Analysis and Exploitation of Automatically Generated Scene Structure from Aerial Imagery

by

David R. Nilosek

B.S. Rochester Institute of Technology, 2008

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Chester F. Carlson Center for Imaging Science
College of Science
Rochester Institute of Technology

March 31st, 2014

Signature of the Author _____

Accepted by _____
Coordinator, Ph.D. Degree Program Date

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE
COLLEGE OF SCIENCE
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

Ph.D. DEGREE DISSERTATION

The Ph.D. Degree Dissertation of David R. Nilosek
has been examined and approved by the
dissertation committee as satisfactory for the
dissertation required for the
Ph.D. degree in Imaging Science

Dr. Carl Salvaggio, Dissertation Advisor

Dr. David Messinger

Dr. Nathan Cahill

Dr. Steven LaLonde

Date

Analysis and Exploitation of Automatically Generated Scene Structure from Aerial Imagery

by

David R. Nilosek

Submitted to the
Chester F. Carlson Center for Imaging Science
in partial fulfillment of the requirements
for the Doctor of Philosophy Degree
at the Rochester Institute of Technology

Abstract

The recent advancements made in the field of computer vision, along with the ever increasing rate of computational power has opened up opportunities in the field of automated photogrammetry. Many researchers have focused on using these powerful computer vision algorithms to extract three-dimensional point clouds of scenes from multi-view imagery, with the ultimate goal of creating a photo-realistic scene model. However, geographically accurate three-dimensional scene models have the potential to be exploited for much more than just visualization. This work looks at utilizing automatically generated scene structure from near-nadir aerial imagery to identify and classify objects within the structure, through the analysis of spatial-spectral information. The limitation to this type of imagery is imposed due to the common availability of this type of aerial imagery. Popular third-party computer-vision algorithms are used to generate the scene structure. A voxel-based approach for surface estimation is developed using Manhattan-world assumptions. A surface estimation confidence metric is also presented. This approach provides the basis for further analysis of surface materials, incorporating spectral information. Two cases of spectral analysis are examined: when additional hyperspectral imagery of the reconstructed scene is available, and when only R,G,B spectral information can be obtained. A method for registering the surface estimation to hyperspectral imagery, through orthorectification, is developed. Atmospherically corrected hyperspectral imagery is used to assign reflectance values to estimated surface facets for physical simulation with DIRSIG. A

spatial-spectral region growing-based segmentation algorithm is developed for the R,G,B limited case, in order to identify possible materials for user attribution. Finally, an analysis of the geographic accuracy of automatically generated three-dimensional structure is performed. An end-to-end, semi-automated, workflow is developed, described, and made available for use.

Acknowledgements

I joined the Digital Imaging and Remote Sensing Laboratory as a graduate student in the Winter of 2008. A numerous amount of people have helped me get to where I am today, and I would like to acknowledge and thank them all.

First and foremost, my thesis advisor Dr. Carl Salvaggio. Carl has officially been my advisor for about six years, but really he has been helping me along since before I completed my undergraduate degree. Beyond being an excellent academic mentor, I also consider him to be a good friend. He has helped me in both my professional and personal life, and I will be forever grateful for all he has done.

I would also like to thank my thesis committee: Dr. Dave Messinger, Dr. Nate Cahill, and Dr. Steve LaLonde. Dr. Messinger and Dr. Cahill were very helpful in shaping the direction of my dissertation work. They asked thought-provoking questions and made a number of very useful suggestions. I am grateful for their expertise and willingness to give advice. I am also very grateful for Dr. LaLonde's being the external chair of my committee. We have a very limited academic relationship, however, I am astounded by the generosity and support he gave towards my dissertation process.

I would like to thank Dr. Derek Walvoord from ITT Exelis for advising me on a number of subjects throughout my work, and for always being able to answer any question I had. There have been a number of staff members in the DIRS Lab, whom have helped me along the way. I would like to thank Jason Faulring, for his unending technical support and expertise. I would also like to thank Scott Brown for his leadership, while I was working under the ESRI grant. I also would like to thank Mike Richardson, who helped me hone my presentation skills time and again. Thanks also goes to Bob Krzaczek for introducing me to git, asciidoc, and his numerous bits of coding advice.

Many thanks must be given to Cindy Schultz, without whom many DIRS graduate students, including myself, would be hopelessly lost. I would also like to thank Sue Chan for making sure I submitted everything on time (even when I was late). I'd also like to thank the many professors I had during my coursework, I am grateful to have attended an institution with such a wealth of knowledge and talent.

I've been lucky enough to have multiple generations of officemates, all of whom have helped me along my path. I'd like to thank Sarah for saving me from drowning at my former desk and convincing Cindy to let me move into the office. She helped me with my classwork, research, and was a great officemate. Thanks also goes to May for helping

me feel welcome in the office, and along with Sarah, helping turn my last name into a verb for computer malfunction. I would like to thank Shaohui and Mike for the many casual and technical conversations we had. They were helpful making office life enjoyable. I would also like to thank Katie, whom joined me in the quest for understanding C++ and Computer Vision. She helped me with my classwork as well as my research. I greatly appreciate all the help and friendship she's given me over the years.

Many thanks also goes to Steve Schultz, from Pictometry Corp., whom not only gave me my first internship experience, but also hired me even before I had completed my dissertation. Thanks also to Amy Galbraith for being my mentor during my internship at Los Alamos National Laboratory.

I want to thank my friends and family. Mom, Dad, Andrea, and Courtney, thanks for all your support and always making me feel like I can accomplish anything. Uncle Ed and Lucy, thank you for supporting me through my undergraduate and graduate degrees, I would not have been able to do it without you. Curtis and Pat, you guys were some of the best roommates I could have asked for, thanks for helping me keep my life entertaining.

Lastly, I would not have made it here without the unyielding support of my fiancée, Elena. I cannot thank her enough for helping me get to where I am today. She was there for me throughout the long nights of writing, working, and frustration. I am beyond lucky to have such a person to rely on and partner with throughout life.

To everyone who has been there, thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, with the increase of computational power and speed, many advancements have been made in the automation of photogrammetry through the application of computer vision methods with aerial imagery. Photogrammetry exploits the geometric properties of imagery in order to make highly accurate measurements of objects within the scene. Computer vision exploits not only the geometric properties of imagery, but also the spatial, spectral, and statistical properties with the attempt to intelligently detect and describe the objects within the imagery. Algorithms and processes in the field of analytical photogrammetry developed as early as the 1960s, while computer vision evolved much later. Due to the separation in age and the difference in goals, these two fields have shared very little with each other, while their combination has significant potential.

This work will focus on taking a computer vision-based approach to extracting geometrically and physically accurate structures from aerial imagery, a goal of photogrammetry. The addition of computer vision-aided techniques allows additional information to be extracted through the combination of multiple modalities of imagery. The goal of this research is to extract geometrically and physically accurate models from multi-view visible (RGB) aerial imagery. Physical accuracy, in this case, is not just the surface structure of the model but also knowledge of the surface reflectance spectra. This knowledge can lead to a higher understanding of the material properties of each model. The end goal from this work will be the production of a geometrically and physically accurate model, along with a semi-automatic end-to-end workflow to produce the model.

The methods required to reach theses goals can be split into two major parts. The first part is the extraction of accurate scene structure from multi-view RGB imagery. Scene structure is defined here as a collection of discrete three-dimensional measurements spread

across the entire scene. These measurements form a three-dimensional point cloud, the basis for further structure modeling. Scene structure alone cannot be used to create a physically accurate three-dimensional model. The second part of the methods used in this work focus on the modeling of the extracted scene structure. The modeling process uses the scene structure along with additional scene information to estimate a physical model for specific objects within the scene.

1.1 Accurate Structure Extraction

Identifying objects within a scene is a key goal in the field of computer vision. One method of describing objects within a scene is to identify their structure through analysis of their motion between multiple images, a process commonly referred to as Structure from Motion (SfM). This technique of analyzing objects has its roots in traditional photogrammetry, though the standard goal of SfM techniques lies in object identification rather than mensuration. To that end, the SfM algorithm chain assumes little or no information about the imaging platform. The computer vision community has developed a number of complex processes to estimate camera pose, *i.e.*, the sensor position and orientation, but these methods are limited to estimating parameters in a relative sense [25, 32]. Consequently, any estimated object structure is in the same relativistic coordinate system. Precise geographic measurements of objects, cannot be directly extracted in this coordinate system without additional information.

With the goal of accurate physical modeling in mind, precise geographic measurements of scene structure are desired. An estimate of the scene structure measurements in an accurate Earth-based coordinate system can be made through the use of additional information. In many practical SfM systems, it is assumed that the camera's calibration information is known; this includes the focal length, pixel pitch, and sensor size [25, 49]. Knowledge of this information allows for a metric reconstruction of both the camera pose and object structure [25]. Consequently, the relationship between the SfM-based world coordinate system and the desired Earth-based coordinate system can be described by a simple, seven degrees of freedom similarity transform, as they are both metric coordinate systems.

Computer vision techniques for extracting scene structure combined with additional information for geographic registration provides the required methods for extracting ac-

curate scene structure from multi-view imagery. Objects contained in this scene structure can be further processed to extract physically accurate three-dimensional models.

1.2 Physical Model Estimation

Geographically accurate image-based three-dimensional structure measurements provide the basis for further analysis and modeling of target objects within the scene of interest. These targets tend to be man-made structures (*e.g.* buildings, houses, large structures). This work focuses on the physical model estimation of these types of structures.

An assumption can be made when focusing on these types of structures, more generally called the "Manhattan-world" assumption [9]. This assumption is that structures in a three-dimensional Cartesian coordinate system are primarily made up of large planar faces and the structure tends to orient itself in three orthogonal directions. The original assumption stated that the camera is assumed to be approximately in the horizontal plane, having Z map with the vertical lines in the imagery. In the case of aerial imagery, the camera is assumed to be approximately orthogonal to the horizontal plane (or near the nadir viewing direction).

A voxel-based modeling process is used in this work, in order to estimate the physical model's surface structure of a specific man-made target. Using the "Manhattan-world" assumption it can be assumed that man-made structures tend to have horizontal planes connected to vertical walls. This assumption aligns itself very well with voxel-based structure modeling.

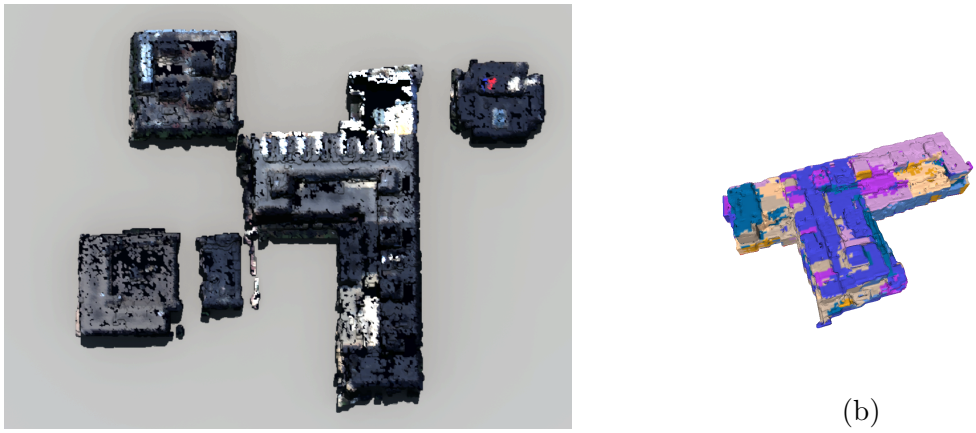
The surface structure is only half of the physical model. The second half requires spectral attribution of the surface. There are two scenarios this work examines; when only R,G,B spectral information is available and when hyperspectral information is available. The latter scenario is the simpler case, as all it requires is a direct mapping from the model's surface to an atmospherically corrected hyperspectral image. When only R,G,B information is known for the scene, estimating a high resolution reflectance spectra becomes very difficult. Instead of attempting to fully estimate the reflectance spectra for each facet on a surface model, this work looks at methods of facet classification. Given knowledge of surface classes, attempts could be made to identify the reflectance spectra through database matching. Given the large nature of that problem, this work is limited

to creating methods to identify facet classes.

1.3 Summary

Physical simulation is a direct application of the automatic generation of physically accurate three-dimensional models. Three-dimensional physical simulation of imagery is the process of synthetically replicating interactions of light with three-dimensional matter and processing those interactions such that a radiometrically accurate image can be produced, given a set of imaging parameters. Simulation of this nature can be very powerful for testing and analysis of novel image processing algorithms, these simulations could even be used for surveillance-based modeling. The basis for physics-based simulation, is an accurate three-dimensional model attributed with material properties. These models are painstakingly created by hand, and take many hours to complete. For this reason, this type of physical simulation is limited in the scenes it can simulate and consequently its applications. Automatically creating these physical models from multi-view aerial imagery would provide a convenient method of model generation, and significantly broaden the applications of physics-based image modeling.

This work takes a step in that direction by developing a process to extract the surface structure of target objects, as well as attempt to label materials on the extracted surface. As explained in Section 1.2, two situations are examined. Given additional atmospherically compensated hyperspectral imagery of the scene, reflectance can be directly mapped to structure facets for physical modeling within the spectral range of the hyperspectral reflectance imagery. An example of this type of modeling is shown in Figure 1.1.



(a) (b)

Figure 1.1: *Physical modeling is performed in this work with two scenarios in mind. The first is with the addition of hyperspectral reflectance imagery, allowing for a direct mapping of spectra onto facets. This allows for physical modeling to be done easily within the spectral range of the hyperspectral imagery, a physical simulation of five structures is shown in (a). Given only R,G,B imagery, estimating the surface reflectance becomes a very difficult problem. To this end, this work attempts to classify different materials on the surface of a structure using spatial-spectral information. The classes allow for user-assisted attribution, an example of a class-mapped three-dimensional surface produced by this work is shown in (b).*

Given only R,G,B imagery, estimating the surface reflectance of a model becomes an ill-posed problem. It is very difficult to discriminate between materials with such low spectral resolution sampling. Instead of identifying specific materials, potential material classes can be identified on the surface structure through analysis of the surface's spatial-spectral properties. Figure 1.1 shows a classified three-dimensional surface created through this process. This gives the user an estimate of the structure's surface as well as the location of different materials classes, which can be attributed by the user.

The remainder of this document is split into four chapters; Chapter 2 details the background work and topics needed to understand the methods used in this work, Chapter 3 discusses the methodology used in performing this work, Chapter 4 shows the datasets used and results obtained through the developed methodology, and Chapter 5 discusses the results, applications, and future work.

1.4 Contributions

The work performed here for geographically accurate structure extraction and physical model estimation makes a number of contributions. A well-known process for generating geographically accurate scene structure from multi-view imagery is presented and a Linux-based, end-to-end, scripted, workflow was developed. A method for analysis of the extracted structure's geoaccuracy was developed, and, used to evaluate the performance of several common methods for structure georegistration.

This work specifically addresses the usage of nadir-looking imagery for scene structure reconstruction. Extracted structure from nadir imagery often contains a significant amount of noise and holes. A voxel-based noise reduction, surface estimation, and interpolation process is developed for estimating the surface of target objects from the extracted structure.

In order to generate physically attributed models, estimated surface facets must be attributed with physical characteristics. The characteristics which are attributed are dependent on the amount of additional data that can be utilized. Two processes are developed for characteristic attribution. The first process describes a methodology for registering additional geolocated information, such as orthorectified imagery, to the surface facets through the orthorectification process. Secondly, if no additional geolocated data is available, surface characteristics can still be attributed through spatial-spectral analysis of the original R,G,B nadir-looking imagery with the reconstructed model. To this end, a spatial-spectral segmentation process was developed for identifying regions of spectrally similar surface facets.

Chapter 2

Background

The process of extracting structure from multi-view imagery is one that has been well developed in the computer vision community [62]. The entire process falls into only a few steps, as shown in Figure 2.1. These steps include; feature detection, description, and matching within imagery, camera pose estimation, structure triangulation, and optimization. Processes to perform each of these steps are reviewed in Sections 2.1 through 2.4.

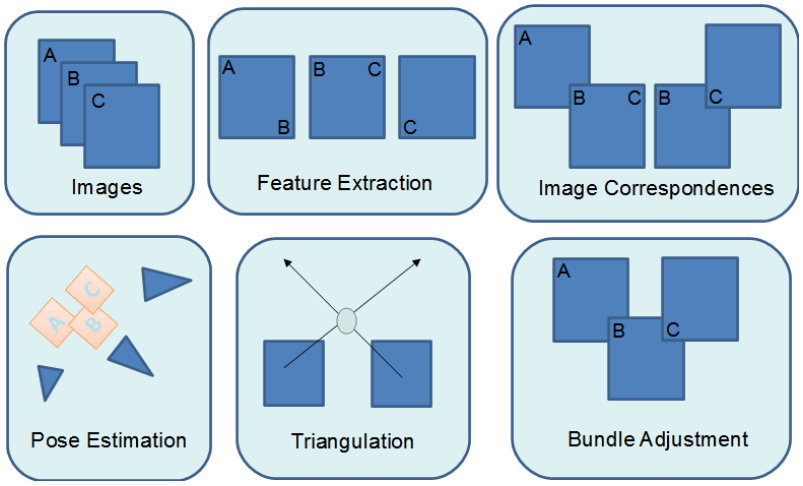


Figure 2.1: *Structure from Motion can be broken down into a few steps: Image feature detection and matching, camera pose estimation, structure triangulation, and optimization.*

Structure from Motion (SfM) provides a methodology for extracting discrete measure-

ments of three-dimensional structure in a relative world coordinate system. A transform can be derived to bring the relative structure measurements to a fixed Earth-based coordinate system. The derivation of this transform is discussed in Section 2.5. This chapter will cover the material which is necessary to understand the three-dimensional reconstruction process used in this work, as well as review previous work done in each area. The discussions presented here will shed light on how an automatic three-dimensional reconstruction can be obtained.

2.1 Epipolar Geometry

Understanding the geometry behind multi-view imagery is critical to understanding the methods that exploit this geometry. There is a nomenclature convention presented by Hartley and Zisserman [25] that the following discussion, as well as the rest of this work, will adhere to. This section will present the basic epipolar (stereo) geometry needed in order to further understand the algorithms used for three-dimensional reconstruction.

2.1.1 Projective Geometry

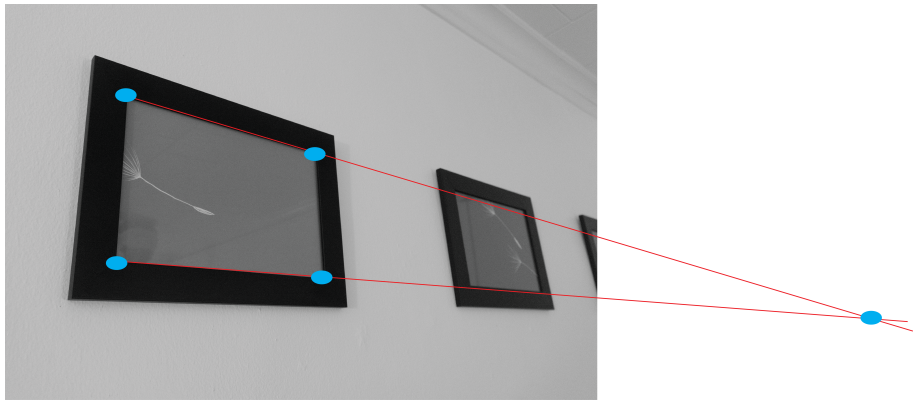


Figure 2.2: *Parallel lines viewed from a angled perspective often appear as though if extended they will eventually intersect (this intersection point is called the point at infinity).*

The notion of projective geometry has been around for centuries [23], that is, an attempt to quantify and model Euclidean geometry that contains a perspective view. It is a natural progression from basic Euclidean geometric modeling, since it is how humans view the world. Figure 2.2 shows a perspective view of an object on a wall. Here, there are lines within the frame that humans know to be parallel which appear to converge at some point, called the point at infinity. In order to represent this point in a coordinate system without causing mathematical errors, a homogeneous coordinate system is used. Projective geometry is essentially the attempt to quantify the projection of a higher-dimensional coordinate system onto a lower-dimensional coordinate system. An example and widely used application of this is the projection of a three-dimensional scene onto a two-dimensional camera frame.

Homogeneous Coordinates

The homogeneous coordinate system was first used in projective geometry by August Möbius [61]. Essentially, this coordinate system is the Euclidean coordinate system with one extra dimension, which allows for the quantification of the projected coordinates. As an illustration, consider the algebraic definition of a line in 2-D space, shown in Equation 2.1.

$$Ax + By + C = 0 \tag{2.1}$$

The definition of a line can be vectorized by representing the line as a set of parameters A, B, and C. The vectorization of this equation is shown in Equation 2.2.

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = 0 \tag{2.2}$$

The left-hand vector of Equation 2.2 is considered to be a point in 2-D space, $\mathbf{x} = [x, y, 1]$, which falls on a line in 2-D space represented by a set of parameters, $\mathbf{l} = [A, B, C]^T$, such that $\mathbf{x}^T\mathbf{l} = 0$. The third dimension of $\mathbf{x}$ allows for the representation of all points that might possibly fall on line $\mathbf{l}$. Parameterizing $\mathbf{x}$, $k\mathbf{x} = [kx, ky, k]$, allows for the representation of all possible set of homogeneous points which represent the same point falling

on line $\mathbf{l}$, this can be seen by substituting this parametrization back into Equation 2.2,

$$\begin{bmatrix} kx & ky & k \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = k0 = 0 \tag{2.3}$$

This also shows that the degrees of freedom for the homogeneous point on a line is equal to 2, the same as it would be in Euclidean space. This representation allows for a projective representation for the point $\mathbf{x}$, illustrated in Figure 2.3.

The previous example gave two important properties of homogeneous coordinates in projective geometry. The first is that in homogeneous coordinates, if a point $\mathbf{x}$ falls on line $\mathbf{l}$, their dot product must be equal to zero. The second important property is the mapping from projective space to Euclidean space. For a 2-D point (x, y) the conversion is $\left(\frac{x}{k}, \frac{y}{k}\right)$ where k is the third dimension of the projective point. This is extended to higher dimensional spaces by simply dividing the extra dimension into the previous dimensions. In order to eliminate confusion with Euclidean coordinates, homogeneous point coordinates will henceforth be represented as $\mathbf{x} = [x_1, x_2, x_3]$ for two-dimensional Euclidean space, and $\mathbf{X} = [x_1, x_2, x_3, x_4]$ for three-dimensional Euclidean space. Euclidean coordinates will be represented using $\mathbf{x} = [x, y]$ and $\mathbf{X} = [x, y, z]$.

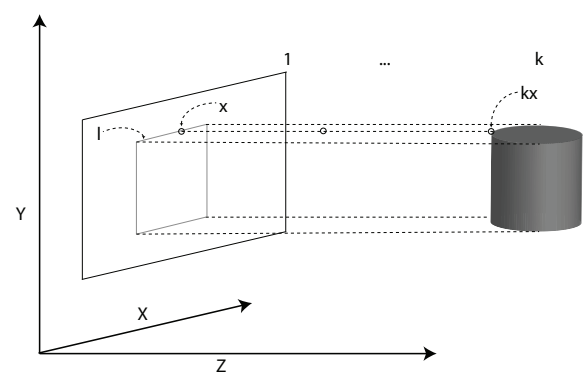


Figure 2.3: *Homogeneous coordinates can be used to represent projections from higher dimensional spaces to lower dimensional spaces. In this situation, the point* $\mathbf{X}$ *can originate anywhere between the projected line and three-dimensional shape. Describing the line* $\mathbf{l}$ *in terms of homogeneous coordinates allows for the representation of all possible points which would be projected onto that line.*

Homographies

A projective transformation, or homography, is defined as an invertible transformation such that $\mathbf{x}' = H(\mathbf{x})$ [25]. The homography transforms the point $\mathbf{x}$ to the projective plane of $\mathbf{x}'$. For example, in Figure 2.3, a homography could be derived to transform the point $\mathbf{x}$ on the on the projective plane $k = 1$ to any projective plane $k > 1$. The use of homogeneous coordinates allows for a matrix representation of the function H, shown in Equation 2.4.

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{2.4}$$

As shown in Equation 2.3, the parameterization of the projective space does not add a degree of freedom to the homogeneous coordinate system. Since the matrix $\mathbf{H}$ is defined in homogeneous coordinates, it is only defined up to this scaling factor, and therefore has eight degrees of freedom. Constraining elements of this matrix allows for affine, similarity, and Euclidean transformations to be constructed. These transformations can be useful in the manipulation of coordinate systems using homographies. The homography can be extended into higher dimensions by adding the appropriate number of rows and columns to the matrix $\mathbf{H}$.

2.1.2 Camera Model

A camera can be mathematically represented as a mapping from a three-dimensional to a two-dimensional coordinate space. This can be done easily by using projective geometry with homogeneous coordinates.

Pinhole Camera Model

The simplest camera to model using projective geometry is the pinhole camera. Figure 2.4 shows an example of a pinhole style projection of a vertical image. While the true exposure is taken at the image negative, it is more useful to define the geometry in terms

of the image positive. Using similar triangles lengths x and y can be defined as follows:

$$x = f\frac{X}{Z}$$
$$y = f\frac{Y}{Z} \tag{2.5}$$

Equation 2.5 assumes the coordinate center to be in the center of the camera, which requires knowledge of the principle point offset. For orthogonal projections, such as the one shown in Figure 2.4, the principle point offset is the offset that brings the coordinate system to the center of the image. Equation 2.6 shows Equation 2.5 updated to include the principle point offset.

$$x = f\frac{X}{Z} + p_x$$
$$y = f\frac{Y}{Z} + p_y \tag{2.6}$$

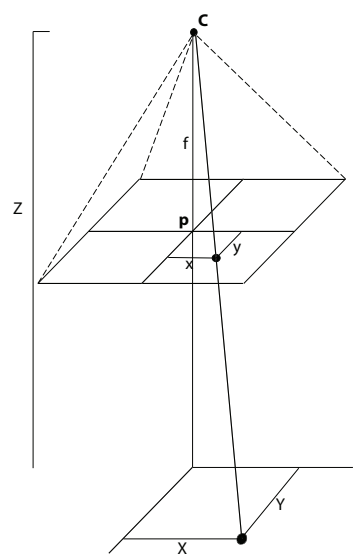


Figure 2.4: *An orthogonal image projection is one where the camera frame has no rotations between the world coordinate system and the frame coordinate system. The principle point offset is included in the orthogonal image projection.*

Equation 2.6 can be represented in matrix form using homogeneous coordinates. This

is shown in Equation 2.7.

$$\begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.7}$$

The left-hand side of the equation is the homogeneous representation of Equation 2.6. This can be verified by using the method for homogeneous to Euclidean coordinate conversion presented in Section 2.1.1. The projection matrix shown in Equation 2.7 is known as the camera calibration matrix, **K**.

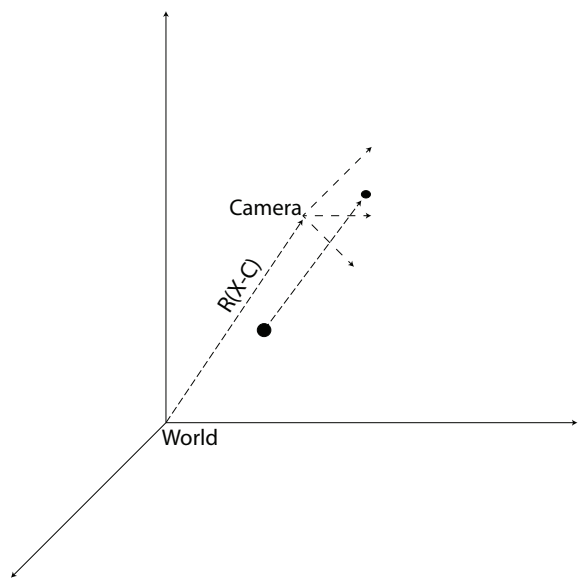


Figure 2.5: *The camera frame is often rotated relative to the world frame. In order to bring a world point into the camera frame, the point must be rotated and translated into the frame.*

World to Camera Frame Transformation

The camera shown in Figure 2.4 assumes that the camera center and world point could be represented in the same frame. In most cases the camera center and world point have to be rotated and translated to be in the same frame, as seen in figure 2.5. In Euclidean

coordinates, the world point $\mathbf{X}$ can be represented in the camera frame by subtracting the camera center from the world point, and then rotating the frame to that of the camera frame. This operation is shown in Equation 2.8,

$$\mathbf{X}_{cam} = \mathbf{R}(\mathbf{X} - \mathbf{C}) \tag{2.8}$$

where $\mathbf{R}$ is a 3x3 rotation matrix that rotates the world frame to the camera frame. This can be represented in homogeneous coordinates as shown in Equation 2.9.

$$\mathbf{X}_{cam} = \begin{bmatrix} \mathbf{R} & -\mathbf{RC} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.9}$$

Camera Projection Matrix

By combining the camera calibration matrix with the world-to-camera frame transformation, the camera projection matrix can be formed. Shown in Equation 2.10, the matrix is split into two sections, a 3x3 block representing the rotation, and a 3x1 block representing the translation. The translation $\mathbf{t}$ represents $-\mathbf{RC}$.

$$\mathbf{P} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}] \tag{2.10}$$

In order to use this model with digital cameras, the camera calibration matrix $\mathbf{K}$ must be modified so that the units are all the same. This requires multiplying each unit by a scale factor,m, which represents the number of pixels per unit length. The modified calibration matrix is shown in Equation 2.11.

$$\mathbf{K} = \begin{bmatrix} mf & 0 & mp_x \\ 0 & mf & mp_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

Given Equation 2.10 combined with Equation 2.11, the relationship between a world point $\mathbf{X}$ and image point $\mathbf{x}$ can be defined using Equation 2.12,

$$\mathbf{x} = \mathbf{PX} \tag{2.12}$$

where $\mathbf{P}$ is known as the camera projection matrix.

2.1.3 Stereo Geometry and the Fundamental Matrix

Stereo geometry is also known as epipolar geometry. This section will cover the basic properties of epipolar geometry. Figure 2.6 shows a basic representation of two images observing the same point in space. With $\mathbf{C}$, and $\mathbf{C}'$ as the camera centers, $\mathbf{X}$ as the point in three-dimensional space, and the points $\mathbf{x}$, and $\mathbf{x}'$ as the projections of $\mathbf{X}$ onto their respective cameras.

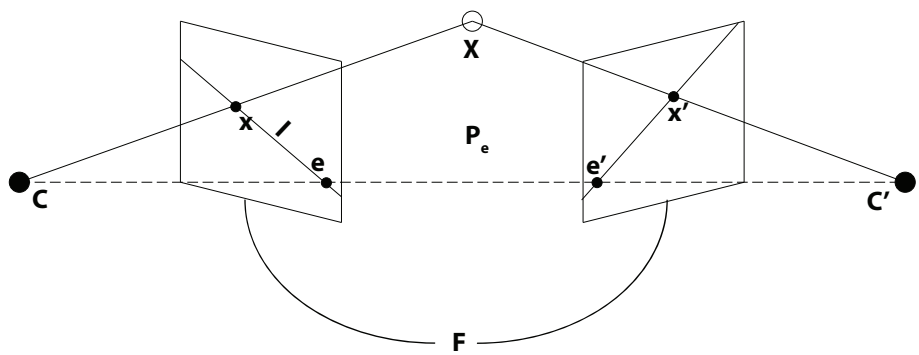


Figure 2.6: *Stereo (epipolar) geometry is the building block for all multi-view reconstruction processes. This figure shows the positions of the epipoles (e,e'), epipolar lines (l,l'), epipolar plane (P), world point ($\mathbf{X}$), and camera centers (C,C'). The relationship between the imagery can be described using the fundamental matrix (F).*

The three-dimensional plane made from points $\mathbf{C}$, $\mathbf{C}'$, and $\mathbf{X}$ represents the epipolar plane, $\mathbf{P}_e$. Rays projected from the camera center to $\mathbf{X}$ are coplanar with the epipolar plane. The intersection points of the ray between $\mathbf{C}$ and $\mathbf{C}'$ constitute the epipoles for each image. The epipolar plane intersects each image at two points, the back projection of $\mathbf{X}$ onto the image, and each image's epipole. The epipole for each set of images remains constant for all corresponding points $\mathbf{x}$ and $\mathbf{x}'$.

Based on the geometry in Figure 2.6, another geometric construct called the epipolar line can be described. The epipolar line is the line between the image epipole and a corresponding point. It can be thought of as an image of the projection of the corresponding point in the opposing image. For example, in Figure 2.6, $\mathbf{l}'$ is the image of the ray

projected from $\mathbf{C}$ through $\mathbf{x}$. This relationship is called the epipolar line correspondence condition.

Given that there is some relationship between $\mathbf{x}$ and $\mathbf{x}'$ using the epipolar line correspondence, it can be inferred that there is a homography that relates the two points.

$$\mathbf{x} = \mathbf{H}\mathbf{x}' \tag{2.13}$$

The cross product of $\mathbf{x}'$ and $\mathbf{e}'$ represents the epipolar line $\mathbf{l}'$, substituting this into Equation 2.13 gives,

$$\mathbf{l} = \left[\mathbf{e}'\right]_\times \mathbf{H}\mathbf{x}' \tag{2.14}$$

The fundamental matrix $\mathbf{F}$ here is defined as [25],

$$\mathbf{F} = \left[\mathbf{e}'\right]_\times \mathbf{H} \tag{2.15}$$

Fundamental Matrix Properties

The fundamental matrix $\mathbf{F}$, is a matrix of rank two with seven degrees of freedom. It has nine elements defined up to a single scale, which removes one degree of freedom. The fundamental matrix also satisfies the constraint,

$$\det\left(\mathbf{F}\right) = 0 \tag{2.16}$$

which removes the last degree of freedom. The fundamental matrix has a number of properties that can be exploited for three-dimensional reconstruction. The first property comes by substituting $\mathbf{F}$ into Equation 2.14, which results in the algebraic definition of the epipolar line correspondence condition,

$$\mathbf{l} = \mathbf{F}\mathbf{x}' \tag{2.17}$$

This equation is useful for image correspondence calculation, which will be discussed in Section 2.2.4. Another useful property is the fundamental matrix correspondence condition. Using the line property discussed in Section 2.1.1, a relationship between a point and the epipolar line it falls on can be described using,

$$\mathbf{x}^T\mathbf{l} = 0 \tag{2.18}$$

Substituting Equation 2.17 into Equation 2.18 gives the equation for the correspondence condition,

$$\mathbf{x}^T\mathbf{F}\mathbf{x}' = 0 \tag{2.19}$$

All corresponding points between two images described by $\mathbf{F}$ must follow this condition. This condition can be used as a model for solving for the fundamental matrix as well as using it as a model for optimization.

2.1.4 Fundamental Matrix Derivation

The fundamental matrix can be derived from two images in two ways. Each way requires having prior knowledge of the image properties. The first method derives the $\mathbf{F}$ matrix from known image correspondences. The second method derives the $\mathbf{F}$ matrix from the known camera projection matrices.

Using Correspondence

The fundamental matrix can be calculated using the correspondence condition shown in Equation 2.19. A single equation can be formed by vectorizing the fundamental matrix into a 1 by 9 vector. Doing this transforms Equation 2.19 into Equation 2.20.

$$x_1x_1'f_{11} + x_1x_2'f_{12} + x_1f_{13} + x_2x_1'f_{21} + x_2x_2'f_{22} + x_2f_{23} + x_1'f_{31} + x_2'f_{32} + f_{33} = 0 \tag{2.20}$$

Here $\mathbf{x}$ and $\mathbf{x}'$ and represented using the $\mathbf{x} = [x_1, x_2, x_3]^T$ form, and making the assumption that $x_3 = 1$. The fundamental matrix is represented using f_{nm} notation. Equation 2.20 can be formed into a linear system of the form $\mathbf{Ax} = 0$, which can be solved using values for $\mathbf{x}$ and $\mathbf{x}'$. The form of the linear system is,

$$\mathbf{Af} = \begin{bmatrix} x_{1_1}x'_{1_1} & x_{1_1}x'_{2_1} & x_{1_1} & x_{2_1}x'_{1_1} & x_{2_1}x'_{2_1} & x_{2_1} & x'_{1_1} & x'_{2_1} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{1_n}x'_{1_n} & x_{1_n}x'_{2_n} & x_{1_n} & x_{2_n}x'_{1_n} & x_{2_n}x'_{2_n} & x_{2_n} & x'_{1_n} & x'_{2_n} & 1 \end{bmatrix} \mathbf{f} = \mathbf{0}. \tag{2.21}$$

Matrix $\mathbf{A}$ is formed for all points 1 to n. The matrix $\mathbf{A}$ is of rank 8 or less, therefore, to solve for the fundamental matrix at least 8 corresponding points have to be used. A least squares solution can be found for the fundamental matrix by using singular value

decomposition (SVD) [25].

SVD is an efficient way of calculating a least squares solution while constraining the solution to have a magnitude of 1. The solution is found as the right-hand singular vector in the SVD output which corresponds to the smallest singular value. In other words, if $SVD(\mathbf{A}) = \mathbf{U\Sigma V}^T$, the solution for f is the last column of $\mathbf{V}$.

Since the Fundamental matrix has only seven degrees of freedom, it is possible to estimate the matrix using seven point correspondences. The solution to $\mathbf{Af} = 0$ will have a two-dimensional null space of the form,

$$\mathbf{F} = \alpha\mathbf{F_1} + (1-\alpha)\mathbf{F_2} \tag{2.22}$$

where the matrices $\mathbf{F_1}$ and $\mathbf{F_2}$ correspond to the last two columns of $\mathbf{V}$. Using the determinant constraint of the fundamental matrix (Equation 2.16), with Equation 2.22 the following can be created,

$$\det\left(\alpha\mathbf{F_1} + (1-\alpha)\mathbf{F_2}\right) = 0 \tag{2.23}$$

The variable α can be solved for and will have three roots. The non-complex roots can be substituted back into Equation 2.22 to solve for the matrix $\mathbf{F}$. This will result in one to three possible fundamental matrices.

These equations are based on perfect image correspondence, and in real-world applications the image correspondence often has a number of outliers. The fundamental matrix can be robustly estimated in this scenario using an model fitting algorithm which estimates a model in the presence of outliers, such as RANSAC [17]. The usage of RANSAC for estimation of the matrix $\mathbf{F}$ is discussed in Section 2.4.1.

Using Cameras

If the camera projection matrices for each image, as defined in Equation 2.12, are known, then the fundamental matrix can be derived using this information. Referring back to Figure 2.6, the world point, $\mathbf{X}$, and the image point, $\mathbf{x}$, are related through the camera matrix, $\mathbf{P}$, as defined in Equation 2.12, which can be reformed as,

$$\mathbf{X} = \mathbf{P}^{\dagger}\mathbf{x} \tag{2.24}$$

Where $\mathbf{P}^{\dagger}$ represents the pseudo-inverse of the camera matrix $\mathbf{P}$. The relationship of image point $\mathbf{x}'$ and $\mathbf{X}$ can also be defined using Equation 2.12, shown as,

$$\mathbf{x}' = \mathbf{P}'\mathbf{X} \tag{2.25}$$

where $\mathbf{P}'$ represents the camera projection matrix for the camera corresponding to the camera center $\mathbf{C}'$. The world point defined in Equation 2.24 can be substituted into Equation 2.25, yielding,

$$\mathbf{x}' = \mathbf{P}'\mathbf{P}^{\dagger}\mathbf{x} \tag{2.26}$$

Due to scale ambiguity, Equation 2.24 actually represents a family of possible solutions for $\mathbf{X}$. This can be thought of as the ray of possible points $\mathbf{X}(\lambda)$ projected from $\mathbf{x}$, where λ is the unknown scale,

$$\mathbf{X}(\lambda) = \mathbf{P}^{\dagger}\mathbf{x} + \lambda\mathbf{C} \tag{2.27}$$

where $\mathbf{C}$ is the camera center of the camera associated with $\mathbf{P}$. Two known points can come from this parameterization: $\mathbf{P}^{\dagger}\mathbf{x}$ at $\lambda = 0$, and $\mathbf{C}$ at $\lambda = \infty$. There two points can be imaged by the camera associated with $\mathbf{P}'$, as,

$$\mathbf{P}'\mathbf{P}^{\dagger}\mathbf{x} \tag{2.28}$$

$$\mathbf{P}'\mathbf{C} \tag{2.29}$$

Using concepts discussed in Section 2.1.3, it is known that the epipolar line $\mathbf{l}'$ can be defined as the cross product between the epipole $\mathbf{e}'$ and the a point $\mathbf{x}'$, shown in Equation 2.30, using the skew-symmetric representation of the cross product,

$$\mathbf{l}' = \left[\mathbf{e}'\right]_{\times}\mathbf{x}' \tag{2.30}$$

The epipolar $\mathbf{e}'$ is the image of $\mathbf{C}$, calculated with Equation 2.29. The point $\mathbf{x}'$ is calculated using Equation 2.28, yielding,

$$\mathbf{l}' = \left[\mathbf{P}'\mathbf{C}\right]_{\times}\mathbf{P}'\mathbf{P}^{\dagger}\mathbf{x} \tag{2.31}$$

Using the epipolar line relationship defined in Equation 2.17, a definition for the fundamental matrix can be derived from Equation 2.31,

$$\mathbf{F} = \left[\mathbf{P}'\mathbf{C}\right]_{\times}\mathbf{P}'\mathbf{P}^{\dagger} \tag{2.32}$$

This representation of the fundamental matrix can be useful when the camera projection matrices are already known and the fundamental matrix is needed.

2.1.5 Relative Camera Pose Estimation

When absolute camera information is not known or available, it is possible to estimate the camera position and orientation information relative to each other using image point correspondences. This is done by using point correspondences to solve for the essential matrix and derive the camera information through matrix decomposition.

The Essential Matrix

The essential matrix, $\mathbf{E}$, is a special case of the fundamental matrix, where the calibration information is known. Given knowledge of the calibration matrix, it is possible to apply the inverse of that matrix to the image point, yielding image points which have the camera intrinsic distortions removed. This can be applied to the fundamental matrix correspondence condition (Equation 2.19), to give the correspondence condition for the essential matrix,

$$\mathbf{x}'^T \mathbf{K}'^{-T} \mathbf{E} \mathbf{K}^{-1} \mathbf{x} = 0 \tag{2.33}$$

It would follow that the relationship between the fundamental matrix and the essential matrix can be defined as

$$\mathbf{E} = \mathbf{K}'^T \mathbf{F} \mathbf{K} \tag{2.34}$$

In order to derive the relationship between the essential matrix and the camera projection matrices, consider two cameras $\mathbf{P}$ and $\mathbf{P}'$, as defined in equations 2.35 and 2.36. The origin of $\mathbf{P}$ is the center of the world coordinate system.

$$\mathbf{P} = \mathbf{K}\,[\mathbf{I}|0] \tag{2.35}$$

$$\mathbf{P}' = \mathbf{K}'\,[\mathbf{R}|\mathbf{t}] \tag{2.36}$$

The matrices $\mathbf{K}$ and $\mathbf{K}'$ are the calibration matrices as defined in Equation 2.11. The matrix $\mathbf{I}$ is a 3x3 identity matrix. The 3x3 matrix $\mathbf{R}$ and the 3x1 vector $\mathbf{t}$ represent the rotation and translation of $\mathbf{P}'$ away from $\mathbf{P}$. The fundamental matrix can be derived

from these two cameras as shown in Equation 2.32. The camera center for $\mathbf{P}$ is defined as the center of the world coordinate system, $\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$, in homogeneous coordinates. The equation for the fundamental matrix for $\mathbf{P}$ and $\mathbf{P'}$ is shown below (the vector $\mathbf{0}$ represents a 3x1 vector of zeros).

$$\begin{aligned} \mathbf{P}^{\dagger} &= \begin{bmatrix} \mathbf{K}^{-1} \\ \mathbf{0}^T \end{bmatrix} \\ \mathbf{F} &= \left[\mathbf{P'C}\right]_{\times} \mathbf{P'P}^{\dagger} \\ \mathbf{F} &= \mathbf{K'}^{-T} \left[\mathbf{t}\right]_{\times} \mathbf{RK}^{-1} \end{aligned} \tag{2.37}$$

Using the relationship described in Equation 2.34, the essential matrix for $\mathbf{P}$ and $\mathbf{P'}$ is given by,

$$\mathbf{E} = \left[\mathbf{t}\right]_{\times} \mathbf{R} \tag{2.38}$$

Equation 2.38 can be used to estimate the camera rotation and translation through matrix decomposition [25].

One important property of the essential matrix is that it is of rank two, just like the fundamental matrix. Also, the two non-zero singular values are equal to each other. This leads to the relationship [32],

$$\mathbf{EE}^T\mathbf{E} = \frac{1}{2} tr\left(\mathbf{EE}^T\right)\mathbf{E} \tag{2.39}$$

This can be shown since the singular value decomposition of $\mathbf{E}$ is $\mathbf{E} = \mathbf{U\Lambda V}^T$, where $\mathbf{\Lambda}$ is defined as,

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \tag{2.40}$$

where λ_1,λ_2, and λ_3 are the eigenvalues. All the singular values are greater than 0, the trace of $\mathbf{EE}^T$ can be defined as,

$$tr\left(\mathbf{EE}^T\right) = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \tag{2.41}$$

The left side of Equation 2.39 can be defined in terms of the SVD by,

$$\mathbf{E}\mathbf{E}^T\mathbf{E} = \mathbf{U}\mathbf{\Lambda}^3\mathbf{V}^T \tag{2.42}$$

The n elements of $\mathbf{\Lambda}^3$ can be derived as shown below. This equation can be derived because it is known that $\lambda_1 = \lambda_2$, and $\lambda_3 = 0$, for the essential matrix.

$$\lambda_n^3 = \frac{1}{2}\left(\lambda_1^2 + \lambda_2^2 + \lambda_3^2\right)\lambda_n \tag{2.43}$$

Given this, Equation 2.41 can be factored out of Equation 2.42 to give,

$$\mathbf{E}\mathbf{E}^T\mathbf{E} = \frac{1}{2}tr\left(\mathbf{E}\mathbf{E}^T\right)\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \tag{2.44}$$

Equation 2.44 is then shown to be equivalent to Equation 2.39 as $\mathbf{E} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$.

Five-Point Solution to the Essential Matrix

Using properties of the essential matrix it is possible to estimate relative camera pose using five image point correspondences. The calibrated point correspondences $\mathbf{q}$ are related to the uncalibrated correspondences as shown in Equation 2.45.

$$\mathbf{q} = \mathbf{K}^{-1}\mathbf{x} \tag{2.45}$$

The essential matrix correspondence condition in Equation 2.33 can be vectorized and reformed giving,

$$\tilde{\mathbf{q}}^T\tilde{\mathbf{E}} = 0 \tag{2.46}$$

The vectors in Equation 2.46 are vectorized in the form.

$$\tilde{\mathbf{q}} = \begin{bmatrix} q_1q_1' & q_2q_1' & q_3q_1' & q_1q_2' & q_2q_2' & q_3q_2' & q_1q_3' & q_2q_3' & q_3q_3' \end{bmatrix}$$
$$\tilde{\mathbf{E}} = \begin{bmatrix} E_{11} & E_{12} & E_{13} & E_{21} & E_{22} & E_{23} & E_{31} & E_{32} & E_{33} \end{bmatrix}$$

The vector $\tilde{\mathbf{q}}^T$ can be put into $\mathbf{A}\mathbf{x} = 0$ form by stacking the $\tilde{\mathbf{q}}$ vectors for each correspondence, which forms a 5x9 matrix. Singular value decomposition can be used to find the null space basis vectors which solve the $\mathbf{A}\mathbf{x} = 0$ for $\mathbf{x}$. Using this, four vectors that form the basis of the right null space can be computed. These vectors, which represent $\tilde{\mathbf{E}}$, can be formed back into 3x3 matrices and used to describe $\mathbf{E}$ in a linear combination.

$$\mathbf{E} = x\mathbf{X} + y\mathbf{Y} + z\mathbf{Z} + w\mathbf{W} \tag{2.47}$$

In Equation 2.47, x, y, z, and w represent the weights for the linear combination of the 3x3 matrices $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and $\mathbf{W}$. Just like the fundamental matrix, the essential matrix can only be defined up to a scale factor because of its derivation in homogeneous coordinates. Therefore it can be assumed that one weight can be set to any value, for simplification purposes w is set to equal 1 [11].

The constraint equation shown in Equation 2.44 can be reformulated into a system of equations which can provide a method of solving for the essential matrix. The reformulation is

$$\mathbf{E}\mathbf{E}^T\mathbf{E} - \frac{1}{2}tr\left(\mathbf{E}\mathbf{E}^T\right)\mathbf{E} = 0 \tag{2.48}$$

Given that the basis vector representation of $\mathbf{E}$ is of three variables, when inserted into Equation 2.48, nine cubic polynomial functions can be extracted. Each of the nine functions correspond to an element of $\mathbf{E}$. A tenth constraint can be added by using the fact that the essential matrix is rank deficient, the determinant of the $\mathbf{E}$ must be equal to 0. These ten constraints form a system of ten equations which can be used to exactly solve for the scalar values of x, y and z [42]. These values, along with $w = 1$, are substituted back into Equation 2.47 to yield a solution for the essential matrix. The cameras rotations and translations are described in relation to the essential matrix in Equation 2.38. The following sections discuss the decomposition of the essential matrix to retrieve the rotations and translations.

Camera Pose Retrieval From the Essential Matrix

The essential matrix can be derived completely from a single camera's rotation and translation as described in Equation 2.38. This can only be the case if one camera is assumed to be at the world coordinate origin, so that the second camera is described relative to the first. These cameras are shown in Equations 2.35 and 2.36. The rotation and translation of the second camera can be decomposed from the essential matrix using SVD. Given that $\mathbf{E} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$, four possible camera matrices for $\mathbf{P}'$ can be derived, as shown in the following [25, 11],

$$\begin{aligned}
\mathbf{P'}_0 &= \left[\mathbf{UWV}^T | u_3\right] \\
\mathbf{P'}_1 &= \left[\mathbf{UWV}^T | -u_3\right] \\
\mathbf{P'}_2 &= \left[\mathbf{UW}^T\mathbf{V}^T | u_3\right] \\
\mathbf{P'}_3 &= \left[\mathbf{UW}^T\mathbf{V}^T | -u_3\right]
\end{aligned} \tag{2.49}$$

Where u_3 represents the third column of the matrix $\mathbf{U}$, and also represents the translation from $\mathbf{P}$ to $\mathbf{P'}$. The matrix $\mathbf{W}$ is an orthogonal matrix as defined by

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.50}$$

The equations shown in 2.49 represent four possible orientations that the camera $\mathbf{P'}$ could take. $\mathbf{P'}_0$ and $\mathbf{P'}_1$ are related by a reverse translation along the baseline between $\mathbf{P}$ and $\mathbf{P'}$, as are $\mathbf{P'}_2$ and $\mathbf{P'}_3$. The cameras $\mathbf{P'}_0$ and $\mathbf{P'}_2$ are related by a 180 degree rotation about the baseline [25]. Only one transformation of $\mathbf{P'}$ is the correct one.

The only correct orientation of $\mathbf{P'}$ is the one in which the points being viewed correspond to a three-dimensional point which is in front of both cameras. The other three orientations will represent a three-dimensional point which is behind one or both of the cameras. This concept is called cheirality, and can be enforced using the cheirality inequalities [25, 11]. Given a pair of corresponding points, a three-dimensional point, $\mathbf{X}$, can be found using methods described in Section 2.3.2 with $\mathbf{P}$ and $\mathbf{P'}_0$. The cheirality inequalities state that if $X_3X_4 < 0$, then the point is behind the first camera, if $(\mathbf{P'}_0X_3)\,X_4 < 0$ then the point is behind the second camera. If both the previously mentioned inequalities are greater than zero, then the point is in front of both cameras and $\mathbf{P'}_0$ is the correct orientation. If both the inequalities are less than zero, that corresponds to $\mathbf{P'}_1$ and that camera is used. If $X_3X_4\,(\mathbf{P'}_0X_3)\,X_4 < 0$, then the rotated case $\mathbf{P'}_2$ is used, and the calculation is done again. If the inequalities are both less than zero again, then $\mathbf{P'}_3$ is the correct configuration.

2.2 Feature Detection, Description, and Matching

The first critical step in all Structure from Motion (SfM) processes is identifying points of interest in each image that will be tracked between each image. There are a number

of ways to do this, this section will discuss a sample of feature detection algorithms that appear in SfM workflows. The scale invariant feature transform (SIFT) is a widely used feature detection and description algorithm within the computer vision community. Affine-SIFT is a feature detection, description, and matching algorithm which tries to add affine invariance to the SIFT algorithm, an issue that arises in wide-baseline image matching. DAISY is a method of feature description which also tries to attack the issue of wide-baseline matching. Finally, the patch-based feature detection and matching method used in the Patch-based Multi-view Stereo (PMVS) algorithm is presented.

2.2.1 SIFT

SIFT Feature Detection and Description

The scale invariant feature transform is one of the most used feature detector and descriptor algorithms used in the field of computer vision [36], it is also one of the earliest developed. SIFT attempts to detect and describe features in a space that is invariant to rotation, translation, and scale [13]. In order to provide invariance to scale, the image is converted into a scale-space using Gaussian convolutions. The actual feature detection is done by taking the difference between each scale-space image, essentially performing a difference of Gaussians (DoG) filter. The DoG filter is a known feature detector which provides extrema along image edges. The scale space conversion and DoG filter is illustrated in Figure 2.7. Features are identified by searching the image and scale space for local extrema so that features within the image at different scales are detected.

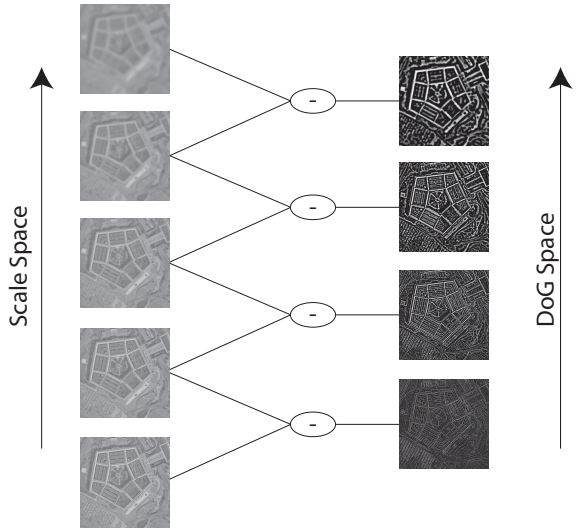


Figure 2.7: *SIFT calculates the position of the features in x,y, and scale space (convolved with Gaussians) for each image. This is done by taking the difference between each scale-space image. The resulting space provides extrema along image edges at different scales. Features are detected within each image and between each scale.*

The initial detection provides a large number of features, many of which are poor. This results since the DoG filter will provide high response along poorly defined lines, and in areas of low contrast due to noise. Poorly defined lines are detected by comparing the horizontal and vertical principle curvatures, a strong line will show low differences between principle curvatures [13]. Features along poorly defined lines as well as in low contrast areas are discarded.

In order to achieve rotational invariance, the rotation of the feature must be quantified. This is done by calculating a gradient value and magnitude for each feature. The orientation and magnitude are calculated using Equation 2.51 and 2.52.

$$m\left(x,y\right)=\sqrt{\left(I\left(x+1,y\right)-I\left(x-1,y\right)\right)^{2}+\left(I\left(x,y+1\right)-I\left(x,y-1\right)\right)} \tag{2.51}$$

$$\Theta\left(x,y\right)=tan^{-1}\left(\frac{I\left(x,y+1\right)-I\left(x,y-1\right)}{I\left(x+1,y\right)-I\left(x-1,y\right)}\right) \tag{2.52}$$

Where $I\left(x,y\right)$ represents the image intensity at pixel position $\left(x,y\right)$. The orientation and magnitude are calculated for all neighboring pixels for a detected feature. These

orientations are then formed into a thirty-six bin histogram which is used to find the dominant orientation for each feature, defined by the largest response in the histogram. For histograms with multiple maxima a separate feature is generated for each one. With knowledge of the dominant orientation and scale of each feature, the feature descriptor can be derived for each feature.

The feature descriptor is essentially a description of the orientation of the area around the detected feature. A 16-by-16 region around the feature is looked at in the scale-space of the detected feature, as shown in Figure 2.8. The grid is broken up into smaller 4-by-4 pixel regions, and an 8-bin orientation histogram is calculated for each region in the same fashion as the 36-bin orientation histogram, using Equations 2.51 and 2.52. The orientation is calculated relative to the dominant orientation of the feature. The histograms for each region are concatenated resulting in a scale, translation, and rotation invariant feature descriptor.

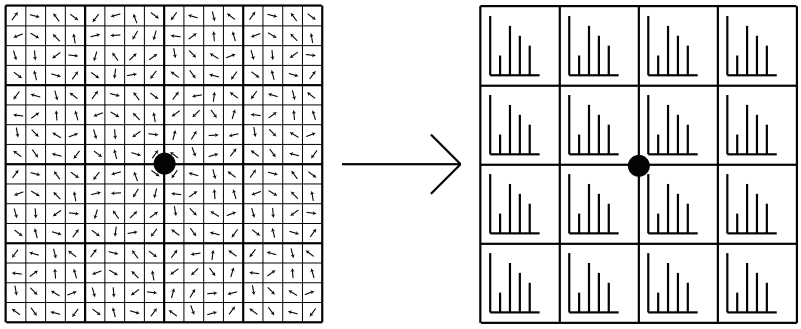


Figure 2.8: *The SIFT feature is calculated over a 16-by-16 region around each detected point. The gradient direction and magnitude is calculated for each bin, the bin size is determined by the scale of the feature. The gradient angles are binned into 8-bin histograms for each 4-by-4 area, and concatenated to form a 128-element feature vector.*

SIFT Feature Matching

Once SIFT features are calculated for a set of images, the features can be matched to find corresponding image points. This matching process is simply a brute-force nearest-neighbor matching algorithm [13]. Each SIFT feature is 128-elements in length. A dot product between each feature in one image and all the features in another image is calculated to determine how similar they are in a 128-dimensional space. This process can

be optimized by only taking matches that are found when matching from one image to the other, and then in reverse. Figure 2.9 shows an example of two images with matching SIFT features. While this provides an estimate of feature correspondence, there is still a significant amount of error that can be removed using optimization techniques which will be discussed in Section 2.4.1.

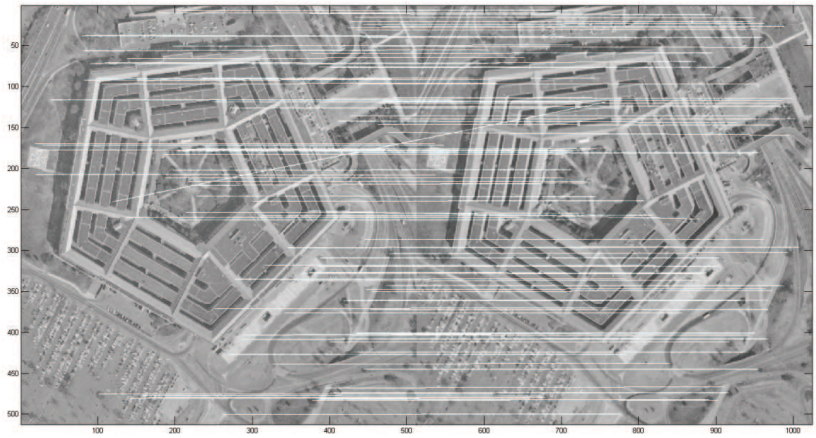


Figure 2.9: *The SIFT matching process can find a very large number of matches, however, not all of the matches will be good.*

2.2.2 Affine-SIFT

While SIFT can handle images that differ by rotation, scale, and translation, it poorly handles images which relate by a projective transformation. Affine-SIFT attempts to modify the SIFT algorithm to perform well on images with an affine distortion. This is done through a computationally rigorous process of simulating image tilting around the image x and y axis. Each simulation is processed with the original SIFT algorithm, providing a rotation, scale, and translation invariant description of the image. The original SIFT matching process is used to match features at different tilts. Features that are matched consistently between the simulated images are kept as good matches [48]. Figure 2.10 shows an example of A-SIFT matching versus SIFT matching.

Affine-SIFT proves to have significantly better matching results than SIFT with images that have a projective relationship. However, this comes at the expense of computational

processing time [48]. Images are sub-sampled in order to speed this process up, and the algorithm can also be parallelized. The computational complexity of A-SIFT is higher than that of SIFT.

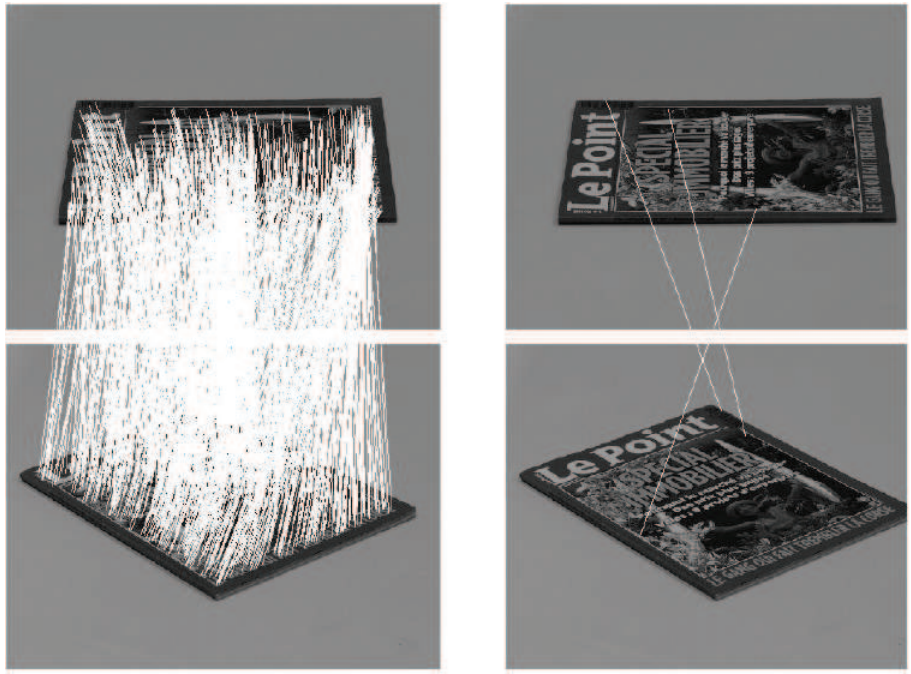


Figure 2.10: *A comparison between A-SIFT and SIFT with image matching between a set of images that have an extreme affine transform relationship. The A-SIFT results are shown on the left and the SIFT results are shown on the right [48].*

2.2.3 DAISY

When matching images that exhibit a large baseline, SIFT tends to have a difficult time finding correspondences [64]. As the baseline widens between the images, the transform that relates the two images tends to become more projective. The goal of the DAISY feature descriptor is to efficiently describe a region around every feature that is invariant to this type of transform [65]. The DAISY feature is just a descriptor, so it can be combined with any type of feature detector.

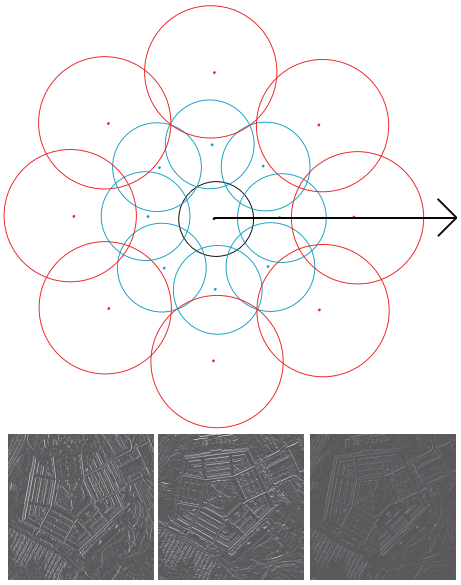


Figure 2.11: *A representation of the DAISY feature descriptor which is applied to each orientation map. Each circle represents a Gaussian kernel with the size proportional to the Gaussian scale. The kernels radiate outwards in a set number of iterations. This feature is calculated on each orientation map. [64].*

The DAISY descriptor is similar to that of the SIFT descriptor described in Section 2.2.1, in that it attempts to quantify the orientation of the region around a given feature. This is done by generating orientation maps instead of orientation histograms. These maps are generated by calculating the gradient intensity in a specific direction for all chosen directions. Then each orientation map is convolved with a series of Gaussian functions of varying scale. The different Gaussian convolutions represent the scale intensity in each direction, thus providing a scale invariance to the descriptor. The descriptor is normalized so it can be matched between images. Figure 2.11 shows the layout of the DAISY feature descriptor. There are a number of parameters that can be altered in the descriptor. These include the radius of the whole descriptor, the number of orientation samples that are collected in each direction, the number of samples that are collected in a single orientation layer, and the number of orientation maps [64].

Invariance in the DAISY descriptor is provided by the scaling of the Gaussian functions as well as the characterization of the orientation intensities. While SIFT does attempt to quantify the orientation intensity, DAISY has proven to be a better and faster descriptor

for features which differ significantly in orientation and scale [65].

2.2.4 Epipolar Line Matching

Given an initial correspondence with a feature detection and matching algorithm, such as SIFT, A-SIFT, or DAISY, the fundamental matrix can be calculated using RANSAC with the seven-point fundamental matrix algorithm, as described in Section 2.1.4. This allows for further exploitation of geometry in order to generate a denser correspondence. This dense correspondence can be used to generate a denser reconstruction. This method is based on the epipolar line constraint described in Equation 2.17. Every point in a given image will correspond to an epipolar line in the corresponding image, related by the fundamental matrix F.

The epipolar line constraint serves as a way to minimize the correspondence search between images from the entire image to just a single line in the image. In order to further minimize the search area, this method requires a user-given region of interest (ROI) in each image over the target of interest [51]. Figure 2.12 shows an example of this process (the user given ROIs are shown as the red boxes in the image).

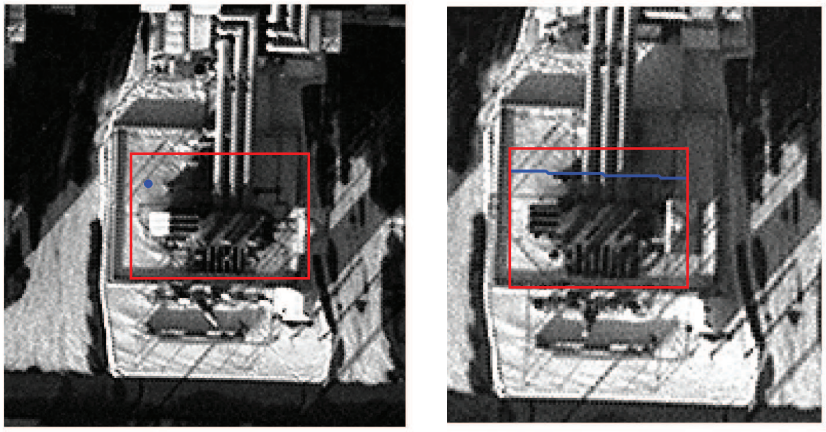


Figure 2.12: *An example of using the epipolar line constraint along with user provided ROIs to minimize a correspondence search for two images. This search is performed between every point in the user provided ROI.*

A region around the point in the original image is then matched along the epipolar line in the corresponding image. This is done by taking a 3x3 region centered around the point and vectorizing the pixels. A vector is generated for every point along the corresponding epipolar line as well as two pixels above and below the line. The vectors are matched between the two images using a brute-force matching by finding the smallest angle between the generated vectors using the dot product,

$$\theta = \cos^{-1}\left(\frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}\right) \tag{2.53}$$

where $\mathbf{a}$ and $\mathbf{b}$ represent the vectorized regions between the two images. This process is repeated for every point in the original image's ROI, so that every single point has a match [51].

There are a few errors which arise in this process and which can be overcome by pre-processing the imagery and post-processing the correspondences. One error is that this method does not take into account any rotation within the imagery. In order to remove the rotation invariance, the images have to be rectified so that their epipolar lines are parallel with each other. This requires transforming the corresponding image so that the epipole goes to infinity [25]. Another error which arises is that this process will have some error in matching due to the fundamental matrix estimation. One way to reduce this error is to perform this calculation, then perform the calculation again using the corresponding image as the original image. The correspondences which match in both directions should be kept. Figure 2.13 shows an example of one image ROI matching to a corresponding image.

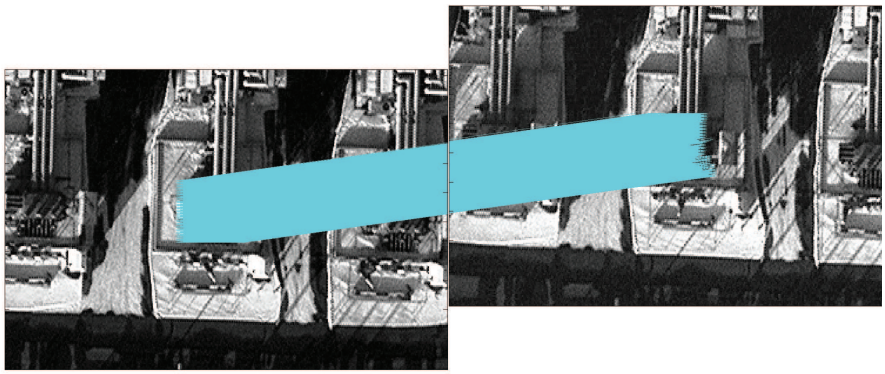


Figure 2.13: *A user-selected ROI being matched between an original and corresponding image. The correspondence is very dense as it was performed for every pixel in each ROI.*

2.2.5 Patch-Based Model

In many SfM applications, a very dense point reconstruction is desired for modeling purposes. This process exploits epipolar geometry in a similar fashion to the one described in Section 2.2.4. This section describes the dense image correspondence method used in the PMVS algorithm described in Section 3.1. This process attempts to match every pixel within an image to another image using a patch-based region growing method, by taking advantage of epipolar geometry constraints provided by knowledge of each image camera projection matrix [18].

There are two major steps in this process, the first step is the initial detection of features. This is done through a combination of two feature detectors that are designed to detect blob and corner features. The first feature detector is the difference of Gaussian (DoG) filter. This is the same filter that is used in SIFT. Here, it is used to detect edges in all directions. The filtered image is then passed to the Harris corner detector which will find only the directional changes along the detected edges. A square grid is put over the whole image, and each grid element which contains a detected local maxima is labeled as a feature. This process is shown in Figure 2.14

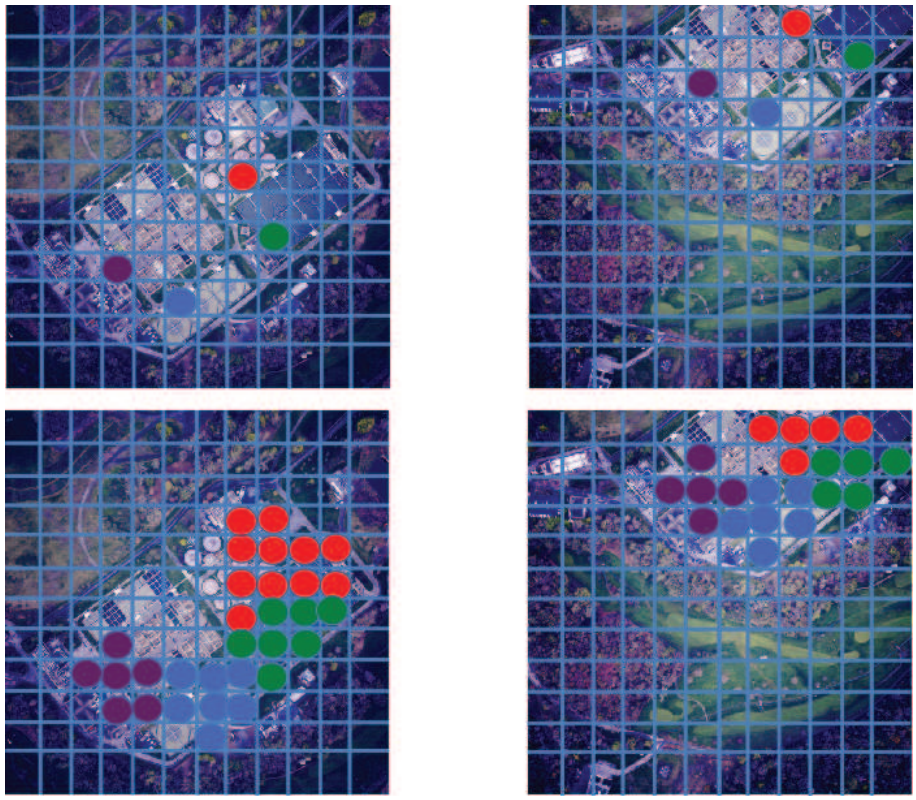


Figure 2.14: *The PMVS feature detection and matching process; The images at the top show the image grid with initial features detected and matched using the epipolar line matching method. The images at the bottom show the expansion process in which patches are expanded and optimized based on information from the nearest reconstructed patch.*

An initial feature matching is done after the inital detection of features. This is where the knowledge of each camera's projection matrix becomes valuable. As shown in Section 2.1.4, the fundamental matrix between two images can be derived from their projection matrices. The epipolar line (Section 2.1.3) concept becomes useful here. Each detected feature in one image will correspond to an epipolar line in the other image. The line can be found by deriving the fundamental matrix from the known camera projection matrices as described in Section 2.1.4. The epipolar line is calculated, then all features that fall within two pixels of that line are collected. These features are considered potential matches for the original feature.

The potential matches are tested to see which features reconstruct in the best manner.

The patch model is used here to determine the best reconstruction. Each feature is triangulated using the feature position and known camera information, this denotes the center of the patch. The normal to the patch is calculated as the vector between the calculated center of the patch and the known center of the corresponding camera. This normal is then compared to the vector between the calculated center of the patch, and the known center of the original camera. The vectors that differ the least, denoted by a certain qualifying threshold, are chosen to be the matching pair [18].

In order to generate a dense reconstruction, every grid element which does not contain a feature is then reconstructed by using information from the nearest reconstructed patch. The patch center and normal are initialized from the nearest reconstructed patch, and are then refined. The refining process minimizes the re-projection error between the patch center and the center of the empty image cell, by adjusting the geometric position of the initialized patch. This process produces a very dense reconstruction.

This method has proven to be extremely effective in dense reconstruction, provided that an accurate representation of a camera system can be obtained for the calculation of the camera projection matrices [18].

2.3 Reconstruction Techniques

The previous feature detection, description, and matching algorithms provide a image-to-image correspondence for each matched image. The next step in the SfM process is to use those correspondences to reconstruct a three-dimensional point in the world-coordinate system. There are a number of different methods to do this reconstruction. This section will focus on methods which have been tested and used within this work.

2.3.1 Photogrammetric Approach

For many decades the photogrammetry community have developed geometric methods for triangulating three-dimensional points. These methods are designed specifically to work with aerial imagery. An advantage that aerial images have in this process is that the camera position data is often available for each image. Therefore no prior estimation of camera positioning needs to take place. Figure 2.15 shows the basic geometry for a set of stereo images taken from an airborne platform.

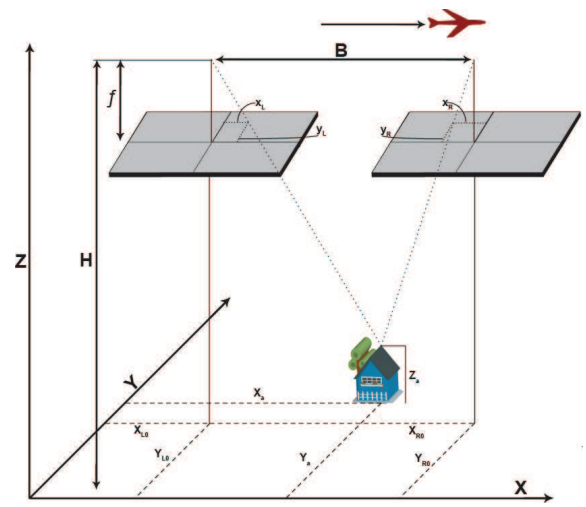


Figure 2.15: *The photogrammetric approach for point reconstruction uses the parallax equations. The geometry for these equations assumes that each camera's focal plane are coplanar. It also assumes the flight line follows the x-dimension of the imagery.*

The three-dimensional coordinates X,Y and Z can be calculated using parameters shown in Figure 2.15. The baseline, B, is the distance between the camera centers for each image. The flying height is represented as H, and the focal length as f. The image coordinates are represented by x and y. The subscripts l and r refer to the left and right images, respectively. The following Equations 2.54, 2.55, and 2.56 describe the calculations for finding the X,Y and Z coordinates. These equations can be derived using the similar triangles found in Figure 2.15 [73].

$$X = \frac{Bx_l}{x_l - x_r} \tag{2.54}$$

$$Y = \frac{By_l}{y_l - y_r} \tag{2.55}$$

$$Z = H - \frac{Bf}{x_l - x_r} \tag{2.56}$$

These equations make two major assumptions about the data. The first is that the flight line is along the horizontal (x) dimension. The second assumption is that the camera focal plane is flat and level to the aforementioned flight line. Imagery taken on an aerial

platform will never conform to both of these assumptions, so some coordinate modifications have to be made in order to force the data to conform to these assumptions. The flight line direction assumption can be simply corrected for by transforming the coordinate system so that the horizontal dimension falls along the recorded flight line.

In order to correct for the flat and level assumption, the images must be transformed such that they appear to be level in relation to each other. This can be accomplished using the recorded camera position information to project the image onto a flattened image plane, as shown in Figure 2.16 [38].

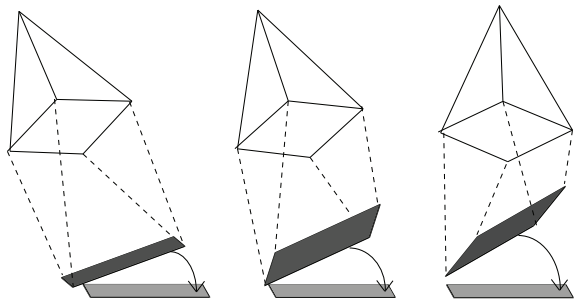


Figure 2.16: *In order to compensate for the uneven focal planes, each image is projected to a new focal plane by reversing the measured pitch and roll of the aircraft, and then reorienting the x-axis for each image to be along the flight path.*

The roll and pitch of the aircraft is recorded for each image frame, along with the camera center Using the focal length, the distance from the camera center to image plane can be calculated. The new image plane is calculated using the pitch and roll of the aircraft, and the original image is projected into this new frame.

2.3.2 Linear Triangulation

This triangulation method comes from the linear manipulation of Equation 2.12. The goal of this process is to form an $\mathbf{AX} = 0$ linear equation that can be solved using SVD, in the same manner that is described in Section 2.1.4. In homogeneous coordinates, the cross product of a point with itself must equal zero, given the known relationship described in equation 2.12, the following equation can be formed.

$$[\mathbf{x}]_\times \mathbf{PX} = 0 \tag{2.57}$$

Equation 2.57 can be expanded to form three separate equations, using the definitions $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ and $\mathbf{X} = \begin{bmatrix} X_1 & X_2 & X_3 & 1 \end{bmatrix}^T$, shown here,

$$x_1 \left(p_{31}X_1 + p_{32}X_2 + p_{33}X_3 + p_{34}\right) - \left(p_{11}X_1 + p_{12}X_2 + p_{13}X_3 + p_{14}\right) \tag{2.58}$$

$$x_2 \left(p_{31}X_1 + p_{32}X_2 + p_{33}X_3 + p_{34}\right) - \left(p_{21}X_1 + p_{22}X_2 + p_{23}X_3 + p_{24}\right) \tag{2.59}$$

$$x_1 \left(p_{21}X_1 + p_{22}X_2 + p_{23}X_3 + p_{24}\right) - x_2 \left(p_{11}X_1 + p_{12}X_2 + p_{13}X_3 + p_{14}\right) \tag{2.60}$$

Where p_{nm} represents nth row and the mth column of the 3x4 camera projection matrix $\mathbf{P}$. Equation 2.60 is linearly dependent on Equation 2.58 and 2.59, and for the purpose of forming a linear equation can be removed [25] The $\mathbf{A}$ matrix can be formed using Equations 2.58 and 2.59 as shown in Equation 2.61.

$$\mathbf{A} = \begin{bmatrix} x_1 \begin{bmatrix} p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}^T - \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \end{bmatrix}^T \\ x_2 \begin{bmatrix} p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}^T - \begin{bmatrix} p_{21} & p_{22} & p_{23} & p_{24} \end{bmatrix}^T \\ x'_1 \begin{bmatrix} p'_{31} & p'_{32} & p'_{33} & p'_{34} \end{bmatrix}^T - \begin{bmatrix} p'_{11} & p'_{12} & p'_{13} & p'_{14} \end{bmatrix}^T \\ x'_2 \begin{bmatrix} p'_{31} & p'_{32} & p'_{33} & p'_{34} \end{bmatrix}^T - \begin{bmatrix} p'_{21} & p'_{22} & p'_{23} & p'_{24} \end{bmatrix}^T \end{bmatrix} \tag{2.61}$$

Where x'_n and p'_{nm} represent a corresponding image frame. This can be formed into the linear equation shown in Equation 2.62, and solved using the SVD methods previously mentioned.

$$\begin{bmatrix} x_1 \begin{bmatrix} p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}^T - \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \end{bmatrix}^T \\ x_2 \begin{bmatrix} p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}^T - \begin{bmatrix} p_{21} & p_{22} & p_{23} & p_{24} \end{bmatrix}^T \\ x'_1 \begin{bmatrix} p'_{31} & p'_{32} & p'_{33} & p'_{34} \end{bmatrix}^T - \begin{bmatrix} p'_{11} & p'_{12} & p'_{13} & p'_{14} \end{bmatrix}^T \\ x'_2 \begin{bmatrix} p'_{31} & p'_{32} & p'_{33} & p'_{34} \end{bmatrix}^T - \begin{bmatrix} p'_{21} & p'_{22} & p'_{23} & p'_{24} \end{bmatrix}^T \end{bmatrix} \mathbf{X} = 0 \tag{2.62}$$

One advantage of this method is that it can be easily extended to N corresponding cameras, shown in Equation 2.63.

$$\begin{bmatrix} x_1^0 \begin{bmatrix} p_{31}^0 & p_{32}^0 & p_{33}^0 & p_{34}^0 \end{bmatrix}^T - \begin{bmatrix} p_{11}^0 & p_{12}^0 & p_{13}^0 & p_{14}^0 \end{bmatrix}^T \\ x_2^0 \begin{bmatrix} p_{31}^0 & p_{32}^0 & p_{33}^0 & p_{34}^0 \end{bmatrix}^T - \begin{bmatrix} p_{21}^0 & p_{22}^0 & p_{23}^0 & p_{24}^0 \end{bmatrix}^T \\ x_1^1 \begin{bmatrix} p_{31}^1 & p_{32}^1 & p_{33}^1 & p_{34}^1 \end{bmatrix}^T - \begin{bmatrix} p_{11}^1 & p_{12}^1 & p_{13}^1 & p_{14}^1 \end{bmatrix}^T \\ x_2^1 \begin{bmatrix} p_{31}^1 & p_{32}^1 & p_{33}^1 & p_{34}^1 \end{bmatrix}^T - \begin{bmatrix} p_{21}^1 & p_{22}^1 & p_{23}^1 & p_{24}^1 \end{bmatrix}^T \\ \vdots \\ x_1^N \begin{bmatrix} p_{31}^N & p_{32}^N & p_{33}^N & p_{34}^N \end{bmatrix}^T - \begin{bmatrix} p_{11}^N & p_{12}^N & p_{13}^N & p_{14}^N \end{bmatrix}^T \\ x_2^N \begin{bmatrix} p_{31}^N & p_{32}^N & p_{33}^N & p_{34}^N \end{bmatrix}^T - \begin{bmatrix} p_{21}^N & p_{22}^N & p_{23}^N & p_{24}^N \end{bmatrix}^T \end{bmatrix} \mathbf{X} = 0 \tag{2.63}$$

This method is very simple to implement, however it is prone to error when $X_4 \neq 1$, or requires a projective reconstruction [25].

2.4 Optimization Techniques

As with many algorithms, actual implementation of concepts requires dealing with noisy and difficult data. In the case of SfM, the feature correspondence data contains the error which needs to be minimized. Many of the previously described algorithms depend on this data as input, and are best solved using some form of optimization routine. This section will discuss some of the optimization routines which are often used to generate quality output from noisy input.

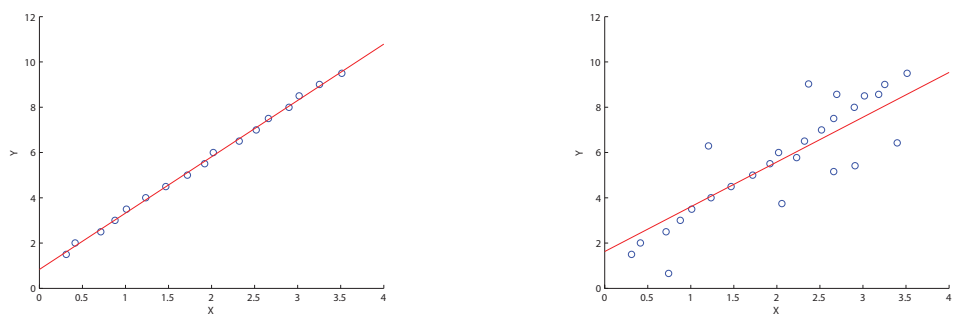
2.4.1 Feature Matching Optimization Using RANSAC

The feature matching problem discussed in Section 2.2 often contains large numbers of false matches after the initial feature matching process. In order to find false matches, a model which describes the correspondence relationship can be fit. This model is shown in Equation 2.19, and is known as the fundamental matrix correspondence condition. This process will not only find false matches in a feature matching set, it will also effectively derive the fundamental matrix from correspondences, as described in Section 2.1.4.

The model-fitting algorithm called RANdom SAmple Consensus (RANSAC) will be used as the optimization routine to achieve the model fitting. RANSAC is an algorithm that is designed to find the best fitting model parameters in the presence of outliers. In

the feature matching case, the outliers will be the false matches. As indicated in its name, RANSAC uses a random process to iteratively find the best fitting model [17].

Consider a simple line fitting model, $y = Ax + B$, which contains two parameters; the slope A and the intercept B. This algorithm can be exactly solved with at least two data points using a least squares line fitting method. As the noise in the data increases the ability to accurately model the line using standard model fitting fails. Figure 2.17-a shows a simple least squares line fit to data which contains very little noise, Figure 2.17-b shows what happens to this least squares fit when noise is added to the system. RANSAC attempts to overcome this problem, by fitting a model to the data while simultaneously removing the outliers from the dataset.



(a) *No Outliers* (b) *Some Outliers*

Figure 2.17: *Simple linear regression can be significantly impacted in the presence of outliers. Both lines here are fit with standard linear regression.*

The RANSAC algorithm consists of essentially three steps. The first step is to randomly select the minimum number of points required to solve for the model parameters. In the case of line fitting, this would be two points. The next step is to use those points to solve for the parameters and generate a potential model. The third step is to take that potential model and determine from the set of all data points, how many points fit in the model within a predefined tolerance measure. This process is repeated a predefined number of times until a model with enough inliers has been found or the process has repeated a set number of times.

It is important to properly set the predefined thresholds based on the input data. The distance tolerance measure can be set based on any type of assumed distribution. For example, for the line fitting algorithm, a Gaussian distributions is assumed and a distance

threshold is computed. The distance measure is the square of the Euclidean distance between the points, and the final distribution is a sum of squared Gaussian variables. This is modeled using a χ^2_{DOF-1} distribution, where the DOF is equal to the number of input parameters to the model, in this case, two. The probability that a random variable is less than a given variable is modeled using the cumulative distribution function [47]. The inverse cumulative distribution function can be used to find a factor of the variance of the data, σ^2, which will be used as the threshold. This is defined as,

$$\tau = F^{-1}_{DOF-1}(\alpha)\,\sigma^2 \tag{2.64}$$

where F^{-1}_{DOF-1} is the inverse cumulative distribution function. The probability α is usually chosen to be 95%, so that an incorrect rejection of an inlier only happens 5% of the time [25].

Another threshold required is the total number of expected inliers. This requires making an assumption about the proportion of the data which contains outliers. It is best to choose a conservative estimate, such as 20% or 30%. The stopping threshold can then be defined as shown in Equation 2.65 [25], where ϵ is the assumed proportion of the data which contains outliers and n is the size of the data.

$$T = (1-\epsilon)\,n \tag{2.65}$$

The maximum number of samples can also be defined using probability. The probability of selecting all data points within τ is w^n, where w is the probability of selecting one data point within τ. To ensure with a probability of p that at least one selection, k, contains all inliers, the following equation can be stated [25].

$$(1-w^n)^k = (1-p) \tag{2.66}$$

Equation 2.66 can be solved for k,

$$k = \frac{\log(1-p)}{\log(1-w^n)} \tag{2.67}$$

where k is the number of samples required. This number can be updated iteratively within the RANSAC algorithm by defining w as the ratio of inliers to the total number of points. Algorithm 1 shows the full RANSAC algorithm in pseudocode. Figure 2.18 shows an example of a line fit to the data in Figure 2.17-b using RANSAC.

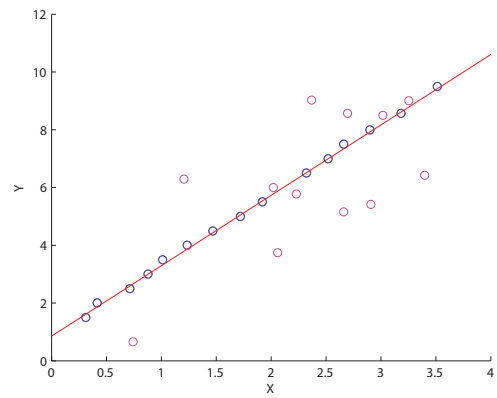


Figure 2.18: *The data in Figure 2.17-b fit with a line using RANSAC. The points in magenta were chosen by RANSAC as the outliers. RANSAC proves to be a robust model fitting approach given the presence of outliers.*

RANSAC can be used to solve the correspondence problem, as mentioned in Section 2.4.1. In this case, the model is the fundamental matrix, $\mathbf{F}$. The parameters of the model, the elements of the fundamental matrix, can be calculated from image correspondences as shown in Section 2.1.4 [25]. The distance function would be the correspondence condition shown in Equation 2.19.

The matrix $\mathbf{F}$ is calculated using seven corresponding points, using the method described in Section 2.4.1. If there is more than one solution for $\mathbf{F}$, each solution is tested, and the matrix with the largest number of inliers is kept. There is a computational advantage to using a seven-point solution for the fundamental matrix over an eight-point solution, which can be illustrated using Equation 2.67, shown in Figure 2.19.

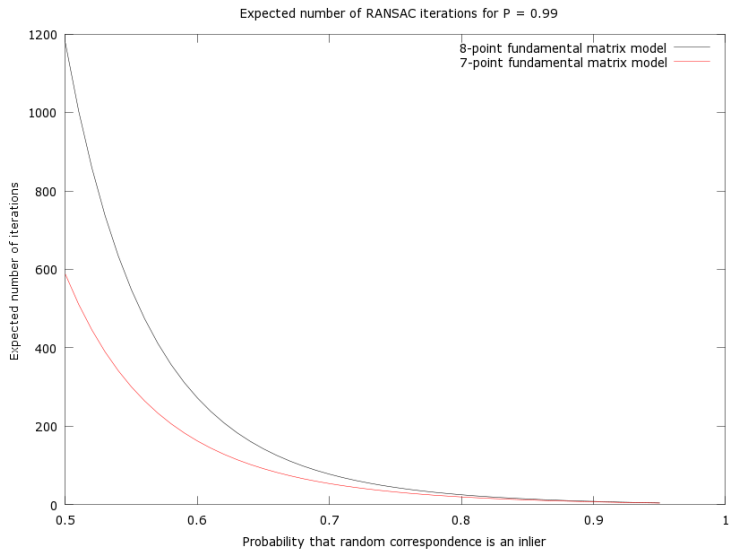


Figure 2.19: *The expected number of RANSAC iterations for the calculation of a fundamental matrix can be estimated using Equation 2.67. This plot shows the expected number of iterations for an eight and a seven-point fundamental matrix model as a function of the probability of a randomly selected point being an inlier to the model. The computational advantage of using the seven-point model can be seen as the probability of a point being an inlier decreases.*

Using a seven-point solution to the fundamental matrix means that a smaller set of RANSAC iterations must be performed in order to solve for the fundamental matrix than would be needed using an eight-point solution. This reduction in the number of iterations requires having to test one to three possible matrices in each RANSAC iteration, instead of just one. As shown in Figure 2.19, the gap between iterations grows much larger as the probability of inliers decreases. This is an important note specifically for aerial imagery.

Aerial imagery tends to have many small features, especially over urban scenes. The large number of features causes the error in feature matching to be high. When using RANSAC to determine the fundamental matrix between two aerial images, it should be assumed that the probability of a randomly selected image correspondence being an inlier should be low. A conservative estimate of probability would be around $w = 0.6$, which would require approximately 272 RANSAC iterations for the eight-point solution, and 163 iterations for the seven-point solution. This work used a very conservative threshold of

2048 iterations for fundamental matrix estimation with RANSAC.

Using RANSAC for this problem yields an estimate for the fundamental matrix from a set of image correspondences as well as the image correspondences that have been identified as false (outliers). Figure 2.20 presents an example of a set of SIFT correspondences that have been optimized using RANSAC.

The camera pose estimation problem presented in Section 2.1.5 can also use RANSAC to generate a reliable estimate of the camera pose using a noisy set of image correspondences. In this case, the essential matrix (Equation 2.46) can be used as the RANSAC model.

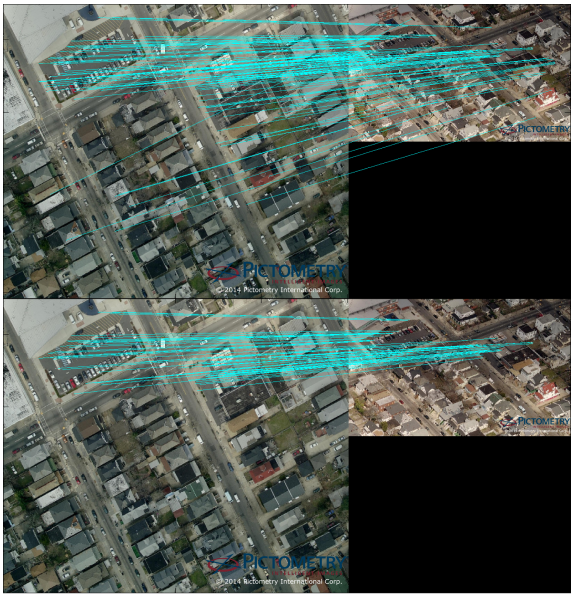


Figure 2.20: *SIFT correspondences optimized with RANSAC. The top image pair shows the correspondences prior to the optimization, and the bottom image pair shows what RANSAC found as the best correspondences. (Imagery courtesy of Pictometry Inc.)*

Algorithm 1 RANSAC Algorithm

```
τ =Number chosen as defined in Equation 2.64
ϵ =Assumed proportion of data that contains outliers
T = (1 − ϵ) size(data)
p = Probability of choosing a dataset with all inliers (Usually 0.99)
n = Number of parameters in model
k = ∞
iterations = 0
bestModel = null
bestConsensusData = null
while k > iterations do
    randIndicies = randomGenerator()
    consensusData = data[randIndicies]
    proposedModel = modelFittingFuction(consensusData)
    for all data not in consensusData do
        if distanceFunction(data, proposedModel) < τ then data is added to consensusData
        end if
    end for
    if size(consensusData) > size(bestConsensusData) then
        bestConsensusData = consensusData
        bestModel = propsedModel
        if size(bestConsensusData) > T then Break Loop
        end if
    end if
    w = size(consensusData)/size(data)
    k = log (1 − p) / log (1 − w^n)
    iterations = iterations + 1
end while
model = modelFittingFuction(consensusData)
```

2.4.2 Bundle Adjustment

The process of three-dimensional reconstruction using real-world data can produce error in many different ways. In order to minimize this error, after the initial reconstruction, a global optimization process is carried out. The optimization process used for this is called a bundle adjustment, it is widely used in the computer vision community to achieve this goal [25, 44, 50]. The notion of a bundle adjustment was developed in the 1960s with the growth of analytical photogrammetry [73]. The process reduces simply to minimizing the re-projection error, as defined in Equation 2.68.

$$\epsilon = \sum_{j=1}^{m} \sum_{i=1}^{n} d\left(\hat{\mathbf{x}}_i, \mathbf{P}_j \mathbf{X}_i\right)^2 \tag{2.68}$$

The function d is a distance function, which represents the distance between the known feature point $\hat{\mathbf{x}}$ and the projected point calculated from the camera projection matrix and the reconstructed three-dimensional point, $\mathbf{x} = \mathbf{PX}$. The total error is calculated for all n points and across all m cameras. In photogrammetric terms the projection of a three-dimensional point is considered a ray passing through the camera. The optimization process adjusts the bundle of rays which pass through each camera to minimize the re-projection error, hence the term bundle adjustment.

Gauss-Newton Iteration Solution

The issue of minimizing the re-projection error can be represented as a nonlinear optimization problem, to which there are a number of known solution methods. The re-projection error can be defined as,

$$\epsilon\left(\beta\right) = \left\| f\left(\beta\right) - x \right\| \tag{2.69}$$

where $f\left(\beta\right)$ represents the nonlinear function f which produces an image projection given a vector of parameters β, and x represents the known feature point. The parameters in this case are the elements of the camera projection matrix along with the three-dimensional point. In order to solve the nonlinear minimization process an iterative approach is taken assuming that an initial approximation which is close to the minimum value can be found. It is also assumed that the function f can be represented as a linear function with small changes in β. Therefore, an iterative relationship, $\beta_n + 1 = \beta_n + \Delta\beta$, can be formed. The

cost function, g, associated with minimizing the sum of squares is

$$g(\beta) = \frac{\|\epsilon(\beta)\|^2}{2} \tag{2.70}$$

The iterative relationship can be substituted into the cost function shown in 2.70, which can then be expanded using a Talyor-series expansion truncated at the third term [25].

$$g(\beta_n + \Delta\beta) = g(\beta_n) + g'(\beta_n)\Delta\beta + \frac{\Delta\beta^T g''(\beta_n)\Delta\beta}{2} + \dots \tag{2.71}$$

In order to minimize the cost function using the iterative relationship, an expression is needed for $\Delta\beta$, such that when $\Delta\beta$ is added to β_n the function heads towards to minima. This can be found by finding an expression for the minimum of the cost function $g(\beta_n + \Delta\beta)$, with respect to $\Delta\beta$. In other words, finding the condition in which the iterative change to β_n equals zero. This is done by taking it's derivative and setting the equation equal to zero,

$$g'(\beta_n) + g''(\beta_n)\Delta\beta = 0 \tag{2.72}$$

Solving for $\Delta\beta$ in Equation 2.72, gives an expression for $\Delta\beta$,

$$\Delta\beta = -\frac{g'(\beta_n)}{g''(\beta_n)} \tag{2.73}$$

In terms of calculation, it is convenient to express g' and g'' in terms of the error function shown in Equation 2.69. The first derivative of the cost function is [25].

$$g'(\beta_n) = \epsilon'(\beta_n)^T \epsilon(\beta_n) = J(\beta_n)^T \epsilon(\beta_n) \tag{2.74}$$

Where $J(\beta_n)$ is the Jacobian vector of $f(\beta)$ evaluated at β_n. The second derivative can be calculated from $g'(\beta_n)$, and is given by,

$$g''(\beta_n) = \epsilon'(\beta_n)^T \epsilon'(\beta_n) + \epsilon''(\beta_n)^T \epsilon(\beta_n) \tag{2.75}$$

Making the assumption that $f(\beta)$ can be approximated to be linear in small increments, the second derivative of $\epsilon(\beta_n)$ would go to zero. As shown in Equation 2.74, the first derivative of $\epsilon(\beta_n)$ is the Jacobian of $f(\beta_n)$, Equation 2.75 reduces to

$$g''(\beta_n) = J(\beta_n)^T J(\beta_n) \tag{2.76}$$

The second derivative approximation of the cost function as shown in Equation 2.76 is the defining quality of the Gauss-Newton method [25][1]. The equation for $\Delta\beta$ can be updated in terms that are computationally simpler to calculate, shown in Equation 2.77.

$$\Delta\beta = -\left(J\left(\beta_n\right)^T J\left(\beta_n\right)\right)^{-1} J\left(\beta_n\right)^T \epsilon\left(\beta_n\right) = -J\left(\beta_n\right)^\dagger \epsilon\left(\beta_n\right) \tag{2.77}$$

Where the products of the Jacobians in Equation 2.77 is recognized as the pseudo-inverse of the Jacobian matrix. The iterative relationship is then expressed in these terms that yield the Gauss-Newton update equation [25], namely,

$$\beta_{n+1} = \beta_n - J\left(\beta_n\right)^\dagger \epsilon\left(\beta_n\right) \tag{2.78}$$

Using Equation 2.78 to solve for β_{n+1}, will converge on a minimum of $f\left(\beta\right)$, given that the assumptions made hold true for function and initial estimate of β. In review the assumptions are 1) the second derivative of the cost function can be approximated by $J^T J$, and 2) the initial estimate is close to the minimum. In terms of minimizing re-projection error, the initial estimate of the parameters can sometimes be far away from the minimum re-projection error. In this case, other methods can be used.

Steepest Descent Iteration

A simpler approach to the optimization problem is called the Steepest Descent (or Gradient Descent) optimization algorithm. The idea behind this algorithm is that by always taking a step in the downward gradient direction, the algorithm will eventually converge to a minimum. This method can be used to minimize the cost function defined in Equation 2.70, by using the negative gradient, $g'\left(\beta\right)$. An expression for $g'\left(\beta\right)$ is shown in Equation 2.74. The expression for $\Delta\beta$ is shown below,

$$\Delta\beta = -\lambda g'\left(\beta\right)\Delta\beta = -\lambda J\left(\beta\right)^T \epsilon\left(\beta\right) \tag{2.79}$$

The step size for $\Delta\beta$ is controlled by λ. The method of steepest descent is computationally simple to execute, and methods of adaptively controlling λ can improve convergence from positions far from the minimum [45]. However, for functions which are poorly scaled, the number of iterations required to find the minimum becomes very large. This method can be combined with the Gauss-Newton method presented in the previous section, which

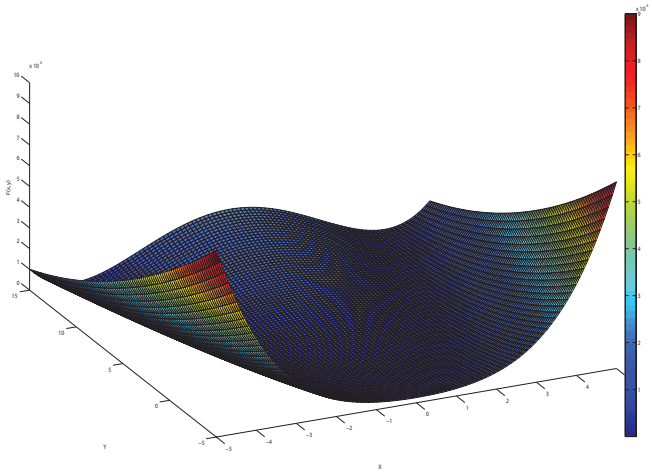
is the basis for the Levenberg-Marquardt optimization routine.

Levenberg-Marquardt Optimization

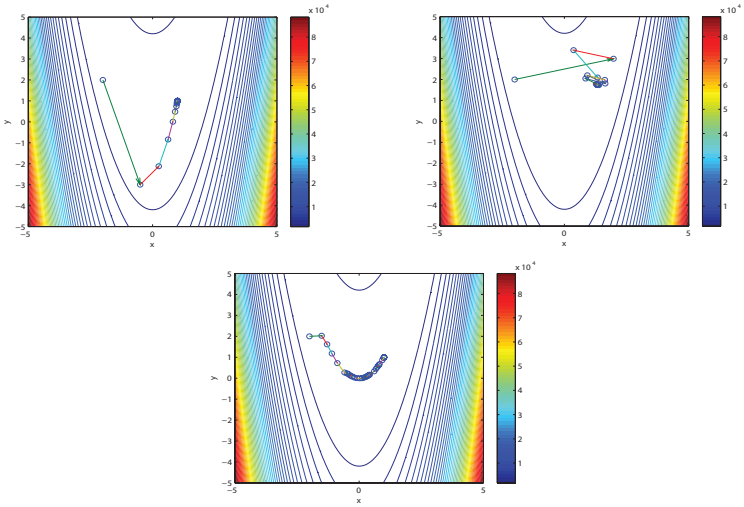
The Levenberg-Marquardt optimization routine is very popular in nonlinear numerical optimization, and used widely to solve the bundle adjustment problem [25]. It can be thought of as a combination of the Gauss-Newton and steepest descent optimization algorithms. The equation shown in 2.77 is reformed as

$$\left(J\left(\beta\right)^T J\left(\beta\right) + \lambda \mathbf{I}\right) \Delta\beta = -J\left(\beta\right)^T \epsilon\left(\beta\right) \tag{2.80}$$

where $\mathbf{I}$ is the identity matrix, and λ is some chosen scalar value. The combination of Gauss-Newton and steepest descent methods can be illustrated by observing the effect λ has on the step size $\Delta\beta$. As λ grows very large in comparison to $J\left(\beta\right)^T J\left(\beta\right)$, the solution for $\Delta\beta$ is essentially the solution shown in Equation 2.79 for steepest descent. As λ becomes very small, the solution for $\Delta\beta$ becomes the one shown in Equation 2.77 for the Gauss-Newton method. The step size λ is initially chosen to be the average if the diagonal elements of $\left(J\left(\beta\right)^T J\left(\beta\right)\right)$ multiplied by 10^{-3}. For each iteration that minimizes the error, λ is divided by a factor of ten. In this manner, the Levenberg-Marquardt algorithm acts as the steepest descent method when far from the minimum, and as Gauss-Newton when close to the minimum. This optimization routine provides very fast convergence by using this combination of methods [14]. A comparison of Gauss-Newton, steepest descent, and Levenberg-Marquardt optimization on the Rosenbrock function is shown in Figure 2.21. The steepest descent method takes the largest number of iterations, as expected, and the Levenberg-Marquardt algorithm is quicker and converges faster than both the Gauss-Newton and steepest descent methods.



(a) *Rosenbrock Function*



(b) *Comparision of optimization routines with starting point [-2,2].*

Figure 2.21: *A comparison of Gauss-Newton, steepest descent, and Levenberg-Marquardt optimization algorithms using the Rosenbrock Function (Minimum at [1,1]). In (b), Top Left: Gauss-Newton, Top Right: Steepest Descent, Bottom: Levenberg-Marquardt. The steepest descent method takes the largest number of iterations, while the Levenberg-Marquardt algorithm is quicker and converges faster than the others. The following data was collected for each of the optimization methods during the calculations shown in (b): Gauss-Newton; 56 iterations in 2.09 ms, Steepest Descent; 300 iterations in 9.58 ms, Levenberg-Marquardt; 30 iterations in 2.04 ms.*

If the current iteration of λ causes a decrease in error, then it is accepted as a good iteration and λ is divided by ten. However, if the current iteration of λ causes the error to increase, then λ is multiplied by ten and ΔP is recalculated. This is repeated until a λ is found that causes a reduction in error. The Levenberg-Marquardt algorithm is shown in Algorithm 2

Algorithm 2 Levenberg-Marquardt Algorithm

β = initialEstimate /* *Initial Estimate of Parameters* */
$\mathbf{X}$ = values /* *Values for solution of F(P)=X* */
F = @function /* *Handle to nonlinear function F() to be minimized* */
J = @functionJacobian /* *Handle to Jacobian of F()* */
I = $identity(length(P_0))$ /* *Identity matrix the size of* $J^T J$ */
$\lambda = \frac{diag\left(J(\beta)^T J(\beta)\right)}{length(X)} * 10^{-3}$ /* *Initial estimate of lambda* */
$\epsilon = F(\beta) - X$
$\alpha = 10^{-15}$ /* *Some error threshold in which LM should stop iterating* */
while $\left(\frac{\epsilon^T \epsilon}{2}\right) > \alpha$ **do**
  $\Delta\beta = \left(J(\beta)^T J(\beta) + \lambda \mathbf{I}\right)^{-1} J(\beta)^T \epsilon$
  $\beta_t = \beta - \Delta\beta$
  $\epsilon_t = F(\beta_t) - X$
  while $\left(\frac{\epsilon^T \epsilon}{2}\right) < \left(\frac{\epsilon_t^T \epsilon_t}{2}\right)$ **do**
    $\lambda = \lambda * 10$
    $\Delta\beta = \left(J(\beta)^T J(\beta) + \lambda \mathbf{I}\right)^{-1} J(\beta)^T \epsilon$
    $\beta_t = \beta - \Delta\beta$
    $\epsilon_t = F(\beta_t) - X$
  end while
  $\beta = \beta_t$
  $\epsilon = \epsilon_t$
  $\lambda = \frac{\lambda}{10}$
end while

Bundle Adjustment with Sparse Levenberg-Marquardt Optimization

The Levenberg-Marquardt optimization method is the preferred method for solving the bundle adjustment problem. This is due to its fast convergence to a minimum even when the starting point is far off. The standard Levenberg-Marquardt algorithm can still

be too slow for a real-world bundle adjustment. The number of parameters for a bundle adjustment using the camera-projection matrices with every point seen in each image, would be $n * k + m * j$, where n is the number of cameras, k is the number of camera parameters, m is the number of three-dimensional points, and j is the number three-dimensional point parameters. For a projective bundle adjustment, k would be equal to twelve and j would be equal to four. A normal bundle-adjustment could have around 20 cameras and $1,000$ points. This would result in a 4240×4240 Hessian matrix which would have to be inverted multiple times for each iteration of Levenberg-Marquardt [25]. This can become computational infeasible, however, due to the lack of correlation between each image projection, the Hessian matrix has a definable sparse structure.

The Jacobian structure for an image projection can be separated into two distinct parts; the camera and three-dimensional coordinate parameters. If the projection is thought of as a nonlinear function $f(\beta)$, β would be a vector containing a set of parameters for each projection. In the case of a projective system, there would be twelve camera parameters and four point parameters. The Jacobian matrix would then have the structure $\mathbf{J} = [\mathbf{A}|\mathbf{B}]$, where $A_{kn} = \dfrac{\partial f\left(P_k^n\right)}{\partial P_k^n}$ a $mn{\times}12n$ matrix and $B_{jm} = \dfrac{\partial f\left(X_j^m\right)}{\partial X_j^m}$ a $mn{\times}4m$ matrix. This yields a Jacobian structure shown below, as shown for two cameras and three points

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f(P_1^1)}{\partial P_1^1} & \cdots & \frac{\partial f(P_{12}^1)}{\partial P_{12}^1} & 0 & \cdots & 0 & \frac{\partial f(X_1^1)}{\partial X_1^1} & \cdots & \frac{\partial f(X_4^1)}{\partial X_4^1} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \frac{\partial f(P_1^1)}{\partial P_1^1} & \cdots & \frac{\partial f(P_{12}^1)}{\partial P_{12}^1} & 0 & \cdots & 0 & 0 & \cdots & 0 & \frac{\partial f(X_1^2)}{\partial X_1^2} & \cdots & \frac{\partial f(X_4^2)}{\partial X_4^2} & 0 & \cdots & 0 \\ \frac{\partial f(P_1^1)}{\partial P_1^1} & \cdots & \frac{\partial f(P_{12}^1)}{\partial P_{12}^1} & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \frac{\partial f(X_1^3)}{\partial X_1^3} & \cdots & \frac{\partial f(X_4^3)}{\partial X_4^3} \\ 0 & \cdots & 0 & \frac{\partial f(P_1^2)}{\partial P_1^2} & \cdots & \frac{\partial f(P_{12}^2)}{\partial P_{12}^2} & \frac{\partial f(X_1^1)}{\partial X_1^1} & \cdots & \frac{\partial f(X_4^1)}{\partial X_4^1} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \frac{\partial f(P_1^2)}{\partial P_1^2} & \cdots & \frac{\partial f(P_{12}^2)}{\partial P_{12}^2} & 0 & \cdots & 0 & \frac{\partial f(X_1^2)}{\partial X_1^2} & \cdots & \frac{\partial f(X_4^2)}{\partial X_4^2} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \frac{\partial f(P_1^2)}{\partial P_1^2} & \cdots & \frac{\partial f(P_{12}^2)}{\partial P_{12}^2} & 0 & \cdots & 0 & 0 & \cdots & 0 & \frac{\partial f(X_1^3)}{\partial X_1^3} & \cdots & \frac{\partial f(X_4^3)}{\partial X_4^3} \end{bmatrix} \tag{2.81}$$

where each $\dfrac{\partial f\left(P_k^n\right)}{\partial P_k^n}$ is a $1{\times}k$ vector and each $\dfrac{\partial f\left(X_j^m\right)}{\partial X_j^m}$ is a $1{\times}j$ vector. This sparse structure creates another definable sparse structure for the normal equations $\mathbf{J}^T\mathbf{J}$, which can be defined using four sections, as shown in Equation 2.82.

$$\mathbf{J}^T\mathbf{J} = \begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V} \end{bmatrix} \tag{2.82}$$

The normal equations shown in Equation 2.82 contain a block structure with matrices $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$. The matrices $\mathbf{U}$ and $\mathbf{V}$ contain a $nk{\times}nk$ and a $mj{\times}mj$ sparse symmetric

block structure as shown in Equations 2.83 and 2.84.

$$\mathbf{U} = \begin{bmatrix} U_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & U_n \end{bmatrix}$$

$$U_n = m \begin{bmatrix} \frac{\partial f(P_1^n)}{\partial P_1^n}^2 & \cdots & \frac{\partial f(P_1^n)}{\partial P_1^n}\frac{\partial f(P_k^n)}{\partial P_k^n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(P_1^n)}{\partial P_1^n}\frac{\partial f(P_k^n)}{\partial P_k^n} & \cdots & \frac{\partial f(P_k^n)}{\partial P_k^n}^2 \end{bmatrix} \tag{2.83}$$

$$\mathbf{V} = \begin{bmatrix} V_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & V_m \end{bmatrix}$$

$$V_m = n \begin{bmatrix} \frac{\partial f(X_1^m)}{\partial X_1^m}^2 & \cdots & \frac{\partial f(X_1^m)}{\partial X_1^m}\frac{\partial f(X_j^m)}{\partial X_j^m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(X_1^m)}{\partial X_1^m}\frac{\partial f(X_j^m)}{\partial X_j^m} & \cdots & \frac{\partial f(X_j^m)}{\partial X_n^m}^2 \end{bmatrix} \tag{2.84}$$

The matrix $\mathbf{W}$ has a non-symmetric $nk \times jm$ dense structure as shown in Equation 2.85.

$$\mathbf{W} = \begin{bmatrix} \frac{\partial f(P_1^1)}{\partial P_1^1}\frac{\partial f(X_1^1)}{\partial X_1^1} & \cdots & \frac{\partial f(P_1^1)}{\partial P_1^1}\frac{\partial f(X_j^1)}{\partial X_j^1} & \cdots & \cdots & \frac{\partial f(P_1^1)}{\partial P_1^1}\frac{\partial f(X_j^m)}{\partial X_j^m} \\ \vdots & \ddots & \vdots & & & \vdots \\ \frac{\partial f(P_k^1)}{\partial P_k^1}\frac{\partial f(X_1^1)}{\partial X_1^1} & \cdots & \frac{\partial f(P_k^1)}{\partial P_k^1}\frac{\partial f(X_j^1)}{\partial X_j^1} & \cdots & \cdots & \frac{\partial f(P_k^1)}{\partial P_k^1}\frac{\partial f(X_j^m)}{\partial X_j^m} \\ \vdots & & \vdots & \ddots & & \vdots \\ \vdots & & \vdots & & \ddots & \vdots \\ \frac{\partial f(P_k^n)}{\partial P_k^n}\frac{\partial f(X_1^1)}{\partial X_1^1} & \cdots & \frac{\partial f(P_k^n)}{\partial P_k^n}\frac{\partial f(X_j^1)}{\partial X_j^1} & \cdots & \cdots & \frac{\partial f(P_k^n)}{\partial P_k^n}\frac{\partial f(X_j^m)}{\partial X_j^m} \end{bmatrix} \tag{2.85}$$

When combined $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ create a sparse block structure for the normal equations $\mathbf{J}^T\mathbf{J}$. Figure 2.22 shows this structure for the two camera and three point example, and for the original real-world twenty camera and one-thousand point example. It can be noted that in real-world examples the matrix $\mathbf{V}$ tends to be much larger than $\mathbf{U}$.

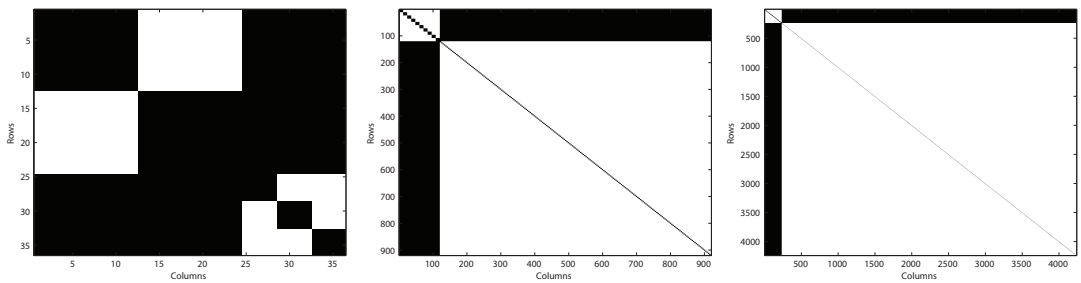


Figure 2.22: *Illustration of the normal equations* $\mathbf{J}^T\mathbf{J}$ *for three examples for camera and point sets. The black areas contain equations while the white areas contain zeros. Normal equation parameters left to right: 2 cameras and 3 points, 10 cameras and 200 points, 20 cameras and 1000 points.*

The Levenberg-Marquardt optimization method uses an augmented set of normal equations, as shown in Equation 2.80. The augmented operation adds a constant along the diagonal of $\mathbf{J}^T\mathbf{J}$, which corresponds to adding the constant to the matrices $\mathbf{U}$ and $\mathbf{V}$. The augmented diagonal matrices will be represented as $\mathbf{U}^*$ and $\mathbf{V}^*$, so the form of the Levenberg-Marquardt equation (2.80) in sparse structure form, is given by

$$\begin{bmatrix} \mathbf{U}^* & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V}^* \end{bmatrix} \Delta\beta = -[\mathbf{A}|\mathbf{B}]^T \epsilon(\beta) \tag{2.86}$$

The top right corner of the normal equations can be eliminated by multiplying both sides of the equation by $\begin{bmatrix} \mathbf{I} & -\mathbf{W}\mathbf{V}^{*-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$. This results in a new set of equations [25, 44]

$$\begin{bmatrix} \mathbf{U}^* - \mathbf{W}\mathbf{V}^{*-1}\mathbf{W}^T & 0 \\ \mathbf{W}^T & \mathbf{V}^* \end{bmatrix} \begin{bmatrix} \Delta\beta_P \\ \Delta\beta_X \end{bmatrix} = \begin{bmatrix} \epsilon(\beta_P) - \mathbf{W}\mathbf{V}^{*-1}\epsilon(\beta_X) \\ \epsilon(\beta_X) \end{bmatrix} \tag{2.87}$$

where the vector $\Delta\beta$ has been broken into its camera and coordinate parameter $\Delta\beta_P$ and $\Delta\beta_X$. The matrix expression in Equation 2.87 can be parsed into it's top and bottom halves, which allows the solving of $\Delta\beta_P$ and $\Delta\beta_X$ separately. The solution for $\Delta\beta$ now

becomes the system of equations shown below.

$$\begin{aligned}
\Delta\beta_P &= \left(\mathbf{U}^* - \mathbf{W}\mathbf{V}^{*-\mathbf{1}}\mathbf{W}^T\right)^{-1}\left(\epsilon\left(\beta_P\right) - \mathbf{W}\mathbf{V}^{*-\mathbf{1}}\epsilon\left(\beta_X\right)\right) \\
\Delta\beta_X &= \mathbf{V}^{*-\mathbf{1}}\left(\epsilon\left(\beta_X\right) - \mathbf{W}^T\Delta\beta_P\right) \\
\Delta\beta &= \begin{bmatrix} \Delta\beta_P \\ \Delta\beta_X \end{bmatrix}
\end{aligned} \tag{2.88}$$

Computationally, the sparse variant of the Levenberg-Marquardt algorithm is much simpler than the general algorithm. The general algorithm requires the repeated inversion of a $(nk+mj)\times(nk+mj)$ matrix. The sparse algorithm requires the repeated inversion of a $nk\times nk$ matrix and a $mj\times mj$ symmetric matrix. In real-world situations, it is true that $n << m$, so the inversions in the sparse algorithm are much more computationally efficient than those in the general algorithm for practical usage. In modern SfM applications the sparse variant of the Levenberg-Marquardt optimization algorithm is most often used to solve the bundle adjustment problem [44, 50]

2.5 Deriving Geo-Accurate Structure Measurements

In order to generate geographically accurate scene structure, the structure estimation from the SfM processes must be registered to a fix Earth-based coordinate system. Geo-registration of point clouds to obtain absolute orientation has existed for many years and is well documented [4, 46]. The simplest georegistration method uses just GPS information to extract a georegistering similarity transform, $\mathbf{T_s}$, by matching the estimated camera pose to its corresponding GPS location [50]. Given the noisy nature of most low-cost GPS devices, as those found in common hand-held imagery, noise-reducing and optimizing algorithms such as RANSAC (Section 2.4.1). The error in this estimation alone is often too large, and additional information is used to further refine the transform. Wang *et al.* [71] incorporate Google Street View imagery and Google Earth models to further refine the position of SfM derived structure. Use of accurate digital surface models can also reduce positional error in ground-based SfM point clouds, as shown by Wendel *et al.* [72] It is even possible to incorporate the geolocations into the SfM process itself. Crandall *et al.* [10] utilize Internet-based imagery with geolocations to aid the estimation of camera pose

and orientation [10]. They propose a novel SfM technique that solves the problem using Markov random fields (MRF) by incorporating the geotags into pose and orientation estimation using an energy function that minimizes the noise present in these types of geotags. This work focuses on only using the simplest method of georegistration, calculating the similarity transform $\mathbf{T_s}$.

2.5.1 Calculating $\mathbf{T_s}$

Points contained within two different Euclidean coordinate systems can be related using a similarity transform, $\mathbf{T}_s$. A set of inhomogeneous points $\mathbf{X}_A$ in a metric coordinate system A, can be transformed to the desired metric coordinate system D using a seven DoF transform as shown,

$$\mathbf{X}_D = s\mathbf{R}\mathbf{X}_A + \mathbf{t} \tag{2.89}$$

where s is a uniform scale, $\mathbf{R}$ is a 3-by-3 rotation matrix derived from three rotation parameters, and $\mathbf{t}$ is a 3-by-1 translation vector. This transform can be calculated by using a set of known corresponding points, $\mathbf{x}_a$ and $\mathbf{x}_d$, such that $\mathbf{x}_a \in \mathbf{X}_D$ and $\mathbf{x}_d \in \mathbf{X}_D$.

The first step in solving for this transform is calculating the scale. This is done by moving the centroid of each set of points, $\mathbf{x}_a$ and $\mathbf{x}_d$, to the origin of their respective coordinate systems,

$$\mathbf{C}_{a,i} = \mathbf{x}_{a,i} - \bar{\mathbf{x}}_a \tag{2.90}$$

$$\mathbf{C}_{d,i} = \mathbf{x}_{d,i} - \bar{\mathbf{x}}_d \tag{2.91}$$

The scale value can then be determined by the ratio between the mean lengths of each zero-centered points set. Figure 2.23 shows an example of $\mathbf{C}_{a,i}$ and $\mathbf{C}_{d,i}$.

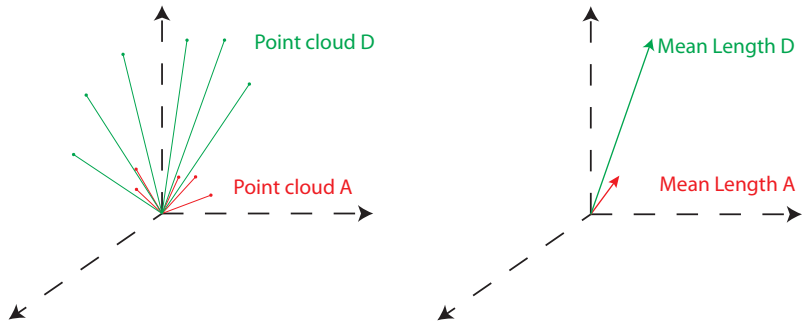


Figure 2.23: *This figure shows an example of two sets of zero centered point clouds. The figure on the right shows the mean vector length of each set of points. The ratio of the mean lengths can be used to calculate the scale between the two data sets.*

The scale relating the two point sets, $\mathbf{C}_{a,i}$ and $\mathbf{C}_{d,i}$, can be calculated through the ratio of the mean vectors lengths for each point set, as shown below and pictured in Figure 2.23 [34].

$$s^2 = \frac{\sum_{i=1}^{n} \parallel C_{d,i} \parallel^2}{\sum_{i=1}^{n} \parallel C_{a,i} \parallel^2} \tag{2.92}$$

The next step is to compute the rotation matrix $\mathbf{R}$ which relates $\mathbf{C}_{a,i}$ to $\mathbf{C}_{d,i}$. This can be done using the Kabsch algorithm, which is a method for calculating an optimal rotation matrix between two sets of points in a least squares sense [69, 68]. After the rotation matrix and uniform scale value are calculated the translation vector can be calculated using the centroids, as shown in Equation 2.93.

$$\mathbf{t} = \bar{\mathbf{x}}_a - s\mathbf{R}\bar{\mathbf{x}}_d \tag{2.93}$$

Equation 2.89 can be formated into a 4x4 transform $\mathbf{T_s}$ in homogeneous coordinates as,

$$\mathbf{T_s} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.94}$$

This method alone is sensitive to error in the corresponding points $\mathbf{x}_a$ and $\mathbf{x}_d$. The similarity transform can calculated using a model fitting method robust to outliers such as the RANSAC algorithm described in Section 2.4.1. Methods for extracting the corre-

sponding points are discussed in Section 3.2.

2.6 Discussion

The goal in Structure from Motion is to estimate the pose of a series of images through the estimation of the three-dimensional scene structure viewed by the images. This section covered the basis for what is needed in order to perform SfM in a variety of different ways, as well as the optimization routines to remove the error generated in the SfM steps. This process can be broken into five essential steps; feature detection, feature matching, pose estimation, structure reconstruction, and error reduction. Methods for each step were presented in this section, in the order in which they are often used. Each one of these steps uses concepts derived from epipolar geometry. Figure 2.24 shows a common example a SfM workflow using methods previously described.

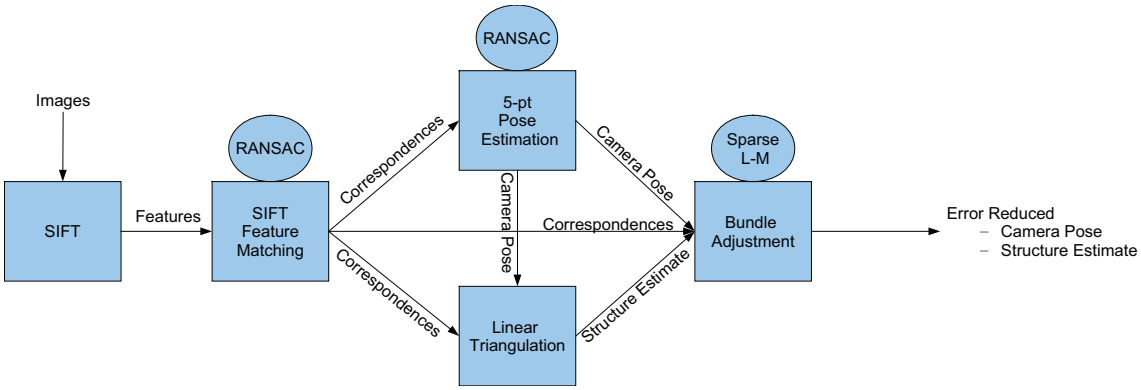


Figure 2.24: *A common SfM workflow. The circles represent the optimization method used for the connecting methods within.*

The workflow presented in Figure 2.24 is just one possible configuration. Given the application, it might be more suitable to use DAISY or A-SIFT in place of SIFT as the feature detector and matcher. Often aerial imagery is taken with IMU/GPS estimates which can be used as the initial estimate for the camera pose instead of using pose estimation. For near-nadir aerial imagery, a photogrammetric estimation of the three-dimensional point provides a better estimate of the three-dimensional structure than a linear estimation. The

one step in every SfM process that tends to be constant is the bundle adjustment [44]. Although, if desired, certain parameters of the bundle adjustment could be held constant, if the user had more confidence in specific parameters.

Structure from Motion methods provide a viable way to automatically generate three-dimensional structure from multi-view imagery. These methods can potentially have a strong impact in applications such as photogrammetric modeling, which is often heavily user-assisted. The goal of this work is to make steps towards fully automated and attributed physical modeling from aerial imagery, and SfM provides the basis for generating the physical geometry for every model.

Chapter 3

Methodology

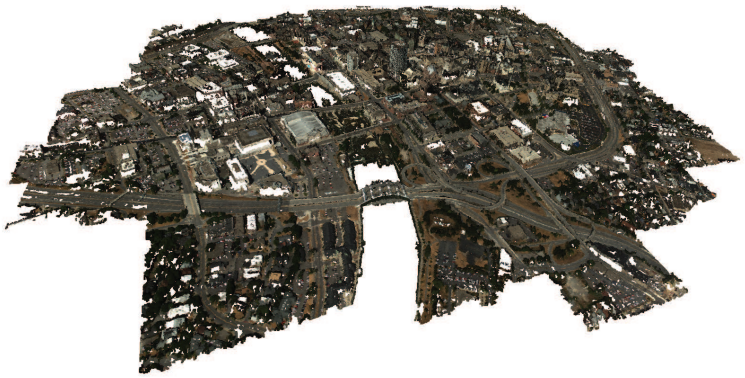


Figure 3.1: *A geoaccurate dense point cloud reconstruction of downtown Rochester, NY, generated using processes described in Chapter 2.*

The objective of this work is to extract physically accurate three-dimensional models from multi-view imagery. The background provided in Chapter 2 is necessary for understanding how the structure of the physically accurate models can be obtained. Methods described in the previous chapter can be used to formulate a workflow to generate geographically and physically accurate structure, such as the one shown in Figure 3.1. The physically accurate model is estimated from the output of this workflow. This chapter will review the methods used to estimate this model:

- Section 3.1 reviews the SfM open-source software which is used to estimate the scene structure from multi-view imagery
- Section 3.2 describes methods of estimating the georegistration transform described in Section 2.5
- Section 3.3 details methods tested and used for extracting the object's surface from the structure measurements, as well as methods for assigning a confidence metric to the estimated surface, and extracting individual structures from the entire scene structure.
- Section 3.4 presents methods for surface material spectral attribution and classification.

3.1 Software

Structure extraction is a very active and ongoing area of research in the computer vision community, consequently the methods presented in the previous chapter only touches the surface of a very large research area. Due to the large amount of interest, a collection of open-source software has surfaced through the community. This open-source software can be put together to formulate a workflow to extract structure from multi-view imagery, as discussed in Section 2.6.

In practice, the SfM steps are often repeated and exploited in order to generate reliable dense three-dimensional point clouds. There exists a number of open-source applications that have implemented different parts of the SfM workflow. Three applications are commonly used in conjunction to generate dense three-dimensional point clouds from imagery using SfM techniques; siftGPU, Bundler, and PMVS with CMVS. Figure 3.2 shows the order in which they are used. The following sections will provide some details on each software package.

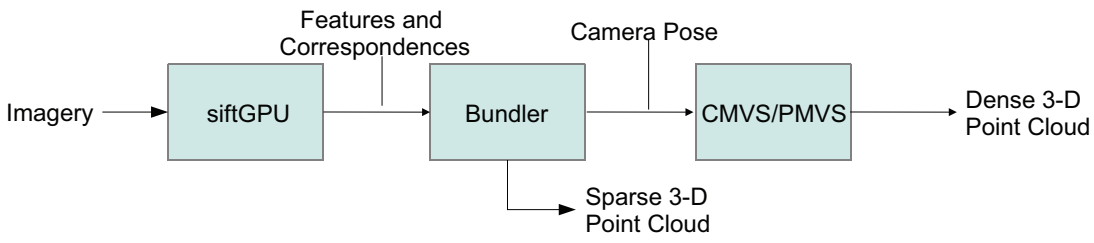


Figure 3.2: *Flow of the SfM open-source applications, with the data input and output for each application. This is the software used in this work.*

siftGPU

This software package is an implementation of SIFT as described in Section 2.2.1 which runs on the GPU, written by Changchang Wu [67]. There are many advantages to running SIFT on the GPU, as many stages of the SIFT process can be restructured to fit the parallel processing power of the GPU [27]. Aerial imagery tends to have very large pixel counts, and as the number of image pixels increases the number of detected SIFT features increases almost exponentially [67]. This becomes a major issue in the feature matching stage when doing a one-to-all matching for every feature in every image, which has a $O\left(n^2\right)$ time complexity. The matching process can also be parallelized and run on the GPU to significantly decrease the run time.

For the goal of urban modeling, very massive datasets often need to be processed. The SIFT feature extraction and matching process can take many hours or even days when run on a CPU. This time can be reduced to just a few hours by processing on the GPU. For this work, densely collected high resolution imagery was used for processing, and therefore it was only computationally feasible to use siftGPU.

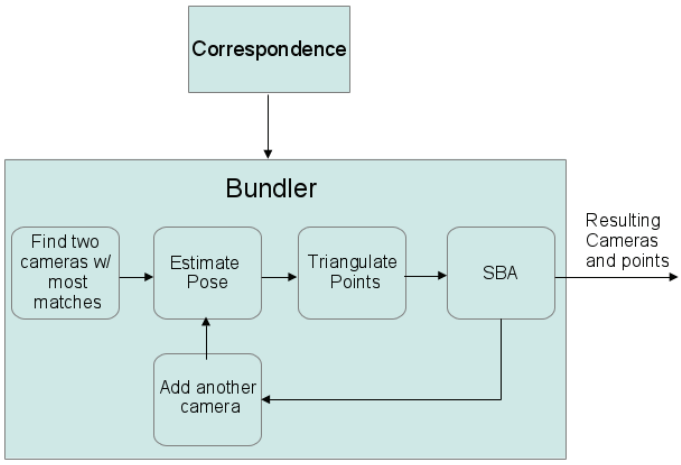


Figure 3.3: *Bundler is a packaged SfM workflow which uses an iterative bundle-adjustment approach for error reduction. The feature optimization, camera pose estimation, triangulation, and bundle adjustment are all performed in this software.*

Bundler

Bundler is a powerful SfM algorithm written by Noah Snavley [49]. This algorithm uses the initial correspondence estimate provided by siftGPU as input. The fundamental workflow within Bundler is shown in Figure 3.3.

This algorithm takes an iterative approach to its optimization process. Here, a sparse bundle adjustment (SBA) is used to perform the optimization. The input to the SBA is generated using the initial correspondences. The camera pose is estimated using the five-point pose estimation algorithm described in Section 2.1.5. Using that pose, each correspondence is triangulated for each image using a linear triangulation estimation described in Section 2.3.2. The correspondence, camera pose, and triangulated points are used as inputs to the SBA, which provides a solution based on optimizing the minimizing reprojection error as described in Section 2.4.2. This is done initially for the two cameras with the most correspondences, and repeats adding the camera with the next highest number of correspondences. The goal of using the iterative approach is to reduce the propagation of error in the SBA. The result of the Bundler process is a sparse point cloud which is generated from the initial SIFT correspondences, as well an error minimized set of camera pose information for each image.

PMVS with CMVS

Bundler provides a sparse reconstruction of the scene. However, a dense reconstruction is often desired. This is where the Patch-based Multi-View Stereo software (PMVS) can be used [19, 74]. This algorithm uses the camera pose provided by Bundler to narrow down a correspondence search, which allows for a large number of pixels to be reconstructed. The feature detection and matching method PMVS uses is described in Section 2.2.5.

The PMVS algorithm is a multi-core implementation which allows for the simultaneous processing of many images to occur quickly. However, when the image set grows large, this processing time can still become very long. A clustering algorithm called Clustering Views for Multi-view Stereo (CMVS) is implemented to break down the image set in to managed clusters [74]. CMVS is a graph-cut based clustering algorithm which clusters the cameras according to their pose [20]. The goal is to find clusters of cameras which are observing the same region of the scene, and run PMVS on just the corresponding images. This not only reduces runtime for PMVS, but also the error in the resulting point cloud. The final output of PMVS when run with CMVS is a dense three-dimension point cloud. This point cloud is what can be exploited as the base physical structure for modeling.

3.2 Obtaining an Accurate Coordinate System

Physically accurate modeling requires that the coordinate space of the point cloud reconstruction is quantifiable in some manner. The relative estimation of camera positions shown in Section 2.1.5 generates a camera coordinate system which is in an arbitrary relative coordinate space. The estimated structure based on these cameras are also in the same coordinate system. Applying a bundle adjustment to the whole system will move the cameras and three-dimensional points to an error reduced space, however this space has no physical meaning. Ideally, these three-dimensional points should be registered to a fixed Earth-based coordinate system. A method for deriving a similarity transform between two Euclidean coordinate systems was discussed in Section 2.5. In this case, the two Euclidean coordinate systems are the SfM based world coordinate system, and the fixed Earth-based coordinate system. In order to calculate this transform, a set of corresponding three-dimensional points must be obtained between the two coordinate systems. This section will discuss two methods of obtaining these corresponding points. A third method using linear triangulation as described in Section 2.3.2 is used for georegistration and also

analyzed in this work. An analysis of each methods accuracy is presented in Section 4.1.

3.2.1 Using Camera Position Estimates

The method most commonly used in the computer vision community uses geo-tags from imagery [62]. Estimated camera centers derived from the SfM process are used as the points from set $\mathbf{X}_A$, and the GPS-located camera centers are used as the points from set $\mathbf{X}_D$ (from Equation 2.89). The transformation process, which is henceforth referred to as *the camera centers approach*, is shown in Figure 3.4.

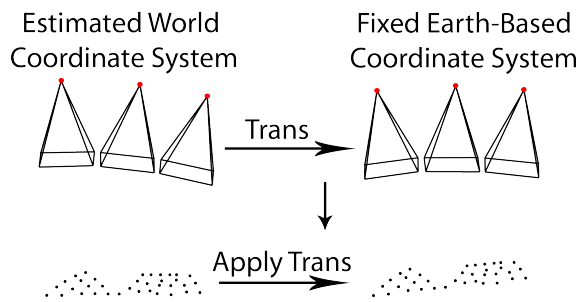


Figure 3.4: *The most common method for obtaining corresponding points utilizes the camera centers. Estimated three-dimensional coordinates from the SfM process and GPS-located coordinates are used to calculate the transform. This transform is then used to convert the SfM estimated structure into the GPS coordinate system.*

While this process requires only knowledge of the GPS information from each camera, it is highly susceptible to error. Given error in the GPS location or error in the camera pose estimation, the resulting transform is susceptible to significant inaccuracies. These errors can be mitigated using an outlier removal algorithm such as RANSAC [10].

3.2.2 Using the Camera Model and Image Correspondence

Another method for deriving the correspondence between these two coordinate systems relies on using the camera systems forward projection model [70], referred to as the *augmented camera model transform*. This method assumes that the imagery has been

acquired alongside highly accurate inertial navigation and global position systems (INS/GPS), which is very common with aerial imagery. Furthermore, it is assumed that the camera system has been well calibrated and tested so that the full ground-to-image function is known, again very common with aerial imaging platforms.

Using highly refined image correspondence, points are triangulated in both coordinate systems; In the SfM world coordinate system (WCS) using the estimated camera pose, and in the fixed Earth-based WCS using the GPS/INS measurements. These triangulated points yield the correspondence needed to estimate the georegistration transform, as shown in Figure 3.5.

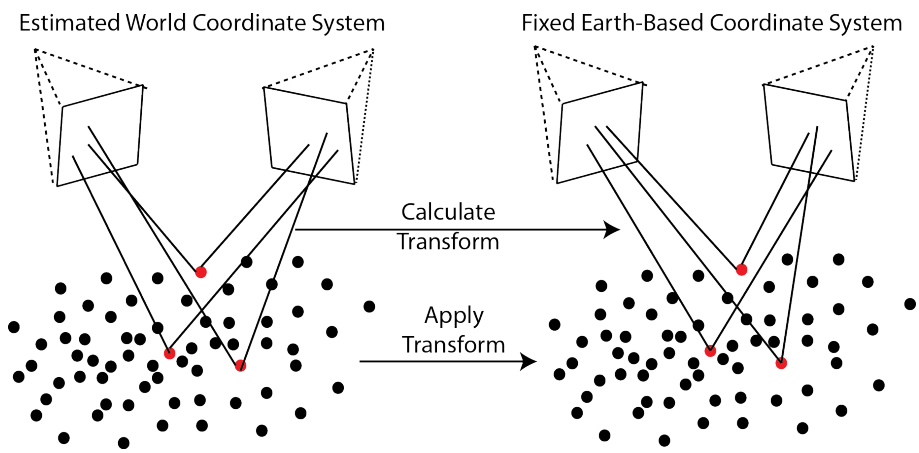


Figure 3.5: *The augmented camera model transform uses refined image correspondence, along with a known and calibrated camera model and accurate GPS/INS information, to calculate correspondence between the SfM WCS and the fixed Earth-based WCS.*

An additional step of error reduction is built into this method. By triangulating highly refined image correspondence with potentially noisy GPS/INS information, the error which would be introduced by the GPS/INS is minimized in the estimation of the 3-dimensional point. Section 2.3.2 describes the triangulation method used in this process, a process which essentially estimates the best structure point given the constraints shown in Equation 2.12. This additional error reduction provides some robustness to GPS/INS noise that the camera centers method cannot provide.

The highly accurate image correspondence can be determined by using a feature matching method with a very high threshold. Another method for quantifying accuracy would

be to use the residual error information from the SfM bundle adjustment. The points with the lowest residual error across the images will likely be the highest quality image correspondences.

3.2.3 Using the Camera Model Directly

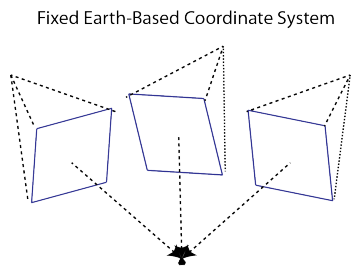


Figure 3.6: *With measured GPS/INS information and a known camera model, every image correspondence can be directly triangulated in the Earth-based coordinate system.*

Given the method presented Section 3.2.2, it is logical to try and triangulate every known image correspondence using measured GPS/INS information. This can be done using the algorithm presented in Section 2.3.2, and is referred to as the *direct triangulation* method. This is the simplest approach for georegistering the SfM extracted structure, however, it is also the most sensitive to error. The triangulation method is affected by both error in the image correspondence as well as error in the camera model, which can be caused by error in the GPS/INS measurements.

It might also seem logical to perform an optimization step once three-dimensional structure has been triangulated using the GPS/INS information, such as a bundle adjustment. This could be a viable option if there was known and measured ground control, which would require intensive manual intervention. However, using just GPS/INS information as well as the triangulated three-dimensional points in a bundle adjustment will likely cause the solution to converge to a local minimum. Adjusting the GPS/INS information based on measured image correspondence alone cannot guarantee convergence into the "true" Earth-based coordinate system without highly accurate, fixed, known ground control.

3.3 Surface Reconstruction Methods

After georegistration, each structure is processed on an individual basis, that is, a user has selected and isolated a specific target structure in the three-dimensional point cloud for modeling. The ideal output from a surface model looks like the hypothetical model shown in Figure 3.7. This model contains a polygon or facet-based representation of the objects surface, with each polygon or facet material identified.

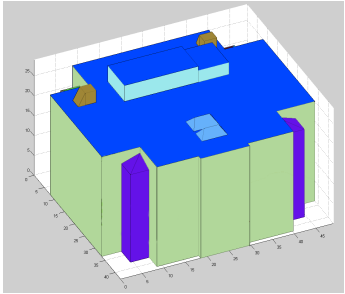


Figure 3.7: *The ideal output for physical modeling would be a CAD-like model in which each surface type is segmented or attributed.*

This section will discuss methods of estimating the surface structure, Section 3.4 will discuss surface attribution or segmentation. Two methods of surface estimation are explored. The first method attempts to decompose the extracted structure into geometric primitives in order to identify polygons and structures within the object point cloud. The second method takes a voxel-based approach for cleaning and estimating the surface structure. A confidence metric for the voxel-based reconstruction approach is developed and described as well. Finally, a method for automatically extracting individual structures, from user selected imagery, is presented.

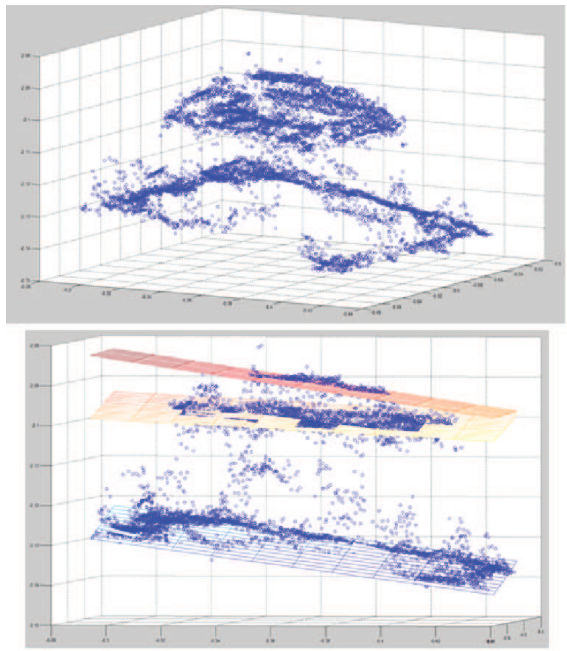


Figure 3.8: *Fitting of planes to a point cloud using the iterative RANSAC plane-fitting approach.*

3.3.1 Model Extraction Using RANSAC Plane Fitting and Alpha Shapes Boundary Extraction

The RANSAC model fitting algorithm described in Section 2.4.1 provides a suitable method for finding geometric primitives within a point cloud. This section will look at one type of geometric primitive, a plane. The plane geometry was chosen due to the high occurrence of planar structures in most urban structures. The inhomogeneous plane model shown in Equation 3.1 is used for the RANSAC model. This model requires three points to solve in $\mathbf{Ax} = 0$ form.

$$ax_1 + bx_2 + cx_3 = 0 \tag{3.1}$$

It is assumed that the point cloud for each region will be made up of a number of planes, therefore the RANSAC plane fitting process is iterative. The dominant plane is found initally, and all points falling on and near that plane will be removed from the point cloud. The process repreats itself until no planes can be fit to the remaining points using

the RANSAC process. Figure 3.8 shows an example of an object point cloud fitted with planes using this iterative model fitting approach.

A set of planes can be defined for each region, described by a collection of plane equations. To finalize the plane geometry, a boundary for the plane has to be defined using the object point cloud. This is done using a boundary extraction algorithm called Alpha Shapes, which has shown good results for point cloud boundary extraction [41]. The two-dimensional process for boundary extraction using Alpha Shapes can be thought of simply as a circle with a defined radius being 'rolled' around a set of two-dimensional points. When two points are touching the circle and there are no points within the circle, those two points are considered part of a boundary. Otherwise, when points are contained within the circle, the two points are not considered a boundary section [16]. Figure 3.9 shows an example of the boundary extraction process used in Alpha Shapes.

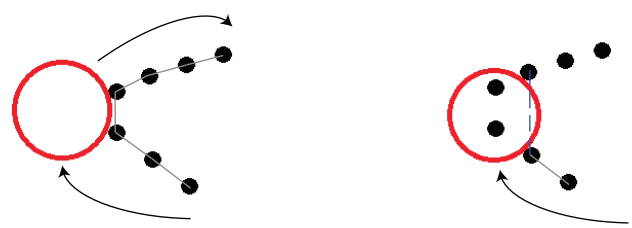


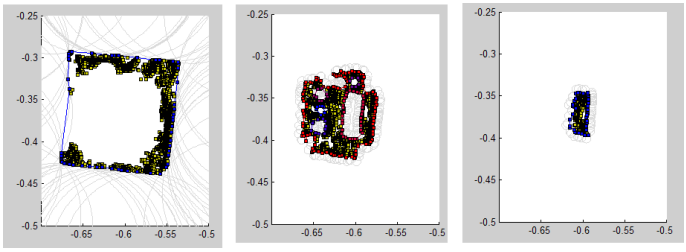
Figure 3.9: *Simplified boundary extraction process performed by Alpha Shapes. The points on the left will be defined as the boundary. The points on the right show how surface on the inside of the boundary will not be connected, because there are points contained within the circle.*

The Alpha Shapes boundary extraction process is used to extract a boundary for the points in each extracted plane. The points are projected into the plane to form a two-dimensional representation of the planar layer in the point cloud. The boundary is then calculated for that two-dimensional representation. The height values for the boundary is calculated by solving for the undefined coordinate in Equation 3.1,

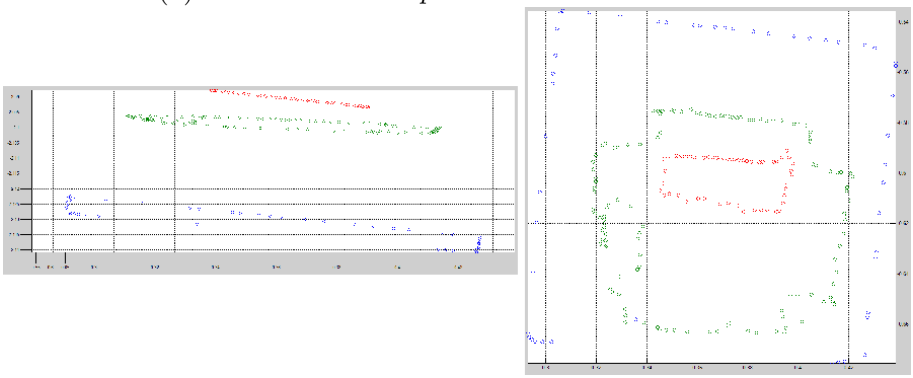
$$x_3 = \frac{-(ax_1 + bx_2)}{c} \tag{3.2}$$

Figure 3.10 shows an example of this process performed on the object planes shown

in Figure 3.8. Given a three-dimensional boundary for each plane, each boundary can be treated as a set of points which define a polygon. These polygons exist at varying heights, and represent layers of an urban structure. It is assumed that these layers are roof structures at different levels, and the building sides can be represented as vertical planes. For each polygon, starting from the topmost, a vertical plane is dropped to the next highest polygon. This is repeated until a plane is dropped to the bottommost polygon. This generates a primitive geometric model for the object point cloud, made up of planes with defined boundaries. Figure 3.11 shows an example of this primitive geometric model for the object shown in Figure 3.8, along side a hand-constructed CAD model of the same building. The process of iterative RANSAC plane fitting with Alpha Shape boundary extraction results in a primitive geometric model defined for a set of segmented regions within a point cloud.

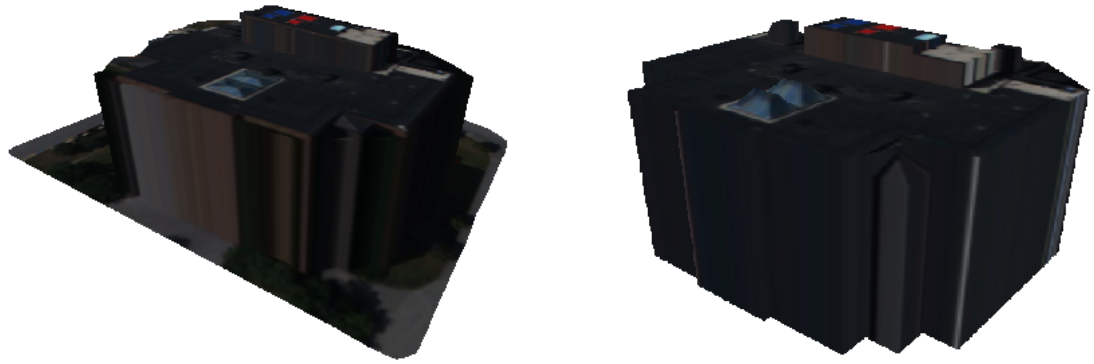


(a) *Three RANSAC plane boundaries extracted*



(b) *The boundaries project onto the three-dimensional plane*

Figure 3.10: *The detected boundaries of the extracted planes from Figure 3.8 (a) and those boundaries projected back into the three-dimensional plane to form a three-dimensional boundary (b).*



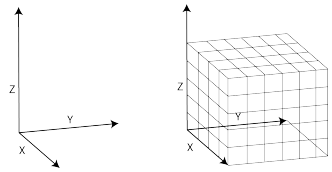
(a) *Derived primitive model* (b) *Handmade model*

Figure 3.11: *The derived primitive geometric model (a) for the object shown in Figure 3.8 alongside a hand-made CAD (b) model of the same building for comparison.*

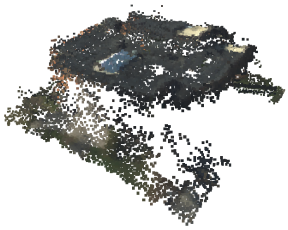
While this approach works in certain situations, it is limited to only plane-based surface estimations. Furthermore, even with the noise-reducing RANSAC approach, it is still significantly effected by noise. This method may work in some situations, but it is not a very robust approach to surface estimation from scene structure.

3.3.2 Voxel-Based Surface Estimation

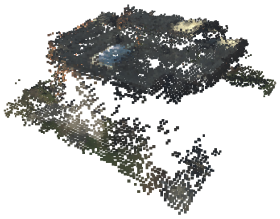
The best surface estimation method would be able to extract a facet-based representation of an object's surface, without imposing many constraints on the representation. Many methods exist which attempt to estimate the surface of unordered three-dimensional point clouds, such as Delaunay Triangulation, three-dimensional Alpha Shapes, or Poisson Surface Reconstruction [30, 15, 39]. However, these methods often have issues with noisy point clouds, which may contain large holes. This type of point cloud is fairly characteristic of three-dimensional structure which is automatically extracted from aerial imagery. By making a few assumptions about the characteristics of the point cloud, a simplified voxel-based method can be developed.



(a) *Voxelization of a continous space*



(b) *Unorded estimated structure*



(c) *Voxelized structure*

Figure 3.12: *Voxels are a discrete method of representing a continuous three-dimensional coordinate system. The process of voxelizing an unordered point cloud is inherently noise reducing.*

Voxelization is the representation of a continuous three-dimensional coordinate system as discrete volumes, akin to the two-dimensional representation of the pixel. The process of voxelizing an unordered point cloud is essentially a sampling operation, which is inherently noise reducing. This is an advantage when working with noisy three-dimensional structure, similar to what is often generated with aerial imagery. Conversion of an unordered point cloud to an ordered voxel cloud also allows for further noise reduction processes to occur.

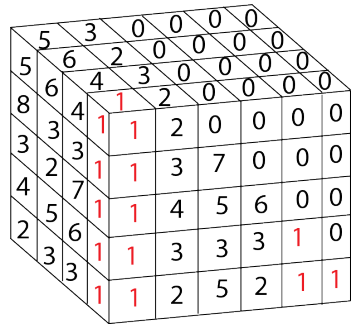


Figure 3.13: *Density filtering is an additional noise reduction process that voxelization can provide. In a situation as the one pictured here, voxels with only one point contained inside would be removed.*

Two noise reduction processes are performed beyond the inherent reduction in the voxel sampling. The first eliminates voxels that contain a low point density. The second process looks at a radius of voxels around each voxel center, and filters out voxels which do not contain a specified number of neighbors. While both these processes are very similar, they perform different tasks. The elimination of low point density voxels removes spurious three-dimensional structure which might have been caused by noise (Figure 3.13). The second process removes small clusters of points which would otherwise not be removed by the first process, a process known as radius filtering [6]. An example of this process is shown in Figure 3.14.

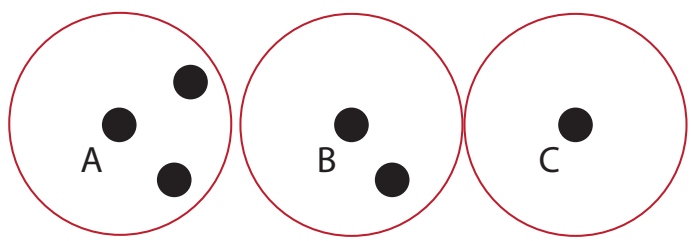


Figure 3.14: *Example of the radius search algorithm. Given a set radius and a threshold of at least two neighbors, voxel A would be accepted and voxels B and C would be labeled as outliers. If the threshold was one neighbor then A and B would be accepted and C would not.*

This work is focused on extracting and analyzing structure from aerial imagery. The structure of interest in this type of imagery tends to be man-made objects and larger structures, such as buildings. Structure of this nature tends to have similar features that can be exploited to make some assumptions and interpolate empty areas of the voxel cloud. The assumptions made here are derived from the Manhattan-world assumption [9], an assumption used by many researchers dealing with aerial reconstructions.

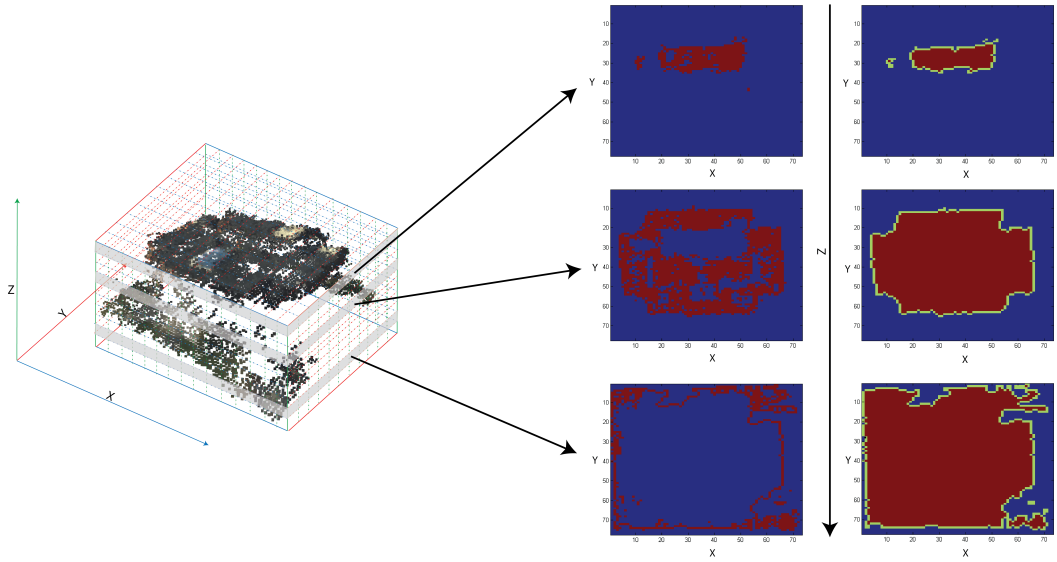


Figure 3.15: *Assuming that connected components in each X-Y orthogonal direction in the voxel cloud are likely part of the same surface, a top-down approach to voxel cleaning is taken. Each X-Y planar slice is treated as a binary image, and morphological cleaning and filling operations are performed. The boundary of the structure in each slice is identified and assumed to be the connection point to the Z orthogonal direction (*i.e. *the wall). The boundary is extended down to the next Z-level until the bottom level is reached.*

Using the Manhattan-World Assumption for Surface Estimation

The Manhattan-world assumption states that most objects contained in a scene are comprised of edges and planes which can be defined using three orthogonal directions [9]. A voxelized space is very conducive to representing objects using this assumption. This work assumes that for a single object, connected components in every voxel X-Y layer are likely part of the same surface. Structure is cleaned and interpolated in the X-Y orthog-

onal directions along the Z orthogonal direction from the top of the voxel cloud to the bottom. Processing each voxel level in this manner allows for surface interpolation and hole filling as well as boundary detection.

The voxel Z-level processing shown in Figure 3.15 is primarily executed through morphological operations, treating each Z-level as a binary image. An initial noise-reduction process, using the hit-or-miss algorithm, is used to identify and remove isolated voxels in the X-Y plane. After this, a morphological closing is performed using a 3-by-3 rectangular structuring element. Each isolated interior hole is filled. This is done by filling the exterior space, taking its inverse, and adding the inverted space to the cleaned voxels. Finally, a morphological closing then opening is performed, using a 3-by-3 rectangular structuring element, to clean the edges of the estimated surface.

The hit-or-miss transform is a simple pattern recognition algorithm using morphological operations. Specifically, it can be used to identify all pixels which match the shape of kernel K_1 and find all pixels which don't match the shape of K_2. Formally, the hit-or-miss transform is defined for an image I_z,

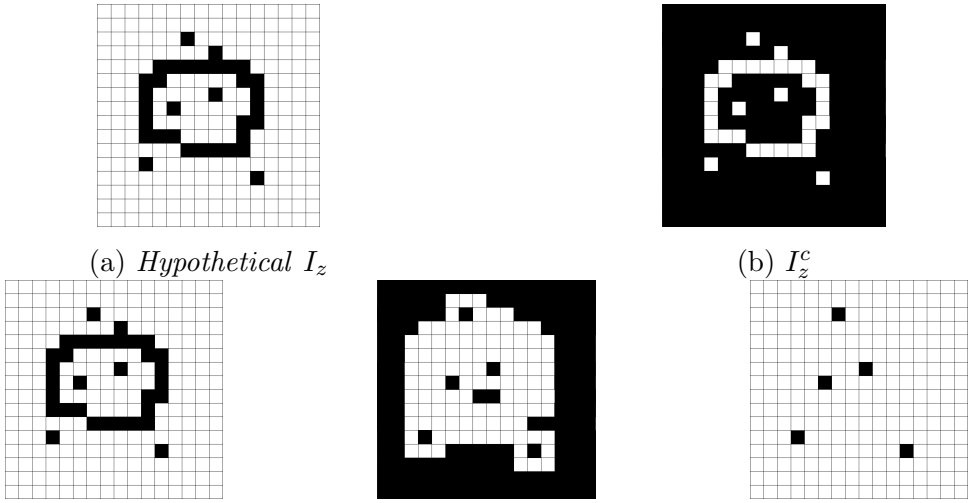
$$I_z \odot K = (I_z \ominus K_1) \cap (I_z^c \ominus K_2) \tag{3.3}$$

The hit-or-miss transform is the intersection of the erosions of I_z by K_1 and the compliment of I_z by K_2. The compliment of an image is where all nonzero elements become zero, and vise versa. This transform will identify specific pixels which satisfy an arraignment as defined by K_1 and K_2. This transform can be used to identify isolated pixels given the following kernels,

$$K_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.4}$$

$$K_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.5}$$

This process is shown step by step in Figure 3.16.



(a) *Hypothetical* I_z (b) I_z^c

(c) $(I_z \ominus K_1)$ (d) $(I_z^c \ominus K_2)$ (e) $(I_z \ominus K_1) \cap (I_z^c \ominus K_2)$

Figure 3.16: *The hit-or-miss transform can be used to identify isolated pixels in an image. Here, the transform is split into steps for visualization. The kernels uses here are the kernels presented in Equations 3.4 and 3.5*

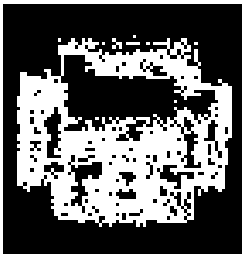
The identified pixels, such as the ones shown in Figure 3.16-(e), are then removed from the image I_z. This transform could be used to remove other patterns, however, for this work removing isolated pixels is sufficient for noise cleaning. The next step in the X-Y voxel plane cleaning process is a closing, using a rectangular structuring element R. This process fills small holes in the plane. Large holes are identified by filling the exterior space of I_z, using a small structuring element S and initializing from the exterior space of I_z. The compliment of the result is taken to identify the holes. These identified holes in I_z are then filled. The final step is a closing then opening operation using R to clean the edges of the estimated surface. This entire process is described as,

$$I_z' = (I_z - (I_z \odot K)) \bullet R \tag{3.6a}$$

$$X_k = (X_{k-1} \oplus S) \cap I_z' \tag{3.6b}$$

$$I_z'' = \left(\left(I_z' \cup X_k^c\right) \bullet R\right) \circ R \tag{3.6c}$$

where X_0 is pixel from the exterior space of I_z', and is iterated from 0 to k and X_k^c is the compliment of X_k. X_k is iterated until $X_k = X_{k-1}$. The resulting I_z'' is the cleaned X-Y voxel plane. Figure 3.17 shows this process for one of the levels in Figure 3.15.

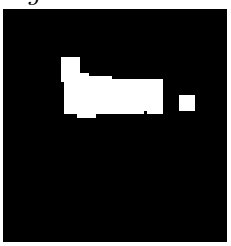


(a) *Inital X-Y voxel plane*

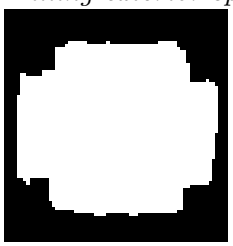
(b) *Hit-miss to remove isolated voxels*

(c) *Closing to remove small holes*

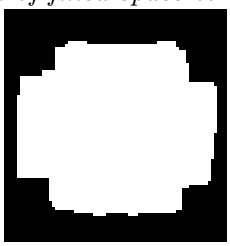
(d) *Filling exterior space*



(e) *Inversion of filled space to identify holes*



(f) *Holes filled*

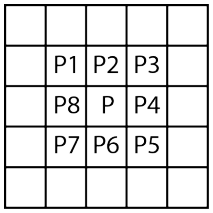


(g) *Closing and Opening to clean edges*

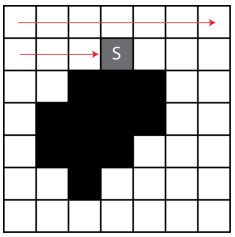
Figure 3.17: *The X-Y voxel plane is cleaned and interpolated using binary morphological operations. Initially a hit-or-miss algorithm is used to identify isolated voxels, then a closing is performed to fill small holes. The large holes are identified by filling the exterior space, and then inverting the filled space. The identified holes are added back to the closed image. Finally a closing and opening is done to clean the edges of the estimated surface.*

The boundary of I_z'' is identified using the Moore-Neighbor tracing algorithm. This algorithm does a raster search to identify a starting pixel P on the border of the estimated

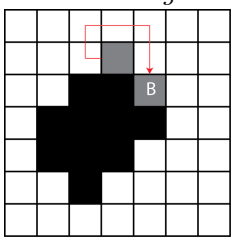
surface in I_z'', this pixel is added to the boundary set B. Then starting from the pixel preceding the current filled pixel found in the search. the Moore-neighborhood is searched in a clock-wise fashion to see if another filled pixel is found. If a filled pixel is found, this pixel is added to B and the search starts again from the preceding pixel in the most recent clockwise search of the Moore-neighborhood. This process is shown in Figure 3.18.



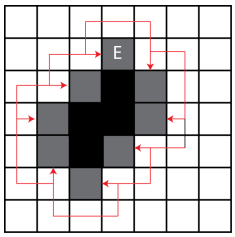
(a) *A Moore-neighborhood*



(b) *Find starting point*



(c) *Moore-neighborhood search*



(d) *Search iterated until S*

Figure 3.18: *A Moore-neighborhood is shown in (a) with the starting pixel in the upper right hand corner of the neighborhood. The starting pixel is found through a raster search, shown in (b). A Moore-neighbor search is performed, starting on the pixel prior to the starting pixel, the next pixel found in the search is added to the boundary set, B, as shown in (c). This search is repeated until the starting pixel is reached again, shown in (d).*

Once the set of boundary pixels B is found in I_z, each boundary pixel is then added to the following z-level,

$$I_{z-1} \cup \left(B \subseteq I_z''\right) \tag{3.7}$$

This process assumes that the boundary is the connection with the next z-level, or in terms of the object's surface structure, it is assumed to be the edge of the wall. This process is repeated for I_z to $I_{z=0}$. A pseudo-code for the entire process is shown in Algorithm 3.

Algorithm 3 Z-level surface estimation algorithm

R *is a rectangular structuring element*

for $z = maxZ$ to 0 **do**
  $I'_z = (I_z - (I_z \odot K)) \bullet R$

  $X_0 = p$ *Where, p is a pixel on the outside of the surface estimation in* I'_z
  while $X_k \mathrel{!=} X_{k-1}$ **do**
    $X_k = (X_{k-1} \oplus S) \cap I'_z$
  end while

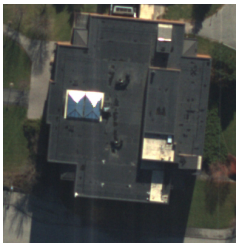
  $I''_z = ((I'_z \cup X_k^c) \bullet R) \circ R$

  Find $B \subseteq I''_z$ *using Moore-neighborhood boundary search*

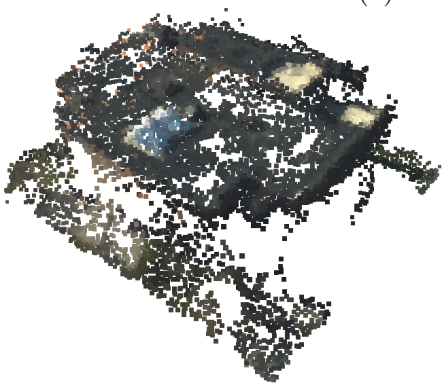
  if $z \mathrel{!=} 0$ **then**
    $I_{z-1} \cup (B \subseteq I''_z)$
  end if

end for

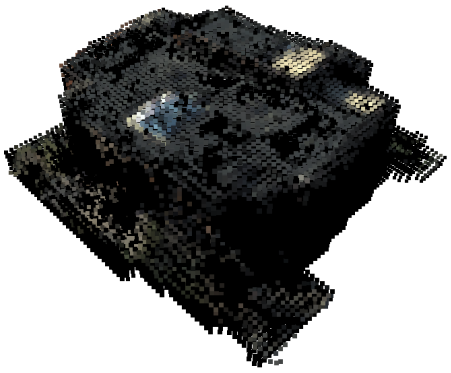
This algorithm estimates the surface of a noisy three-dimensional point cloud using a top-down voxel-based approach. An example of the estimated surface for the structure shown in Figure 3.15 is illustrated in Figure 3.19.



(a) *A view of the object*



(b) *The reconstructed points*



(c) *The estiamted surface voxel centers*

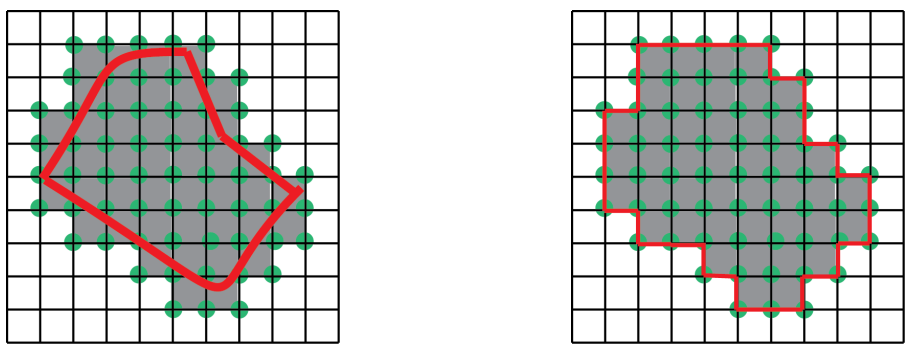
Figure 3.19: *An example of the result of voxel Z-level cleaning as described in Algorithm 3. This structure is the same structure used in Figures 3.15 and 3.17. An aerial view of the reconstructed building is shown in (a). The three-dimensional reconstructed structure is shown in (b), and the estiamted surface is shown in (c). The points shown in (c) are the centers of the estimated voxels. The voxel centers without color are the voxels which were interpolated in the estimation process.*

Extracting Facets from Surface Estimation

Given a voxel-based estimation of an object's surface, a facetized model is desired. There are many methods for estimating the facets of an ordered point cloud, also known as estimating the polygons of the voxel's isosurface. Methods such as marching cubes, dual contouring, and 2.5D dual contouring can be applied [43, 35, 76]. Each method attempts to estimate and facetize the isosurface of the voxel set. This work uses a moving-least squares (MLS) variant of the marching cubes algorithm to estimate the facets for the set of estimated voxels.

Marching cubes (MC) is a simple algorithm that traverses the isosurface of a set of voxels, testing each of the eight corners of each surface voxel to estimate the type of poly-

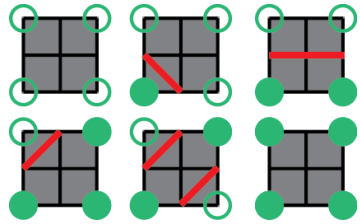
gon that should replace the isosurface. The concept behind MC is that the isosurface itself must be partitioned in some manner that will create a smooth surface. If the isosurface alone was used to generate a set of facets, the resulting model would have a very jagged surface. In order to better describe the three-dimensional surface fitting process, the two-dimensional case is examined. Given a set of two-dimensional pixels sampling a shape, as shown in Figure 3.20-(a), the pixels that are not surrounded by other points could be considered the isosurface (isoline) of the figure. If each of these points were connected, as shown in Figure 3.20-(b), a jagged representation of the figure would be made.



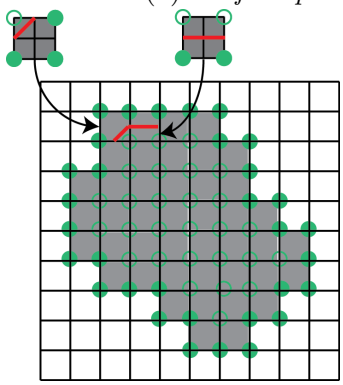
(a) *2-D samples of a figure* (b) *Connected surface points*

Figure 3.20: *A sampling of a two dimensional figure, shown in (a) can be used to estimate the true surface of the figure. The simplest method would be to connect the edges of each sample for which the area is not fully surrounded by other samples, as shown in (b).*

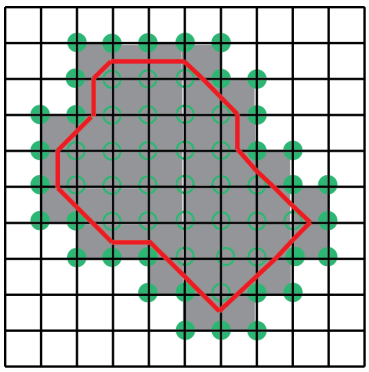
The marching cubes approach to this problem defines a set of surface primitives which are combined in such a manner so as to create a smoother approximation of the surface. These primitives are defined by the corners of the sampling structure, for example, in the two-dimensional case these are the corners of pixel. A set of unique shapes can be made by determining if the corner of the sample is contained in the isosurface, on, or above the isosurface. The sampling element is partitioned in such a manner that the shapes can be positioned at halfway points along the edge of the sampling element. Figure 3.21-(a) shows the set of surface primitives where corners on, beyond, or below the isosurface are identified. Rotated variants of these surface primitives are combined together to form a surface estimation, shown in Figure 3.21-(b).



(a) *Surface primitives used for surface estimation*



(b) *Two steps of the estimation*



(c) *Surface estimation using surface primitives*

Figure 3.21: *Using a set of surface primitives (a), and rotated variants of these primitives, a more accurate representation of the shape's surface can be created (b). The fully estimated surface is shown in (c). The corners of the primitives in (a) are either below or above the isosurface. The filled circles represent pixel corners beyond or on the isosurface and the unfilled circles represent corners below the isosurface.*

The same process can be extended to three dimensions by using the eight corners of each voxel. Figure 3.22 shows the set of fifteen polygon primitives which are used to estimate the voxel cloud surface.

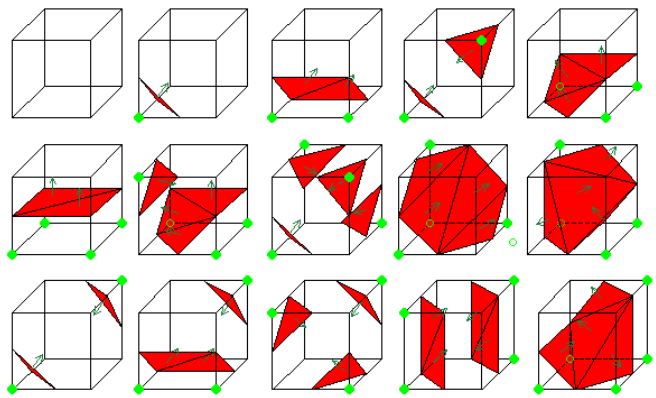
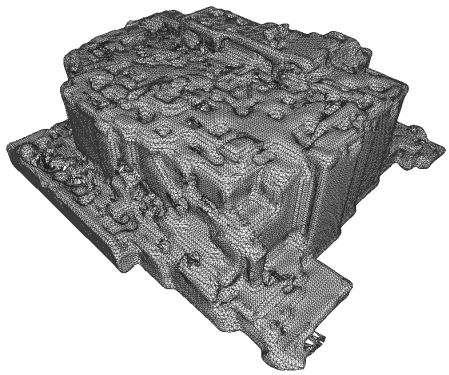
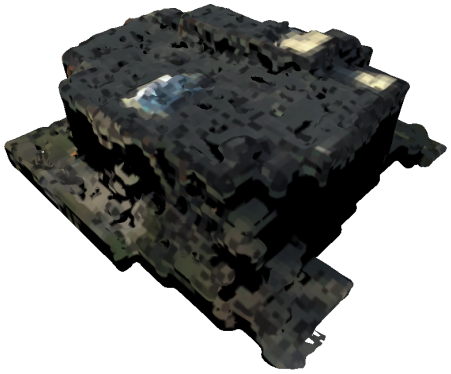


Figure 3.22: *The three-dimensional extension to the two-dimensional set of primitives shown in Figure 3.21-(a). These fifteen primitives are combined to estimate the surface of the set of voxels contained on the estimated surface.*

Marching cubes provides the necessary primitives to create a fully facetized three-dimensional model from the voxel-based surface estimation process. Figure 3.23 shows the facetization of the structure from Figure 3.19.



(a) *A facetized model*



(b) *The facetized model colored*

Figure 3.23: *The marching cubes algorithm is used to facetize the voxel surface estimates from the voxel cleaning and estimation process shown in Algorithm 3. This shows an example of the facetization process of the structure shown in Figure 3.19. The facetized model is shown in (a) and a colorized model is shown in (b)*

3.3.3 Constructing a Confidence Metric for Voxel-Based Estimated Surface Structure

The previous section described methods for extracting and estimating the surface of an object from discrete three-dimensional measurements. There were a number of assumptions made in the reconstruction process, as well as interpolation of surface estimates. Given the assumptions and interpolations, it would be advantageous to assign each voxel a confidence value. This metric would allow users to quickly determine regions of the estimated surface structure which may of been poorly estimated.

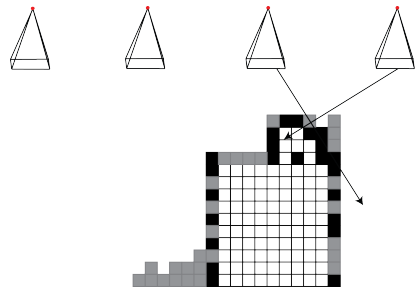
The confidence metric is calculated for each voxel in the estimated surface, and then transferred to it's corresponding facet. There are two properties of the estimated surface structure which are taken into account for a confidence measure, 1) the variance of color within each voxel, and 2) the proximity to a non-interpolated point. The variance of color within each voxel comes from measuring the color in the projected area of the voxel, for each image used to create the three-dimensional structure. Since many voxels in the voxel-based reconstruction process are interpolated, the proximity metric is used to reduce confidence in areas which have been largely interpolated with few regions of known structure. The known structure is defined as the initially filled voxels prior to any interpolation process (Figure 3.17-(a)). An energy function is defined and used as the confidence metric for this process, shown in Equation 3.8.

$$E(V) = \begin{cases} e^{-\frac{\sigma_V}{\omega_\sigma}} e^{-\frac{d_V}{\sigma_d}} & \sigma_V > 0, d_V > 0 \\ -1.0 & \sigma_V = 0 \\ 1.0 & d_V = 0 \end{cases} \tag{3.8}$$

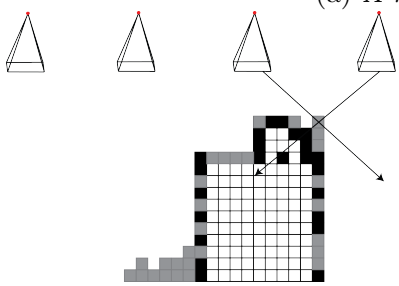
where σ_V is the variance within voxel V, and d_V is the distance from V to a non-interpolated voxel. The variables ω_σ and ω_d are used to adjust the weighting of each value, such that the weighting is approximately equal. The value of $E(V)$ will range from 0 to 1, where 0 energy represents the lowest confidence possible. If σ_V is equal to zero, this means that there were zero unoccluded projections of V into the imagery for color measurement. In nadir-looking imagery this could occur on vertical structures. This case is indicated by assigning the voxel confidence to -1.0. If the distance d_V of V is equal to zero, this means that V was not interpolated, and therefore has the highest confidence.

This means that the confidence in the voxel-based structure is only high when the interpolated voxel is near a non-interpolated voxel, and the variance of color in the interpolated voxel is very low. If the interpolated voxel is far away from the known voxels, or if

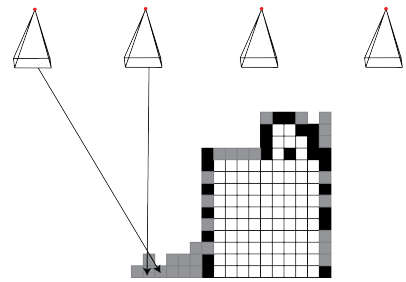
the interpolated voxel color variance is too high, a low confidence is produced. Situations representing each are shown in Figure 3.24. Part (a) of this figure shows a high confidence measure for the voxel in question. Figure 3.24-(b) shows a high color variance situation, where the projected voxel is going to likely have the color of both the surface structure and the ground structure in the same voxel. Figure 3.24-(c) shows a situation where the color variance will be high, but the distance to an uninterpolated voxel is low.



(a) *A high confidence voxel*



(b) *Low confidence caused by large* σ_V



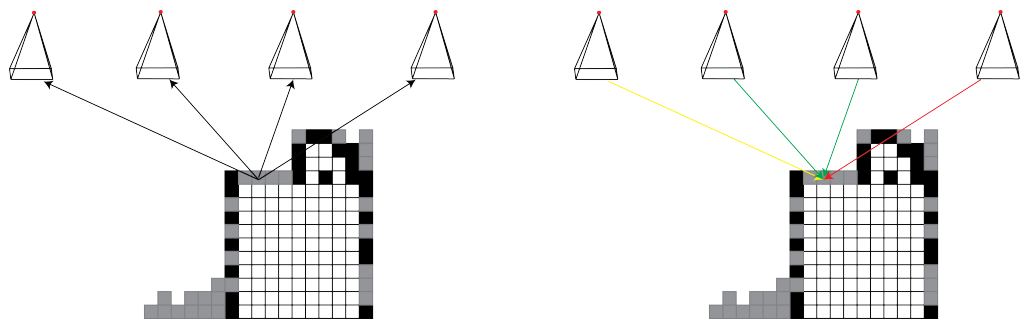
(c) *Low confidence caused by large* d_V

Figure 3.24: *This shows three situations of the voxel confidence metric shown in Equation 3.8. In each situation, the voxel in question has it's image projected rays shown. The uninterpolated voxels for each situation are shown in black, while the interpolated voxels are shown in grey. The situation shown in (a) represents a high confidence situation, both the color variance and* d_V *are low. Situation (b) shows one in which the color variance is likely going to be high, because one of the projected rays does not intersect the structure. Situation(c) shows one where the color variance may be low the* d_V *value is high, resulting in a lower confidence.*

Occlusion Handling

The value calculated through Equation 3.8 is calculated for every surface voxel. However, there may be situations where a surface voxel is occluded in one image, but unoccluded in

another. Occlusion is mapped through a forward and backward projection of the voxel, as shown in Figure 3.25. The measurement points of each projected voxel are cast back into the voxel space through ray-casting techniques. The number of intersected voxels prior to the projected voxel are counted, and the voxel is considered occluded in this image if this number is greater than one. This eliminates error in the σ_v measurement which would otherwise be caused by occluding surface structure.



(a) *Voxel projected image measurments* (b) *Occlusion test by ray casting*

Figure 3.25: *Situations arise where surface structure may be visible in one image, but not in another. A large amount of error would be introduced to σ_V, If the color measurements were taken for voxels in occluded images. Occlusion tested by projecting each voxel into every image, and then casting that projection back into the voxel space. The number of voxels which are intersected prior to the projected voxel are counted. If this number is greater than a defined threshold, then the voxel is considered occluded in that image and the measurement is removed. In this situation the far right and far left projections would be removed.*

The occlusion handling process, as well as the color measurement process requires that the voxel can be projected back into the imagery. This is possible using the derived camera projection matrix, however, the structure of interest is one that has been transformed via a georegistration transform (Section 2.5). In this case, the projection matrices must also be transformed such that they are in the new coordinate system. Appendix A describes how to perform this transformation.

3.3.4 Using a Depth Map for Structure Segmentation

It has been shown in many sections in Chapter 2 that it is possible to use a camera projection matrix to project a three-dimensional point into an image. Attributes of this projection can also be applied to the image through this known correspondence. Every projected point corresponds to a three-dimensional point some distance away from the camera center, $\mathbf{C}$. A value can by added to the projected point which is equal to the distance between the camera center and the three-dimensional point, $\|\mathbf{C} - \mathbf{X}\|$, or the depth of the point from the camera. Calculating this value for every projected point in a single image will yield a sparse depth map. An example of such a map is shown in Figure 3.26.

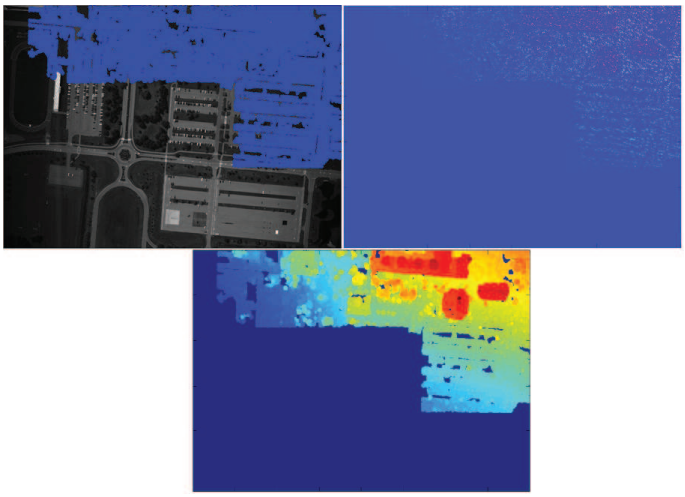


Figure 3.26: *A point projection onto a single image, the calculated sparse depth map, and the interpolated depth map.*

The sparse depth map can be interpolated by using the morphological dilation operator to generate a full depth map for the image, an example of this is also shown in Figure 3.26. Using a structuring function that is at least as large as the smallest distance between two points in the sparse depth map, a good representation of the full depth map can be obtained. This information can be used to detect object of interest in the point cloud using a local thresholding technique. This is done by searching the image for areas which have a large difference in depth relative to their neighbors. Each area can be individually segmented from the depth map as a possible structure using the method of connected

components [28], as shown in Figure 3.27.

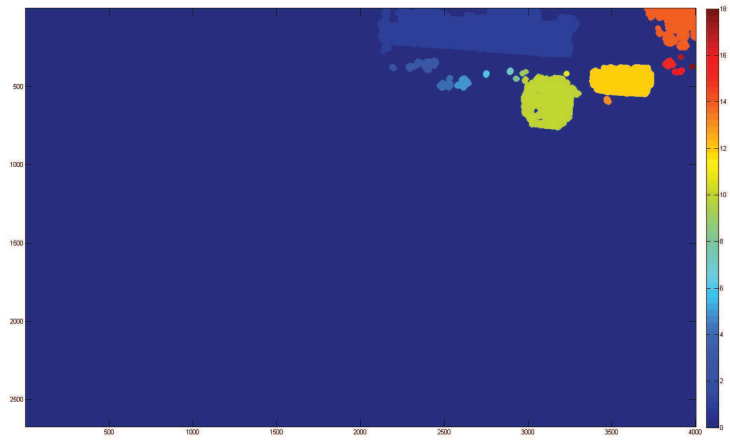


Figure 3.27: *Segmentation of objects from the depth map using local thresholding and connected components, each object is labeled by color.*

Three-dimensional points from the point cloud can be segmented using the areas in the segmented depth map shown in Figure 3.27 by simply determining which point projects into which region. This generates a set of segmented three-dimensional point clouds that are objects of interest. Further processing is then required to determine if the object is a man-made or vegetation structure.

Vegetation Removal

The objective in the modeling process is to extract and model man-made structures, therefore it is important to detect and remove structure which is generated from vegetation. This is possible by using both information from the RGB imagery and the depth to determine if the object is vegetation or man-made. Color and size of the point cloud can be used to identify vegetation. Each connected component is examined, and labeled as either vegetation or man-made. The average red, green, and blue digital count values from the color image are computed for each component region. The green-to-red and green-to-blue ratios are calculated, as shown in Equation 3.9. If either ratio is greater than unity, the object is determined to be vegetation, otherwise, it is labeled as man-made. This ratio

exceeding unity indicates that the amount of green color in the point cloud is larger than either the red or blue colors for this component, indicating it is likely vegetation.

$$\begin{aligned} \frac{\overline{DC}_g}{\overline{DC}_r} \geq 1 \vee \frac{\overline{DC}_g}{\overline{DC}_b} \geq 1 \Rightarrow vegetation \\ \frac{\overline{DC}_g}{\overline{DC}_r} \leq 1 \wedge \frac{\overline{DC}_g}{\overline{DC}_b} \leq 1 \Rightarrow manmade \end{aligned} \tag{3.9}$$

This process works for most green vegetation. A secondary process looks at the size of the point cloud for the remaining regions. If the size is small compared to the other regions then the small region is removed, discarded as an object of interest. This is done by calculating the average size of the regions and removing regions which fall below two standard deviations of the average. The reasoning is that small areas are either small patches of vegetation which was not detected by the first process, or that it represents some error within the segmentation process. Figure 3.28 shows the vegetation and small region removal process for the segmented objects shown in Figure 3.27.

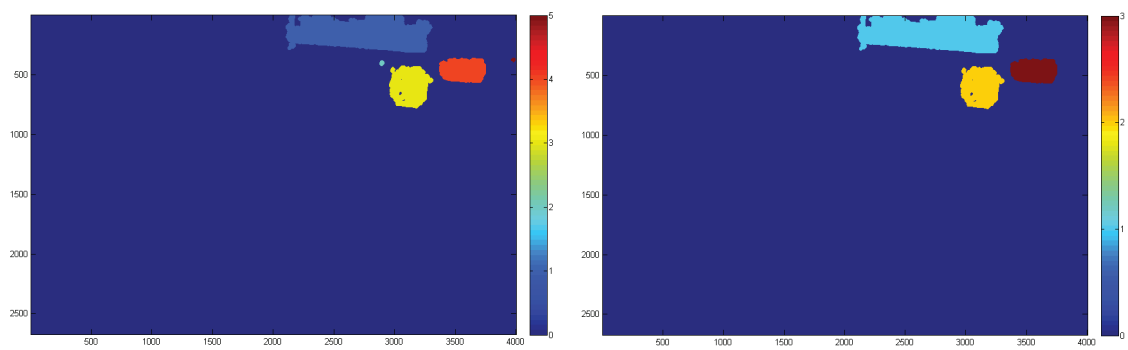


Figure 3.28: *Removal of vegetation regions and then the removal of small regions, from the segmented regions shown in Figure 3.27.*

The remaining connected components represent different regions in the source point cloud which can be go through the surface estimation process, described in the previous sections. Recently, other methods have been developed for identifying vegetation in a point cloud [59], based on distribution of point normals. While using normal distributions works well, often image-derived structure will contain a significant amount of noise. This

noise will cause errors in vegetation detection. The methods presented in [59] use a graph-cut based approach to vegetation segmentation. The graph weights are calculated using surface curvature and normal variation.

3.4 Surface Attribution and Classification

To this point the structure modeling process has been similar to the goals of photo-realistic modeling. While the approaches presented here are primarily for aerial nadir-looking imagery, they are similar in their goals. Three-dimensional surface structure can be used as an additional source of data when trying to discriminate material types on the surface of a target of interest. This structure can be combined with additional imagery to assign spectra to facets for modeling purposes. If additional information is not available, the original R,G,B spectra used to generate the structure can be further analyzed to classify possible materials on the surface of the structure. Each of these methods lead to a better understanding of the structure under analysis, and could be used as input to a physical modeling process.

The goal of this work is to examine methods of exploiting the three-dimensional structure generated from near-nadir aerial imagery. Ultimately, the output of these processes could feed a larger physical modeling process, which could have many different applications. The physical modeling software which inspired this work, and many other works [57], is known as the Digital Imaging and Remote Sensing Generation (DIRSIG) model. This model was created by researchers from the Digital Imaging and Remote Sensing (DIRS) laboratory at the Rochester Institute of Technology [58]. DIRSIG is a first principles synthetic image generation model which can produce imagery ranging from the visible to thermal infrared spectrum. The synthetic image generation process requires a spectrally attributed three-dimensional model of scene to be imaged. These attributed models are created by hand and require a large amount of man-power to model large scenes [29]. Consequently, rapid modeling of any scene is impossible. If this were possible, it would open up a large number of applications in process modeling, that is, the near real-time modeling of scenes.

Three-dimensional surface structure can be incorporated into segmentation and classification algorithms to perform a three-dimensional spatial-spectral classification. This type of classification could be used to discriminate materials on the surface of the esti-

mated structure. It can also be used to identify distinct materials which could be manually attributed with properties. These manually attributed models could be used in physical simulation software such as DIRSIG. While this process is partially manual, it is far faster than creating each model by hand.

This section will cover processes of further analyzing surface structure, under two scenarios. Given the addition of hyperspectral imagery of the same scene, taken at the same time, surface estimation facets can be directly mapped to spectra from this imagery. If the hyperspectral imagery is calibrated and atmospherically compensated such that the data is in reflectance space, then the facets can have reflectance measurements mapped directly on to them. This allows for direct spectral modeling of these structures. The second scenario considered, is if no additional information was available. This would mean only the R,G,B spectra of the high-resolution color imagery used to generate the structure would be available for spectral analysis. Considering the difficulties in estimating a full reflectance spectra from R,G,B values, a semi-automated approach for identifying surface materials is taken. The spectra combined with the surface model can be used to develop a classification algorithm. The identified clusters of spectra on the surface estimation can be used to define classes for manual material attribution.

3.4.1 Reflectance Attribution Through Hyperspectral Imagery

Incorporating additional spectral information along side the R,G,B information, which is contained within the original imagery, could prove to be beneficial in both material identification and object reconstruction. Spectral information is necessary for physical modeling of scenes. In order to use spectral imagery with image-derived surface structure, the high resolution imagery used to create the point clouds must be registered with the spectral imagery. Since image derived point clouds can be reprojected into the imagery, they can also be projected into any other registered data. Registration of high resolution imagery taken from a framing camera and lower resolution imagery taken from a line-scanning camera is done by taking each image to a common coordinate system. This is done through orthorectification. The process of orthorectification removes the distorting effects of terrain within an image, projecting the image as though it was on a flat plane orthogonal to the imaging direction. The terrain must be predefined using a digital elevation map (DEM), which provides a specific height above the geoid for a particular geographic position at a given resolution. By using a geographically accurate data source,

orthorectified imagery can provide a geographic position for every pixel within an image.

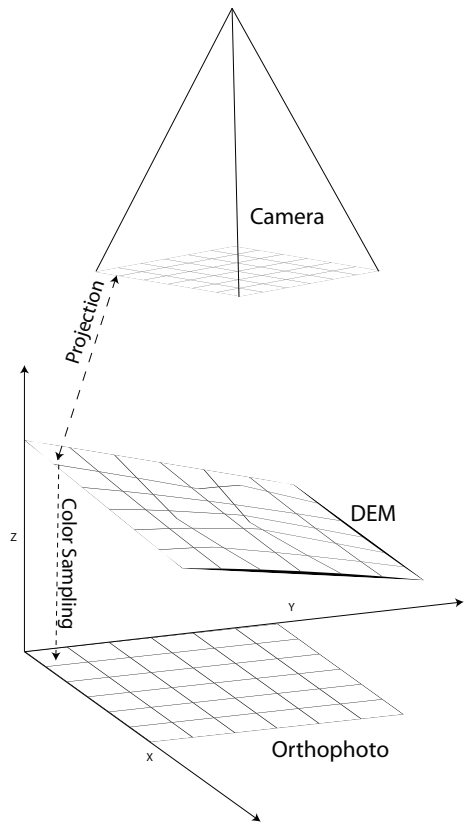


Figure 3.29: *The process of orthorectification through direct georeferencing. Using the known camera position and orientation, along with a DEM, terrain points can be projected into the camera and sampled to generate an orthophoto.*

Orthorectification Using Direct Georeferencing

The process of orthorectification removes the distortion caused by the movement of the sensor as well as the elevation of the terrain. This is performed using a digital elevation map (DEM), which contains terrain elevation and location information. The DEM is sampled to to have a corresponding terrain elevation and location point for each pixel in the unorthorectified image. The camera position and orientation information is used to project each terrain point into the image. This projection provides an image point which

can be used for color sampling. The sampled data is then placed into a raster array corresponding to the sampled DEM. Performing this operation for every point in the sampled DEM generates the orthophoto. This process is shown in Figure 3.29.

Prior to accurate GPS and IMU systems, the camera position and orientation was calculated through camera resectioning [73] using ground control points. The information from the GPS and IMU systems provides enough information to create a camera projection matrix (such as the one shown in Section 2.1.2), which allows for the projection of three-dimensional points, in this case the DEM values, into the image. This allows for significantly faster and cheaper creation of orthophotos without the use of ground control.

Mapping Image-Derived Point Clouds to Orthorectified Imagery

In this work, the high resolution imagery used for point cloud derivation is orthorectified. A mapping process is performed to allow for the projection of the derived image points into the orthorectified image through the original image. This also allows for the registration of the image derived point cloud with all other georeferenced data sources.

The map is generated using a color image of the same dimensions as the original imagery. One channel contains the physical horizontal pixel locations, the other channel contains the vertical pixel locations, and the third channel is unused. This image is orthorectified using the same parameters as the corresponding original image, the resulting image can be used as a mapping between the orthorectified image and the original image. In order to remove any error from sampling, nearest neighbor sampling is used, an example of this process is shown in Figure 3.30. The map image is essentially a look up table for pixel locations from the original image.

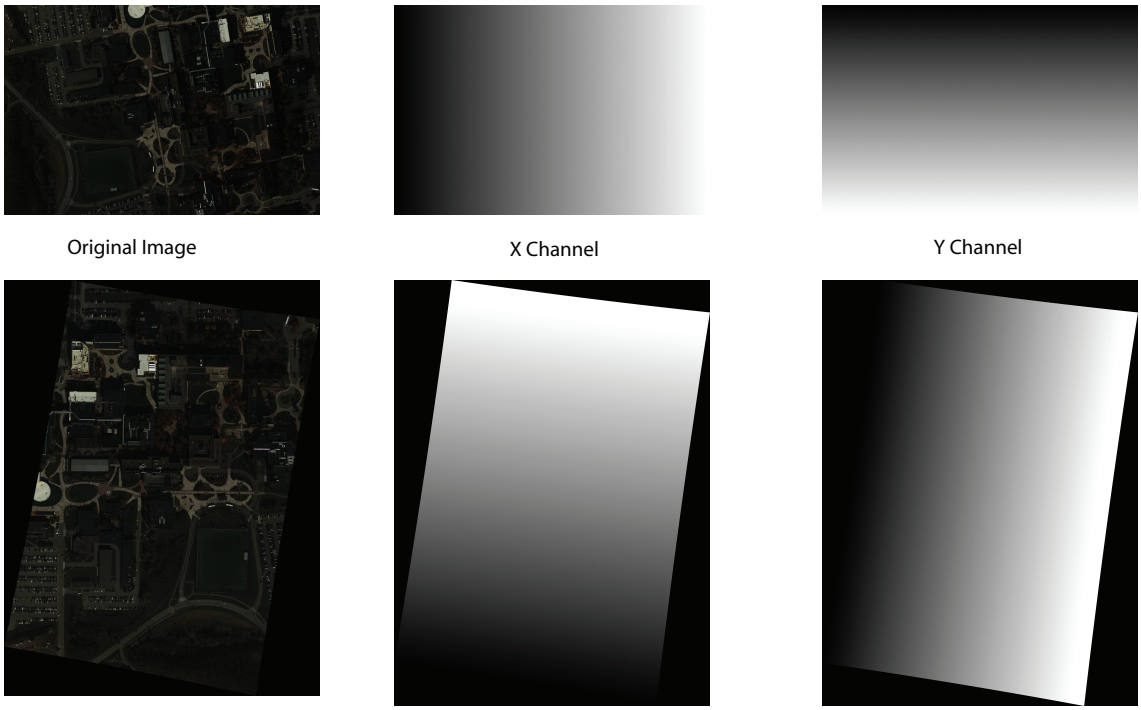
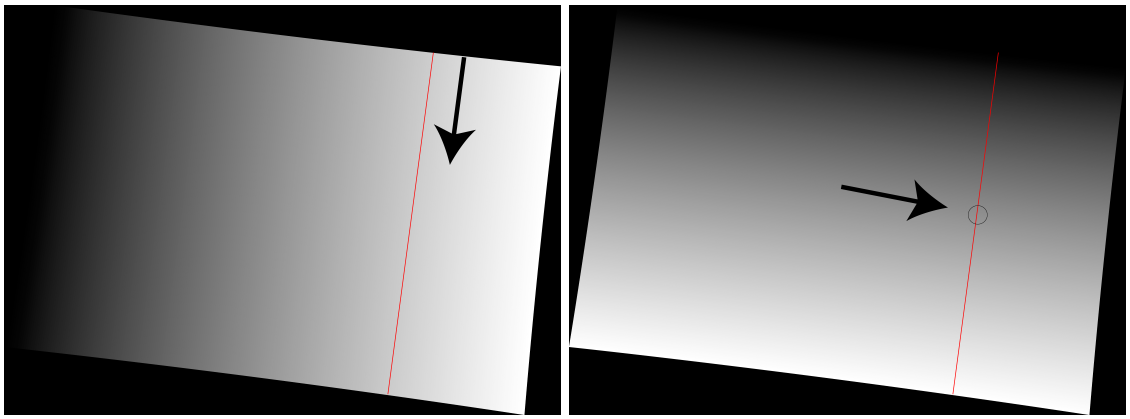


Figure 3.30: *An example of the orthorectification mapping process. The top row is pre-orthorectification and the bottom row is post-orthorectification. These maps are used to track the position of the original image pixels into the orthorectified image.*

Efficient Searching Through the Orthorectification Map

Searching a two-dimensional map for a specific pair of (x, y) values for every mapping would become computationally expensive. A couple of properties of these orthorectifcation maps can be exploited to increase the speed of the map search. The search is done in two steps, the first step searches through the x-Channel, and the second searches through the y-channel. Given the format of the orthorectification maps, it can be assumed that any one x or y value will be adjacent to one or more pixels with the same value. Using this assumption, a raster search is done to find the first instance of a value x in the x-channel. From this pixel, each of the eight pixels surrounding this pixel is searched for the same value of x. This process is iterated until every instance of the value x is found. This constitutes a set of pixels P_x in which the corresponding y value must be contained. A second search is done on just the set of pixels P_x, in the y-channel, for the value y.

This pixel position (X, Y) is the corresponding pair that can be used for mapping the un-orthorectified pixel position (x, y), to the orthorectified image. Figure 3.31-(a) shows an example x-channel with one x value highlighted. These pixel locations are then searched in the t-channel for the corresponding y value, shown in Figure 3.31-(b).



(a) A search along all equal x values (b) A search along these values for y

Figure 3.31: *A two-dimensional search through the x and y-channel maps would be very computationally expensive. An efficient method of searching can be done by exploiting the fact that each unique x value is always adjacent to another equal x value. Finding every x value can be simplified using this property of orthorectification maps, shown in (a). Once every unique x pixel is found, these pixels are searched in the Y-channel for the corresponding y value.*

Model Attribution

With a one-to-one mapping of hyperspectral data to R,G,B pixels in the original imagery, the estimated surface structure of an object can be directly attributed with hyperspectral reflectance data. This is performed on a per-facet basis, where each facet is assigned a spectra. The spectra is assigned by projecting the facet into the original imagery, and then using the orthorectification map registration process to map to a reflectance spectra, which is then attributed to the projected facet. The average spectra of the projected area is used for attribution. In terms of data representation, each facet is assigned a different material, and each material corresponds to a single reflectance spectra in the hyperspectral imagery. This is conceptually shown in Figure 3.32.

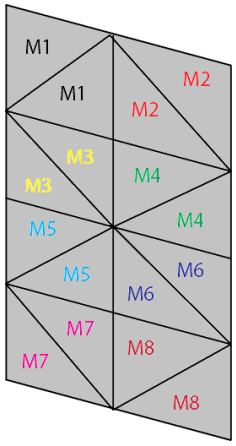


Figure 3.32: *Each facet has a spectra assigned to it via projection through the orthorectification map process.*

3.4.2 Surface Material Segmentation with R,G,B Spectral Information

In many cases atmospherically compensated hyperspectral imagery will not be available for reflectance attribution. Determining such information for just R,G,B spectral information can be a significant challenge. From this perspective, a semi-automated approach for material identification and attribution is taken. Using the R,G,B spectral information combined with spatial information from the three-dimensional estimated surface structure, segmentation algorithms can be created. These classification algorithms can be used to identify distinct clusters of materials on the surface of the three-dimensional structure. These clusters can be treated as separate classes, which a user can then use to assign material information.

One approach for segmentation, using graph theory, is known as the normalized cuts algorithm. This algorithm can be used to identify and segment structures. The weighted graph can be built from the three-dimensional points in the reconstructed structure, and the weighted connections can be created using a spatial-spectral approach. Another type of segmentation algorithm which is examined, takes a region-growing approach to identifying segments. By seeding the initial segments with a distribution of points across the structure, these seeds can be grown and merged within a specific set of spatial-spectral rules in order to identify unique segments on the surface of the structure. These two approaches are examined in this work.

Segmentation Using Normalized Cuts

It has been shown that using graph-based approaches for classification or segmentation with high spatial and spectral resolution data produces good results [22]. Similar graph-based approaches have been successfully used to segment and model three-dimensional point clouds with and without additional information [63, 60, 66]. It follows that utilizing spectral-graph theory for segmenting three-dimensional point clouds is possible with good results.

Normalized cuts (NC) is an algorithm which finds the minimum normalized cut in an undirected weighted graph. Often the minimum cut in a graph will provide an undesirable cut. This algorithm weights each segment by the total weight of all edges connected to that segment, thereby reducing the impact of larger segments in the minimization process. The normalized cut can be calculated using spectral graph theory [33], a detailed description of Normalized Cuts is shown in Appendix B.

The weights for each edge in the graph representing the point cloud is defined in Equation 3.10. The weights are in the form of an $N \times N$ matrix, where N is the total number of points. Weights are calculated for k points surrounding every point, the surrounding points are determined using a k-nearest neighbor approach. The minimum cut is recursively solved in order to segmented multiple classes.

$$w_{i,j} = \begin{cases} e^{\frac{-\left|\mathbf{I}_i - \mathbf{I}_j\right|}{\sigma_I}} \cdot e^{\frac{-\left|\mathbf{X}_i - \mathbf{X}_j\right|}{\sigma_X}} & \forall j \in k_i \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

Using a color attributed point cloud, I will be the vector of R,G,B values for points i and j. The value $\mathbf{X}$ is the vector of x, y, and z values for i and j. The values of σ_I and σ_X control the sensitivity of the similarity metric. Figure 3.33 shows an example of a segmented point cloud using Normalized Cuts. A simple build segmentation is shown in Figure 3.33-a, where there are very few materials and significant spatial variation. In Figure 3.33-b, a slightly more difficult segmentation is shown. In this point cloud there are multiple materials in the building roof which are difficult to identify using just the surrounding structure, shown in the circles. Due to the spectral similarity metric used in the normalized cut algorithm, two of the materials were totally segmented, and one of the materials was partially segmented.

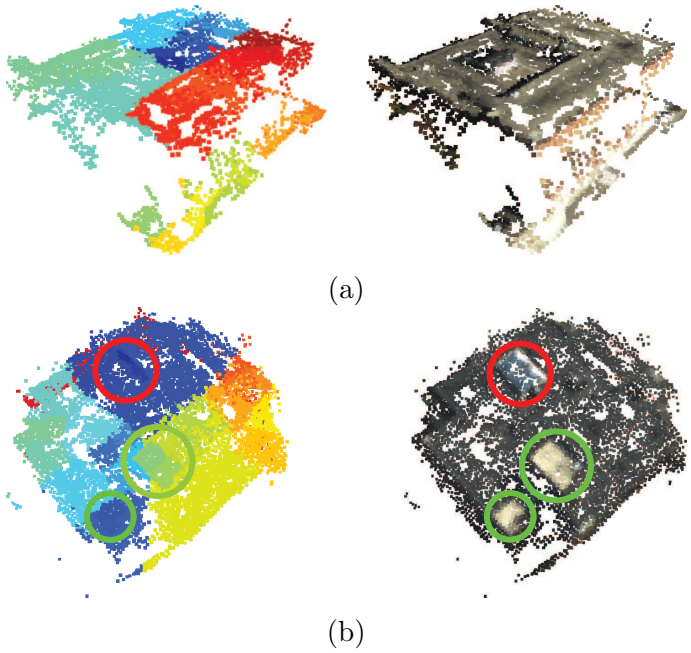


Figure 3.33: *A point cloud segmented using Normalized Cuts. The original point cloud is on the right and the segmented point cloud on the left (Colors corresponding to different segments). This segmentation process can segment different materials from a point cloud using R,G,B attributes.*

This segmentation process is capable of segmenting multiple different materials from a point cloud with R,G, and B attributes. The spatial segmentation is performed in all dimensions, therefore materials that are large may still be segmented into different parts due to their size. A secondary recombination process needs to be performed in order to merge segments which belong together. This approach can be extended to the voxelized estimated surface (Section 3.3.2), which would allow for a quicker k-nearest neighbor search.

While the normalized cuts segmentation approach is very powerful, it does not scale well with the potential size of reconstructed point clouds. Solving for the minimum cut requires the repeated solving for eigenvectors of an N-by-N matrix, where N is the number of points in the point cloud. This value can become very large, depending on the structure. A better approach would be able to exploit spatial-spectral relationships, but also be able to scale to large structures.

Segmentation Using a Region-Growing Approach

Region growing is another method for performing a spatial-spectral segmentation. This approach grows segments starting from seeds, based on some similarity criterion.

In many segmentation algorithms, it can be useful to partition the lightness from the chroma information. This partitioning can be used to group colors which may have the same chroma but different lightness. A practical example of this situation is a shadowed material on a structure's surface, the material class should still be the same. For this reason, the colorspace used for this segmentation process is the hue, saturation and lightness (H,S,L) color space. Through comparison of just the hue and saturation, shadowed effects can be minimized.

The region growing segmentation algorithm requires seeds to start growing segments, these seeds should be chosen such that the total number of segments is minimized. The curvature of every three-dimensional point is calculated relative to it's neighbors, and then each point is sorted from lowest to highest curvature. The local principle curvature is calculated for each point, through principle component analysis of the distribution of point vertex normals [75].

By selecting the lowest curvature points as seed points, the growth will originate in the structurally flat areas on the object and reduce the total number of segments [6]. Surface normals can also be computed and compared for the region growing process. By making sure the surface normals for each segment pointing in the same direction (within some tolerance), the region can be prevented from growing beyond a point on a structure where there is a change from a horizontal surfaces to vertical surfaces (*i.e.* from the structure's roof to the walls).

The region growing process compares two properties between three-dimensional structure when growing from seeds. The first property is the angular difference between the hue/saturation vectors, and the second property is the angular difference between the three-dimensional normals of the points. Seeds are recursively added from the minimum curvature list and grown. Segments are merged, and if the grown segment it too small, it is removed. This is done until all points are part of a segment, or have been removed. Neighboring segments are merged using the same criterion. A visualization of the possible outcome for a set of surface points is shown in Figure 3.34.

(a) A hypothetical surface point (b) The result of the region growing process

Figure 3.34: *A region growing process is used for segmenting the surface points through comparison of spatial-spectral properties. The set of points are first sorted according to minimum curvature. From this list, seeds are chosen and the neighbors of these seeds are compared using two tests. The first test checks the angular difference between the hue/saturation vector of the two points. The second test checks the angular difference between the three-dimensional normals of the two points. If both these tests are below a set threshold, then the neighboring point is added to the segment. Part (a) of this figure shows an example colored surface set of points, and part (b) shows the hypothetical segmentation.*

The region growing process works well for segmenting points all over a given structure. The angular requirement of the set of normals prevents the regions from growing over edges. This is an advantage in the presence of color noise. However, there could be segments on surfaces with opposing normals that should be part of the same class. Therefore, a final k-means clustering is done on the average hue and saturation of each segment. This reduces the overall number of segments and yields a result such as the one shown in Figure 3.35.

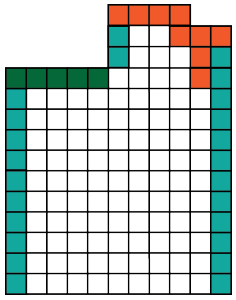
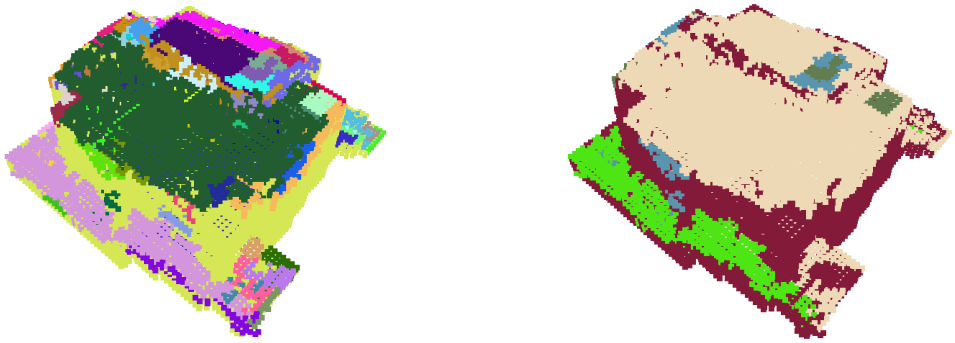


Figure 3.35: *The results shown in Figure 3.34-(b) provides a good segmentation, but a number of surfaces which should belong to the same class are segmented differently. Therefore, a final k-means segmentation is done on the average hue and saturation values for each segment. The clustered segments are merged and the hypothetical result of this is shown here.*

Taking a region-growing approach to segmentation may be a simplified approach as compared to the normalized cuts approach presented in the previous section, however, this approach scales very well to a large number of points and can still provide a good segmentation. The results of the segmented point cloud shown in Figure 3.33-(b) segmented using the region growing approach are shown in Figure 3.36.



(a) Before final clustering (b) After final clustering (k = 5)

Figure 3.36: *The example results of the same object shown in Figure 3.33-(b). These surface points have been further estimated using the voxel-based estimation approach.*

3.5 Discussion

The methodology presented in this chapter builds on the foundation provided by Chapter 2, in order to estimate and analyze the surface structure of reconstructed three-dimensional points. These points are transformed into a geoaccurate coordinate system, then modeled through application of the Manhattan-world assumption, and finally the surface structure is analyzed to either attribute or classify surface materials. This results in a three-dimensional surface structure that has some form of material attribution, which can be used as the basis for a physical model.

A number of approaches are presented in this chapter, however, a specific set of approaches are recommended. An in-depth analysis of each georegistration approach is presented in Chapter 4, it will be shown that the *augmented camera model* transform performs the best and should be used for the georegistration. The voxel-based surface reconstruction method is recommended for surface estimation. It is not limited to specific primitive types, and can use the Manhattan-world assumption to interpolate surface structure. Finally, for R,G,B limited classification, the region-growing segmentation method is the preferred approach for segmentation. It is a simpler approach than the normalized-cuts method, but can scale easily to large point cloud sets. Combining each one of these processes with the SfM algorithms presented in Section 3.1, results in a full workflow to extract and analyze the materials on the surface of geoaccurate three-dimensional structure. The complete workflow is shown in Figure 3.37.

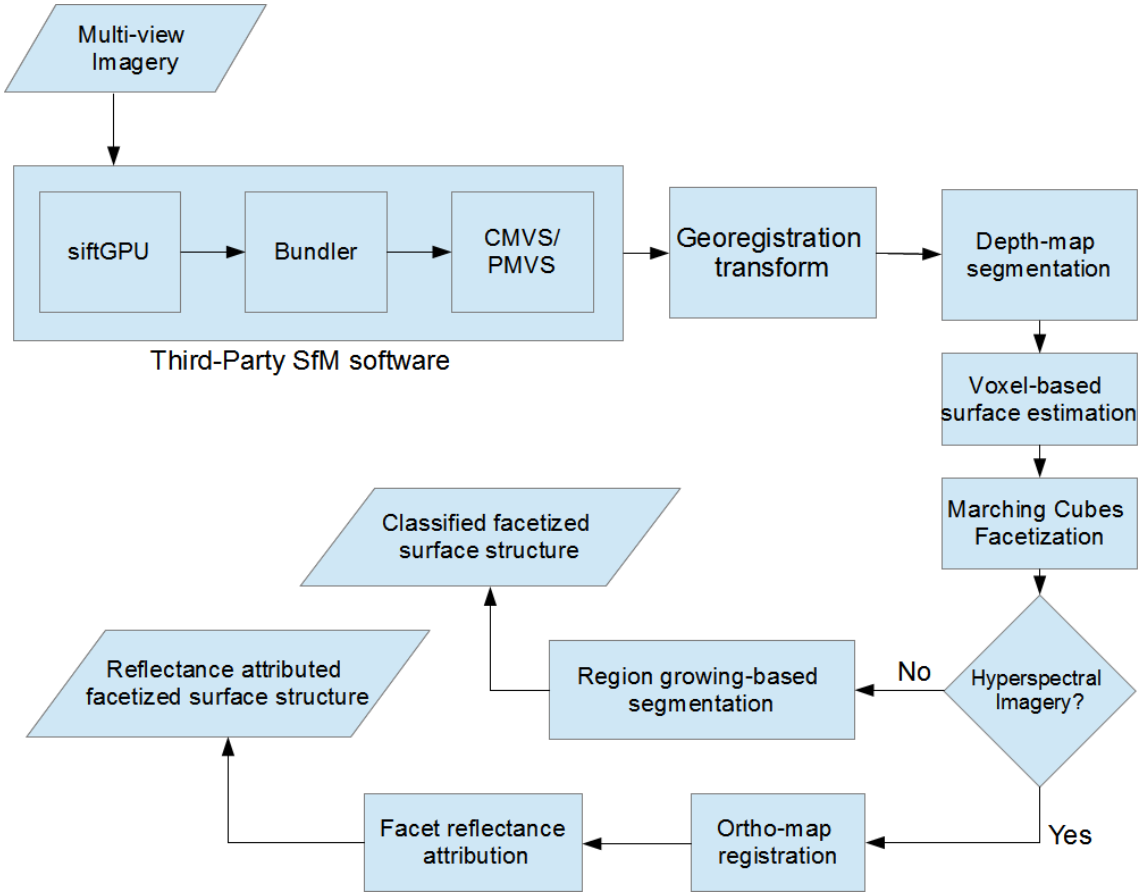


Figure 3.37: *This shows the full workflow to exploit three-dimensional structure to generate estimated surface structure which is either attributed with reflectance properties or classified for attribution.*

Chapter 4

Results and Analysis

This chapter will review the results produced by the methodology presented in Chapter 3. An analysis of each georegistration method presented in Section 3.2 is presented in Section 4.1. Examples of the voxel-based reconstruction method with the confidence metric calculation are shown in Section 4.2. Examples of surface structure attributed with reflectance spectra are shown in Section 4.3. Finally, an analysis of the region-growing based segmentation process is shown in Section 4.4. A description of the datasets used in this work can be found in Appendix C.

4.1 Georegistration Analysis

The ability to control every parameter within an image collection is necessary to fully understand how each georegistration method performs. To this end, a synthetic aerial image dataset was generated using the DIRSIG model described in Section 3.4. This work utilized a variant of Megascene 1, a hand-created three-dimensional scene of a small suburban region in northeast Rochester, New York [29]. Larger block-style apartment buildings were added to the scene to increase the depth of targets within the scene [37].

DIRSIG uses a ray tracer to calculate the sensor response at each pixel in every synthetic frame, the pixel to ground intersections for each ray is recorded and can be used as ground truth. Each synthetic frame is explicitly defined with calibration parameters and an exterior orientation, which is used to create a noiseless sensor model. Error analysis for each georegistration method is performed using absolute truth for every pixel and a perfect sensor model using this DIRSIG image dataset. An example image from the data used in this work along with a SfM derived point cloud is shown in Figure 4.1.

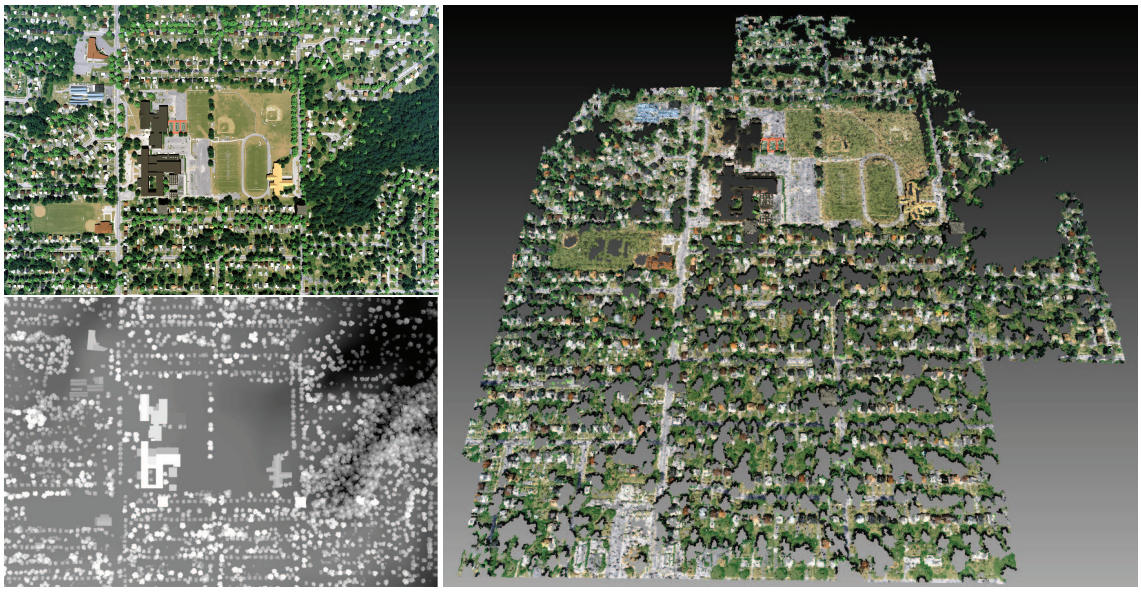


Figure 4.1: *DIRSIG is used to generate synthetic imagery which is input to the SfM process. Shown here on the left is a sample DIRSIG image created for this work along with the corresponding truth. On the right is a view of the SfM derived point cloud.*

Every three-dimensional point in the SfM process is created from corresponding pixels in multiple images. These pixels are tracked and used with the DIRSIG truth imagery to calculate the error in each three-dimensional point. The average of every truth measurement is used for error calculation. The Euclidean distance between the georegistered three-dimensional point and it's corresponding truth point is taken as the error.

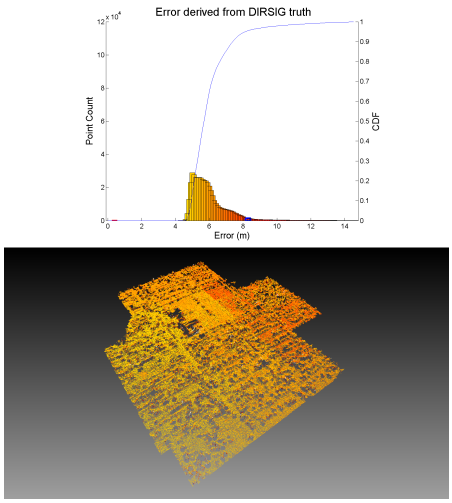
Each method for calculating a georegistration transform from Section 3.2 is analyzed using the DIRSIG truth. Finally, the georegistration error for each method is calculated again after random noise was added to the sensor model in order to simulate a more realistic scenario. The amount of random noise was determined based on RMS error reported in the Applanix POSTrack performance summary [3].

4.1.1 Georegistration Error Using DIRSIG Noiseless Sensor Model

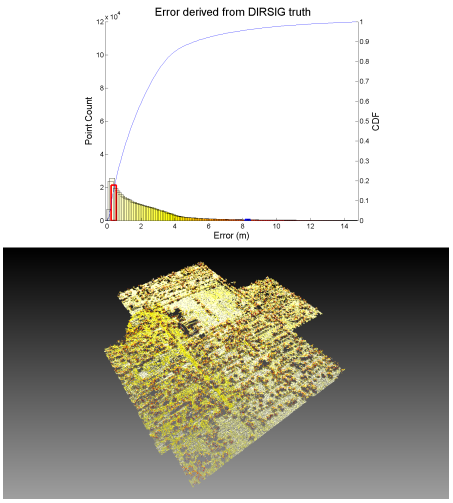
Two major sources of error are analyzed in this work. The first source of error comes from the SfM process itself. The effects of this error are isolated by using noiseless DIRSIG sensor models in the georegistration process. Error in geoaccuracy caused by the SfM process often manifests itself in poor image correspondence. Optimized image correspondence can still contain some small amount of error. This can cause small errors in camera pose estimation as well as point triangulation. The synthetic DIRSIG test imagery is taken from a nadir-looking direction, and therefore has a very low base-to-height ratio (B/H). This low B/H causes poor image correspondence to have significantly more effect on the error in the Z-dimension [73].

Error is calculated for each dimension X, Y and Z, however, for illustrative purposes the error is displayed as the Euclidean distance from the truth. Given the low B/H, the error resulting from the SfM process will primarily be in the Z-dimension. However, if the georegistration transform is flawed the error will be in every dimension. Figure 4.2 shows error for each method calculated using the DIRSIG noiseless sensor models.

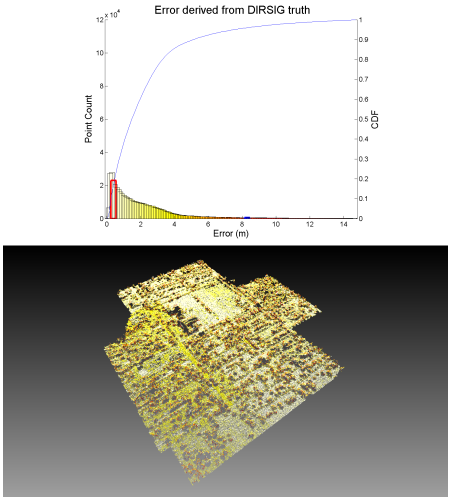
Figure 4.2-(a) shows a significant amount of error, caused by a poor estimation in the georegistration transform. Small errors in the relative camera pose estimation are amplified in the scale change between the relative and absolute coordinate systems. The other two methods shown in Figure 4.2-(b) and (c) show approximately the same amount of error, almost completely in the Z-dimension. Figure 4.3 shows the X,Y, and Z error distributions for Figure 4.2-(b). Given the random distribution of the error in the Z-dimension it is likely caused by error from the SfM process. This result is expected when using noiseless sensor models.



(a) *Camera centers*

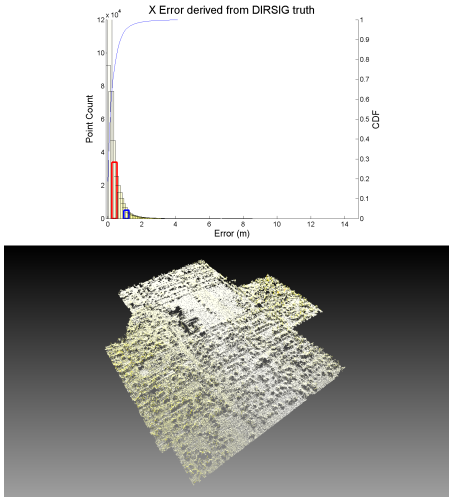


(b) *Augmented Camera Model*

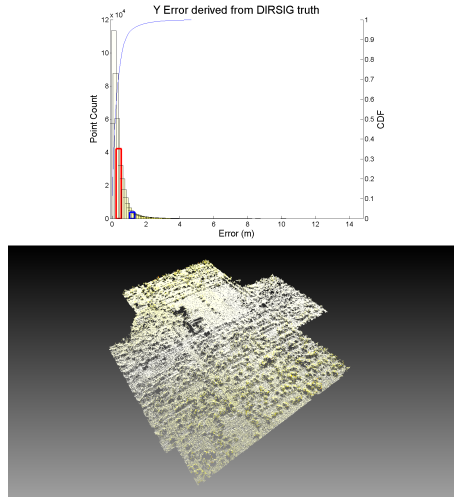


(c) *Direct Triangulation*

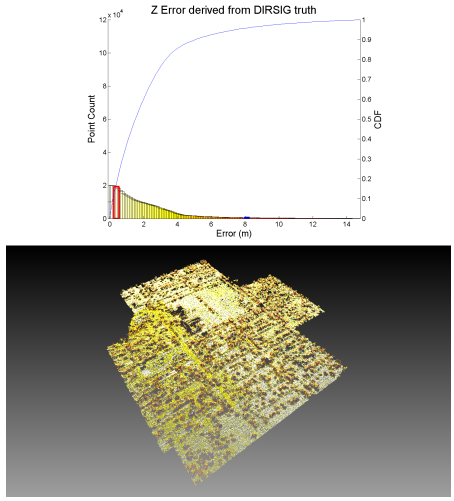
Figure 4.2: *Error is calculated using DIRSIG truth and represented as the Euclidean distance from the truth. The georegistration methods are processed using the noiseless DIRSIG sensor models. Three methods are tested, (a) is the error found using the camera-center based georegistration transform, (b) is the error found using the camera model based transform, and (c) is simply triangulating correspondence using the sensor model. Each plot shows a histogram of error with the bar representing 1m of error highlighted in red, as well as the bar corresponding to the 95% cumulative distribution value. The histogram's CDF is plotted over the histogram as well.*



(a) *Error in X*



(b) *Error in Y*



(c) *Error in Z*

Figure 4.3: *Error is calculated in all three dimensions, however, for visualization purposes, only displayed as Euclidean distance. The error for the methods shown in Figure 4.2-(b) and (c) have error primarily in the Z dimension. This suggests that the error is mostly due to error in image correspondence. The X,Y, and Z error for Figure 4.2-(b) is shown here.*

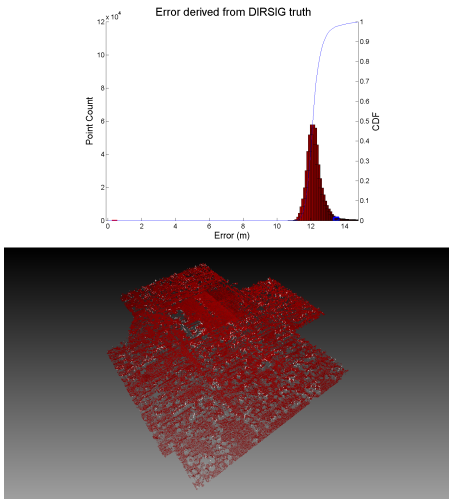
4.1.2 Georegistration Error Using DIRSIG Noisy Sensor Model

A more realistic scenario would be when the sensor models contain a small amount of random noise. The error analysis was repeated adding random error to the sensor position and pointing information, the amount of error was determined from Applanix specifications [3]. The RMS error in position is 0.1 meters in horizontal directions, 0.2 meters in the vertical. The error in pointing is 0.015 degrees for roll and pitch, and 0.040 degrees for heading. Figure 4.4 shows the error calculated for each method.

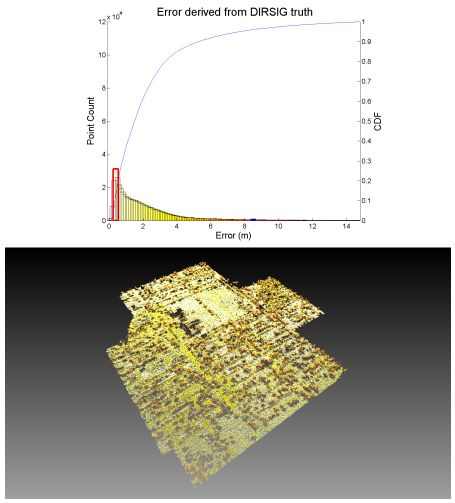
Similar to the noiseless sensor, the camera center based transform has a significant amount of error, the addition of noise increased the error further. Adding noise to the sensor information had a significant impact on directly re-triangulating each point, seen in Figure 4.4-(c). A small change in pointing information from a long distance will drastically alter the projection of each pixel in each frame, causing the triangulation solution to contain significantly greater error. The augmented camera model based transformation remained only mildly affected by the noise. This can be seen by measuring the change in position of the 95% cumulative distribution value relative to the noiseless sensor position, shown here in Table 4.1.

Table 4.1: A comparison of the 95% cumulative distribution values for each georegistration approach between noiseless and noisy sensors

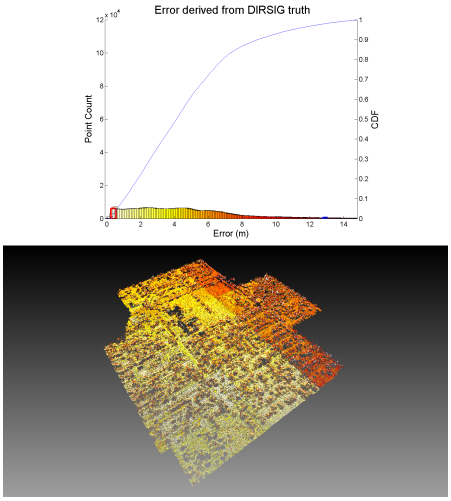
	95% CDF Value Noiseless (m)	95% CDF Value Noisy (m)	Change (m)
Camera Centers Approach	8.3	13.5	5.2
Augmented Camera Model Approach	8.3	8.5	0.2
Direct Triangulation Approach	8.3	12.9	4.6



(a) *Camera Centers*



(b) *Augmented Camera Model*



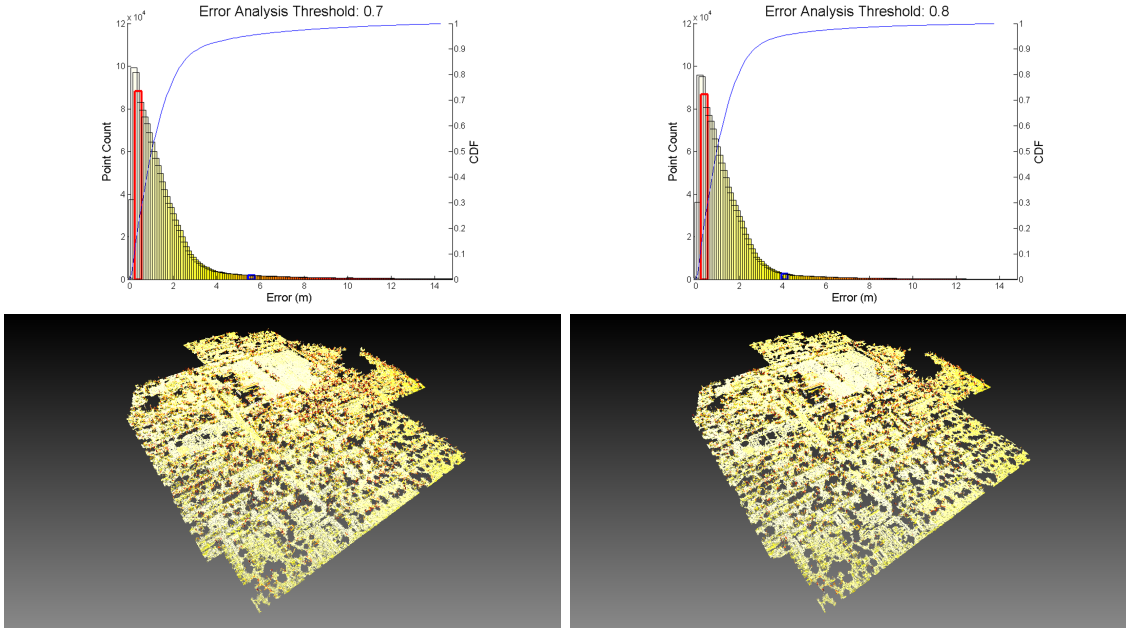
(c) *Direct Triangulation*

Figure 4.4: *Adding noise to the DIRSIG sensor models yields a more accurate representation of a real life scenario. The georegistration methods are processed using a noisy DIRSIG sensor model. Three methods are tested, (a) is the error found using the camera-center based georegistration transform, (b) is the error found using the camera model based transform, and (c) is simply triangulating correspondence using the sensor model. Each plot shows a histogram of error with the bar representing 1m of error highlighted in red, as well as the bar corresponding to the 95% cumulative distribution value. The histogram's CDF is plotted over the histogram as well.*

4.1.3 Reducing the SfM Error

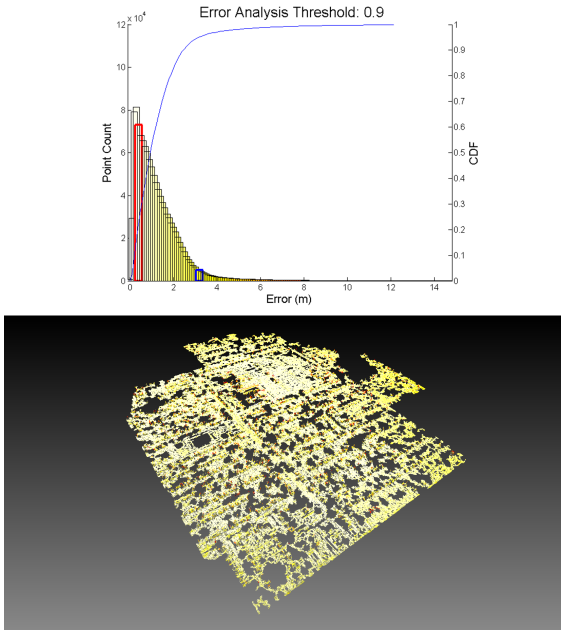
Two major sources of error are prevalent within this error analysis. The first source of error is addressed in the previous sections, which originates from the georegistration process itself. The second major source of error comes from the SfM process, in which image features may be inaccurately matched between images. A mismatch of a few pixels may translate into several meters of error in triangulation due to the small base-to-height ratio in near-nadir imaging. Often SfM algorithms threshold the image correspondence based on a threshold correspondence value, this threshold can be adjusted to filter the corresponding points accordingly [19].

The algorithm used to generate these point clouds has one threshold metric which filters image correspondence based on photometric consistency, using a normalized cross correlation [18]. The software uses this threshold to determine the quality of an image correspondence, with a range of -1 (bad) to 1 (good). The consistency value used in the creation of every point cloud shown in this analysis so far has been 0.8, error analysis for consistency values of 0.7, 0.8, and 0.9 are shown in Figure 4.5.



(a) *Threshold 0.7*

(b) *Threshold 0.8*



(c) *Threshold 0.9*

Figure 4.5: *Error from the SfM process often comes from mismatch in image correspondence. In the process used in this work, image correspondence is filtered using normalized cross correlation photometric consistency measurement. This figure shows error analysis done using the augmented sensor transform for different consistency threshold values.*

The histograms shown in Figure 4.5 depict a trend as the consistency threshold is increased. The total number of points decreases with the increase, which can be seen by either the relative total area of each histogram or by the decreasing number of visible points in each error visualization. However, it is noted that the 95% point in each histogram, shown by the highlighted blue bar, is moving closer to zero with the increase of the consistency threshold. This means that even though the total number of points is decreasing with the increase of threshold, the number of points with large geographic error is decreasing. It can be concluded that the extreme error values in the georegistration process are likely due to triangulation error in the SfM process caused by error in image correspondence.

4.1.4 Using a Large Number of Images

A larger set of images may reduce the error in transformation estimation, the error analysis performed in Sections 4.1.1 and 4.1.2 was done on a dataset containing 10 images. A second dataset containing 100 images was generated and analyzed. For this particular dataset there is a large amount of noise in the camera pose estimation and this remained true with the larger dataset. Figure 4.6 shows the camera centers for each camera as well as the estimated camera centers transformed using the method described in 3.2.1.

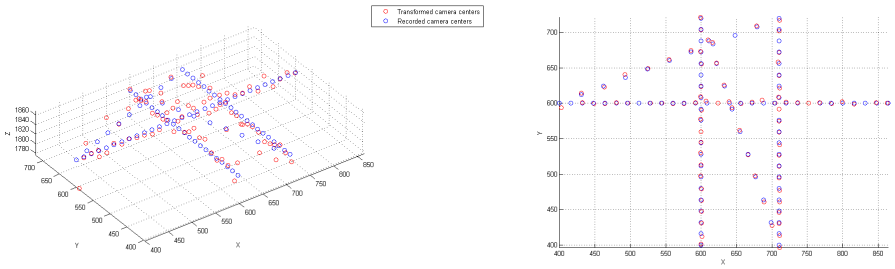
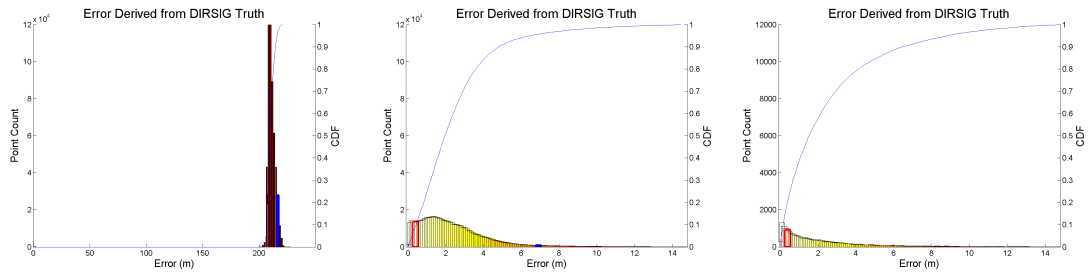


Figure 4.6: *This synthetic dataset had a significant amount of error in camera pose estimation, and this remained true for the larger dataset. This figure shows the known camera centers alongside the estimated camera centers which have been transformed using the method described in Section 3.2.1.*

The small errors in the relative pose estimation are amplified with the scale change from the SfM WCS to the Earth-based WCS. This causes the scale estimation to contain a larger amount of error, and therefore the error in transformation increased. Error analysis using the augmented sensor transform as well as the direct triangulation method is shown in Figure 4.7.



(a) *Camera Centers* (b) *Augmented Camera Model* (c) *Direct Triangulation*

Figure 4.7: *Increasing the number of images in the SfM process should reduce the amount of error in the transformation. While this may be the case for many datasets, for this synthetic dataset the error in the transformation increased, likely due to poor image correspondence. However, conclusions from this experiment can still be drawn. The camera center transform method is highly sensitive to error in the camera pose estimation, while the camera model based method as well as the retriangulation method remain stable. The error in (a) is likely caused by the large scale change between the SfM WCS and the Earth-based WCS.*

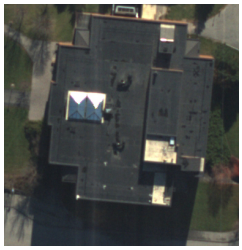
This error analysis was performed using noiseless sensor information. The results for augmented sensor transform and direct triangulation method are similar to those shown in 4.1.1. While it is logical to assume that a larger dataset would contain more redundancy, higher image overlap, and therefore, lower error in image correspondence; this specific dataset showed the opposite traits. It is likely that this was caused by the high occurrence of image correspondence error. This was caused by the synthetic nature of the imagery, which contains regions of low texture and regions of highly symmetric texture. While this large amount of image correspondence error might not be the case for all datasets, a conclusion can be drawn from this. The camera center transform method is highly sensitive to error in relative pose estimation, while the augmented sensor transform stays more robust to the increased error.

4.2 Voxel-Based Surface Reconstruction

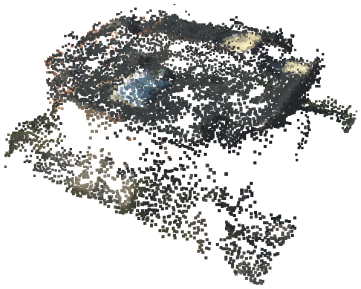
Two datasets were processed using the workflow presented in section 3.5, both datasets are detailed in Appendix C. The first dataset covered the Rochester Institute of Technology (RIT) campus in Henrietta, NY. Four buildings with varying levels of complexity were chosen for analysis. The second dataset covered downtown Rochester, NY. This dataset is significantly different from the RIT dataset due to the very tall buildings within the scene. Four of these buildings were chosen for analysis.

Each structure had its surface estimated using the voxel-based method presented in Section 3.3.2, and the confidence map for each surface was calculated as described in Section 3.3.3. The surfaces are shown here facetized with the Marching Cubes algorithm described in Section 3.3.2. The facets are colorized using the estimated surface structure color and confidence metric. The confidence metric is colored using a grayscale color scheme with white as the highest confidence. The -1 confidence value associated with colorless voxels is shown in red.

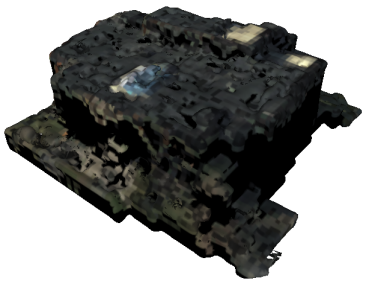
4.2.1 Buildings From the RIT Dataset



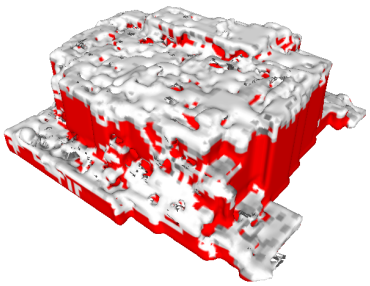
(a) *View of structure*



(b) *The intial reconstructed points*



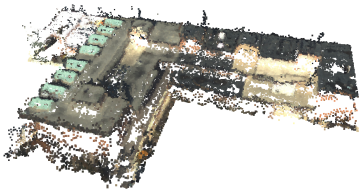
(c) *Facetized surface*



(d) *Confidence metric*

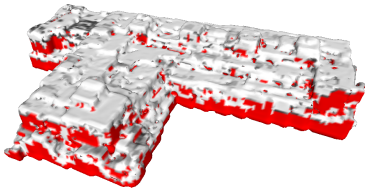
Figure 4.8: *The voxel-based modeling results for Building 76 from the RIT dataset*

(a) *View of structure*



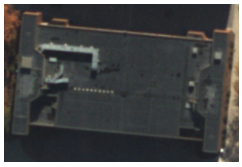
(b) *The intial reconstructed points*

(c) *Facetized surface*



(d) *Confidence metric*

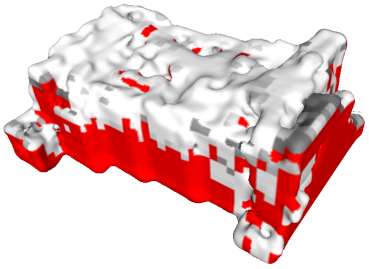
Figure 4.9: *The voxel-based modeling results for Building 7 from the RIT dataset*



(a) *View of structure*

(b) *The intial reconstructed points*

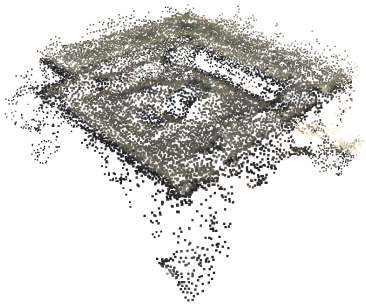
(c) *Facetized surface*



(d) *Confidence metric*

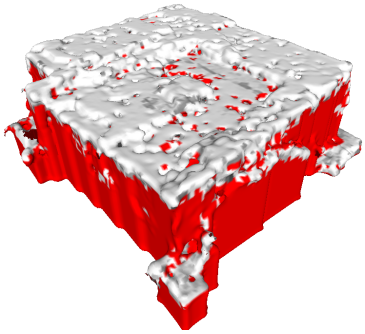
Figure 4.10: *The voxel-based modeling results for Building 6 from the RIT dataset*

(a) *View of structure*



(b) *The intial reconstructed points*

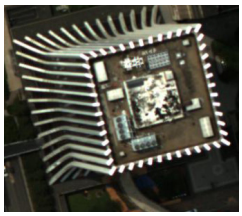
(c) *Facetized surface*



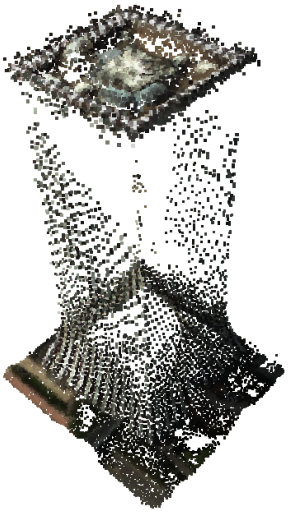
(d) *Confidence metric*

Figure 4.11: *The voxel-based modeling results for Building 5 from the RIT dataset*

4.2.2 Buildings From the Downtown Rochester Dataset

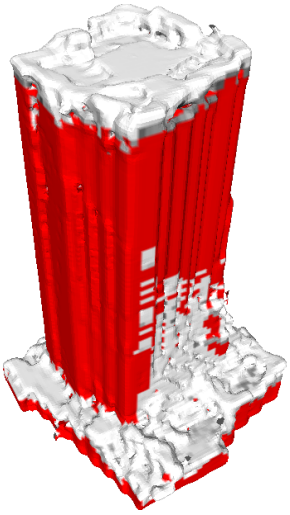


(a) *View of structure*



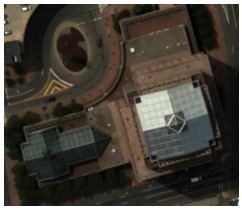
(b) *The intial reconstructed points*

(c) *Facetized surface*

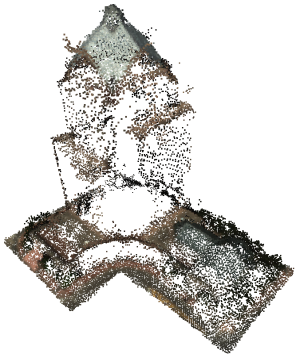


(d) *Confidence metric*

Figure 4.12: *The voxel-based modeling results for the Chase Tower from the downtown Rochester dataset*

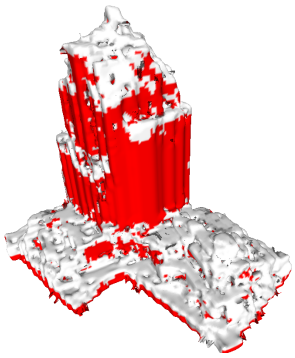


(a) *View of structure*



(b) *The intial reconstructed points*

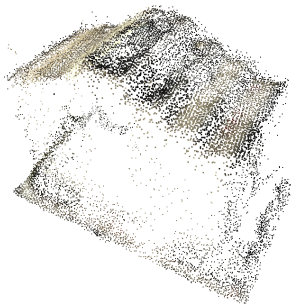
(c) *Facetized surface*



(d) *Confidence metric*

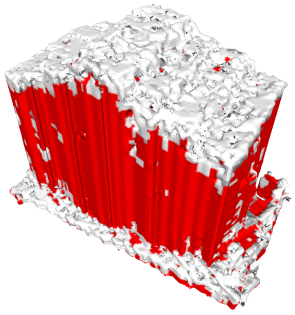
Figure 4.13: *The voxel-based modeling results for the Bausch & Lomb Place from the downtown Rochester dataset*

(a) *View of structure*



(b) *The intial reconstructed points*

(c) *Facetized surface*

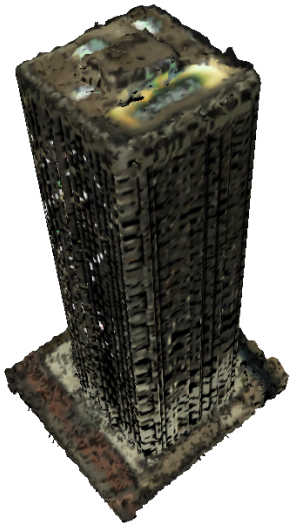


(d) *Confidence metric*

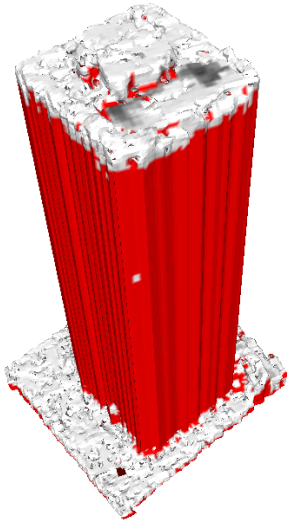
Figure 4.14: *The voxel-based modeling results for the Clinton Square Building from the downtown Rochester dataset*

(a) *View of structure*

(b) *The intial reconstructed points*



(c) *Facetized surface*

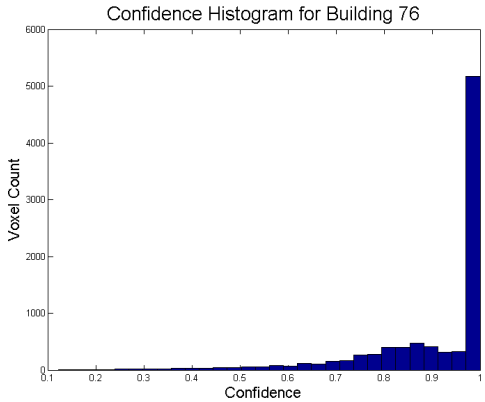


(d) *Confidence metric*

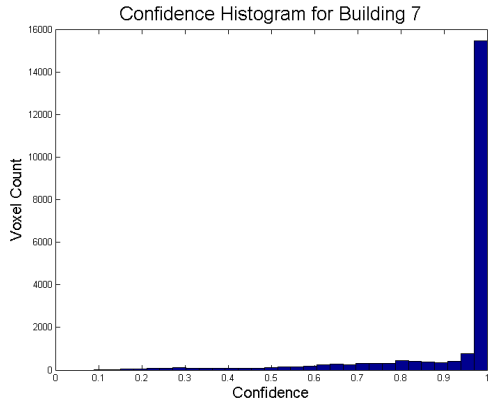
Figure 4.15: *The voxel-based modeling results for the Xerox Tower from the downtown Rochester dataset*

4.2.3 Confidence analysis

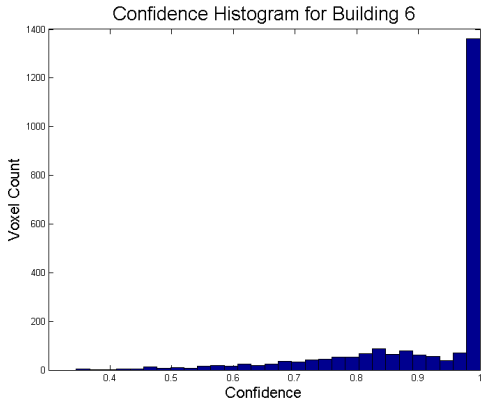
The histogram of positive confidence values are plotted for each reconstruction. The following figures show the histograms for each of the reconstructions shown in Figures 4.8 through 4.15. The confidence is very high for most surfaces, however, poorly reconstructed regions are highlighted by the values less than 0.9.



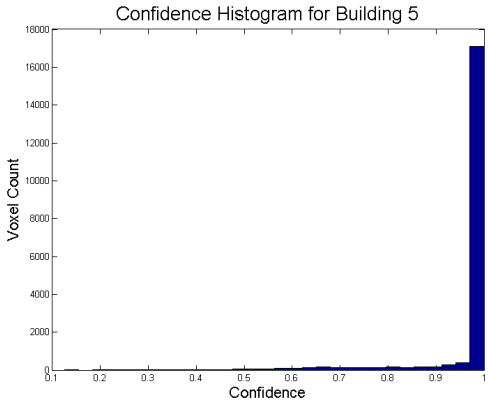
(a) *Building 76*



(b) *Building 7*

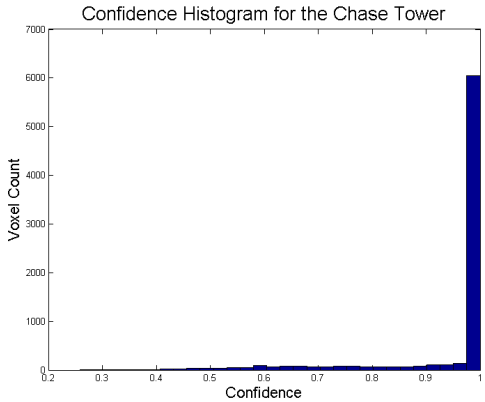


(c) *Building 6*

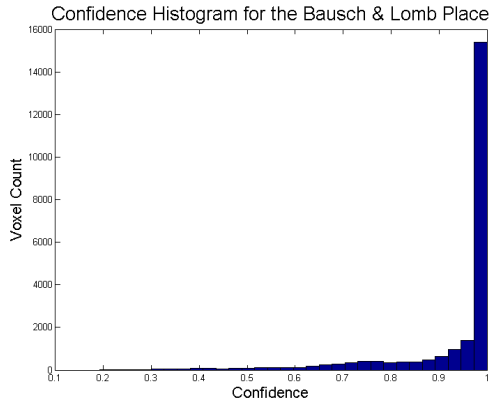


(d) *Building 5*

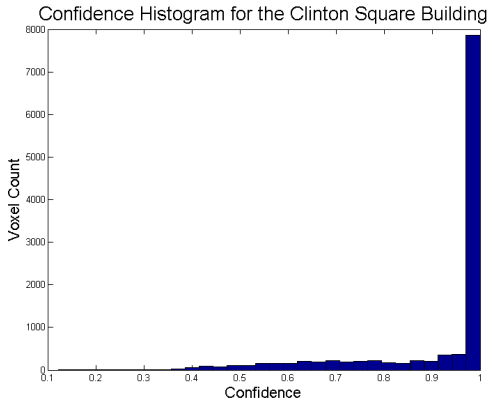
Figure 4.16: *The positive confidence values are plotted as a histogram for the buildings shown in Figures 4.8- 4.11. The confidence histograms are plotted in bins of* 0.02 *from* 0 *to* 1. *The confidence is high for most reconstructions, with lower confidence corresponding to poorly reconstructed regions.*



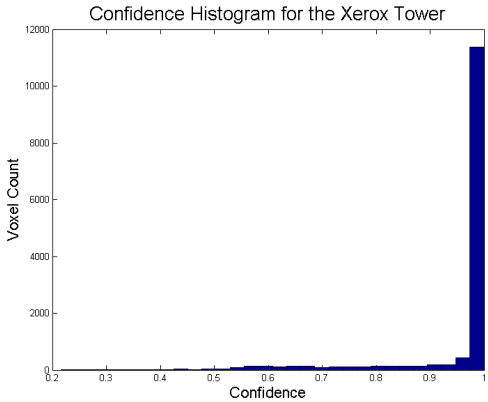
(a) *Chase Tower*



(b) *Bausch & Lomb Place*



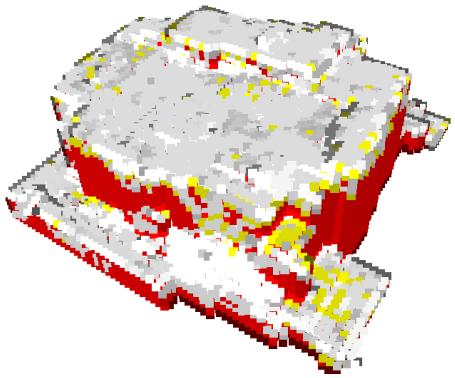
(c) *Clinton Square Building*



(d) *Xerox Tower*

Figure 4.17: *The positive confidence values are plotted as a histogram for the buildings shown in Figures 4.12- 4.15.The confidence histograms are plotted in bins of* 0.02 *from* 0 *to* 1. *The confidence is high for most reconstructions, with lower confidence corresponding to poorly reconstructed regions.*

A thresholding of these histograms could be used to automatically identify voxels with regions of poor reconstruction. An example of this is shown in Figure 4.2.3, where a thresholded example of two buildings are shown.

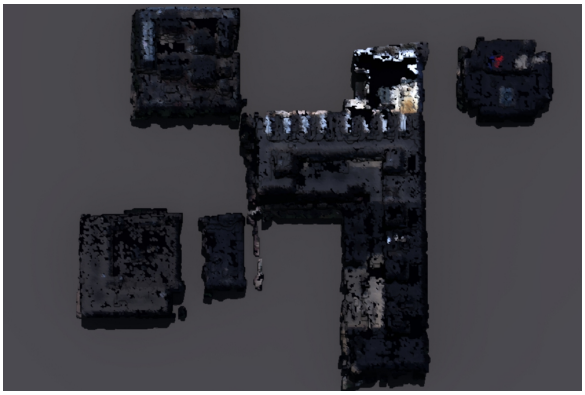


(a) *Building 76 thresholded voxel cloud*

(b) *Xerox Tower thresholded voxel cloud*

Figure 4.18: *The histograms shown in Figures 4.2.3 and 4.2.3 can be used to automatically identify voxels with regions of poor reconstruction. Here two buildings, with histograms corresponding to 4.2.3-(a) and 4.2.3-(d) are thresholded at confidence equal to* 0.85. *The yellow regions represent the thresholded voxels.*

The yellow voxels in Figure 4.2.3 represent voxels that were assigned less than 85% confidence, and it can be seen that these areas are in regions of poor reconstruction. In Figure 4.2.3-(a), a number of structures on the roof of the building are missing, this led to a high amount of color variation, since those voxels were considered part of the main roofing material, therefore resulting in low confidence in those regions. In Figure 4.2.3-(b), the voxel-based surface reconstruction method filled a void in the roof of the tower that was actually part of the structure. This also led to a high amount of color variation, and consequently, low confidence in the reconstruction region.



(a) *Simluated R,G,B image*

(b) *Simluated NIR image*

(b) *Simluated SWIR image*

Figure 4.19: *Reflectance attributed models processed using DIRSIG. (a) shows an example synthetic image processed with an R,G,B imaging platform, (b) shows an example synthetic image processed with a NIR platform (0.7 μm to 1.0 μm), and (c) shows an example processed with a SWIR platform (1.8 μm to 2.4 μm).*

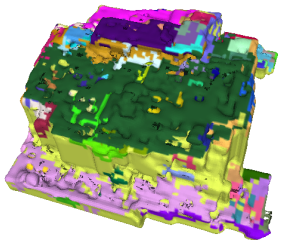
4.3 Reflectance-Attributed Facetized Surface Structure

A hyperspectral dataset was collected over the RIT campus, which can be used for the reflectance mapping described in Section 3.4.1. Further information about this dataset can be found in Appendix C. Each processed building from the RIT dataset from the previous section had each facet mapped with reflectance spectra. Having this spectra mapped to each facet allowed for the creation of a model which could be used for physical simulation. An example of the physical simulation of these buildings is shown in Figure 4.19. This

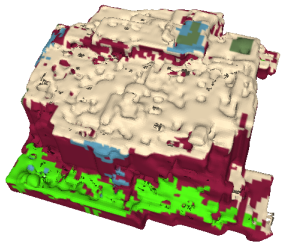
simulation was performed using the DIRSIG software. Since only reflectance spectra was mapped to the facets, the simulation can only be performed within the bandpass of the hyperspectral imagery. The simulation pictured is intended to replicate the R,G,B platform which originally captured the data.

4.4 Classified Facetized Surface Structure

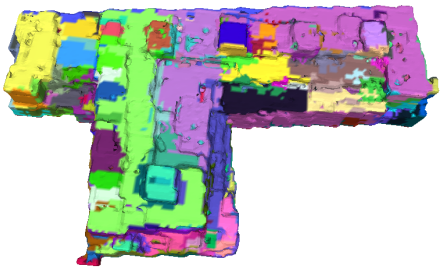
Each structure shown in Section 4.2 was segmented using the region-growing approach described in Section 3.4.2. A user-provided estimate of the number of materials was used for the final k-means clustering step in the segmentation process. The initial region-growing segmentation and the final k-means clustering segmentation are shown for each structure. The results are shown in Figures 4.20 and 4.21.



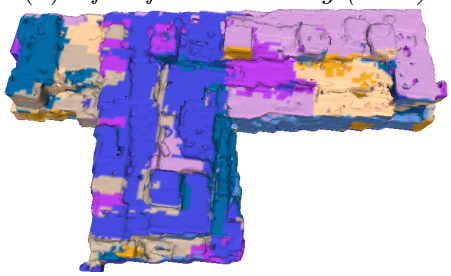
(a) *Inital Segmentation*



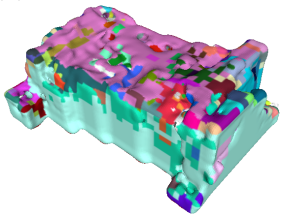
(b) *After final clustering (K=5)*



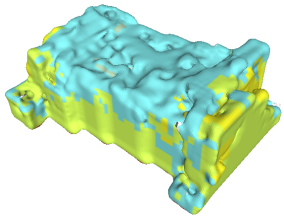
(c) *Inital Segmentation*



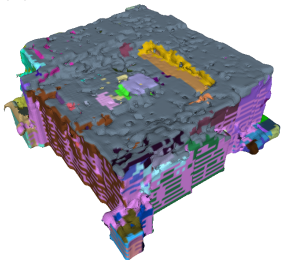
(d) *After final clustering (K=9)*



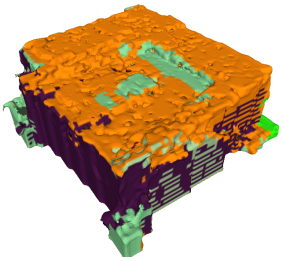
(e) *Inital Segmentation*



(f) *After final clustering (K=2)*

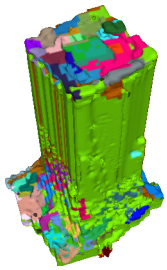


(g) *Inital Segmentation*

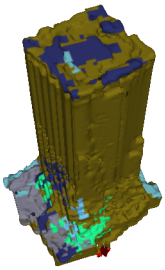


(h) *After final clustering (K=5)*

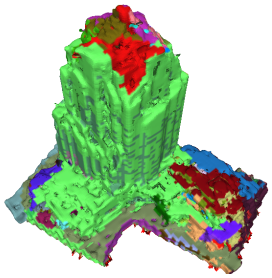
Figure 4.20: *(a)-(b): Building 76, (c)-(d): Building 7, (e)-(f): Building 6, (g)-(h): Building 5*



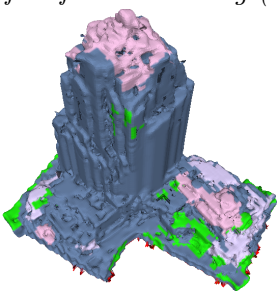
(a) *Inital Segmentation*



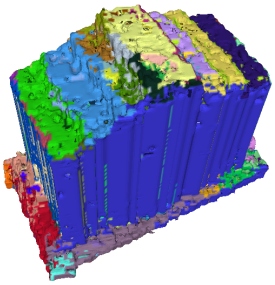
(b) *After final clustering (K=6)*



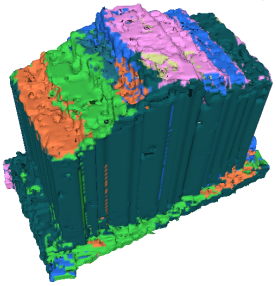
(c) *Inital Segmentation*



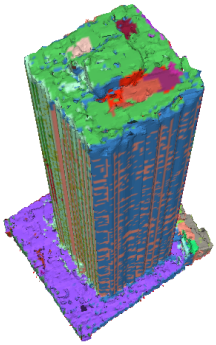
(d) *After final clustering (K=5)*



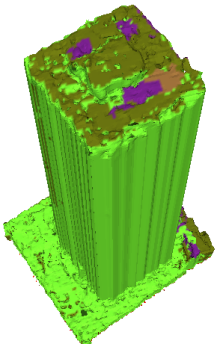
(e) *Inital Segmentation*



(f) *After final clustering (K=6)*



(g) *Inital Segmentation*



(h) *After final clustering (K=5)*

Figure 4.21: *(a)-(b): Chase Tower, (c)-(d): Bausch & Lomb Place, (e)-(f): Clinton Square Building, (g)-(h): Xerox Tower*

4.4.1 k-Means Clustering Sensitivity Study

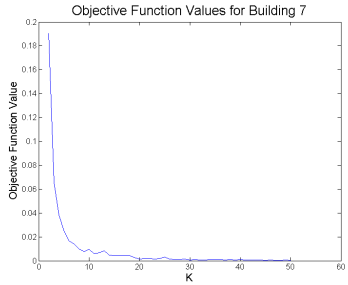
The k-means algorithm attempts to cluster points by minimizing the objective function [31] shown in Equation 4.1. That is, by minimizing the average squared Euclidean distance of each point from their cluster centers.

$$J = \sum_{j=1}^{k} \sum_{i=1}^{x} \|x_i^j - c_j\|^2 \tag{4.1}$$

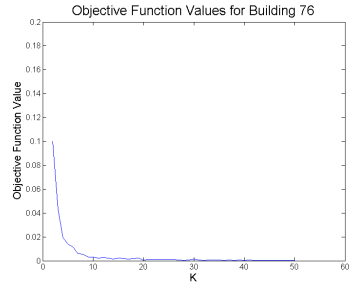
Where x_i^j is the ith element in the jth cluster, and c_j is the centroid of cluster j. The closer the objective function value J is to zero, the more compact the clustering will be. Of course, as k goes to infinity, J will go to zero. However, there will be a point at which increasing the value k has a diminishing effect on the value of J. The best clustering scenarios will have a relatively low k value resulting in a strong minimization of the objective function J.

An analysis was performed on each structure presented in this section. A range of k values between zero and fifty was used in the final k-means clustering step of the region-growing based segmentation. The objective function was measured for each structure across the range of k values. Figure 4.22 shows the results for each structure.

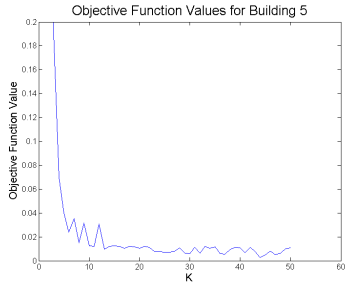
It can be seen from Figure 4.22 that each structure had a relatively good clustering with less than ten classes. This means that the region-growing based clustering scenario is well posed for a quality classification.



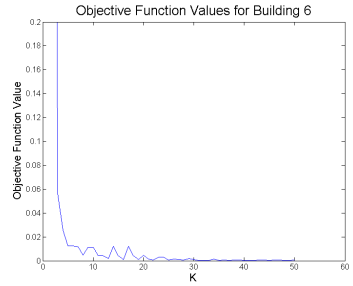
(a) *Building 7*



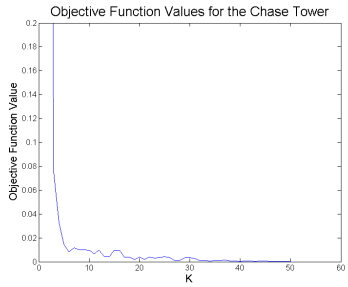
(b) *Building 76*



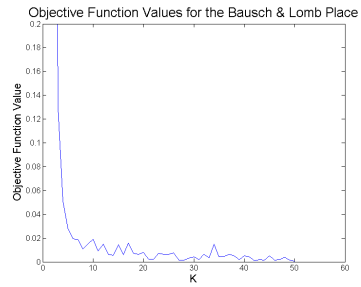
(c) *Building 5*



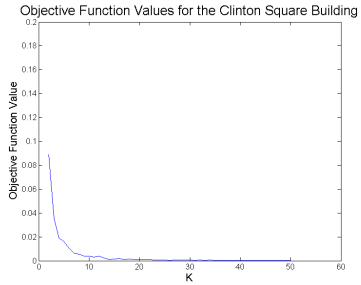
(d) *Building 6*



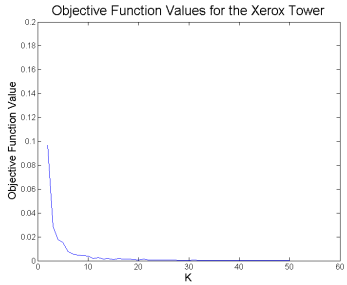
(e) *Chase Tower*



(f) *Bausch & Lomb Place*



(g) *Clinton Square Building*



(h) *Xerox Tower*

Figure 4.22: *The objective function value vs. k value for each structure is shown here. The structures in (a)-(d) are from the RIT dataset, and the structures in (e)-(h) are from the downtown Rochester dataset. It can be seen that each structure minimizes it's objective function with a relatively low k value.*

Chapter 5

Discussion

Geographically and physically accurate models of man-made structures are used as the basis of many modeling applications. Automatically generating these models from aerial imagery would provide new opportunities for applications and research. The automated extraction of these types of models is split into two major sections; 1) the automated extraction of the geographically accurate structure using imagery, and 2) the modeling and analysis of this extracted structure. The computer vision community has developed a strong understanding of automated structure extraction processes, through a workflow known as Structure from Motion (SfM). The development of this understanding has produced many methods for implementing this workflow, including processes that have been made open source and available to the public. Geographic accuracy requires knowledge of additional geographic information from the imaging platform. The use of aerial imagery provides a distinct advantage is this area, due to the common usage of highly accurate INS and GPS systems which record positional information for each image. In addition to this, aerial imaging platforms tend to be very well calibrated and characterized. These attributes make aerial imagery an ideal candidate for geographically accurate and automated structure extraction.

Structure extracted from aerial imagery does not provide a complete understanding of a scene. These structures are simply discrete measurements of the three-dimensional world. These measurements can be processed, interpolated, and analyzed to develop a better understanding. Through the application of certain assumptions, such as the Manhattan-world assumption, man-made object surfaces can be estimated from the structure measurements. The surface extraction is an incremental step towards scene understanding. The following step is classification of materials of the object's surface. This step combines

spectral information with the three-dimensional spatial information to develop methods for surface understanding.

Each process required to estimate a geographically and physically accurate model has been discussed or developed in this work. Structure extraction is achieved from multi-view aerial imagery through popular open-source SfM algorithms. This structure is combined with INS/GPS measurements to generate a georegistration transform, bringing the estimated structure to an Earth-based coordinate system. The geoaccurate structure is further analyzed by estimating and interpolating an object's surface using geometric primitives. The approach developed in this work utilizes a voxel-based methodology, which allows for estimation and interpolation using common morphological processes. A confidence metric for the interpolated surface estimation was developed so users could further analyze the estimated surfaces and determine regions and causes of error. Finally, incorporation of spectral information from the imagery is used to develop methods of material attribution and segmentation. Surfaces are mapped with reflectance spectra if additional high-spectral resolution calibrated imagery is available. If this is not the case, the surface is mapped with original R,G,B spectral information, and segmented using properties of the three-dimensional surface along with the mapped spectral information. These segments can be treated as classes for a user, in order to identify and attribute materials.

5.1 Georegistration

This work has analyzed three methods of SfM point cloud georegistration, namely the *camera centers approach* (Section 3.2.1), the *augmented camera model* approach (Section 3.2.2), and the *direct triangulation* approach (Section 3.2.3). Each method requires additional information beyond the imagery, which is taken from INS/GPS systems. All three methods use this additional information to varying degrees. Two main sources of error contribute to the geoaccuracy of a georegistered SfM point cloud.

The first source of error comes from the SfM process itself, often from error in image-to-image correspondence. Many factors can contribute to error in image correspondence, for example, wide-baseline camera geometry lends itself to error due to the large difference in object appearance. Poor texture definition or repetitive texture on the object's surface can also lead to mismatch. This is also the case with other imaging modalities, object in thermal infrared imagery have rather poor texture and definition, which would prove a

challenge to most feature correspondence algorithms. Error in correspondence propagates throughout the SfM process, causing error in camera pose estimation and scene structure triangulation. For the photogrammetric applications of the SfM process it is important to understand the limitations that are inherent in this process. It was shown that adjusting a single parameter to minimize error in correspondence can have an impact on the total amount of error. However, all aspects that contribute to correspondence error should be considered when using the SfM process for photogrammetric purposes.

The second source of error is within the georegistration transform, which is affected by error in the SfM process, but is also affected by error in the additional information used to generate the transform. For this work, additional information is found in the INS/GPS data. Given no error in this information, each approach was only affected by SfM error. The SfM error in the test dataset for this work was substantial, and this translated to a significant amount of the error in the *camera centers approach*. The *augmented camera model* approach and the *direct triangulation* approach remain significantly less affected. When random instrument error is added to the INS/GPS readings, the *direct triangulation* approach fails. This is expected, as the instrument error directly affects the triangulation. The random error increases the total error in both the *camera centers* and *augmented camera model* approaches, however the latter was shown to be far more robust than the former.

5.2 Surface Estimation and Analysis

A CAD-like model is the desired output for physical modeling. That is, a model that is defined by a number of geometric primitives. Two methods for extracting these primitives were presented, a RANSAC-based and a voxel-based method. The RANSAC-based extraction worked well for decomposing an object into planes, however, it is limited in the shapes in which it can represent. The voxel-based reconstruction is far more robust for representing any arbitrary shape. Furthermore, the nature of the voxel space allows for interpolation of an object's surface.

Scene reconstructions through nadir-looking aerial imagery tend to be noisy. Voxel-based processing is inherently noise-reducing as it is a sampling of a continuous space, although, a number of other noise reduction processes are implemented. Point density and voxel proximity information are used for further noise reduction.

Using Manhattan-world assumptions, the vertical surfaces of the man-made structures

are interpolated. Furthermore, using the same assumption, horizontal surfaces containing holes are interpolated. A confidence metric is developed to help a user determine the quality of a given surface reconstruction. This metric is important to have, since most of an object's surface in a noisy point cloud will be interpolated using the orthogonality assumptions.

The surface estimation was augmented with spectral information for further analysis. Given additional atmospherically compensated hyperspectral imagery, each facet on the surface estimation is attributed with a reflectance spectra. The attribution is performed by registering each pixel from the source and hyperspectral imagery to a digital elevation map. This attribution allows for the direct physical modeling of these surfaces, within the bandpass for the hyperspectral sensor.

It unlikely that many data collects will have coincident hyperspectral and high-resolution multi-view imagery for three-dimensional reconstruction and analysis. For this reason, a segmentation algorithm, using just the high-resolution R,G,B values from the multi-view imagery, was developed. This segmentation algorithm analyzes the spatial-spectral properties of the surface to determine the segments. When used in conjunction with the voxel-based surface estimation process, the interpolated voxel's color is estimated though projection into the source imagery.

The spatial-spectral segmentation algorithm's final step is a spectral k-means clustering. This is to ensure that similar materials which may be separated by a significant distance, or may be on opposing sides of a surface, are part of the same segmentation class. An analysis of the chosen k-value was performed to determine if this final k-means clustering could provide a quality classification. It was shown that for every structure tested, the quality of classification neared a maximum under ten classes. Using a relatively small number of classes, a quality classification can be obtained. This segmentation approach allows a user to automatically determine the potential classes of materials on the surface of a reconstructed structure, using just the inherent R,G,B values.

5.3 Limitations

As with every process that makes some number of assumptions, there are limitations to the performance of the algorithms. From a SfM perspective, there are a number of well-known limitations, generally all related to image correspondence. Given imagery which

has a significantly large baseline between the camera centers, the objects contained in the scene will be difficult to match between views. Also, the content of a scene may provide issues. A scene with very little texture, or a very large amount of random texture, can provide difficulty for feature matching algorithms. Feature matching error propagates itself throughout the entire SfM process and to subsequent processes. The quality of the feature matching algorithm can be a limiting factor in surface reconstruction processes, therefore, the algorithm should be chosen carefully.

Georegistration transforms are derived from measured INS/GPS information. There is inherent measurement error in this information, which can contribute to error in the transformation calculation. This can be limiting when the measurement errors become large.

The voxel-based surface reconstruction process presented here is robust to noisy point clouds and object shape. However, given the assumptions used in the interpolation processes, there are some limitations. One major limitation is the inability to close large holes on sloped surfaces that are not closed during the Z-level morphological processing. This process will have problems with poorly defined sloped roofs. A small amount of this error can be seen in Figure 4.13-(c) on the sloped roof of the building. The other limitation, which is also considered an advantage, is the hole-filling process. It is assumed that all connected components in each Z-level in the morphological processing step contains the same structure. This is a fairly accurate assumption for most buildings, however, if a structure has a "hole" in the surface, it will be filled. This can be seen in both Figures 4.9 and 4.15. This situation provides further reasoning for a confidence metric. The calculated confidence for these "hole" regions on the estimated surface is much lower than for the quality surface estimations. An enlarged view of this is shown in Figure 5.1.

(a) *Structure with "hole"* (b) *The filled "hole"* (c) *The filled hole's confidence*

Figure 5.1: *The voxel-based surface reconstruction process assumes that there are no holes on the surface of any structure. Therefore, if these are "holes", there regions will be filled in. This error can be detected in the confidence map for the surface reconstruction.*

The confidence metric can be used to identify surfaces which have been poorly reconstructed. A histogram of the confidence values (Section 4.2.3) shows that most of the reconstructed surfaces have a very high confidence value, however, in many regions such as the ones shown in 5.1 the confidence is lower.

The spectral attribution and segmentation methods have a few limitations as well. The largest limitation is in the reflectance spectra mapping, through orthorectification. The orthorectification process removes any amount of parallax that might be in the image. Conceptually, this means that all the structure is projected directly onto the ground from the point of view of the camera. If the object undergoing registration through the orthorectification map is too far off-nadir in the scene, then there will be some registration error caused by this parallax. This can be mitigated by only registering objects through images in which the objects are near the principle point (or most-nadir pointing part) of the image.

The final limitation is in the last step of the spatial-spectral segmentation process, the spectral k-means clustering. This step requires a user to identify the number of materials expected to be on the surface of the structure. While this is a limitation inherent in k-means classification, it is one that may become an issue when trying to obtain full automation.

5.4 Future Work

This work leads to a wide range of possible future work to be done in this same area. The areas that require consideration are:

1. It is very important that any surface estimation process is seeded with the highest quality structure reconstruction. Therefore, it would be useful to determine if other reconstruction methods can provide better quality structure estimation than the SfM software presented in this work. A couple related algorithms, probabilistic voxel modeling [55], and semi-global image matching [24], should be tested. These algorithms are not as widely used in the computer visions community, but have significant potential to be very powerful.

2. The structure extraction process should be extended to include off-nadir imagery. This limitation is reasonable due to the large amount of near-nadir aerial imagery already in existence. However, off-nadir imagery can provide significantly more

information about vertical structures in a scene (such as walls). A higher fidelity reconstruction would allow for some of the assumptions to be relaxed.

3. All structure extraction processes must extract initial feature correspondences. This work uses a GPU-based implementation of the SIFT algorithm. The GPU implementation was used for computational speedup due to the large number of features often generated by aerial images. The computer vision community has moved past SIFT towards other feature detection and description algorithms. A couple were discussed in Chapter 2 (ASIFT, DAISY), however, there are many more which should be explored. Algorithms such as SURF [5] and HoG [12] have become the replacement algorithms for SIFT. More recently, binary descriptors have become popular using algorithms such as ORB [56] and FREAK [2]. All these feature detection and description algorithms should be tested in the SfM framework to determine the best option to use. A better feature extraction and matching will lead to lower overall error throughout the process.

4. The surface estimation method presented in this work allows for the estimation of surface structure without significant constraint on the shape of the structure. This, however, causes edges, planes, and corners to smooth out or become noisy. A stronger assertion of the Manhattan-world assumption could be implemented to make sure that all edges, corners, and planes follow the orthogonality rules. This would results in a more visually pleasing surface estimation.

5. The surface confidence metric could more than just a visualization tool. Confidence could be used to guide surface estimation processes.

6. The segmentation methods explored in this work touch upon a whole new dimension of classification algorithms. Through incorporation of a third spatial dimension, objects and their materials can be further discriminated. There are many types of spatial-spectral segmentation algorithms that should be explored and tested in the framework of three-dimensional data.

7. For testing other segmentation methods, it would be useful to have a known, accurate, test dataset to be used with these types of algorithms. A full set of classified objects and multi-view imagery for each of these objects would provide a good amount of truth for classification algorithm testing.

8. An automated process for identifying materials from the segmented surface structure could be explored. Using spectral statistics from the segmented class, as well as some spatial statistics, it may be possible to narrow down a search to determine the type of surface material.

9. A higher fidelity mapping of color to the surface structure would allow for more thorough segmentation. Instead of assigning a color to each facet, if each facet was UV-mapped back to the known imagery, a number of pixels could be associated to a single facet. From this, texture-based segmentation methods could also be incorporated to further analyze the surface structure.

5.5 Conclusions

The work presented here covers an end-to-end process for extracting three-dimensional reconstructions from multi-view imagery, georegistering the reconstructions, estimating their surface, and attempting to further attribute the surface with spectral information. A number of contributions have been made as a result of this work.

- Open-source SfM algorithms were pulled together to form an end-to-end SfM workflow that was made available to the public. A tutorial on this workflow is shown in Appendix D.

- A methodology for the analysis of georegistration accuracy was developed using synthetic imagery. This methodology can be incorporated into other SfM-based algorithms in order to quantify their geoaccuracy.

- A voxel-based surface estimation and interpolation method was developed, as well as an interpolation confidence metric. This algorithm is open-source and available to the public, a description of installation and usage can be found in Appendix E.

- A method for creating reflectance-attributed physical models of man-made structures, using hyperspectral imagery in addition to multi-view imagery, was developed. Software to perform this registration was made open-source, and can be found in Appendix F.

- A process for surface estimation classification was developed through the application of spatial-spectral segmentation algorithms. This process is incorporated into the

voxel-based modeling software, and is consequently also open-source and available to the public.

This work has explored some of the potential applications of three-dimensional reconstructions from aerial imagery, primarily focusing on the analysis of distinct materials on estimated surface structure. While much the commercial world is focused on photo-realistic scene development, researchers should keep their eyes forward for the potential applications of multi-view information. Incorporating the third-dimension into digital image processing and remote sensing algorithms allows for greater possibilities and applications of future research and development.

Appendix A

Transforming the projection matrix P using the georegistration transform T_S

The transform $\mathbf{T_s}$ can be used to transform one point set $\mathbf{X_a}$ to another point set $\mathbf{X_b}$. It can also be important to understand the effect that transform $\mathbf{T_s}$ may have on projective cameras which may have been used to create the point set $\mathbf{X_a}$. The relationship of $\mathbf{T_s}$ and the point sets is defined as

$$\mathbf{X_b} = \mathbf{T_s}\mathbf{X_a} \tag{A.1}$$

The relationship between the projection matrices, $\mathbf{P_a}$ and $\mathbf{P_b}$ with the set of points $\mathbf{X_a}$ and $\mathbf{X_b}$ is

$$\mathbf{x_a} = \mathbf{P_a}\mathbf{X_a} \tag{A.2a}$$

$$\mathbf{x_b} = \mathbf{P_b}\mathbf{X_b} \tag{A.2b}$$

where $\mathbf{x_a}$ and $\mathbf{x_b}$ are the projections of $\mathbf{X_a}$ and $\mathbf{X_b}$ into the image. The image projections $\mathbf{x_a}$ and $\mathbf{x_b}$ only differ by a scale value in projective space, as they are projections of points that only differ by an affine transform. Furthermore, in inhomogeneous coordinates, they are equivalent. Using this relationship along with Equation A.1, Equation A.2a, and Equation A.2b the following can be defined,

$$\mathbf{P_a X_a} = \mathbf{P_b T_s X_a} \tag{A.3}$$

From this, the desired projection matrix $\mathbf{P_b}$ can be solved for in terms of $\mathbf{P_a}$ and $\mathbf{T_s}$,

$$\mathbf{P_b} = \mathbf{P_a T_s}^{-1} \tag{A.4}$$

To verify this solution, Equation A.2b is examined, by substituting Equation A.1 into Equation A.4, and the following is obtained

$$\mathbf{x_b} = \mathbf{P_b T_s X_a} \tag{A.5}$$

Substituting the solution for $\mathbf{P_b}$ into Equation A.6 yields

$$\mathbf{x_b} = \mathbf{P_a T_s}^{-1}\mathbf{T_s X_a} = \mathbf{P_a X_a} \tag{A.6}$$

Using the relationship $\mathbf{x_a} = \mathbf{x_b}$, Equation A.6 can be shown to be equivalent to Equation A.2a. It is important to note that due to this transformation, the scaling of the projection matrix $\mathbf{P_b}$ may not be correct. Decomposing $\mathbf{P_b}$ into it's fundamental parts gives,

$$\mathbf{P_b} = \mathbf{k}\,[\mathbf{R_b}|\mathbf{t_b}] \tag{A.7}$$

where $\mathbf{k}$ is the camera calibration matrix, $\mathbf{R_b}$ is the rotation matrix, and $\mathbf{t_b}$ is the translation vector. This transformation cannot guarantee a normalized Rotation matrix $\mathbf{R_b}$. Therefore, the transformed projection matrix $\mathbf{P_b}$ should be scaled by the L2 norm of the decomposed rotation matrix, $\mathbf{R_b}$, yielding the normalized projection matrix $\widetilde{\mathbf{P}}_\mathbf{b}$,

$$\widetilde{\mathbf{P}}_\mathbf{b} = \frac{1}{\|R_b\|_2}\mathbf{P_b} \tag{A.8}$$

This solution can be used for projecting the points, $\mathbf{X_a}$, which have been transformed to $\mathbf{X_b}$.

Appendix B

Normalized Cuts

Normalized cuts is a graph partitioning algorithm, which has been shown to be powerful for image-based segmentation [33]. This concept of normalized cuts can be easily extended to a third dimension, and used for point cloud segmentation algorithms, such as the one shown in Section 3.4.2. This appendix will cover the basic normalized cuts algorithm.

B.1 Representing Data as Graphs

Any set of data that has quantifiable relationships between datum can be represented in a graph. For example, a cluster of points, as shown in Figure B.1, can be related by proximity to each other, and represented as a graph.

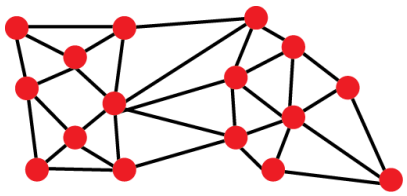


Figure B.1: *A cluster of points can be represented as a graph using their proximity to each other as edges. Here, each point represents a node of the graph and the line connecting the nodes represent an edge between the two nodes.*

A graph, **G**, is defined as a collection of nodes, **N**, and edges between those nodes, **E**. A graph can be represented using an adjacency matrix, **A**, which is an n-by-n binary

matrix where each column and row represent a node in the graph. Each element, $a_{i,j}$ represents a potential connection point between nodes. It is assumed that nodes are not connected to themselves, therefore the diagonal of $\mathbf{A}$ is zero.

$$\mathbf{A} = \begin{bmatrix} 0 & a_{0,1} & \cdots & a_{0,n} \\ a_{1,0} & 0 & \cdots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,0} & a_{n,2} & \cdots & 0 \end{bmatrix} \tag{B.1}$$

Often, edges are represented by a value or weight. Similar to the adjacency matrix, these edges value are represented with the weight matrix, $\mathbf{W}$, in which each non-zero element represents a weighted edged between two nodes.

Another matrix can be defined from $\mathbf{A}$ or $\mathbf{W}$, called the degree matrix, $\mathbf{D}$. This is a n-by-n diagonal matrix in which each element is the degree for the corresponding node. The degree of a node is defined for an unweighted graph as the total number of terminating edges on that node. For a weighted graph, the degree is defined as the sum of all edge weights terminating on that node.

$$\mathbf{D} = \begin{bmatrix} d_{0,0} & 0 & \cdots & 0 \\ 0 & d_{1,1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_{n,n} \end{bmatrix} \tag{B.2}$$

$$d_{n,n} = \deg(n) = \sum_{k=0}^{n} A_{n,k} \tag{B.3}$$

Finally, a matrix called the Laplacian matrix, $\mathbf{L}$, is defined as the difference between the degree matrix and weight matrix.

$$\mathbf{L} = \mathbf{D} - \mathbf{W} = \begin{bmatrix} d_{0,0} & -a_{0,1} & \cdots & -a_{0,n} \\ -a_{1,0} & d_{1,1} & \cdots & -a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n,0} & -a_{n,2} & \cdots & d_{n,n} \end{bmatrix} \tag{B.4}$$

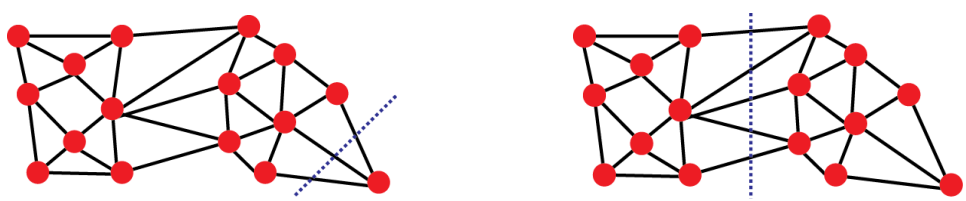
Analysis of the matrix properties (i.e eigenvalues, eigenvectors) of these matrices leads to a mathematical field called spectral graph theory. It is from this analysis that the concept of normalized cuts arises.

B.2 Graph Cuts and Normalized Cuts

A partitioning of the weight matrix represents a form of data segmentation. This partitioning is called a cut, defined as follows,

$$cut\,(A, B) = \sum_{i \in A, j \in B} w_{i,j} \tag{B.5}$$

The partitioning of the weight matrix will be one where the cut is minimized. A detailed description of this process can be found in the literature [7]. This partitioning is very sensitive to outliers, as shown in Figure B.2-(a), where the minimum cut would be the one in which the sum of the weights will be the smallest.



(a) Minimum cut (b) Normalized Cut

Figure B.2: *The solution to a minimum cut problem is very sensitive to outliers. For example in (a) the solution to the minimum cut would likely be shown by the dotted line. A better partitioning would be one which normalizes these outliers so they have less effect on the computation of the cut. Here, (b) shows the ideal cut which could be achieved through normalization.*

A better partitioning would take the relative size of each cut into account, such that outliers have limited effect on the calculation of the cut. This can be done by weighting the cut by the total size of each partition. The total size of each partition can be calculated by summing all of the edge weight terms in the partition, this measure is called the association. For partition A this is defined as,

$$assoc\,(A, V) = \sum_{i \in A, j \in V} w_{i,j} \tag{B.6}$$

where V represents all the nodes in A. Using this, a normalized cut definition can be created,

$$nCut\,(A,B) = \frac{cut\,(A,B)}{assoc\,(A,V)} + \frac{cut\,(A,B)}{assoc\,(B,V)} \tag{B.7}$$

This cut will weight each partition according to the total weight of the partition. Resulting in a cut that is less sensitive to outliers, and that will perform similarly to the cut shown in Figure B.2-(b). Calculating this cut is computationally difficult, but it can be done through spectral graph theory analysis of the Laplacian matrix for the weighted graph [33].

B.3 Calculating the Minimum Normalized Cut

It has been shown that the minimum normalized cut can be found using the graph Laplacian [33],

$$\min nCut\,(A,B) = \min_y \frac{y^T\,(\mathbf{D}-\mathbf{W})\,y}{y^T\mathbf{D}y} \tag{B.8}$$

where y is the partitioning vector, that is, a vector where each element corresponds to a node. It has also been shown that this can be solved using eigenanalysis if the values of y are allowed to take on real values. This is done by solving the eigenvalue system,

$$(\mathbf{D}-\mathbf{W})\,y = \lambda\mathbf{D}y \tag{B.9}$$

The smallest eigenvalue for this system will be equal to zero, and have an eigenvector with values equal to one. The second smallest eigenvalue would provide the solution to Equation B.8. This second smallest eigenvector can be used to partition the graph. This is done by solving Equation B.8 with this eigenvector, and using the solution to partition the eigenvector. This partition can be used to segment the nodes in the graph into two segments. A recursive segmentation algorithm can be created from this process.

1. Create a weighted graph representation of the data
2. Calculate the second smallest eigenvector for the graph Laplacian
3. Use this eigenvector to partition the graph into two smaller graphs
4. Repeat steps 2-4 on the smaller graphs until the cut size becomes too small, as defined by a threshold

Appendix C

Datasets

This appendix presents the details each dataset used in this work. Three real datasets were used for testing the geometrical and physical accuracy of the three-dimensional modeling processes. A fourth synthetic dataset was created for geoaccuracy analysis.

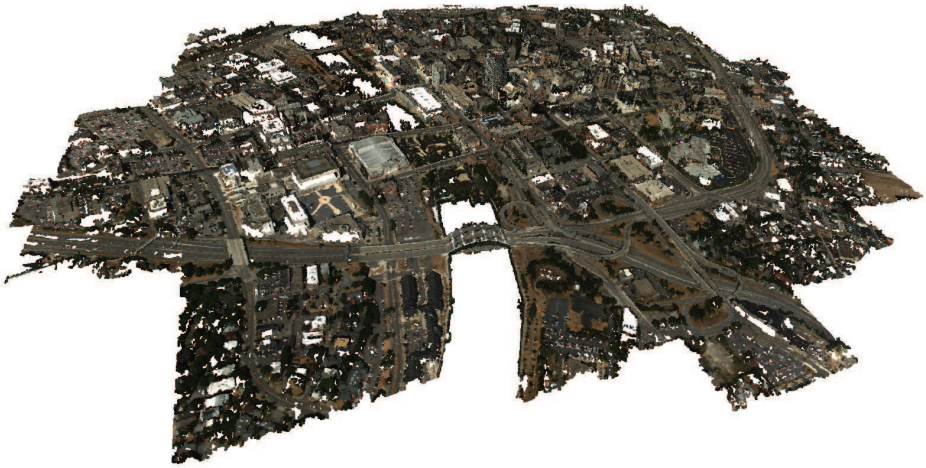
The first two datasets were collected with the RIT Wildfire Airborne Sensor Program (WASP) imaging platform. The WASP platform is comprised of four imaging sensors; a 4000x2672 pixel visible/near infrared (VNIR), a 640x512 pixel short wave Infrared (SWIR),a 640x512 pixel midwave infrared (MWIR), and a 640x512 pixel long wave infrared (LWIR) sensor [40]. The WASP datasets shown in the following sections only take advantage of the VNIR imagery. The final real dataset was provided by SpecTIR using their airborne hyperspectral platform, as part of the SHARE2010 collect [26].

C.1 Downtown Rochester, NY

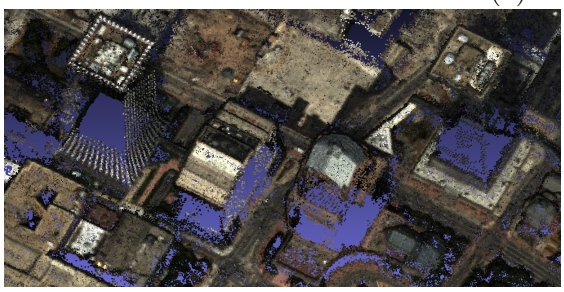
The data presented in this section was collected over downtown Rochester, NY. This data was collected with the intention of use in SfM processing algorithms. Each image was collected with approximately 80% forward overlap and 90% side overlap. Flightlines were flown east-west, north-south, and in each cardinal direction over the city [52]. This provided a very dense collection of imagery over the center of Rochester. Each image is 4000x2672 pixels with a GSD of approximately 0.3 meters. Figure C.1 shows the camera centers drawn over an aerial map of the downtown area.

Figure C.1: *The camera centers for each image capture for the data collect over downtown Rochester, NY (approx. 2 sq. km.). This collect was specifically designed to have high overlap for use with three-dimensional reconstruction algorithms. The yellow triangles represent the densest area of the collect.*

A 120-image subset of the densest section of the collect shown in Figure C.1 was processed using the software described in Section 3.1. Figure C.2 shows the dense geoaccurate point cloud generated using this process.



(a) Full view



(b) Four buildings point cloud

(c) Four buildings image

Figure C.2: *The geoaccurate dense point cloud reconstruction (a) of downtown Rochester, NY using a 120-image subset of the densest section of the collect shown in Figure C.1. The four center buildings that are used for processing are shown in (b), as well as a nadir-looking reference image in (c) for context.*

A manual verification of the accuracy was performed, yielding an average error of approximately 0.3 meters, the GSD for the collection. Four buildings were segmented from the point cloud for model extraction processing, the results of this processing can be seen in Chapter 4.

C.2 RIT Dataset

This dataset was collected over the Rochester Institute of Technology campus with the intention of use with SfM processing algorithms. The imagery was collected with significant overlap over the entire campus. Each image is 4000x2672 with a GSD of approximately $0.3m$, the total area covered is shown in Figure C.3 and the reconstructed dense point cloud is shown in Figure C.4.

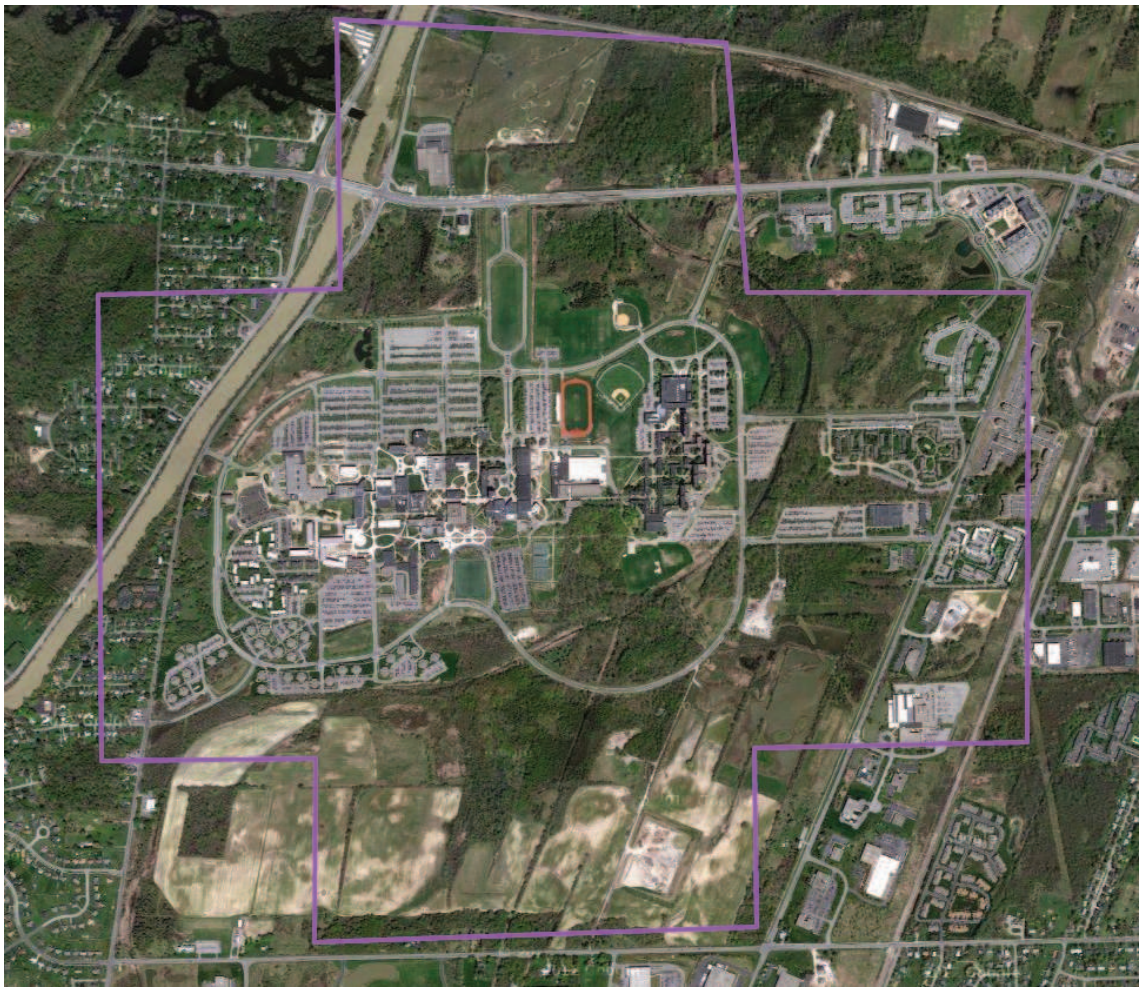
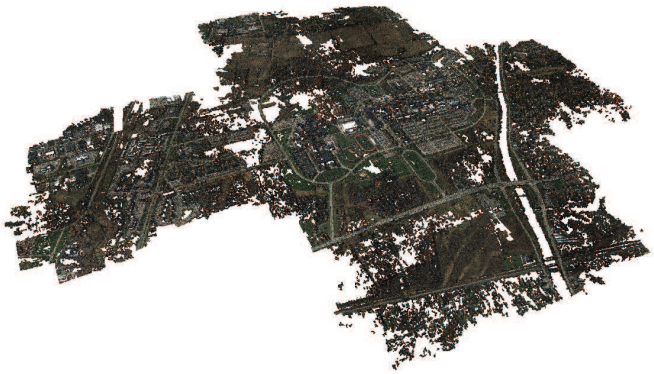
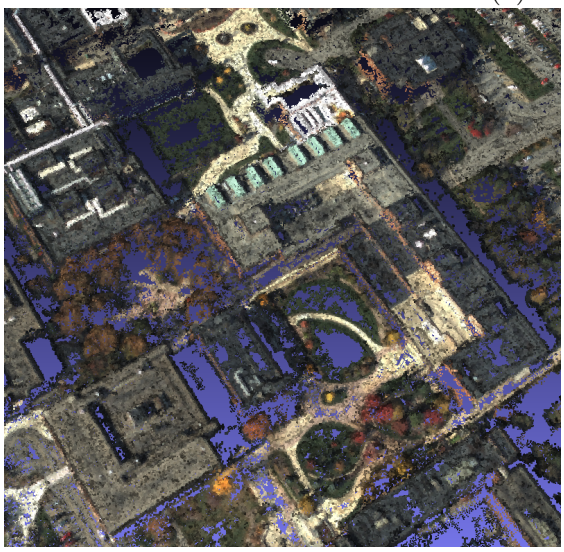


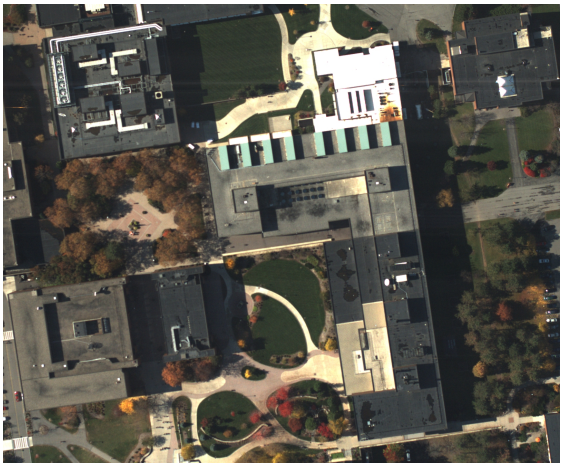
Figure C.3: *The area covered by the RIT dense imagery collect*



(a) Full view



(b) Four buildings point cloud



(c) Four buildings image

Figure C.4: *The geoaccurate dense point cloud reconstruction (a) of RIT. The four center buildings which were used for processing are shown in (b), as well as a nadir-looking reference image in (c) for context.*

C.3 SHARE-2010

Another collection that was incorporated in this work was the SHARE-2010 collection [26]. This collection was an attempt to generate a collection of multi-modal imagery, which included airborne hyperspectral, LiDAR, and high-resolution multispectral imagery. One section of the SHARE-2010 collect was a high resolution hyperspectral collect over the Rochester Institute of Technology campus. Figure C.5 shows the footprints of the data collected. Each flightline was calibrated and atmospherically compensated by the data provider so the imagery is presented in reflectance units. This hyperspectral data was used in combination with the data described in Section C.2 to perform the model reflectance attribution as described in Section 3.4.1.

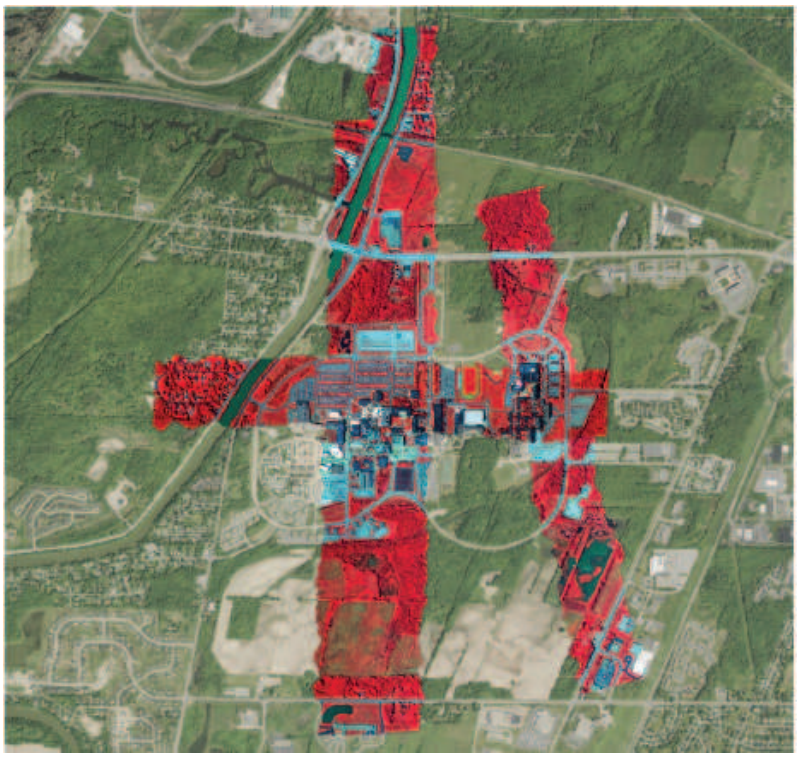


Figure C.5: *The footprint of the hyperspectral collect over RIT from the SHARE 2010 data collect. This figure was provided by SpecTIR.*

C.4 Synthetic DIRSIG Dataset

A synthetic dataset was created for geoaccuracy testing, based on a previously created synthetic dataset[37]. Two versions of this dataset were created, one with 10 nadir-looking images, and one with 100 nadir looking images. Figure C.6 shows a sample of synthetic images.

(a) (b) (c)

Figure C.6: *(a) through (c) show different views of the synthetic image created for geoaccuracy analysis.*

These images were processed through the SfM workflow, extracting three-dimensional structure that was used for testing. A view of both versions of the point cloud is shown in Figure C.7. As expected, the point cloud with more imagery has a denser reconstruction.

(a) Point cloud from 10 images (b) Point cloud from 100 images

Figure C.7: *Two sets of synthetic images were produced, one with ten images and one with one hundred. This shows the point cloud reconstructions from each, (a) 10 (b) 100. As expected, the point cloud which was created using more imagery has a denser reconstruction.*

Appendix D

Structure from Motion Workflow Tutorial

A complete workflow for running the software presented in Section 3.1 was developed for this work. An algorithm which implements the georegistration transform (Section 2.5) using camera centers (Section 3.2.1) was written and implemented as part of this SfM workflow. This tutorial will detail how to install and run this software. This tutorial can also be found online at `http://dirsapps.cis.rit.edu/3d-workflow/?q=3d-workflow`.

It should be noted that the computational requirements for this software are very high. Beyond needing a CUDA-capable GPU, mutlicore CPUs are recommended for reasonable processing times. For reference, this software has been successfully run on the following systems:

- Intel Core i5-2400 3.10Ghz Quad Core, 8 GB RAM, nVidia GTX 460 V2 Fermi 1GB 336 processing cores
- Intel Pentium G6950 2.8Ghz Dual Core, 16 GB RAM, nVidia Tesla C1060 4GB 240 processing cores
- (Dual) Intel Xeon X5680 3.33Ghz Hex Core Hyper-threaded, 72GB RAM, nVidia GT430 2GB 96 processing cores

D.1 Installation

The installation described here is for a machine running the latest version of Fedora (`https://fedoraproject.org/`). A number of supporting third party software packages must be installed for the successful compilation of siftGPU, Bundler and PMVS/CMVS.

```
Update First

$ sudo yum update -y --skip-broken

Reboot if needed then perform the the following command

$ sudo yum install git patch make gcc-c++ freeglut-devel
  libXi-devel libXmu-devel DevIL-devel boost-devel
  gsl-devel libjpeg-devel lapack-devel zlib-devel
  opencv opencv-devel
```

D.1.1 Installing CUDA

The NVIDIA CUDA module must be installed for siftGPU. This requires a number of steps for successful installation.

```
Install the RPM fusion repositories

$ sudo yum localinstall --nogpgcheck
http://download1.rpmfusion.org/free/fedora/
rpmfusion-free-release-stable.noarch.rpm
http://download1.rpmfusion.org/nonfree/fedora/
rpmfusion-nonfree-release-stable.noarch.rpm

Install the NVIDIA drivers

$ sudo yum install kmod-nvidia akmod-nvidia kernel-devel

Back up the initramfs image and generate a new one
```

```
$ sudo mv /boot/initramfs-$(uname -r).img
                        /boot/initramfs-$(uname-r).nouveau.img
$ sudo dracut /boot/initramfs-$(uname -r).img $(uname -r)
```

`Run nvidia-xconfig`

```
$ sudo nvidia-xconfig
```

Once the proper NVIDIA drivers have been installed, the CUDA toolkit should be downloaded (`https://developer.nvidia.com/cuda-toolkit`) and installed.

```
$ chmod +x cudatoolkit
$ sudo cudatoolkit
```

Two files must be edited to complete the CUDA installation.

`In /usr/local/cuda/include/host_config.h`

```
Change Line 80 to:
#if __GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ > 7)
```

`In ~/.bashrc add`

```
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=
                $LD_LIBRARY_PATH:/usr/local/cuda/lib
                                    :/usr/local/cuda/lib64
```

D.1.2 Installing Graclus

Version 1.2 of Graclus needs to be installed for CMVS. This software can be downloaded from their website (`http://www.cs.utexas.edu/users/dml/Software/graclus.html`). If building on a 64-bit system the following change must be made:

`In graclus1.2/Makefile.in, Change line 11 to:`

```
COPTIONS = -DNUMBITS=64
```

Once this is done, Graclus can be built, and installed.

```
$ cd graclus1.2
$ make
$ sudo mv *.a /usr/lib
$ sudo cp -p metisLib/*.h /usr/include
```

D.1.3 Installing the SfM Workflow

The last step in this installation process is to download and install the workflow which was developed as part of this research. This workflow can be downloaded as shown,

```
$ git clone https://github.com/drn2369/workflow-3d.git
$ cd workflow-3d/src
$ sh setup.sh
```

Running setup.sh will download and install Bundler, PMVS and CMVS from their respective websites. Finally, typing `make` in the source directory will build each software package as well as GTransform.

```
In workflow-3d/src/

$ make
```

Finally, the script which runs the workflow must be updated with the full path to the workflow directory.

```
In workflow-3d/scripts/RunProcess.sh

BASEPATH=/full/path/to/workflow-3d
```

D.2 Example Usage

The workflow-3d git repository contains a demonstration set of data. The following directory tree is used for processing,

```
/
├── bin
├── src
├── scripts
├── data
│   └── Rochester-Demo
└── workspace
    └── demo
```

It is recommended that the data and processing sections be kept in separate directories. This allows for multiple processes to be run on large datasets without having to copy imagery between directories. The workspace directory tree is where the processing is done. Every process requires it's own directory.

The RunProcess script located in the script directory must be copied into each processing folder and run from the base level of that folder. Each folder must also contain a file named *list.txt*, that contains the full path to each image in the data directory which will be processed. In order to run the full demo, type the following

```
$ cd workspace/demo
$ sh RunProcess.sh -ag
```

Upon successful completion, the following output should be seen,

```
[- Prepping Data -]
[- Running siftGPU -]
[- Running Bundler -]
[- Preparing PMVS -]
[- Running CMVS -]
[- Running PMVS -]
[- Running GTransform -]
[- Clean Up -]
[- Done -]
```

This will run the full SfM workflow and produce the following directory tree

```
demo
```

```
├── logs
└── results
    ├── Backup
    └── trans
        ├── models
        └── txt
```

The non-georegistered point clouds will be located in the results directory. The georegistered point clouds will be located in the trans/models directory, and the corresponding camera projection matrices will be located in the trans/txt directory. An example of the expected output for each point cloud is shown in Figure D.1.

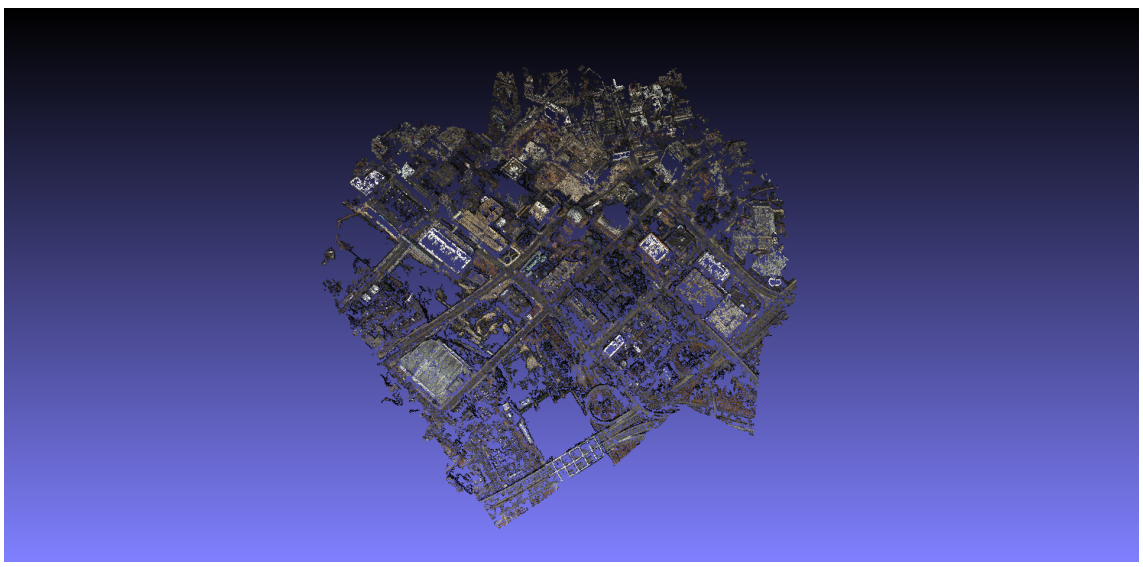


Figure D.1: *This shows the expected point cloud result for running the demo data through the SfM workflow.*

D.2.1 RunProcess.sh Script Parameters

The RunProcess script has a number of parameters that can be altered to adjust the workflow. These parameters are shown in Table E.1.

Table D.1: A description of all the parameters that can be used in the RunProcess.sh script

Parameter	Description
-a	Run the whole workflow (equivalent to -sbp)
-s	Run siftGPU
-b	Run Bundler
-p	Run CMVS/PMVS
-g	Run GTransform
-k	Prevent the script from running cleanup. This will keep all the output of siftGPU, Bundler, CMVS/PMVS, and GTransform.
-d	Set the GPU device number for siftGPU (Default: 0)
-x	Set the maximum dimension for an image for siftGPU, this will cause siftGPU to down sample images larger than this dimension (Default: 2000).
-y	Set the maximum number of features for siftGPU. This is useful for running siftGPU at full resolution but limiting the memory usage (Default: 8000).
-f	Set the focal length of the camera used to take the images, currently assuming all images taken with the same camera at the same focal length. The focal length must be in pixel units (focal length in mm / pixel size in mm) (Default: 6111.11).
-i	Set the focal length constraint weight for the bundler adjustment in Bundler (Default: 0.0001).
-c	Set the max cluster size for CMVS (Default: 30).
-l	Set the level for PMVS (Default: 1).
-e	Set the cell size for PMVS (Default: 2).
-t	Set the threshold level for PMVS (Default: 0.7).
-w	Set the window size for PMVS (Default: 7).
-m	Set the min image number for PMVS (Default: 3).
-u	Set the number of CPUs to use for CMVS and PMVS.

D.2.2 Running additional data

Processing additional imagery which was not provided with the repository is possible. There are a few parameters that must be set based on data from the imagery. First, the maximum resolution must be set such that it does not exceed the resolution of the

imagery. Second, the focal length of the camera must be set.

In order to run GTransform, each image must be accompanied by a POS file. The format of the POS file must be as follows,

```
imageFileName.pos

X-position Y-position Z-position
```

where `imageFileName.pos` is the exact file name for the image file, with the `.pos` extension. Also each X,Y and Z parameter is the location of the camera center for each image. These coordinates should be in the desired Euclidean output coordinate system (*i.e.* UTM, NEU, etc).

Appendix E

Three-dimensional Surface Estimation and Classification Software

This appendix presents the software written for the surface estimation and R,G,B segmentation described in Section 3.3.2 and Section 3.4.2. These two methods are combined into one software code. Steps for installation and running a demo reconstruction and segmentation are presented here. The output of this code are three point clouds. The first point cloud is a colorized point cloud where each point is the voxel center. The second point cloud, also using voxel centers, is colored with the initial region-growing segmentation. The third point cloud is colored with the final segmentation.

E.1 Installation

This software uses the OpenCV [21] and Point Cloud Library (PCL) [6] libraries for processing. These libraries are widely used and available for many platforms. This section will discuss installation on a machine running the latest version of Fedora. These dependencies, along with CMake and Git, can be installed as follows,

```
$ sudo yum install git cmake opencv opencv-devel
  pcl pcl-devel pcl-tools pcl-doc
```

The source code for this software can be downloaded from the git repository,

```
$ git clone https://github.com/drn2369/voxelProcessing.git
```

Lastly, the binary can be built using the following commands:

```
$ cd voxelProcessing/src/
$ mkdir build
$ cd build
$ cmake ..
$ make
```

The binary will be in the `voxelProcessing/bin/` directory. A set of demo data are included in the repository and can be found int he `voxelProcessing/demo/` directory.

E.2 Usage

This code is intended to be used on individual structures separated into their own individual Standford PLY files [54]. This is a manual process for now, and can be done using common point cloud editing software, such as Meshlab [8]. The following arguments are used to run the voxelProcessing software,

```
$ voxelProcessing plyPath transPath visPath voxelSize numMaterials
                depth useDiagonalNorms useLightness
```

A description of each of these parameters is found below.

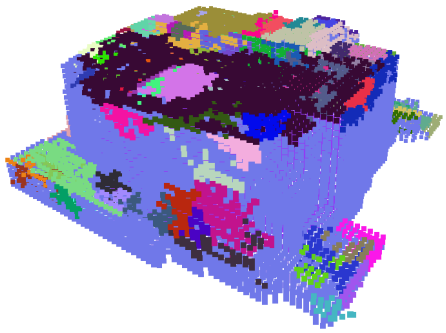
Table E.1: A description of all the input parameters for the voxel processing software

Parameter	Description
plyPath	The path to a single building cropped from a ply file
transPath	The path to a txt file containing the full path to each projection matrix
visPath	The path to a txt file containing the full path to each image
voxelSize	The desired voxel size in world coordinate units
numMaterials	An estimate of the number of materials on the object
depth	The length along a projected ray that will be used to consider if a voxel is occluded or not. Default is 1, 2 is better for taller buildings.
useDiagonalNorms	Flag to turn diagonal norms on, default is 0 (false).
useLightness	use L from HSL color default is 1 (true).

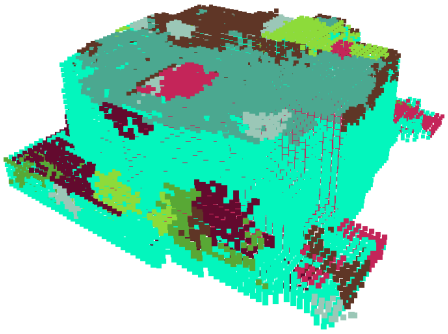
An example output from this software is shown in Figure E.1 and can be reproduced using the included data. After installing the software, use the following commands to run the code:

```
$ cd voxelProcessing/demo/76/
$ ../../bin/voxelProcessing 76-crop.ply trans.txt vis.txt
                                0.004 8 1 1 1
```

The software will run on this code, and produce the three point clouds shown in Figure E.1. The code will produce three point clouds files; *segmentedVoxels_init.ply*, *segmentedVoxels.ply*, and *voxelTrueColor.ply*.



(a) *segmentedVoxels_init.ply*



(b) *segmentedVoxels.ply*

(c) *voxelTrueColor.ply*

Figure E.1: *The voxel processing software produces three point clouds. The points in each of these point clouds are created from the voxel centers. The points in (a) are colorized to show the initial segmentation from the region-growing approach. The points in (b) are colored to show the final segmentation. Finally, the points in (c) represents the true color for all the voxel centers. The uncolored voxel centers were occluded from every image, and therefore do not have color.*

Appendix F

Surface Attribution with Hyperspectral Imagery

This appendix presents the software written for registering and attributing hyperspectral imagery with three-dimensional models, as described in Section 3.4.1. This software is a combination of C++ and MATLAB code. Steps for installation and running the demo are presented here. The output of this code is a DIRSIG [58] model with facets that have been attributed using atmospherically corrected hyperspectral imagery.

F.1 Installation

This software uses OpenCV, GDAL, and MATLAB. The MATLAB scripts could also be run using GNU Octave. This section will discuss installation on a machine running the latest version of Fedora. These dependencies can be installed as follows,

```
$ sudo yum install git opencv opencv-devel
  gdal gdal-devel octave-forge
```

The source code for this software can be downloaded from the git repository,

```
$ git clone https://github.com/drn2369/genMaterials.git
```

The binaries can be built using the following commands,

```
$ cd genMaterials/src/genSpec
$ make
```

The binaries will be built in `genMaterials/bin/`.

F.2 Usage

A demo is included in this software, however, it requires the use of a hyperspectral data collection which must be downloaded from the RIT SHARE 2010 data collect [26]. This data can be downloaded at `http://dirsapps.cis.rit.edu/share-2010/cgi-bin/share-2010.pl`. This demo requires the data file named `001_0729-1929_ref_corr.dat`.

The software consists of two steps, the first step attributes the point cloud with the hyperspectral reflectance data. This is performed using the `genSpec` software, this software uses an options file to read in parameters. The options file has the following format,

```
Path to projection matrix of the ortho map
Base path to orthorectification map
Spec (leave this as is)
Path to ply file of structure to attribute
```

For the demo, the only path which must be changed, is the one which points to the hyperspectral data file `001_0729-1929_ref_corr.dat`. The `genSpec` software can then be as follows,

```
$ cd genMaterials/demo/7
$ ../../bin/genSpec options.txt
```

This will create a `.spec` file, which can then be used with the MATLAB script to generate the input DIRSIG files. This can be done by running the MATLAB script found here at `genMaterials/src/MATLAB/GenerateDIRSIGFiles.m`. This script will generate a set of DIRSIG model files which can be used in a DIRSIG scene.

F.2.1 Use with your own data

The orthorectification map which is used to generate the mapping from the original image to the hyperspectral image must be made. This can be done by using any orthorectification software, along with maps which have been generated in the manner described in Section 3.4.1. This work used the open source software package OSSIM [53] for orthorectification.

There are a number of edits that must be made to the MATLAB script for customization. The input variable section at the top of the script must be edited. This script

was written specifically for the data from the SHARE 2010 collection, however a different hyperspectral sensor could be used. This would require the user to edit the `wavelength` variable at line `367` to be the wavelengths of the spectra generated in the `.spec` file, in microns.

Bibliography

[1] Bjorck A. *Numerical methods for least squares problems*. 51. Society for Industrial Mathematics, 1996.

[2] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. "Freak: Fast retina keypoint". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 510–517.

[3] Applanix. *POSTrack Specifications*. URL: `www.applanix.com`.

[4] Blostein S.D. Arun K.S Huang T.S. "Least-squares fitting of two 3-D point sets". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 9.5 (1987), pp. 698–700.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features". In: *Computer Vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[6] Cousins S. Bogdan R. "3D is here: Point Cloud Library (PCL)". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, 2011.

[7] Zabih R. Boykov Y. Veksler O. "Fast approximate energy minimization via graph cuts". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 23.11 (2001), pp. 1222–1239.

[8] Visual Computing Lab ISTI CNR. *MeshLab*. URL: `http://meshlab.sourceforge.net/`.

[9] Yuille A. Coughlan J. "Manhattan world: Compass direction from a single image by bayesian inference". In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Vol. 2. IEEE. 1999, pp. 941–947.

[10] A. Snavely N. Crandall D. Owens and Huttenlocher D. "Discrete-continuous optimization for large-scale structure from motion". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 3001–3008.

[11] Nistér D. "An efficient solution to the five-point relative pose problem". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 26.6 (2004), pp. 756–770.

[12] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 886–893.

[13] Lowe D.G. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[14] Marquardt D.W. "An algorithm for least-squares estimation of nonlinear parameters". In: *Journal of the Society for Industrial & Applied Mathematics* 11.2 (1963), pp. 431–441.

[15] Mücke E. Edelsbrunner H. "Three-dimensional alpha shapes". In: *ACM Transactions on Graphics (TOG)* 13.1 (1994), pp. 43–72.

[16] Seidel R. Edelsbrunner H. Kirkpatrick D. "On the shape of a set of points in the plane". In: *Information Theory, IEEE Transactions on* 29.4 (1983), pp. 551–559.

[17] Bolles R.C. Fischler M.A. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[18] Ponce J. Furukawa Y. "Accurate, dense, and robust multiview stereopsis". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.8 (2010), pp. 1362–1376.

[19] Ponce J. Furukawa Y. *Patch-based Multi-view Stereo Software (PMVS - Version 2)*. URL: `http://grail.cs.washington.edu/software/pmvs/`.

[20] Seitz S.M. Szeliski R. Furukawa Y. Curless B. "Towards internet-scale multi-view stereo". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE. 2010, pp. 1434–1441.

[21] Bradski G. "OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[22] Bowles J.H. Gillis D. B. "Hyperspectral image segmentation using spatial-spectral graphs". In: *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics. 2012, 83901Q–83901Q.

[23] Coxeter H. *Projective geometry*. Springer Verlag, 2003.

[24] Hirschmuller H. "Accurate and efficient stereo processing by semi-global matching and mutual information". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 2. IEEE. 2005, pp. 807–814.

[25] Zisserman A. Hartley R. *Multiple View Geometry in Computer Vision*. 2008.

[26] Weatherbee O. Messinger D. van Aardt J. Ientilucci E. Ninkov Z. Faulring J. Raqueño N. Herweg J. Kerekes J. and Meola J. "SpecTIR hyperspectral airborne Rochester experiment data collection campaign". In: *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics. 2012, pp. 839028–839028.

[27] Smolic A. Frohlich B. Wiegand T. Heymann S. Muller K. "SIFT implementation and optimization for general-purpose GPU". In: *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*. 2007, p. 144.

[28] Tarjan R. Hopcroft J. "Algorithm 447: efficient algorithms for graph manipulation". In: *Communications of the ACM* 16.6 (1973), pp. 372–378.

[29] Brown S. Ientilucci E. "Advances in wide-area hyperspectral image simulation". In: *AeroSense 2003*. International Society for Optics and Photonics. 2003, pp. 110–121.

[30] Boissonnat J. "Geometric structures for three-dimensional shape representation". In: *ACM Transactions on Graphics (TOG)* 3.4 (1984), pp. 266–286.

[31] MacQueen J. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 281-297. California, USA. 1967, p. 14.

[32] Philip J. "A Non-Iterative Algorithm for Determining All Essential Matrices Corresponding to Five Point Pairs". In: *The Photogrammetric Record* 15.88 (1996), pp. 589–599.

[33] Shi J. and Malik J. "Normalized cuts and image segmentation". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.8 (2000), pp. 888–905.

[34] Schunck B. Jain R. Kasturi R. *Machine vision*. Vol. 5.

[35] Schaefer S. Warren J. Ju T. Losasso F. "Dual contouring of hermite data". In: *ACM Transactions on Graphics (TOG)*. Vol. 21. 3. ACM. 2002, pp. 339–346.

[36] Gwun O. Juan L. "A comparison of sift, pca-sift and surf". In: *International Journal of Image Processing (IJIP)* 3.4 (2009), pp. 143–152.

[37] Salvaggio K. and Salvaggio C. "Automated identification of voids in three-dimensional point clouds". In: *SPIE Optical Engineering+ Applications*. International Society for Optics and Photonics. 2013, 88660H–88660H.

[38] Walli K. "Relating Multimodal Imagery Data in 3D". Ph.D. Thesis. Rochester Institute of Technology.

[39] Hoppe H. Kazhdan M. Bolitho M. "Poisson surface reconstruction". In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. 2006.

[40] DIRS Laboratory. *Wildfire Airborne Sensor Program (WASP)*. URL: `http://lias.cis.rit.edu/projects/wasp`.

[41] Kerekes J.P. Lach S.R. "Multisource data processing for semi-automated radiometrically-correct scene simulation". In: *Urban Remote Sensing Joint Event, 2007*. IEEE. 2007, pp. 1–10.

[42] Hartley R. Li H. "Five-point motion estimation made easy". In: *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*. Vol. 1. IEEE. 2006, pp. 630–633.

[43] Cline W. Lorensen W. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM Siggraph Computer Graphics*. Vol. 21. 4. ACM. 1987, pp. 163–169.

[44] Argyros A.A. Lourakis M.I.A. "SBA: A software package for generic sparse bundle adjustment". In: *ACM Transactions on Mathematical Software (TOMS)* 36.1 (2009), p. 2.

[45] Xie Z. Mathews V.J. "A stochastic gradient adaptive filter with gradient adaptive step size". In: *Signal Processing, IEEE Transactions on* 41.6 (1993), pp. 2075–2087.

[46] Bethel J.S. Mullen R. American Society for Photogrammetry McGlone J.C. Mikhail E.M. and Remote Sensing. *Manual of Photogrammetry*. American Society for Photogrammetry and Remote Sensing, 2004.

[47] Vining G.G. Montgomery D.C. Peck E.A. *Introduction to linear regression analysis*. Vol. 821. Wiley, 2012.

[48] Yu G. Morel J.M. "ASIFT: A new framework for fully affine invariant image comparison". In: *SIAM Journal on Imaging Sciences* 2.2 (2009), pp. 438–469.

[49] Snavely N. *Bundler: Structure from Motion (SfM) for Unordered Image Collections*. URL: `http://phototour.cs.washington.edu/bundler/`.

[50] Snavely N. "Scene Reconstruction and Visualization from Internet Photo Collections". Ph.D Thesis. University of Washington, 2008.

[51] Salvaggio C. Nilosek D. "Applying computer vision techniques to perform semi-automated analytical photogrammetry". In: *Image Processing Workshop (WNYIPW), 2010 Western New York*. IEEE. 2010, pp. 1–5.

[52] Nilosek D. Ontiveros E. Salvaggio C. *3D-Rochester Image and LiDAR Dataset for Point Cloud Reconstruction and Processing Algorithms*. URL: `http://dirsapps.cis.rit.edu/3d-rochester/`.

[53] OSGeo. *OSSIM: Advanced Image Processing and Geospatial Data Fusion*. URL: `http://trac.osgeo.org/ossim/wiki`.

[54] *PLY - Polygon File Format*. URL: `http://paulbourke.net/dataformats/ply/`.

[55] Mundy J. Pollard T. "Change detection in a 3-d world". In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–6.

[56] Ethan Rublee et al. "ORB: an efficient alternative to SIFT or SURF". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2564–2571.

[57] Brown S. *DIRSIG Documentation*. URL: `http://dirsig.org/documentation`.

[58] Brown S. *DIRSIG: The Digital Imaging and Remote Sensing Image Generation model*. URL: `http://dirsig.org/`.

[59] Sun S. "Automatic 3D Building Detection and Modeling from Airborne LiDAR Point Clouds". Ph.D Thesis. Rochester Institute of Technology.

[60] Zara J. Sedlacek D. "Graph Cut Based Point-Cloud Segmentation for Polygonal Reconstruction". In: *ISVC Part II*. Ed. by G. Bebis. 2009.

[61] Merriman M. Smith D. *History of modern mathematics*. J. Wiley & Sons, 1896.

[62] Szeliski R. Snavely N. Seitz S. "Photo tourism: exploring photo collections in 3D". In: *ACM transactions on graphics (TOG)*. Vol. 25. 3. ACM. 2006, pp. 835–846.

[63] Olson E. Strom J. Richardson A. "Graph-based Segmentation for Colored 3D Laser Point Clouds". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2010.

[64] Fua P. Tola E. Lepetit V. "A fast local descriptor for dense matching". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE. 2008, pp. 1–8.

[65] Fua P. Tola E. Lepetit V. "Daisy: An efficient dense descriptor applied to wide-baseline stereo". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.5 (2010), pp. 815–830.

[66] Dijkman S. Vosselman D. "3D building model reconstruction from point clouds and ground plans". In: *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences* 34.3/W4 (2001), pp. 37–44.

[67] Changchang W. *SiftGPU: A GPU Implementation of Scale Invariant Feature Transform (SIFT).* URL: `http://cs.unc.edu/~ccwu/siftgpu/`.

[68] Kabsch W. "A discussion of the solution for the best rotation to relate two sets of vectors". In: *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 34 (1978), pp. 827–828.

[69] Kabsch W. "A solution for the best rotation to relate two sets of vectors". In: *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 32.5 (1976), pp. 922–923.

[70] Paul B. Brower B. Pellechia M. Walvoord D. Rossi A. "Geoaccurate three-dimensional reconstruction via image-based geometry". In: *SPIE Defense, Security, and Sensing.* International Society for Optics and Photonics. 2013, pp. 874706–874706.

[71] Snavely N. Wang C. Wilson K. "Accurate Georegistration of Point Clouds Using Geographic Data". In: *3DTV-Conference, 2013 International Conference on.* IEEE. 2013, pp. 33–40.

[72] Irschara A. Wendel A. and Bischof H. "Automatic alignment of 3D reconstructions using a digital surface model". In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on.* IEEE. 2011, pp. 29–36.

[73] Dewitt B.A. Wolf P.R. *Elements of Photogrammetry: with applications in GIS.* Vol. 3. McGraw-Hill New York, NY, USA, 2000.

[74] Furukawa Y. *Clustering Views for Multi-view Stereo (CMVS).* URL: `http://grail.cs.washington.edu/software/cmvs/`.

[75] Yong-Liang Yang et al. "Robust principal curvatures on multiple scales". In: *Symposium on Geometry Processing*. 2006, pp. 223–226.

[76] Neumann U. Zhou Q. "2.5 D dual contouring: a robust approach to creating building models from Aerial LiDAR point clouds". In: *Computer Vision–ECCV 2010*. Springer, 2010, pp. 115–128.