal.y * planeY) + (f.normal.z * planeZ)

        r = sqrt(planeX**2 + planeY**2)
        faceAngle = atan2(r,-planeZ)

        if (f.select == True) and (dotProd < 0) and (faceAngle >= radians(30)):
            # Calculate the offset for the whole face at a time rather than
            # per vertex. This should prevent smearing

            f_c_r = sqrt( (f_c.x - scanX)**2 + (f_c.y - scanY)**2 )

            centroidU = (atan2( (f_c.y - scanY) , -(f_c.x - scanX)) - scanAngle +pi )/(2.0*pi)
            centroidV = (atan2( f_c_r, scanZ-f_c.z))/(pi)

            if centroidU < 0:
                centroidU+=1

            if centroidV < 0:
                centroidV += 1

            for L in f.loops:

                # Calculate radial distance from center of image
                r = sqrt( (L.vert.co.x - scanX)**2 + (L.vert.co.y - scanY)**2 )

                # IF radial distance is too small, assume values for U and V
                if r < 0.01:
                    U = 0.5
                    if (L.vert.co.z-scanZ) > 0:
                        V = 1.0
```

```
                    else:
                        V= 0.0
                else:
                    # U is the normalized horizontal angle
                    U = (atan2( (L.vert.co.y - scanY) , -(L.vert.co.x - scanX)) - scanAngle +pi
)/(2.0*pi)

                    # V is the normalized vertical angle (zero for straight down, 1 for straight
up)

                    V = (atan2( r, scanZ-L.vert.co.z))/(pi)

                    for i in range(-1,2):
                        if fabs((U+i)-centroidU) < 0.25:
                            U += i
                            break

                    luv = L[uv_layer]
                    luv.uv = (U,V)
        else:
            for l in f.loops:
                luv = l[uv_layer]
                luv.uv = (0.5+random()/10,0+random()/10)

    bmesh.update_edit_mesh(ob.data)

def createGeometricWeightImage(size, ob, scan, uvName):
    bm  = bmesh.from_edit_mesh(ob.data)
    uv_layer = bm.loops.layers.uv[uvName]

    geoWeight = np.zeros(size)

    shape0 = geoWeight.shape[0]
    shape1 = geoWeight.shape[1]

    for f in bm.faces:

        f_c = f.calc_center_median()

        planeX = f_c.x - scan.x
        planeY = f_c.y - scan.y
        planeZ = f_c.z - scan.z
        planeVec = [planeX, planeY, planeZ]

        dotProd = (f.normal.x * planeX) + (f.normal.y * planeY) + (f.normal.z * planeZ)

        angle = atan2(sqrt(planeX**2 + planeY**2), -planeZ)

        faceWeight =  0

        if (dotProd > 0 or (angle < radians(32) and planeZ < 0)):
            continue
        else:
            faceWeight = abs(dotProd/sum(x**2 for x in planeVec))

        x_coords = []
        y_coords = []
        for L in f.loops:
            luv = L[uv_layer]
            x_coords.append( luv.uv[0])
            y_coords.append( luv.uv[1])

        # Barycentric triangle test
        x = np.asarray(x_coords)
        y = np.asarray(y_coords)

        x *= shape0

        y *= shape1

        denom = ((y[1] - y[2])*(x[0] - x[2]) + (x[2] - x[1])*(y[0] - y[2]))
```

```
        if denom == 0:
            continue

        invDenom = 1/denom

        for i in range(int(np.amin(x)), int(np.amax(x)+1.5)):
            for j in range(int(np.amin(y)), int(np.amax(y)+1.5)):

                a =  ((y[1] - y[2])*(i - x[2]) + (x[2] - x[1])*(j - y[2])) * invDenom
                b =  ((y[2] - y[0])*(i - x[2]) + (x[0] - x[2])*(j - y[2])) * invDenom
                c = 1 - a - b
                condition = a<=1 and a>=0 and b<=1 and b>=0 and c>=0 and c<=1

                if i < geoWeight.shape[0] and j < geoWeight.shape[1] and condition:
                    geoWeight[geoWeight.shape[1]-j,i] = faceWeight


    return geoWeight

print('Beginning Texture Baking Procedure')

curDirectory = 'C:\\Users\\Adam\\Documents\\Blender Stuff\\'

textureSize = 4096

# Load the (hardcoded) scan locations
scanPosition = namedtuple('ScanPosition',['name','x','y','z','ang'])

# Future work: Plugin for Faro that loads these values directly
# or even just from a specific file
allScanPoses = [
 scanPosition('scan_001', -3.679692,  3.942834 ,1.393807 , 58.353693),
 scanPosition('scan_002', -0.84634, 7.649932, 1.213567  , 72.287435),
 scanPosition('scan_003', 1.856701, 6.924198, 0.391366 , 36.992808),
 scanPosition('scan_004', 5.289296, 2.512209, -0.695059  , 36.801182),
 scanPosition('scan_005', 6.022316, -1.458117, -1.520517, 103.64301),
 scanPosition('scan_006', 1.60267, -1.84452, -0.8429, 165.616499),
 scanPosition('scan_007', -0.815092, -1.530523, -0.203589 , 117.531972),
 scanPosition('scan_008', -3.13409, 1.85642, 0.87117, 144.713466),
 scanPosition('scan_009', 4.366643, 5.5628, -0.289433, 142.801249),
 scanPosition('scan_010', 3.67919, -3.942931, -1.396275, 126.174026)
 ]

# Associate each scan location with a particular texture
# This assumes that the images are already loaded into Blender
# This adds work that needs to be done before this function is called
# It would be better to include this loading process as part of the function
scanImages = {}

for scan in allScanPoses:
    scanImages[scan] = 'Arctic_'+scan.name+'_resize.png'
    try:
        bpy.data.materials[0].active_texture.image = bpy.data.images[scanImages[scan]]
    except:
        print('Image: '+scanImages[scan]+' not found')

# Get reference to mesh
ob = bpy.context.selected_objects[0]
bpy.ops.object.mode_set(mode='EDIT')
bpy.ops.mesh.select_all(action='SELECT')


# Set the material/texture properties to correct settings
mat = ob.active_material
mat.use_shadeless = False

# Unwrap the mesh's primary UV map

try:
    ob.data.uv_textures['SmartProject'].active = True
except:
```

```
    ob.data.uv_textures.new(name='SmartProject')
    ob.data.uv_textures['SmartProject'].active = True
#bpy.ops.uv.smart_project()

try:
    ob.data.uv_textures['Equi-rectangular'].active = True
except:
    ob.data.uv_textures.new(name='Equi-rectangular')
    ob.data.uv_textures['Equi-rectangular'].active = True

bpy.ops.uv.reset()
# Make sure the light object has the right settings
# i.e. is a point light, casting ray shadows,

if len(bpy.data.lamps) == 0:
    bpy.ops.object.lamp_add(type='POINT')
elif len(bpy.data.lamps) > 1:
    print('Error: More than one lamp in scene')

bpy.data.lamps[0].shadow_method = 'RAY_SHADOW'
bpy.data.lamps[0].energy = 100

# Prepare to store images
scanWeightDict = {}
scanTextureDict = {}
# For each scan
for scan in allScanPoses:
    # Set the current texture and UV, and other settings for material
    try:
        ob.active_material.active_texture_index = 0
    except:
        bpy.ops.texture.new()

    try:
        mat.active_texture.image = bpy.data.images[scanImages[scan]]
    except:
        print('Image '+scanImages[scan]+' not loaded, continueing')
        continue

    # Calculate the current texture's UVs
    print('starting Equirectangular UV map calculation for scan '+ scan.name)
    time1 = time.clock()
    equirectangularuvmap(ob, scan, 'Equi-rectangular')
    time2 = time.clock()
    elapsed = time2-time1
    print("Time spend processing Equirectangular UV map: "+str(elapsed))
    print('Ending Equirectangular UV map calculation for scan '+ scan.name)

    mat.active_texture.extension = 'REPEAT'

    mat.texture_slots[0].texture_coords = 'UV'
    mat.texture_slots[0].uv_layer = 'Equi-rectangular'
    mat.texture_slots[0].use_map_alpha = True

    currentTextureName = scan.name+'_texture_bake'
    # Create new image
    bpy.ops.image.new(name=currentTextureName, width=textureSize, height=textureSize, alpha=True)

    # Set bake margins
    bpy.context.scene.render.bake_margin = 2

    # Set the UV coordinates
    ob.data.uv_textures['SmartProject'].active = True

    # Specify the bake type
    bpy.data.scenes["Scene"].render.bake_type = "TEXTURE"

    # Exit edit mode
    bpy.ops.object.mode_set(mode='OBJECT')

    # Set the target image
```

```
for d in ob.data.uv_textures['SmartProject'].data:
    d.image = bpy.data.images[currentTextureName]

# Enter edit mode
bpy.ops.object.mode_set(mode='EDIT')
bpy.ops.mesh.select_all(action='SELECT')

# Bake the image
bpy.ops.object.bake_image()

# Save the baked texture
bpy.data.images[currentTextureName].update()
bpy.data.images[currentTextureName].filepath_raw = '//'+currentTextureName+'.png'
bpy.data.images[currentTextureName].file_format = 'PNG'
bpy.data.images[currentTextureName].save()

# Set the light to the location of the scan
bpy.data.objects['Point'].location = [scan.x, scan.y, scan.z]

# Create a new texture for the shadow map
currentShadowName  = scan.name+'_shadow_bake'
bpy.ops.image.new(name=currentShadowName, width=textureSize, height=textureSize)

# Bake the shadows
# Set the UV coordinates
ob.data.uv_textures['SmartProject'].active = True

# Specify the bake type
bpy.data.scenes["Scene"].render.bake_type = "SHADOW"

# Exit edit mode
bpy.ops.object.mode_set(mode='OBJECT')

# Set the target image
for d in ob.data.uv_textures['SmartProject'].data:
    d.image = bpy.data.images[currentShadowName]

# Enter edit mode
bpy.ops.object.mode_set(mode='EDIT')
bpy.ops.mesh.select_all(action='SELECT')

# Bake the image
bpy.ops.object.bake_image()

# Save the baked shadow map
bpy.data.images[currentShadowName].update()
bpy.data.images[currentShadowName].filepath_raw = '//'+currentShadowName+'.png'
bpy.data.images[currentShadowName].file_format = 'PNG'
bpy.data.images[currentShadowName].save()

# Convert the shadow map to a greyscale, taking the minimum of
# a colour channel and the alpha channel
shadow = misc.imread(curDirectory+currentShadowName+'.png')
occlusion = (shadow[:,:,0].astype(float))/255
del shadow

occlusion = occlusion>0.5
eroded_occlusion = ndimage.binary_erosion(occlusion)
occlusion = ndimage.binary_dilation(eroded_occlusion)
del eroded_occlusion

occlusion = ndimage.gaussian_filter(occlusion.astype(float), sigma=1)

# Create a new greyscale image, for the weight of each face to belong
print('starting geometric texture bake')
time1 = time.clock()
geometricWeight = createGeometricWeightImage(occlusion.shape, ob, scan, 'SmartProject')
time2 = time.clock()
elapsed = time2-time1
print("Time spend processing Geomitric Weight Image: "+str(elapsed))
```

```
    geometricWeight = ndimage.gaussian_filter(geometricWeight, sigma=1)

    misc.imsave(curDirectory+scan.name+'_geomitricWeight'+'.png',geometricWeight)

    #calculate final weight image
    weightImg = np.multiply(occlusion, geometricWeight)
    scanWeightDict[scan] = weightImg
    scanTextureDict[scan] = misc.imread(curDirectory+currentTextureName+'.png')

sumRed = np.zeros(weightImg.shape)
sumGreen =  np.zeros(weightImg.shape)
sumBlue =  np.zeros(weightImg.shape)

for scan in scanTextureDict:
    sumRed += np.multiply( scanTextureDict[scan][:,:,0], scanWeightDict[scan])
    sumGreen += np.multiply( scanTextureDict[scan][:,:,1], scanWeightDict[scan])
    sumBlue += np.multiply( scanTextureDict[scan][:,:,2], scanWeightDict[scan])

inter = np.dstack(list(scanWeightDict.values()))

sumWeight = np.sum(inter,axis = 2)
sumWeight = np.add(sumWeight, (sumWeight == 0).astype(float)*0.001)

finalTexture = np.zeros([textureSize,textureSize,3])
finalTexture[:,:,0] = np.divide(sumRed,sumWeight)
finalTexture[:,:,1] = np.divide(sumGreen,sumWeight)
finalTexture[:,:,2] = np.divide(sumBlue,sumWeight)

misc.imsave(curDirectory+'finalTexture'+'.png', finalTexture)

bpy.ops.image.open(filepath=curDirectory+'finalTexture.png')
mat.use_shadeless = True
mat.active_texture.image = bpy.data.images['finalTexture.png']
mat.texture_slots[0].uv_layer = 'SmartProject'

print('Done')
# Finally, once all weight matrixs and image textures have been made,
# take a weighted average of them all. This new texture is the final
# texture to be used for the mesh.
```