5-1-2008

Semi-Automated DIRSIG scene modeling from 3D lidar and passive imagery

Stephen R. Lach

Semi-Automated DIRSIG Scene Modeling from 3D Lidar and Passive Imagery

by

Stephen R. Lach

B.S. Villanova Univerity, 1996

M.S. Villanova Univerity, 1998

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
Rochester Institute of Technology

May 23, 2008

Signature of the Author ____________________________________

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

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

CERTIFICATE OF APPROVAL

Ph.D. DEGREE DISSERTATION

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

____
Dr. John Kerekes, dissertation Advisor

____
Dr. John Schott

____
Dr. David Messinger

____
Dr. Vincent Amuso

____
Date

Thesis/Dissertation Author Permission Statement

Title of dissertation: ”Semi-Automated DIRSIG Scene Modeling from 3D Lidar and Passive Imagery”
Name of author: Stephen R. Lach
Degree: Ph.D. Imaging Science

I understand that I must submit a print copy of my thesis or dissertation to the RIT Archives, per current RIT guidelines for the completion of my degree. I hereby grant to the Rochester Institute of Technology and its agents the non-exclusive license to archive and make accessible my thesis or dissertation in whole or in part in all forms of media in perpetuity. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Print Reproduction Permission Granted:

I, Stephen R. Lach, hereby grant permission to the Rochester Institute Technology to reproduce my print thesis or dissertation in whole or in part. Any reproduction will not be for commercial use or profit.

Signature of Author: ______________________________
Date

Inclusion in the RIT Digital Media Library Electronic Thesis & Dissertation (ETD) Archive:

I, Stephen R. Lach, additionally grant to the Rochester Institute of Technology Digital Media Library (RIT DML) the non-exclusive license to archive and provide electronic access to my thesis or dissertation in whole or in part in all forms of media in perpetuity.

I understand that my work, in addition to its bibliographic record and abstract, will be available to the world-wide community of scholars and researchers through the RIT DML. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation. I am aware that the Rochester Institute of Technology does not require registration of copyright for ETDs.

I hereby certify that, if appropriate, I have obtained and attached written permission statements from the owners of each third party copyrighted matter to be included in my thesis or dissertation. I certify that the version I submitted is the same as that approved by my committee.

Signature of Author: ______________________________
Date

DISCLAIMER

Semi-Automated DIRSIG Scene Modeling from 3D Lidar and Passive Imagery

by
Stephen R. Lach

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 Digital Imaging and Remote Sensing Image Generation (DIRSIG) model is an established, first-principles based scene simulation tool that produces synthetic multispectral and hyperspectral images from the visible to long wave infrared (0.4 to 20 microns). Over the last few years, significant enhancements such as spectral polarimetric and active Light Detection and Ranging (lidar) models have also been incorporated into the software, providing an extremely powerful tool for multi-sensor algorithm testing and sensor evaluation. However, the extensive time required to create large-scale scenes has limited DIRSIG's ability to generate scenes ”on demand.” To date, scene generation has been a laborious, time-intensive process, as the terrain model, CAD objects and background maps have to be created and attributed manually.

To shorten the time required for this process, this research developed an approach to reduce the man-in-the-loop requirements for several aspects of synthetic scene construction. Through a fusion of 3D lidar data with passive imagery, we were able to semi-automate several of the required tasks in the DIRSIG scene creation process. Additionally, many of the remaining tasks realized a shortened implementation time through this application of multi-modal imagery.

Lidar data is exploited to identify ground and object features as well as to define initial tree location and building parameter estimates. These estimates are then refined by analyzing high-resolution frame array imagery using the concepts of projective geometry in lieu of the more common Euclidean approach found in most traditional photogrammetric references. Spectral imagery is also used to assign material characteristics to the modeled geometric objects. This is achieved through a modified atmospheric compensation applied to raw hyperspectral imagery.

These techniques have been successfully applied to imagery collected over the RIT campus and the greater Rochester area. The data used include multiple-return point information provided by an Optech lidar linescanning sensor, multispectral frame array imagery from the Wildfire Airborne Sensor Program (WASP) and WASP-lite sensors, and hyperspectral data from the Modular Imaging Spectrometer Instrument (MISI) and the COMPact Airborne Spectral Sensor (COMPASS). Information from these image sources was fused and processed using the semi-automated approach to provide the DIRSIG input files used to define a synthetic scene. When compared to the standard manual process for creating these files, we achieved approximately a tenfold increase in speed, as well as a significant increase in geometric accuracy.

Acknowledgements

I have been fortunate to work with numerous exceptional people during my time at RIT, many of whom have had a profound impact on this body of research. First and foremost, I would like to thank Dr. John Kerekes for serving as my research advisor and mentor. It was through his initial vision, guidance, patience and weekly feedback that this research was able to progress from an abstract initial concept to a refined, useable process. While he let me run with many ideas, he also had the foresight to help me avoid many dead-ends, and in doing so helped keep my research pointed in the right direction. I would also like to thank my other committee members, Dr. John Schott, Dr. David Messinger, and Dr. Vincent Amuso for the many hours they spent discussing my research with me and helping me to stay on track.

I also need to acknowledge the Imaging Science teaching staff that laid the technical foundations upon which I was able to complete this research. Of particular note were Dr. Roger Easton's optics course, Dr. Maria Helgeura's Fourier Analysis courses, Dr. Don Light's photogrammetry course, Dr. Schott's remote sensing sequence, Dr. Kerekes' pattern recognition course, and Dr. Harvey Rhody's image processing courses. It is through exceptional instructors such as these that students develop a passion for this field.

A special thanks also goes out to the DIRS staff, in particular Scott Brown, Dr. Emmett Ientilucci, Dr. Rolo Raqueño and Cindy Schultz. Scott, Emmett, and Rolo are Associate Scientists in the department, and their expertise allowed me to use them as a sounding board and sanity check on numerous occasions. Cindy continually went beyond the call of duty in providing logistical help throughout my entire time at RIT.

I have also received a significant amount of assistance from my fellow students over the past three-and-a-half years. Of particular note are Prudhvi Gurram and Susan Munn, who spent many hours working through the details of modern computer vision techniques with me. Mike Foster and Derek Walvoord were also key to this research, as they both provided many important insights and helped me refine ideas on a wide range of topics.

And of course, I owe a special debt of gratitude to my family, both immediate and extended. In particular, I would like to thank my Uncle Stuart; it has been wonderful having a family member who is able to clearly see solutions for the most difficult technical and mathematical issues I have faced in this research. I also wish to thank my parents for their continual encouragement, which has been consistently present for longer than I can remember. And most of all, I thank my wife Michelle for her constant support throughout the many long hours required by this project. Without her continually going over-and-above these past few years, I would certainly never have been able to complete this program.

Dedication

This dissertation is dedicated to my amazing wife Michelle, without whom this work would not have been possible.

Contents

List of Figures

"Every honest researcher I know admits he's just a professional amateur. He's doing whatever he's doing for the first time. That makes him an amateur. He has sense enough to know that he's going to have a lot of trouble, so that makes him a professional."

Charles Franklin Kettering, U. S. Engineer and Inventor

1 Introduction

In recent years, the remote sensing community has seen significant advances in the development of non-traditional imaging techniques. With the advent of commercial imaging spectrometers, hyperspectral image analysis has been near the forefront of this boom, and algorithms exploiting spectral information continue to evolve. However, in many cases, testing, validation and training of these algorithms requires the use of well-calibrated datasets, the generation of which can be expensive and time prohibitive. As such, quality hyperspectral synthetic imagery can be a tremendous asset, as often the simulated data is able to reduce or eliminate the need for costly real-world data collection campaigns. Additionally, realistic synthetic image generation (SIG) enables engineers to evaluate the final image products given the parameters of the sensor. This may be done to evaluate an existing sensor under a host of illumination, atmospheric and geometric scene conditions, but it may also be done before the sensor is even built. As such, SIG is often a critical component in the design process of new sensors.

Light Detection and Ranging (lidar) systems have undergone a similar maturation in the last decade, and the use of laser-scanning systems to produce three-dimensional imagery is now widespread. However, even with the proliferation of this technology, many researchers are still lacking quality data sets with corresponding truth measurements. This is especially true for those working in the area of multi-modal image fusion, where the lidar data is augmented by imagery from an additional sensor. In this field, accurate registration of the imagery is critical, yet without truth data the quality of this registration is often difficult to ascertain. However, with a good multi-modal SIG model, synthetic combined datasets with associated truth values are easy to obtain. Additionally, synthetic lidar images may be used for the characterization and design of sensors, much as they were in the hyperspectral case. For these reasons, the ability to produce accurate simulated imagery can be of great benefit to the lidar community as well.

The Digital Imaging and Remote Sensing Image Generation (DIRSIG) model described in Section 2.4 is a tool capable of creating this desired synthetic imagery. Given a pre-defined scene, it generates an accurate representation of what a modeled passive electro-optical (EO) or active lidar sensor would detect under specified conditions by modeling the relevant physical processes in the imaging chain. Its use has been well documented in the literature (see [Ientilucci and Brown 2003] and [Schott et al. 1999] for example), and it is currently being used in both academia and industry. However, while it is fairly easy to model different sensor designs, atmospheric conditions and geometries for a given scene, the current process for actually generating a DIRSIG scene is quite involved. As a case in point, the creation of the MegaScene documented in [Ientilucci and Brown 2003] took a team of researchers well over a year to complete.

This work aims to shorten the time required for this process by reducing the man-in-the-loop requirements for many of the aspects of DIRSIG scene construction. Through a fusion of 3D lidar data with passive EO imagery, many of the tasks required in the scene creation process may be at least partially automated. Additionally, most of the remaining tasks will also realize a shortened implementation time through the application of multi-modal imagery. It is anticipated that once the data have been collected, a scene on the order of the Megascene effort could be constructed by a single individual in approximately three weeks, and most of this time would be applied to the final human inspection of the results.

In addition to reducing the time required to generate a DIRSIG scene, such automated techniques also facilitate the creation of an accurate simulated environment for which ground truth is either difficult or impossible to obtain. In many practical scenarios, direct access to the region of interest is restricted, and the scene construction must be completed solely from remotely-sensed data. This is especially true in rapid-response disaster relief efforts and military applications. In these instances, the ability to properly fuse all data sources in a timely manner may be as important as the final fidelity of the resultant model.

The techniques described in this dissertation also have a host of applications outside the realm of synthetic scene construction. Many of these algorithms may be effectively used to generate digital city models used for urban planning, environmental planning and monitoring, and location-based services. Additional applications include cartography, corridor mapping, flood and risk assessment, and forestry management (See [Behan et al. 2000] and [Ma 2004]). With the decreasing costs of lidar and other three-dimensional imaging technologies, traditional photogrammetric efforts are likely to be replaced in many applications by multi-modal fused imaging methods.

This work focuses primarily on the development of the algorithms needed to achieve these objectives, as well as the application of these algorithms to data collected over the RIT campus and the surrounding area. Additional emphasis is placed on describing how well each step in the process was performed, evaluating the effectiveness of the process taken as a whole, and comparing these results to those obtained by traditional methods. While the results in general are most accurate when all data types are present, the methods presented here are often quite flexible, and the approach describes how individual techniques should be modified when certain data sources are degraded or unavailable.

1.1 Research Objectives

This section provides a general overview of the main objectives associated with this research effort. As noted above, the primary goal of this work was to create a spectrally-accurate scene by fusing multiple image modalities in a nearly-autonomous manner. However, early on this goal was broken down into a series of more easily achieved task-oriented objectives. These objectives comprised the minimum set of accomplishments necessary to adequately complete this research. As defined in the research proposal, these tasks were as follows:

1. **Develop or adapt a means to co- and geo-register image data from multiple sensing sources, including high-resolution framed-array imagery, multi-spectral framed-array imagery, hyperspectral line-scanned imagery, and line-scanned lidar point clouds.** The intent of this portion of the research was to provide an accurate method of performing the registration, and automation of this task was not required. Additional difficulties and issues that arose in relation to the specific sensors used in this research were also to be addressed. An example of this was the co-registering of bands from a hyperspectral sensor whose bands exhibit a relative spatial offset, such as the Rochester Institute of Technology's Modular Imaging Spectrometer Instrument (MISI). Although this requirement is discussed briefly in this work, a more complete development is included in [Casey et al. 2007] and [Casey 2008].

2. **Using the image data, extract a bare-earth Digital Terrain Model (DTM) for the desired area.** Although in many cases, an adequate terrain model may be obtained from the U.S. Geological Survey, one of the motivations behind this work was to recreate scenes for which ground truth is unavailable. As such, extraction of the DTM directly from the collected imagery was a requirement. Additionally, the methods chosen to extract this model needed

to be robust with regard to location and image resolution; that is, they should work with data from a variety of sensors operating in both rural and urban environments.

3. **Identify, isolate, characterize and synthetically construct trees.** Although a significant amount of work has been done with regard to identifying and analyzing tree canopy structure, there is very little information available on the fusion of multiple image modalities for individual tree reconstruction in densely-forested areas. This work sought to define a means to combine the height data inherent in a lidar point cloud with traditional optical imagery for improved tree analysis.

   For DIRSIG scene generation, the geometric modelling of trees typically has been done using commercial software such as Tree Professional [Onyx Computing Inc. 2006]. Although such software was used to a limited extent in this research, in most cases tree geometries were selected from a previously-defined library based on image-derived parameters.

4. **Identify, isolate, and reconstruct man-made structures (buildings).** Through a fused-imagery approach, it was anticipated that buildings would be more accurately defined than through traditional photogrammetric techniques. As a starting point in this analysis, techniques relying primarily on lidar data were considered first. Although several general methods for building reconstruction using this point data were pursued, only a variant of the method of intersecting planar faces is fully examined in this report.

5. **Derive a material map for the scene.** Several aspects of the scene creation process require access to a detailed material map. A material map is an indexed image where the pixel value at a each location identifies the material type present at that position. The relation between index values in the material map and associated material types is stored in a look-up table. Traditional hyperspectral classifiers may be used as a starting point in producing these maps, but in practice it has been found that these algorithms provide unsatisfactory results when used in isolation. This research considered additional processing techniques,

often using other imaging sources, to improve the material classification accuracy over the area of interest. However, in most implementations of the process, high-resolution multi-spectral imagery was used in lieu of hyperspectral imagery, as we achieved notably better registration results with the multi-spectral frame-array data.

6. **Derive a means to provide texture (reflective spectral region only) to the scene.** This is typically done using a texture image (or images) that is registered to the scene, and selecting spectral reflectance curves based upon brightness values in these images. Additional work was done in an effort to augment this method with techniques to apply texture when multiple spectral curves for a given material are not available, or when the texture mapping image is not of an adequate resolution to accurately describe the spatial variability of a given class. However, this additional work is still being investigated, and it is not discussed in this dissertation.

7. **Assign spectra to the objects in the scene.** The two approaches for this task are to use the sensor as a field spectrometer, or to use the image data to select spectra for distinct regions from a library of previously collected spectra. This research relies on the former technique in the baseline approach, but handles spectral selection from a library in the cases where hyperspectral imagery is not available.

8. **Specify the quality of the entire process.** One of the more difficult tasks in many technical efforts is to accurately describe how well the requirements were met. In the research effort at hand, this evaluation translates into deriving scene quality metrics that describe how well the resultant scene meets the needs for which it was created. Such metrics have proven difficult to construct when dealing with the geometric nature alone. This difficulty is even greater when dealing with a DIRSIG scene, not only due to the added spectral dimension, but also to the fact that in many instances, accurate scene reproduction (relative to the real world) may be less important than precise scenes that are less indicative of reality but are

carried out at centimeter resolutions. Although an all-encompassing scene metric was beyond the scope of this research, a few performance measures are presented, and the resultant scenes are evaluated according to these criteria.

9. **Apply techniques to real image data.** The techniques developed through this research have been applied to real data collected over RIT and the surrounding area, and the resulting scenes have been compared to a traditionally-produced scene, as well as to measured ground truth.

1.2 Scope

Although many of the algorithms and techniques developed for this thesis may either be applied directly or with slight modifications to a wide variety of image types, this research has only been verified using a limited set of image sources. When available, lidar data are assumed to be given in raw point form, and are only sampled to a raster grid when noted in the text. Multiple return data are assumed as well, as is the availability of point characteristics such as 3D position, return intensity, return number, and time of return. Average point density is typically assumed to be greater than one point per square meter, and in many cases densities on the order of 5-10 points per square meter are used. Points are also assumed to have already undergone whatever initial geometric processing was required to bring the individual scan lines into alignment. This is a reasonable assumption, since this task is usually accomplished by lidar-imaging vendors before data delivery.

Spectral imagery is limited to the 400-2500 nm range. Beyond this region, thermal emission becomes significant, a condition not dealt with in this research. When performing spectral matching to a library of spectral reflectance curves, the spectral imagery is assumed to be accurately calibrated, although a discussion is included as to how to handle the cases when it is not. However, coverage in the full range of 400-2500 nm is not required. This contrasts with using the hyperspectral sensor as a field spectrometer, where absolute calibration is not required if calibration panels are present in the scene, but spectral coverage should be sufficient to encompass the wavelengths required for the scene model.

It is also assumed that distortions of all frame-array imagery have been corrected, so that these imagers are essentially line-preserving central-projection cameras. Although methods to correct for such distortion are well known, these refinements were neither implemented nor used on the actual data. Specifics related to the sensors used to collect the imagery and the parameters describing the quality of the imagery are included as appropriate throughout the dissertation.

Most of the code related to this research was written in Matlab and is available on the CD included in the bound copies of this document. Code written by other researchers is documented in the code only, and no attempt is made to reference this material in the dissertation. Also, it should be emphasized that the code created in support of this research is strictly of the ”proof-of-concept” variety. Although an attempt was certainly made to ensure the programs execute quickly and efficiently, the code was not optimized in any real sense.

1.3 Achievements and Contributions to Knowledge

The research proposed in this document contributes to the field of synthetic scene reconstruction in several ways. Although this field has realized significant gains in recent years, most of the research has been devoted to defining the geometry of a given scene. To date there has been very little work done regarding the semi-autonomous construction of spatially and spectrally-accurate scenes. In a conceptual sense, incorporating this spectral information represents a direct contribution to the field, as such scenes are inherently more descriptive than basic geometric models.

This may similarly be viewed as an advancement to the simulated imagery work being performed at RIT. By semi-automating the scene generation process, large scale scenes will be able to be produced quickly and accurately. Additionally, through the methods proposed here, scenes may be created even when little ground truth is available.

From a more technical standpoint, this work contributes to several areas. Most significant is the novel fusion of lidar data with more traditional imaging sources. This fusion not only includes straightforward techniques such as draping imagery over a lidar-produced CAD model to provide material and texture information, but also contains more integrated techniques such as using the lidar data as a weighted constraint during a photogrammetric bundle adjustment. Additional contributions include a novel method for registering various image types (including a technique to ortho-rectify MISI data), a new technique for performing atmospheric compensation, a modified method for extracting the terrain from lidar data, a refined approach to building and tree extraction and reconstruction, and an improved method of assigning spectral characteristics to vertical surfaces.

1.4 Dissertation Overview and Organization

The remainder of this dissertation is organized as follows. An overview of the various technologies and tools relevant to this work is given in Chapter 2. Chapter 3 builds on this background and introduces several state-of-the-art image processing techniques that the reader may be unfamiliar with. It is here that the technical foundation for the approach will be laid. This includes brief introductions to iterative optimization and robust parameter extraction, as well as more detailed descriptions of modern photogrammetry and computer vision. Chapter 4 develops the baseline approach for this research, and describes the process flow assuming all required image types are available. This chapter also includes a brief overview of current techniques and their limitations for each of the required major tasks. When all data sources are not available, modifications to the baseline approach are often required, and these changes are also detailed in Chapter 4. Chapter 5 presents the results of applying the approach to real data and also provides a description of the quality of the approach. A summary of the research is included in Chapter 6, and ideas regarding future work and potential extensions of this work are given in Chapter 7. The included appendices serve as a tutorial/review for many of the required mathematical processes underlying the proposed techniques, as well as a reference for the basic input files required to define a DIRSIG scene.

"A firm foundation is necessary for any building, institution, or individual to endure."

Russell M. Nelson

2 Background

This chapter will provide an overview of several of the technologies and research fields pertinent to the methods used throughout this research. Although not comprehensive by any means, it should serve as an initial foundation for understanding the rest of this dissertation. Where additional background may be warranted, outside references are given; however, it is anticipated that most readers with a basic technical foundation will be able to proceed without too much difficulty. Specific algorithms and recent advances in these fields as they pertain to 3D object reconstruction will not be included here. Rather, the recent work by other authors in relation to the specific objectives addressed by this report will be presented in the relevant sections of Chapter 4.

2.1 Traditional Remote Sensing and Optical Imaging

The field of remote sensing is primarily concerned with extracting useful information about objects without actually coming into physical contact with them. In practice, this has most frequently referred to analysis of the radiometric content of the imaged scenes, although in a strict sense, photogrammetry and photo interpretation may also be considered a part of the remote sensing field. For years, the primary method of data collection was through the use of airborne film cameras. In recent years however, advances in digital imaging technologies have resulted in the rapid advancement of digital electro-optical (EO) imagers. Rather than recording the image data on silver-halide crystals, digital imagers use an array of solid state sensors such as charge-coupled devices (CCDs) and complementary metal-oxide semiconductor (CMOS) detectors to record the intensity of the incident electro-magnetic radiation. As such, greyscale digital images are represented by a 2D array of picture elements (pixels), with each pixel having a quantized brightness value (grey-level) associated with it. Color and other multi-band digital images may simply be thought of as stacked layers of 2D arrays, where each layer represents the brightness values within a certain spectral band.

EO systems are usually classified according to the general architecture of their image formation. Due to its similarity with traditional film systems, in many ways the digital framing array is the simplest to envision. This device consists of a 2D array of detectors mounted in the focal plane of a camera. To acquire the image, all of the detectors are exposed simultaneously, thereby producing the same type of perspective image that is generated by a film camera. Figure 2.1 shows a schematic illustrating the basic geometry of this design.

Another common architecture for digital imagers is the pushbroom design, which makes use of a linear array of detectors that collects the image as the camera sweeps over the terrain. Since the 2D image is collected as the platform moves forward, the image is acquired in temporal increments. As such, the perspective of these images is similar to the standard perspective image only in the cross-track direction. In the direction of flight, the viewing angle to each line will be

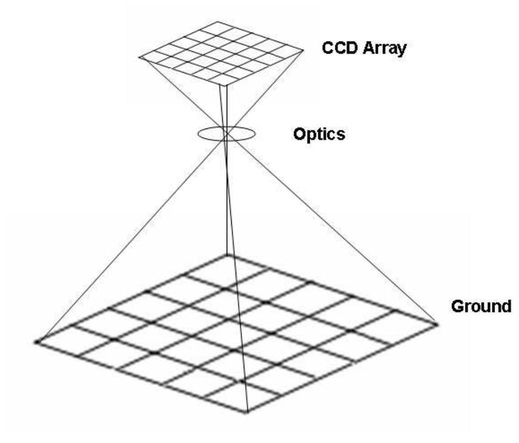


Figure 2.1: Digital framing camera geometry.

constant, and there will be no relief displacement in this dimension. This geometry is termed *line perspective*, and the schematic for such an imager is given in Figure 2.2.

A third method for obtaining digital images is with the airborne line scanner. This system records one pixel at a time, and uses a rotating mirror to scan lines on the ground. The sensor platform advances during the collection of each line, and if the timing is correct, sequential rotations will sweep out adjacent lines on the ground. A schematic of this imaging system is given below in Figure 2.3. Although these systems are comparatively simple in design, they often exhibit

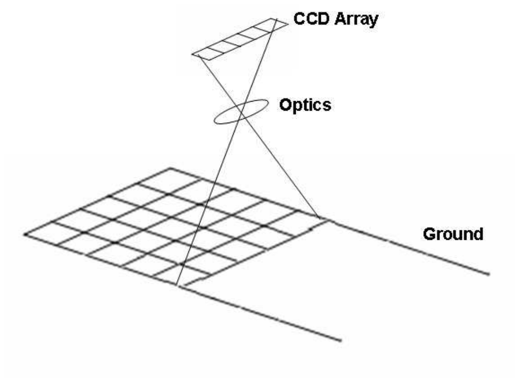


Figure 2.2: Digital pushbroom imager geometry.

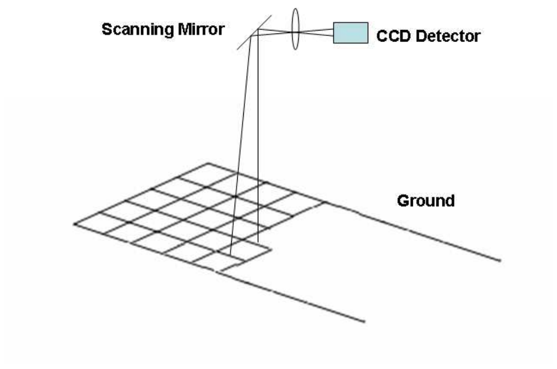


Figure 2.3: Digital line-scanner geometry.

significant geometric distortions which must be corrected before use in many applications.

While many imaging sensors integrate the received radiance across a single spectral band to produce a greyscale image, it is possible to modify the above designs to record multiple spectral bands. This is commonly done to produce standard red, green, blue (RGB) color imagery, but the number of recorded bands need not be limited to three. In general, when several (usually defined as three to approximately ten) bands of data are collected, the system is said to be multi-spectral. Multi-spectral images often offer tremendous advantages over conventional greyscale images when trying to extract scene information. In many cases, objects may be identified or classified merely by processing the relative received radiance in various spectral bands.

Over the past few years, RIT has developed a sophisticated multi-spectral imager in support of the Wildfire Airborne Sensor Program (WASP) [McKeown 2003]. The WASP sensor combines a high-resolution visible RGB-sensing camera with three additional IR cameras operating in the short-wave, mid-wave, and long-wave (SWIR, MWIR, and LWIR, respectively) portions of the IR spectrum. The color camera is an off-the-shelf Pixel Physics mapping design, and it uses a 4000×4000 pixel frame array to record its images. The three IR sensors are from the Indigo Phoenix line, and each uses a 640×512 pixel array. These radiometric sensors are combined with an Applanix inertial measurement unit and an on-board computer processor. When flying at an altitude of 3000 meters, the RGB camera has a ground spatial resolution of 0.5 meters, while the

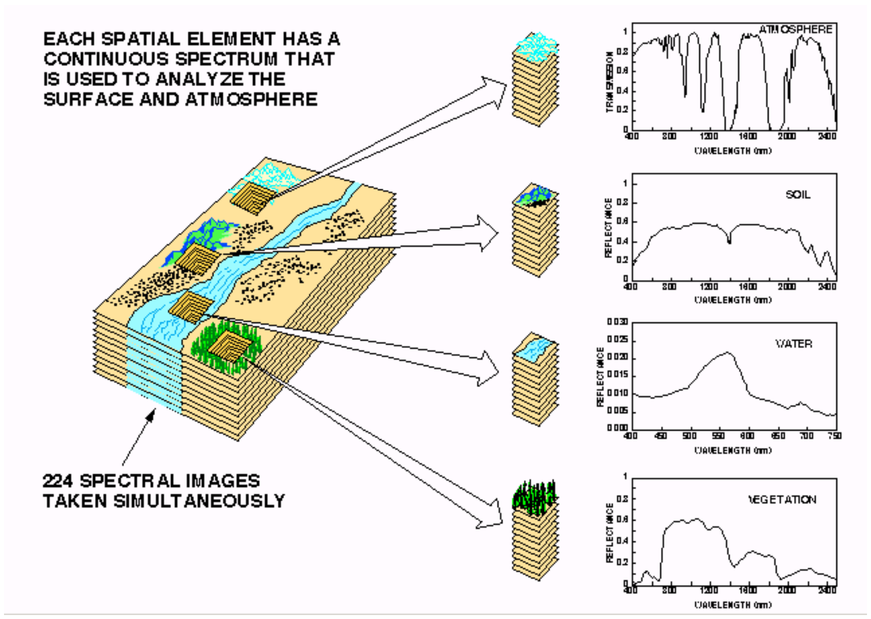

EACH SPATIAL ELEMENT HAS A CONTINUOUS SPECTRUM THAT IS USED TO ANALYZE THE SURFACE AND ATMOSPHERE
224 SPECTRAL IMAGES TAKEN SIMULTANEOUSLY
ATMOSPHERE
SOIL
WATER
VEGETATION
TRANSMISSION
REFLECTANCE
WAVELENGTH (nm)

Figure 2.4: Concept of a hyperspectral image cube [NASA JPL 2006].

IR sensors cover approximately 3 meters.

As the number of discrete spectral bands increases, we transition from multi-spectral to hyperspectral imaging. Hyperspectral sensors, or imaging spectrometers, record the image data at tens to hundreds of narrow, adjacent spectral bands which essentially yields a continuous radiance spectrum at each sample location. By so doing, a wealth of additional information is provided, as subtle spectral features combined with the general spectral shape may be exploited in tasks such as image segmentation and classification, target detection, and material identification. Additionally, in many cases, these analyses may occur at sub-pixel resolutions. In order to describe the image content of a hyperspectral dataset, the term image cube is often used. With this terminology, each spectral band comprises an individual layer of the cube, and if we examine the grey-level values for each layer of a given spatial (pixel) location, the spectral curve for that location is obtained. Figure 2.4 [NASA JPL 2006] displays this concept in relation to the AVIRIS sensor.

The design approaches for multi-spectral and hyperspectral imagers are basically extensions of the traditional designs that were explored above [Schott 2007]. For framing arrays, multiple bands may be recorded by either placing spectral filters in front of individual pixels, or by using a tunable filter and collecting each band's image sequentially. The pushbroom sensor may be modified by introducing a prism or grating device to spatially separate the spectral components of each line, and recording the subsequent spectral-spatial line data on a 2D sensor array. In a similar manner, a linear array may be used with a line-scanning architecture to capture multiple band data.

Of particular interest to this research effort is the array-based COMPact Airborne Spectral Sensor (COMPASS), a hyperspectral imager that captures incident radiation from 400nm to 2350nm on a single focal plane [Simi et al. 2004], [Zadnik et al. 2004]. This sensor was developed at the Army Night Vision and Electronic Sensors Directorate (NVESD), and was used in this research to provide moderately-well calibrated data for the description of target reflectance spectra. The average bandwidth of this sensor is approximately 8nm over the instrument's spectral range, and the spatial resolution was on the order of 1m.

This research also makes use of data from a student-designed hyperspectral line scanner. The Modular Imaging Spectrometer Instrument (MISI) is an airborne line scanning system with modular focal planes that has been under development at RIT for the past 11 years. Originally fitted with a VNIR spectrometer and a LWIR focal plane, it has recently been upgraded to contain a set of SWIR, MWIR and LWIR sensors to support a fire phenomenology program [Raqueno et al. 2005]. The VNIR spectrometer uses a 6-inch rotating mirror in combination with an f/3.3 Cassegranian telescope, and its focal plane is sampled by two optical fiber cables. The fibers lead to two separate 36-channel spectrometers which cover the electro-magnetic spectrum from 0.440μm to 1.020μm in 0.010μm spectral bands. Some of the incident radiation is redirected from the primary focal plane to feed five HgCdTe detectors for LWIR imaging, while secondary focal planes are used for imaging in the SWIR and MWIR. A simple schematic of the MISI design is presented in Figure 2.5 [DIRS Laboratory 2006].

In addition to the geometric distortions inherent to all line scanning designs, the spatial offset

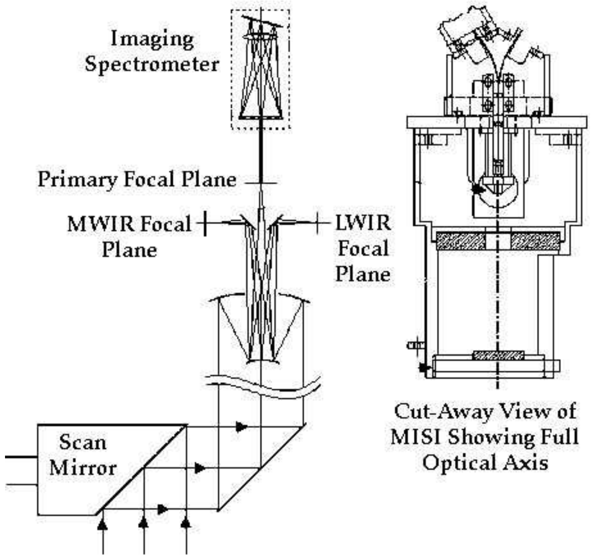


Figure 2.5: MISI optical train.

between the two VNIR fibers in MISI causes an inherent spatial shift between the two sets of spectral bands. This configuration causes a relatively complicated misregistration which cannot be corrected with a simple shift, rotation or translation of the data. Coupled with aircraft roll, pitch, and yaw variations, and elevation changes in the ground surface, these effects make geo-rectification of the imagery quite challenging. The resolution of this issue as related to a set of bands from a single fiber is discussed later in Section 4.3. The mismatch between spectral bands carried on different fibers is discussed in [Casey et al. 2007] and [Casey 2008]. These sources highlight that fact that this phenomenon severely degrades the spectral purity of each pixel for the MISI sensor, an issue that can particularly reduce performance in sub-pixel target detection applications.

2.2 Photogrammetry

While analysis of the radiometric content of a scene falls under the scope of remote sensing, when the focus shifts to geometric measurement of scenes through analysis of imagery, we have entered the domain of photogrammetry. Although in general this field is defined as "the art, science and technology of obtaining reliable information about physical objects and the environment through processes of recording radiant electromagnetic energy" [Wolf and Dewitt 2000], for the purposes of this report, we will limit it to the metric sense, which is primarily concerned with determination of the size, shape, orientation and location of objects in 3-dimensional (3D) space.

In order to use one or more images to extract the three-dimensional structure of objects, we need to select a mathematical model that approximates the geometry of the imaging process. Such a model will enable us to infer the nature of images given camera orientation parameters and the geometry of a scene, as well as to do the 3D scene reconstruction given the images. In order to ensure generality, we will assume that all images may be taken at oblique angles, and that the camera system may have various distortions. However, when these distortions are accounted for, a simple pinhole camera projection model may be assumed. We will also differentiate between the camera coordinate system whose origin is at the projection center (typically the center of the lens), the continuous image coordinate system (termed CIC coordinates here), whose origin is at the center of the image, the pixel-based image coordinate system (relative to a pixel location $(0, 0)$ for digital images, termed PIC here), and the world (object) coordinate system. Unless otherwise specified, all coordinate frames will be right-handed Euclidean systems.

It is common practice to separate the geometry related to the position and orientation of the camera in space from the internal characteristics of the camera itself. Specification of the former requires six parameters (three coordinates of the projection center plus three orientation angles) and is termed exterior orientation (EOP[1]). The interior orientation (IOP) refers to the intrinsic camera parameters which are needed to specify the direction of the projection ray to an object point, given an image point and the exterior orientation [McGlone 2004]. In simple pinhole (projective center) cameras, this may be limited to a description of the focal length; however, in most cases it also

[1]In many references, the more common acronym "EO" is used to specify the exterior orientation. However, in this dissertation, EO is used to mean electro-optical. Therefore, the exterior orientation will be termed EOP, which more precisely refers to the "exterior orientation parameters." In order to maintain a parallel structure, interior orientation will use the acronym "IOP"

contains parameters for principal point (the point in the image normal to the projection center) offset and optical distortion. Interior orientation has traditionally been determined by calibration, although recent techniques have been developed to derive at least some of these parameters from image content. This contrasts with exterior orientation, which is usually determined from the imagery and in-scene control points, although recent advances in aircraft navigation technology also often permit direct measurement of the parameters.

One the EOP and IOP have been recovered, image points representing a common location (termed homologous or corresponding points) may be back-projected through the optical centers of the corresponding cameras (as defined by the EOPs) and intersected in three dimensional space. The location of this intersection point in world coordinates is the location of the object point corresponding to the chosen image point. When these back-projected rays do not intersect due to errors in the process, the point minimizing the orthogonal distance to each ray is often used to define the object point.

The theory underlying these problems is presented in Section 3.1 and covers approaches using both Euclidean and projective geometry.

2.3 Laser-Based Remote Sensing Systems

Lidar (*Light Detection and Ranging* or *Laser Imaging Detection and Ranging*) is an active remote sensing technology that measures the backscattered radiation of a transmitted laser signal to determine properties or position of a distant object. With the continuing advances in laser design, these systems have realized a tremendous development in recent years, and many sectors of the remote sensing field have found uses for this technology. As such, lidar is currently being used in a wide variety of applications, including water depth measurement (bathymetry), ocean research, and topographic mapping of the earth's surface. Lidar also plays a large role in atmospheric studies, as systems that transmit multiple wavelengths are able to determine atmospheric absorption properties and the size distribution of atmospheric particles. Lidar is also used extensively in military

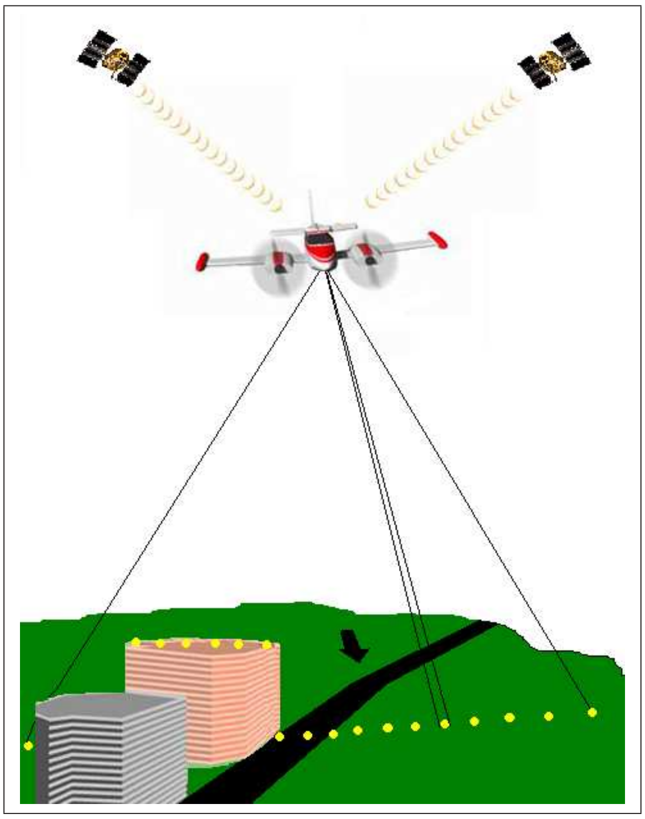


Figure 2.6: A simple cartoon illustrating the basic concepts of lidar data collection (aircraft and building clip art taken from [SIMoN 2006]).

systems to perform such tasks as vehicle tracking, and biological and chemical agent detection, and automatic target recognition.

Topographic lidar systems typically are based on a laser ranging unit coupled with an opto-mechanical line scanner, although other configurations (such as pushbroom or framing array designs) have also been implemented successfully. When used in conjunction with a tightly-coupled GPS/Inertial navigation system, these sensors are able to obtain measurements of the 3D coordinates of a very large number of tightly spaced points in a short amount of time. For line-scanning pulsed-laser systems, as the aircraft flies forward, the scanning mirror continually shifts the viewing angle of the laser ranger. At discrete time intervals, laser pulses are transmitted, reflected off objects, and subsequently re-received by the sensor. By processing the travel time of each pulse in

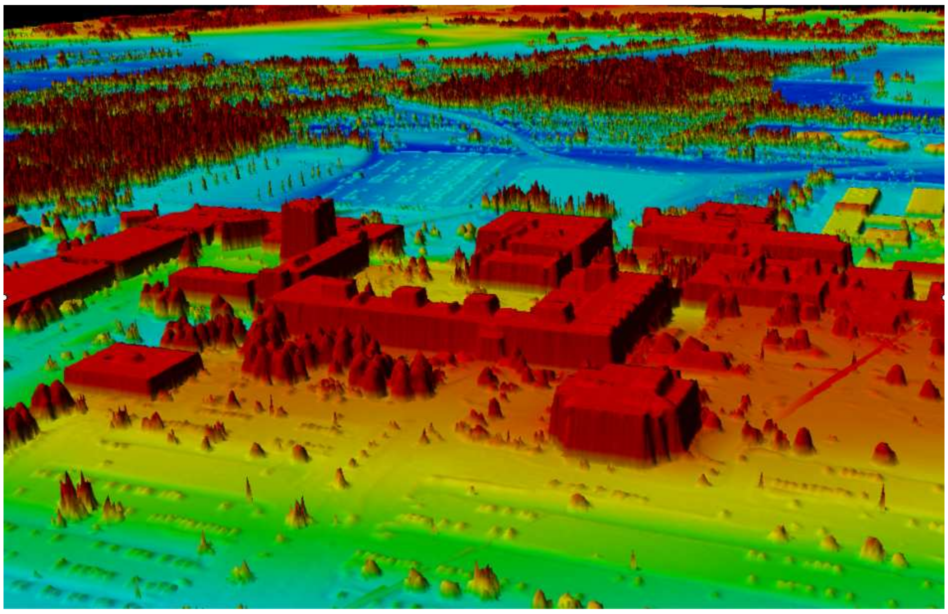


Figure 2.7: Perspective view of 3D lidar data using elevation-based coloring.

conjunction with the pointing angles of transmission, the precise location of the reflecting object may be determined relative to the sensor orientation. When the sensor's location and pointing angles are known in absolute world coordinates, the location of the point of reflection may be expressed in that domain as well. Figure 2.6 shows a cartoon of a generic line-scanning lidar system in operation, and Figure 2.7 shows a resultant 3D mapping for a portion of the RIT campus. The coloring of this image is based on an indexed scheme, where elevations (heights above the coordinate system's (x, y) plane) are depicted as variations in color.

For many topographic lidar sensors, point locations are not the only information extracted by the system. In addition to the range information provided by the delay of the received signal, in many cases the intensity of the received signal is recorded as well. This data yields information related to the reflectance properties (in the spectral region used by the source) of the illuminated object, and may be used in post-processing to aid in point classification and feature extraction. Many lidar sensors are also able to discriminate multiple returns from each transmitted pulse, based on the time delay of the return signal. This often enables the sensor to discriminate points lying at the

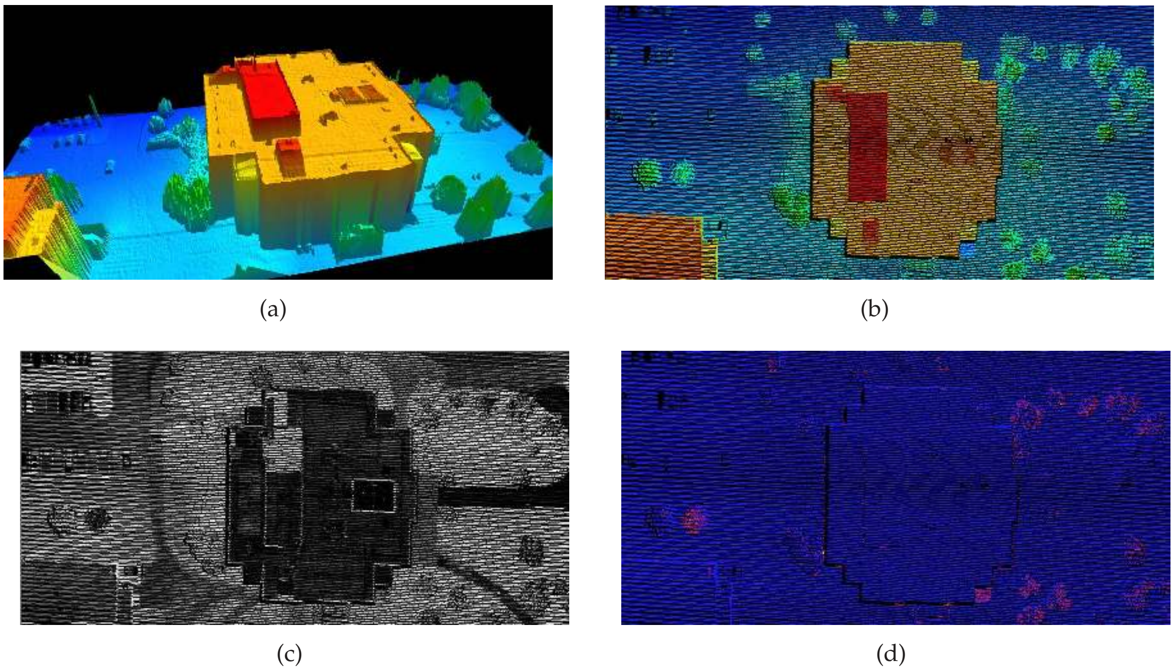


Figure 2.8: Lidar images: (a) Height-based image, perspective view, (b) Height-based image, nadir view, (c) Intensity-based image, (d) Return number-based image.

top of a tree and on the ground plane, and has been used successfully to identify forested regions without relying on hyperspectral imagery or spatial image analysis (see Section 4.2, for example). Figure 2.8 depicts the elevation map in both perspective and nadir orientations, as well as the corresponding intensity and multiple return images for the regions surrounding RIT's Carlson Center for Imaging Science.

For the purposes of this dissertation, much of the theory underlying the operation of the lidar scanner will be ignored, and the system will be treated as a 'black box' capable of producing range and intensity information as noted above. For additional information regarding the operation of these systems, the reader is referred to [Argall and Sica 2003].

It should be noted that in some fields, the term ladar (Laser Detection and Ranging) is preferred when discussing this type of laser ranging technology. A distinction between these acronyms was made in [Burton 2002], where the author specifies that ladar is typically used to denote systems which locate the position of objects, while lidar is used to describe sensors that derive properties of an object. However, in the field of 3D object reconstruction, the term lidar is used almost exclusively. As such, lidar will be the acronym of choice for the remainder of this dissertation.

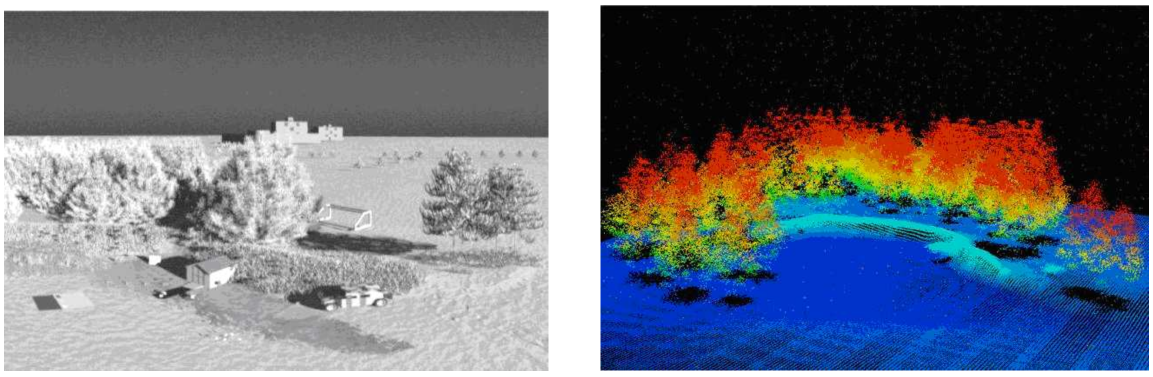
2.4 The DIRSIG Model

The DIRSIG model is a complex synthetic image generation utility that was developed at the Digital Imaging and Remote Sensing (DIRS) Laboratory at the Rochester Institute of Technology (RIT) over the last 20 years. The tool was originally designed to model the thermal infrared region of the electromagnetic spectrum, but was expanded several years ago to cover the full visible to long-wave infrared (0.4 to 20 micron) range. It effectively models broadband, multispectral and hyperspectral imagery using a suite of first-principles based radiation propagation modules. These modules perform specific tasks such as predicting bi-directional reflectance functions (BRDF), computing time and material dependent surface temperature values, and computing the dynamic viewing geometries of scanning instruments on a moving platform. In addition to these DIRSIG-specific modules, the tool leverages several utilities (such as MODTRAN and FASCODE) that have been used extensively by the remote sensing community.

In 2002, a first-principles-based lidar model was incorporated into the passive radiometry framework enabling the model to calculate arbitrary, time-gated photon counts at the sensor for atmospheric, topographic, and volumetric backscattered returns. The DIRSIG lidar model was first developed in [Burton 2002], and it was recently expanded by [Blevins 2005] to handle a wide variety of complicated scene geometries, diverse surface and participating media optical characteristics, and a variety of sensor models. The lidar module now includes the effect of multiple scattering, multiple bounces from topographical targets, and absorption within non-homogeneous finite volumes. Additionally, several noise sources are extensively modeled, such as speckle from rough surfaces and atmospheric turbulence phase effects. As such, DIRSIG is now able to help researchers evaluate design trades for topographic systems and the impact that scattering constituents (e.g. water vapor, dust, sediment, soot, etc.) may have on a Differential Absorption lidar (DIAL) system's ability to detect and quantify constituents of interest within volumes including water and atmospheric plumes [Blevins 2005].

Although the passive and active models used by DIRSIG are valuable in their own right, in many applications the ability to integrate passive EO and lidar simulations using a common scene (and potentially common viewing geometries) is of even greater utility. The multi-modal capabilities of the DIRSIG model allow designers to evaluate both passive and active approaches to solving specific imaging problems, as well as potential fusion techniques using both image modalities. Additionally, the upper performance limit for a given approach may be more easily determined when using a common baseline scene.

Two representative synthetic images produced by the DIRSIG model are included in Figure 2.9 (Courtesy of [Blevins 2005]). The first depicts a near-infrared (passive EO) image of MicroScene1 (a small test range on the RIT campus), and the second is a topographic lidar product of a portion of the same area.



(a) (b)

Figure 2.9: DIRSIG Images (a) Near-infrared simulation of Microscene1 and (b) Simulated topographic lidar image [Blevins 2005].

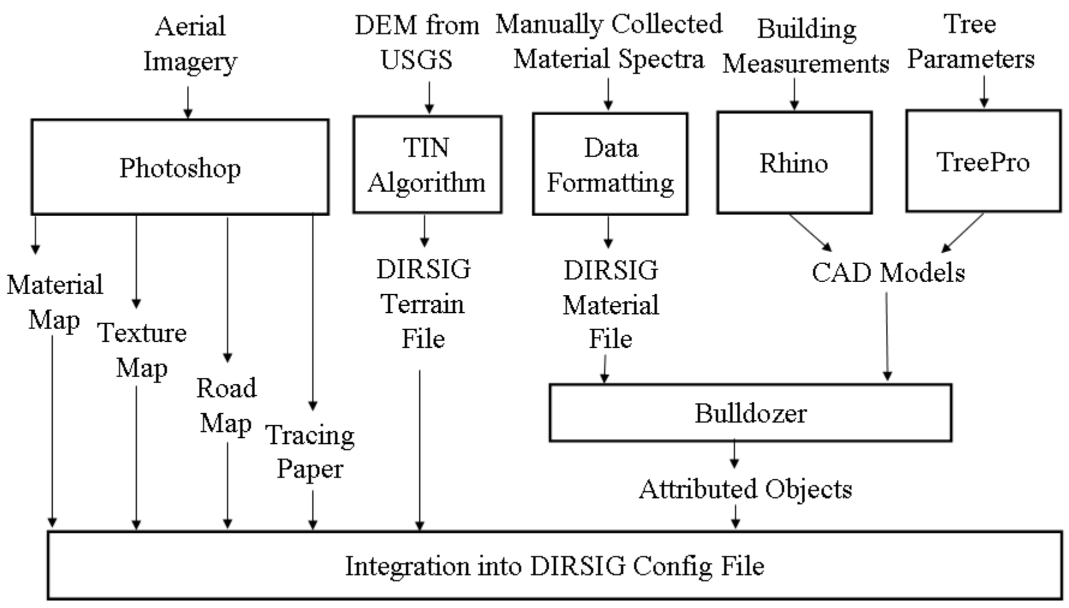


Figure 2.10: Standard process: Block diagram.

2.5 Current Process for DIRSIG Scene Creation

The current method of creating a scene for use in DIRSIG simulations is straightforward, but unfortunately it is also quite labor and time intensive. Figure 2.10 illustrates the process via a simple block diagram. The detailed steps in this figure (and as described below) have been taken from [Brown 2005], [Brown 2006] and [Ientilucci et al. 2003], and represent the techniques used at the Rochester Institute of Technology. Although the specific methods employed in creating scenes may be somewhat different at other organizations, the fundamental principles and basic steps remain the same.

The first step in the generation of a new scene has typically been to acquire a digital terrain model (DTM) of the area to be modeled. For scenes based on real-world locations in the United States, we have used USGS Digital Elevation Models (with a 10 meter ground sample distance) with success, but other sources of these data are also available. For fictional scenes, the terrain model may be designed as desired. Once the terrain geometry has been determined, it is usually up-sampled and registered with any relevant aerial imagery, which facilitates several of the other steps required in the scene specification. Additionally, the terrain model needs to be facetized, which is often done using a triangular irregular network (TIN) generation program for terrains with a near-constant slope. It should be noted that in many cases, the terrain must be smoothed via some form of spatial averaging before the facetized version may be created.

Once the geometric properties of the terrain have been determined, the 3D objects that will appear in the scene must be created. These objects include buildings, trees, vehicles, and any other structures that are not a part of the terrain model. This is often a very time-intensive process, especially if many unique object types are desired. The geometric description of these objects can be generated using a variety of computer aided design (CAD) tools, as long as the specific utility is able to output the final facetized structure as a Alias/Wavefront .obj file. At RIT, most of the geometric models have been manually drawn with the Rhinoceros [Robert McNeel and Associates 2005] CAD package, which has proven to be well suited to the task. The one exception is the modeling of tree structures, for which we have chosen to use Tree Professional [Onyx Computing Inc. 2006]. This software allows the user to render a host of trees, palms, and other plants of various species and sizes and output the result as a facetized model.

Next, the spectral reflectance and emissivity data must be acquired. This is typically accomplished either through a direct collection (using an instrument such as a field spectrometer) or by accessing a spectral data library. This data is then included in a materials file that will be used when accessing the scene.

The geometric objects are then attributed with material characteristics using DIRSIG's Bulldozer tool. This utility allows the user to import the Alias/Wavefront .obj files created earlier and assign material properties to each facet. This task is also quite time consuming, as each portion of an object must be attributed manually.

Road, material, texture, and other background maps (scene layers) are created next [Brown and Schott 2000]. Road maps have traditionally been created in Adobe Photoshop, due to artifacts left behind when segmenting real imagery. These maps are created by opening the scene file, and adding a new layer to the image. The roads are then drawn on the new layer with the paintbrush tool, using the scene image as a guide. The scene layer is then removed, and a black background is added to the road map layer. Materials may be attributed either via a material map or through a direct assignment to individual facets. Material maps usually take the form of traditional classification maps, and are often generated through traditional segmentation algorithms on real remotely sensed imagery. Direct facet assignment of materials is achieved manually through a modification of an object's CAD description.

Texture maps may also be created to drive in-material spatial-spectral variations. Such variations are important in mimicking the real-world phenomenology; without them all pixels of a given material in the scene would have the same spectral characteristics, thereby appearing more uniform than desired. Texture maps are usually just aerial images of the scene to be modeled. In some cases where an image of the scene was not available, a simple noise image was used instead. Use of the texture images is quite simple. If the grass material type has 300 different spectra in its library, the particular spectral curve for each location in the final scene is selected based on a look-up table relating pixel values of the texture image to spectral signatures. This look-up table is created using the statistics of both the texture image and the spectral library.

Finally, the Bulldozer utility is used to place the attributed objects onto the facetized terrain model. If aerial imagery is available, a "tracing paper" layer may be employed to help with object orientation. Once the objects are placed in the scene as desired, a DIRSIG configuration file is created to link together all of the previous files. At this point, the scene has been constructed.

Once the scene has been defined, it is a simple matter to create physically-accurate synthetic images. First, for a standard EO imager, a sensor is defined in terms of its spatial and spectral properties, and an exterior orientation (position and rotation) is specified. A similar set of parameters are defined if a lidar imager is to be used instead. Scene variables such as local time (which dictates the relative solar positioning) and atmospheric conditions are also specified. DIRSIG is then run to invoke the spectral ray-tracing (or photon-mapping, in the case of lidar sensors), and the resultant synthetic imagery is produced. Various truth maps (related to such things as material type) may also be produced, which are frequently used as the baseline when algorithms are applied to the synthetic imagery.

Details regarding the required DIRSIG files and their proper formats are discussed later in Appendix B. In that section, an overview of the DIRSIG file structure is presented, and several example files are included. Additional information regarding the use of DIRSIG and proper formatting of the required input files may be found in [Brown 2005].

As has been demonstrated in this chapter, the current process for creating spectrally-accurate scenes is effective but laborious and time-intensive. With the recent availability of improved lidar, spectral and high-resolution optical sensors, we aim to reduce the effort involved in generating such scenes through a fusion of these image modalities. Such an approach will enable automation of many of the tasks required and will also permit increased accuracy in many instances. The methods proposed for performing this fusion will be discussed in the following chapters.

He who loves practice without theory is like the sailor who boards ship without a rudder and compass and never knows where he may cast."

Leonardo da Vinci

3 Theory

This chapter will provide an overview of several of the technologies and research fields pertinent to the methods used throughout this research. Although not comprehensive by any means, it should serve as an initial foundation for understanding the rest of this dissertation. Where additional background may be warranted, outside references are given; however, it is anticipated that most readers with a basic technical foundation will be able to proceed without too much difficulty. Specific algorithms and recent advances in these fields as they pertain to 3D object reconstruction will not be included here. Rather, the recent work by other authors in relation to the specific objectives addressed by this report will be presented in the relevant sections of Chapter 4.

3.1 Photogrammetry and Computer Vision

In recent years, researchers in the field of computer vision have been attempting to solve many of the problems that have traditionally fallen under the auspices of photogrammetry. Computer vision (CV) is a discipline concerned with discerning properties of the 3D world from one or more 2D digital images. However, unlike photogrammetry, whose interest is primarily in achieving precise measurements in absolute coordinates, CV seeks a less tangible *understanding* of the scene, which may or may not require high accuracy or even a solution in an absolute sense. This difference in requirements arises from differences in application areas; while the primary products of photogrammetric analysis are related to topographic maps, CV tends to apply its methodologies to object recognition, autonomous vehicle navigation, and object modeling [Hartley and Mundy 1993]. Still, it takes little exposure to both fields to see that the similarities outweigh the differences, and that both fields rely heavily on the theory of using a pinhole-type camera to understand a 3D world from 2D projections. As such, both address methods to determine camera calibration and pose, as well as 3D model understanding (and often model construction in absolute world coordinates). In this section, we will present a few of the key concepts from traditional photogrammetry. However, since these techniques are a part of the curriculum for many remote sensing programs, the review will be somewhat brief. More detail will be spent on the newer CV approaches that make use of projective geometry and homogeneous coordinates, as many researchers will be unfamiliar with these techniques.

3.1.1 Traditional (Euclidean-based) Photogrammetry

As noted in Chapter 2, if the interior orientation of the sensing system is known, we may use an ideal (distortionless) pinhole camera model to fully describe the constrained camera geometry. In this case, all rays coming from the object space intersect at the projection center and proceed to the image plane without any change in direction. As such, it is easy to see that this requires any object point, the point's projection in the image, and the exposure station to lie along a straight

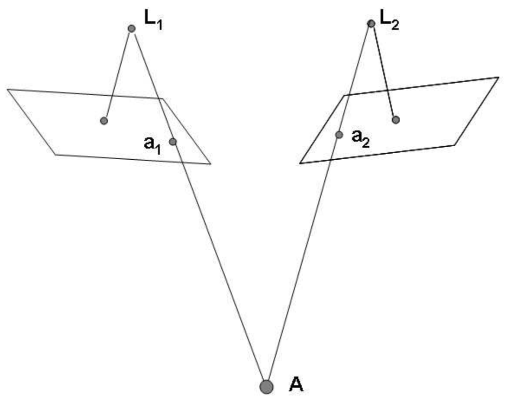


Figure 3.1: Illustration of the geometrical principles of collinearity and coplanarity.

line. This requirement is termed the *collinearity condition*, and is readily seen in Figure 3.1. In the left image (image 1), object point **A**, its image $\mathbf{a}_1$, and the camera location $\mathbf{L}_1$ are all co-linear. The mathematical description of this constraint may be written as

$$\begin{aligned} x &= x_0 - f\frac{r_{11}(X - X_L) + r_{12}(Y - Y_L) + r_{13}(Z - Z_L)}{r_{31}(X - X_L) + r_{32}(Y - Y_L) + r_{33}(Z - Z_L)} \\ y &= y_0 - f\frac{r_{21}(X - X_L) + r_{22}(Y - Y_L) + r_{23}(Z - Z_L)}{r_{31}(X - X_L) + r_{32}(Y - Y_L) + r_{33}(Z - Z_L)}, \end{aligned} \tag{3.1}$$

where x and y are the photo coordinates of the image point, f is the camera's focal length, the r's describe the angular orientation of the camera (see Section A.2), (X, Y, Z) are the object coordinates, and (X_L, Y_L, Z_L) are the object space coordinates of the exposure station. The terms x_0 and y_0 represent the location of the image principal point in image space coordinates. It is no surprise that Equation 3.1 bears a strong resemblance to the equation representing a projective transform, since imaging via central projection is inherently a projective process.

An extension of these concepts will also demonstrate that for two images, object point **A**, its images ($\mathbf{a}_1$ and $\mathbf{a}_2$), and the exposure stations ($\mathbf{L}_1$ and $\mathbf{L}_2$) all lie in a common plane. An illustration of this may also be seen in Figure 3.1, and this condition is termed *coplanarity*. The mathematical model describing this geometry is given by

$$0 = B_X(D_1F_2 - D_2F_1) + B_Y(E_2F_1 - E_1F_2) + B_Z(E_1D_2 - E_2D_1) \tag{3.2}$$

where

$$\begin{aligned}
B_X &= X_{L_2} - X_{L_1} \\
B_Y &= Y_{L_2} - Y_{L_1} \\
B_Z &= Z_{L_2} - Z_{L_1} \\
D &= (r_{12})x + (r_{22})y + (r_{32})f \\
E &= (r_{11})x + (r_{21})y + (r_{31})f \\
F &= (r_{13})x + (r_{23})y + (r_{33})f.
\end{aligned} \tag{3.3}$$

It is noteworthy that in the above notation, the object coordinates are not required, and the equation is completely specified by the relative exterior orientations of the two cameras.

Through the process of *resectioning*, we may apply the collinearity equation to three (or more) control points and solve for the six exterior orientation parameters of each image, provided the interior orientation has been previously measured. It should be noted that although it is possible to use the coplanarity constraint to arrive at this same solution, this is not commonly done [Wolf and Dewitt 2000]. Once the absolute orientations of the cameras are known, we may then reapply the collinearity equations to determine the 3D locations of objects given their image coordinates in each photo. This process is termed *spatial intersection*, as the object point is simply the intersection of the rays $\overrightarrow{L_1a_1}$ and $\overrightarrow{L_2a_2}$. When these rays do not intersect, the point where the rays are closest (in an orthogonal sense) is usually chosen as the solution. A brief tutorial on this technique as applied to multiple images is given in [Slabaugh et al. 2001].

3.1.2 Computer Vision

Unlike traditional photogrammetrists, who typically approach these problems using the tools of analytical Euclidean geometry, CV researchers frequently opt to formulate their models using algebraic projective geometry. As noted in [McGlone 2004], projective geometry is able to overcome the limitations of Euclidean geometry by inherently including points, lines and planes at infinity, which unifies the theory to include what has traditionally been a host of special cases. Additionally, the use of homogeneous coordinates makes most of the pertinent relationships linear. Such an approach has enabled additional insight into many aspects of the theory underlying both sciences, and has directly contributed to many gains in both fields.

In lieu of using control points to obtain an absolute solution in the world coordinate system, a relative solution of the three-dimensional object locations may also be achieved that describes object locations up to an arbitrary projective transform. This is termed the *projective photogrammetric model*, and it holds for the general case of straight-line preserving cameras. If the complete interior orientation of the cameras is also known, this photogrammetric model may be computed in such a way that it is unique up to a similarity transform. In cases where Euclidean 3D coordinates are ultimately required, these more general solutions may be refined by introducing object-image point constraints. However, in many cases it is possible to extract the desired information directly from the projective or similarity photogrammetric models without ever fixing the object coordinates in an absolute sense.

It should be noted that in practice, more mature techniques are almost always used for 3D reconstruction. In several approaches, the interior orientation parameters may be inserted into the collinearity equation and solved for simultaneously with the other unknown terms. Also, it is sometimes possible to use inferred geometric information about the scene (such as specifying parallel and perpendicular lines in the scene) to enable a full Euclidean reconstruction from a single image. Measurement of shadows with a known solar angle may also be used for this purpose [Shufelt 1999].

3.1.3 Photogrammetry Via Projective Geometry

This section will give a very brief introduction to the fundamental concepts of projective geometry as used in modern computer vision and photogrammetric applications. A thorough discussion of these concepts is well beyond the scope of this dissertation; the detail included here will be just enough to describe the methods presented later in Chapter 4. For a more comprehensive review of this topic, the reader should consult the excellent descriptions in [Hartley and Zisserman 2003], [McGlone 2004], [Pollefeys 2002], and [Zhang 1996]. The following discussion is taken primarily from these four sources, and will be presented without further reference.

3.1.3.1 Points, Lines and Planes

Most readers will be familiar with the concept of representing a point in $\mathbb{R}^2$ (2D Euclidean Space) by the coordinate pair (x, y). In the algebraic sense, $\mathbb{R}^2$ may be viewed as a vector space, and the 2D coordinate (x, y) is a vector in this vector space. In order to standardize this mathematical development with the current literature, we will treat coordinate vectors as column vectors, and therefore we describe the Euclidean point $\mathbf{x} = [x, y]^T$.

Geometric entities such as points, lines, planes, and conics are handled somewhat differently in the projective representation. In projective geometry, these entities are described using *homogeneous coordinates*, which describe the entity only up to an arbitrary scalar multiplier. Therefore, the homogeneous representation is not unique, and the entities $\mathbf{a}$ and $k\mathbf{a}$ represent the same thing. In this framework, a point in 2D projective space $\mathbb{P}^2$ is actually a 3-element vector of the form $\mathbf{x} = [kx, ky, k]^T$, where k is an arbitrary scalar constant. It should be clear from this that points may easily be transported from projective to Euclidean coordinates by dividing through by the last (homogeneous) coordinate k and then removing this element.

Lines in the plane may also be written in $\mathbb{P}^2$ and are again written as a 3-vector. The line $ax + by + c = 0$ is defined by its parameters (a, b, c), and multiplying each of these parameters by an arbitrary scalar leaves the resulting line unchanged. Since this retains the same equivalence relationship we defined earlier for homogeneous point coordinates, it is not surprising that the homogeneous representation of a line in $\mathbb{P}^2$ is $\mathbf{l} = [a, b, c]^T$. Like points, lines in $\mathbb{P}^2$ have only two degrees of freedom, since only the ratios of the parameters are important.

We are now in a position to see a small sampling of the usefulness of using homogeneous coordinates in working with an algebraic geometry. In order to see if a point lies on a line, we merely need to verify that $\mathbf{x}^T\mathbf{l} = \mathbf{0}$. If this equation holds, the point is on the line, if not, it is not. In an equally simply fashion, we may find the intersection of two lines $\mathbf{l}$ and $\mathbf{l'}$ according to $\mathbf{x} = \mathbf{l} \times \mathbf{l'}$, where $\times$ represents the cross product. Similarly, the line joining two points is defined by $\mathbf{l} = \mathbf{x} \times \mathbf{x'}$.

In $\mathbb{P}^3$, points are represented as 4-element vectors according to the form $\mathbf{x} = [kx, ky, kz, k]^T$. For vectors with $k \neq 0$, this corresponds to the point $\mathbf{x} = [x, y, z]^T$ in $\mathbb{R}^3$. We may now see that while a projective transform is a non-linear operation in $\mathbb{R}^3$, in $\mathbb{P}^3$ such a transform is linear, and may be represented by $\mathbf{X}' = \mathrm{H}\mathbf{X}$. In this case, H is the homogeneous representation of the projection, and as expected has 15 degrees of freedom, preserves straight lines and retains incidence relations (such as the point of intersection between a line and a plane).

A plane in 3-space may be written as $ax + by + cz + d = 0$ with parameters (a, b, c, d) such that $[a, b, c]$ represents the direction normal to the plane, and $d/ \parallel [a, b, c]^T \parallel$ is the distance of the plane to the origin. Since this plane is the same as the one defined by parameters (ka, kb, kc, kd), it only has three degrees of freedom, and therefore may be represented by the homogeneous 4-vector representation in $\mathbb{P}^3$ according to $\boldsymbol{\pi} = [a, b, c, d]^T$. In a manner similar to that of verifying if a point is on a line in 2-space, we may determine whether or not $\boldsymbol{\pi}^T\mathbf{X} = 0$ to see if a point lies on a plane. When this condition is met, the point is on the plane, when it is not, the point is off the plane. We may also find the plane defined by three points by determining the right nullspace of $[\mathbf{X}_1^T, \mathbf{X}_2^T, \mathbf{X}_3^T]^T$, and the point defined by the intersection of three planes by the right nullspace of $[\boldsymbol{\pi}_1^T, \boldsymbol{\pi}_1^T, \boldsymbol{\pi}_1^T]^T$. It should also be noted that for a point transformation $\mathbf{X}' = \mathrm{H}\mathbf{X}$, planes transform not by H but rather according to $\boldsymbol{\pi}' = \mathrm{H}^{-T}\boldsymbol{\pi}$.

Lines in $\mathbb{P}^3$ are a bit more difficult to represent with 4-vectors, since their 4 degrees of freedom would have a natural homogeneous representation via a 5-vector. Although several workarounds have been implemented successfully (including the use of Plücker matrices and Plücker coordinates, which is beyond the scope of this tutorial), such a representation is not required. For the sake of simplicity, in this research, when 3D lines are required, we will usually return to Euclidean coordinates for the solution.

3.1.3.2 Central Projection in $\mathbb{P}^3$

With the above development,we now have the tools necessary to describe the basic geometry of a general pinhole (central projection) camera using homogeneous coordinates. We will assume a positive-image (image plane in front of the projection center) model geometry, as seen in Figure 3.2. In this model, the center of projection (camera center), **C**, is the origin of the local camera coordinate frame, and the plane $Z = f$, where f is the focal length of the camera, is the image (or focal) plane. In this model, points in 3-space are imaged in the focal plane at the point **x** where the ray connecting **X** to **C** intersects the image plane. This point of intersection may be specified using 3-dimensional camera frame coordinates $[X, Y, Z]^T$, 2-dimensional continuous-space camera frame coordinates using $[x_c, y_c]$, 2-dimensional continuous-space image frame coordinates using $[x, y]$ or 2-dimensional discrete image frame coordinates. Generally, when working with discrete image coordinates, we will divide all image measurements by the pixel pitch, and then round the coordinates to provide the discrete pixel locations.

In Euclidean coordinates, it is fairly straightforward using similar triangles to show that a point at $[X, Y, Z]^T$ maps to 2D camera coordinates $[fX/Z, fY/Z]^T$. In homogeneous coordinates, we may similarly write

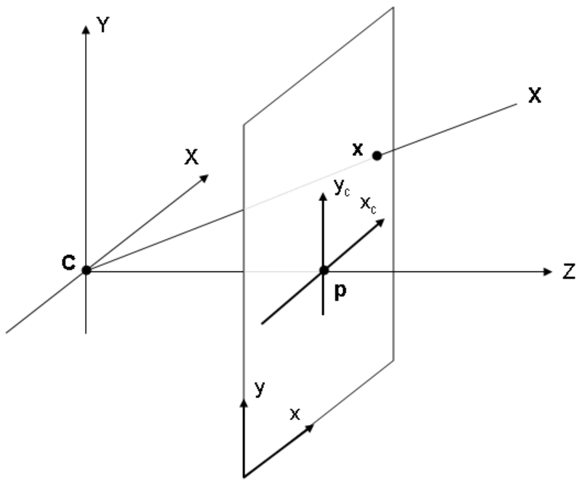


Figure 3.2: Geometry of central projection.

$$\begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = \begin{bmatrix} fX/Z \\ fY/Z \\ 1 \end{bmatrix} \sim \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{3.4}$$

If we wish to use continuous image coordinates instead, we must move the image's origin off of the principal point. This yields the solution

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} (fX + Zp_x)/Z \\ (fY + Zp_y)/Z \\ 1 \end{bmatrix} \sim \begin{bmatrix} (fX + Zp_x) \\ (fY + Zp_y) \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{3.5}$$

This solution may then be extended to pixel-based image coordinates, given (potentially non-square) pixel dimensions d_x and d_y. In this case, the principal point $\mathbf{p} = [p_x, p_y]^T$ may be written in terms of the pixel dimensions according to $x_0 = p_x/d_x$ and $y_0 = p_y/d_y$, and

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} floor(x_{tp}) \\ floor(y_{tp}) \\ 1 \end{bmatrix}, \tag{3.6}$$

where

$$\begin{bmatrix} x_{tp} \\ y_{tp} \\ 1 \end{bmatrix} \sim \begin{bmatrix} f/d_x & 0 & x_0 & 0 \\ 0 & f/d_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathrm{K}[\mathrm{I} \mid \mathbf{0}]\mathbf{X}, \tag{3.7}$$

and $\mathbf{X}$ is in 3D camera coordinates. In this notation, the 3×3 matrix K is called the camera calibration matrix, and it contains all of the required IOPs for a perfect central projection.

If the point location $\mathbf{X}$ is described in world coordinates instead of camera coordinates, we must also account for the translation and angular orientation of the camera in order to properly determine the location of the point's image in the focal plane. This may be simply accomplished by noting that the Euclidean representation of the 3D point in the camera frame, represented as $\tilde{\mathbf{X}}_{cam}$, is equal to $\mathrm{R}(\tilde{\mathbf{X}} - \tilde{\mathbf{C}})$, where $\tilde{\mathbf{X}}$ is the Euclidean representation of the point, $\tilde{\mathbf{C}}$ is the camera center, both in world coordinates, and R is an "active" rotation matrix of the form described in Appendix A.2. This now permits us to write

$$\mathbf{x} = \mathrm{KR}[\mathrm{I} \mid -\tilde{\mathbf{C}}]\mathbf{X} = \mathrm{K}[\mathrm{R} \mid \mathbf{t}]\mathbf{X} = \mathrm{P}\mathbf{X}, \tag{3.8}$$

where $\mathbf{t} = -\mathrm{R}\tilde{\mathbf{C}}$, and the 3×4 matrix P is termed the camera matrix. Determination of P is equivalent to solving the full resection problem first discussed in Section 2.2.

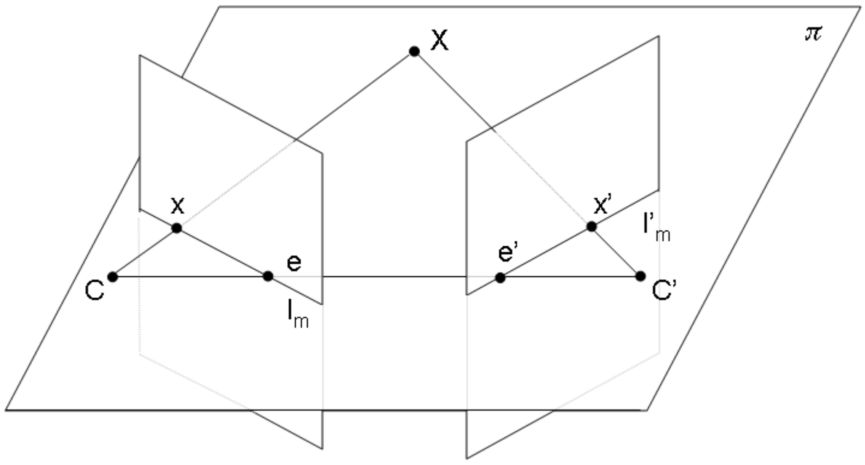


Figure 3.3: Epipolar geometry.

3.1.3.3 Epipolar Geometry and Multiple-Image Object Reconstruction

Epipolar geometry is the description of the fundamental projective relationship between two camera systems. Given two camera centers located at **C** and **C′** as shown in Figure 3.3, we may make some interesting observations. First, for any point **X**, we may observe that **X**, its image in the first camera **x**, and the image in the second image **x′**, **C**, and **C′** are all coplanar. Also, the line connecting **C** to **C′** (termed the *baseline*) is in this plane, therefore the points of intersection between this line and the two image planes (the *epipoles* **e** and **e′**) are also on this plane.

This provides a useful constraint on the relationship between corresponding points in the two images, provided the relative geometry of the cameras is known. A point **x** in the left image effectively defines the plane π via that point, **C** and **e**. This plane in turn defines the *epipolar line* $\mathbf{l}'_{\mathbf{m}}$, on which $\mathbf{x}'$ is guaranteed to lie. When searching for image correspondences, this constraint can greatly reduce the search space.

For uncalibrated cameras (K is unknown), the entire epipolar geometry can be specified via a 7 degree of freedom, 3×3 matrix termed the fundamental matrix, F. This matrix effectively provides a mapping between image points in one image and the corresponding epipolar line in the conjugate image. We may note that for any pair of corresponding points in the two images $\mathbf{x}$ and $\mathbf{x}'$, $\mathbf{x}'^T \mathrm{F}\mathbf{x} = 0$. This means that F may be computed from image correspondences alone, and the epipolar geometry is dependent only on the two camera matrices, P and P′. However, while F is determined uniquely from the two camera matrices (up to a scale factor), the inverse is not true. The matrices P and P′ may only be recovered up to a projective ambiguity, since the cameras PH and P′H yield the same fundamental matrix as P and P′, where H is any 4×4 matrix representing a projective transform in $\mathbb{P}^3$.

This ambiguity has a significant impact when we subsequently wish to perform the process of intersection to determine the coordinates of the object points. Since F is specified by the image correspondences $\mathbf{x} \leftrightarrow \mathbf{x}'$, but the camera matrices may only be recovered up to an arbitrary projective transform, the final 3D points in object space are valid only up to this same ambiguity. That is, given an adequate number of point correspondences in the images with no additional information, the location of a given image pair may be determined to be $\mathbf{X}$. However, due to the projective ambiguity, H$\mathbf{X}$ is also a valid solution, where H is the same for all points. It is only through a knowledge of the IOPs (K matrices), through finding correspondences between the image points and world points, or through a partial knowledge of the scene geometry (e.g. that certain features are parallel or orthogonal) that this ambiguity may be removed.

If the IOPs are known a priori (or have been determined via in-scene techniques), the geometry of Figure 3.3 is further specified. By knowing the principal point offsets and the focal lengths of each camera, the fundamental matrix is further constrained, and is renamed the *essential matrix*, E. E represents the six unknown quantities of the EOPs, but since there is an overall scale ambiguity, the matrix has only five degrees of freedom.

In order to get a metric reconstruction of points in 3-space using the known IOPs, we may proceed along two different paths. We may first process each point via $\hat{\mathbf{x}} = \mathrm{K}^{-1}\mathbf{x}$, where $\hat{\mathbf{x}}$ is the

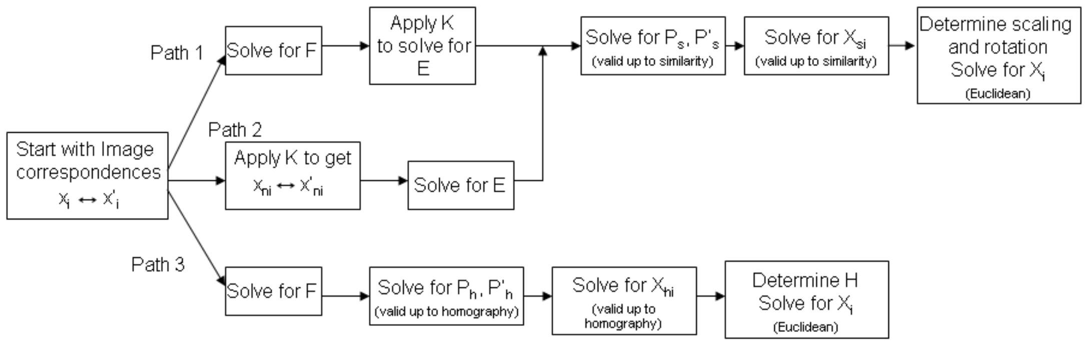

Path 1
Solve for F
Apply K to solve for E
Solve for P_s, P'_s (valid up to similarity)
Solve for X_si (valid up to similarity)
Determine scaling and rotation Solve for X_i (Euclidean)
Path 2
Start with Image correspondences x_i ↔ x'_i
Apply K to get x_ni ↔ x'_ni
Solve for E
Path 3
Solve for F
Solve for P_h, P'_h (valid up to homography)
Solve for X_hi (valid up to homography)
Determine H Solve for X_i (Euclidean)

Figure 3.4: Approaches to 3D point recovery using multiple view geometry.

point expressed in what the literature terms *normalized coordinates*. However, since coordinates may have to be "normalized" in a different sense for numerical conditioning purposes, I prefer to refer to the $\hat{\mathbf{x}}$ as *compensated coordinates*. We may then recover P and P$'$, and although they are defined only up to an arbitrary similarity, they may be used to derive a metric reconstruction.

An alternate approach is to use to uncompensated matching points to first define F, then convert F to E via the relationship $\mathrm{E} = \mathrm{K}'^T\mathrm{FK}$. We then proceed as before. Of course, a third approach is simply to perform the projective reconstruction using F, then using world-to-image correspondences to remove the resultant ambiguity. These alternatives are illustrated in Figure 3.4.

3.2 Iterative Optimization: The Levenberg Marquardt Algorithm

In many practical problems, a 'best' solution is found by assuming that the provided data fits the form of a chosen mathematical model, and that in some sense the cumulative deviations from this model should be minimized. The deviations are typically quantified by specifying a cost function that measures the agreement between the observations and the model, given a set of model parameters. In many cases, the cost function is as simple as the sum of the squared error values at each data point.

In many cases, the cost function may be linearized, and a direct solution may be found through familiar techniques such as a traditional least squares regression or a Demming regression (refer to Appendix A.1 for additional information regarding these techniques). However, when the problem of optimization can not be linearized, iterative algorithms may still be used in an attempt to find the parameters that drive the cost function to a minimum value.

The familiar gradient descent method takes incremental steps that are proportional to the negative of an approximate value of the local gradient at each iteration. For the case of a scalar-valued function, we start with an initial set of parameters $\mathbf{p}_0$. We then compute the gradient of the function f at this location, $\nabla f(\mathbf{p}_0)$ in order to determine the direction of steepest descent. The next value is then chosen in a path along this direction according to

$$\mathbf{p}_{i+1} = \mathbf{p}_i - \eta(i)\nabla f(\mathbf{p}_i), \tag{3.9}$$

where η is a positive scale factor that determines the step size [Duda et al. 2000]. Unfortunately, there are many issues that plague the basic gradient descent approach. Intuitively, we would want to take large steps when the gradient is small and smaller steps when the gradient is steep so as to not overshoot the minimum point. However, gradient descent inherently does the opposite. Ignoring the local curvature is another issue. As noted in [Roweis 1996], if there is a long narrow valley in the error surface, the component of the gradient in the direction along the base will be small relative to the component in the direction of the walls. As such, the dominant motion will be in the direction of the walls rather than directly down to the base.

By taking local curvature into account, a significantly better solution may be realized, and this forms the basis of Newton's method. By expanding the expression of the gradient at $\mathbf{p}_i$ using a Taylor series, ignore the higher-order terms, and solve for the minimum point, we get the improved update rule

$$\mathbf{p}_{i+1} = \mathbf{p}_i - (\nabla^2 f(\mathbf{p}_i))^{-1}\nabla f(\mathbf{p}_i), \tag{3.10}$$

This approach does not need to evaluate the Hessian ($\mathrm{H} = \nabla^2 f(\mathbf{x})$) exactly, and the approximation $\nabla^2 f(\mathbf{x}) = J(\mathbf{x})^T J(\mathbf{x})$ may be substituted, where J represents the Jacobian matrix. In this case, the method is termed the Gauss-Newton algorithm, and convergence is usually fairly rapid.

It can be seen that both the gradient descent and the Gauss-Newton algorithms provide certain advantages. In order to try to capture the best features of each of these approaches, Levenberg created a blended approach, using the update rule

$$\mathbf{p}_{i+1} = \mathbf{p}_i - (\mathrm{H} + \lambda \mathrm{I})^{-1} \nabla f(\mathbf{p}_i). \tag{3.11}$$

In this method, if the error is reduced in a given iteration, the iteration is accepted, and λ is reduced by a scale factor (usually 10) in order to reduce the degree of gradient descent. However, if the error goes up, the iteration is discarded, λ is increased by the scale factor, and the iteration is re-attempted. This increase in λ moves the process more towards gradient descent.

Marquardt sought to improve this composite method by making use of the Hessian matrix (and scaling each component of the gradient according to curvature), even for large values of λ [Roweis 1996]. This modification results in larger movements along the direction where the gradient is smaller, which is intuitively what we desire. In this, the Levenberg-Marquardt (LM) algorithm, the modified update rule is therefore

$$\mathbf{p}_{i+1} = \mathbf{p}_i - (\mathrm{H} + \lambda \, diag[\mathrm{H}])^{-1} \nabla f(\mathbf{p}_i). \tag{3.12}$$

Although this method is not strictly an optimal descent, it tends to work very well in practice, and has been widely used in the field of computer vision over the past several years. In modern photogrammetric research, the LM algorithm is frequently used to refine the solution for camera matrices and computed homographies. These applications are also relevant to this work, and we will also use this method to fit non-planar surfaces to clusters of 3D lidar points.

3.3 Robust Parameter Estimation

The classical least squares approach to parameter estimation optimizes the model relative to all of the given data. Such a technique implicitly assumes that all data points are valid, and in many cases that the noise associated with each point is Gaussian. When these assumptions are not met, the quality of the solution may be significantly reduced.

Robust parameter estimators, on the other hand, seek to provide good solutions, even in the event that the baseline assumptions are not fully met. They typically do this by removing or mitigating the effect that gross outliers have on the final solution. Such outliers often arise due to gross measurement errors or the presence of multiple structure in the data, but other causes are also possible. Such errors are especially common in automated systems, where the data is often the product of error-prone processes. By removing the majority (or all) of the effects of outliers in the estimation, the final solution is much more likely to be useable. This idea is illustrated in Figure 3.5, which was adapted from one of the seminal papers on robust estimation [Fischler and Bolles 1981]. The data in this figure contains 7 well-behaved data points, and a single gross outlier. The resulting models include those from a standard least squares regression (the initial least squares fit), an incorrect approach to robust fitting (described below) and a well-conceived robust algorithm (RANSAC), where the effect of the outlying datum is not included in the final solution.

In order to handle data outliers, a potential technique is to first compute the least squares solution using all of the available data, then discarding those points that lie farthest from the solution. A new least squares solution is then achieved using only the remaining points. This process is iterated until no more outliers remain, or until the model is deemed satisfactory in some other sense.

It is a simple matter to show that this approach may fail when even a single gross error is included in a set of otherwise good data points. Consider again the data shown in Figure 3.5. After the initial least squares fit, the error of point A to the line is actually larger than that for

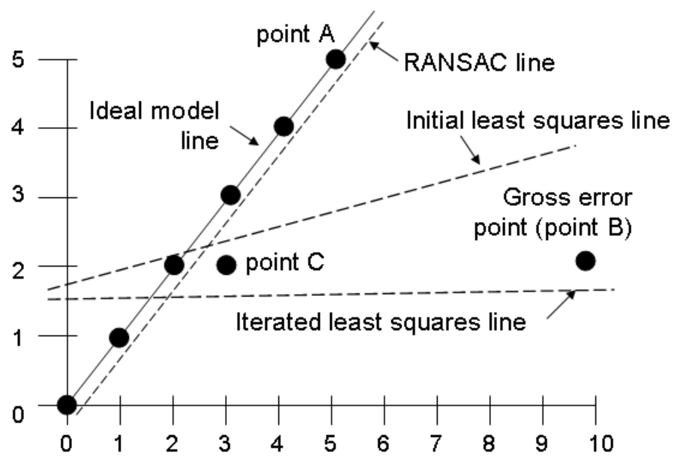


Figure 3.5: Line fitting via RANSAC and iterative least squares (adapted from [Fischler and Bolles 1981].

point *B*. Therefore, this approach would remove point *A*, then perform an additional least squares fit. With point *A* now removed, the regression will be weighted even more towards point *B*, and another 'good' data point is removed. The final iterated least squares fit is indicated by the lower dashed line in the figure. Although this approach is still advocated occasionally, it clearly should be avoided.

The robust method considered in this research, the RANdom SAmple Consensus (RANSAC) [Fischler and Bolles 1981], provides a completely different paradigm for the removal of outliers. Whereas most least squares approaches use as much data as possible for the initial fit then attempt to remove outliers, RANSAC starts with a minimal initial data set, and then evaluates how many data points tend to agree with this model hypothesis. If the agreement rate is high enough, then the points in agreement are considered inliers, and they are (all) used to re-fit the model in a least-squares sense. If the fit does not meet the agreement rate, a new minimal set is selected, and the process is performed again. The iterations continue until the agreement rate is sufficiently high, or a pre-defined number of iterations has been realized, at which point the largest inlier set from the previous iterations is used. Looking again at Figure 3.5, we see that this method, while still including point C in the final regression, fits the data much better than the other two techniques.

In a bit more detail, this algorithm may be described by the following pseudocode.

```
• Assume:
    – $S$ = set of all data points (including outliers)
    – $N$ = minimum number of points to define the parameterized model
• Define:
    – $k$ = maximum number of iterations
    – $t$ = distance threshold for determining if a point is an inlier to
      a given model
    – $d$ = minimum number of inliers to exit the algorithm early
1. Randomly select $N$ points and define the model from these points
2. Determine the set $S_i$ (the consensus) of points that are within $t$ of the model
3. If size of $S_i$ is greater than $d$, re-estimate model using $S_i$ and END
4. If size of $S_i$ is less than $d$, return to Step 1
5. After N trials, select largest $S_i$ so far, re-estimate model using this $S_i$, and END
```

Note that only the parameters N, t, k, and d need to be specified, and N is actually set by the form of the assumed model. The parameter k should be large enough that there is a fairly high probability of selecting at least one set of N points that are all inliers. As noted in [Fusiello 2007], given a fraction of outliers ε, the probability of randomly selecting p inliers in k trials is

$$P = 1 - (1 - (1 - \varepsilon)^p)^k. \tag{3.13}$$

Therefore, by selecting a value of P near 1, we may solve for the required value of k.

In this dissertation, RANSAC will be used for a variety of purposes. In the polyhedral building modeling from lidar data, it will be used to fit planes to data points in 3-dimensions. Additionally, through an analysis of the outliers, multiple planes may be recovered from a single dataset. RANSAC will also be used for less obvious purposes as well. When attempting to establish feature correspondences between a 3D model and a 2D image, RANSAC will enable a robust fit where additional unmatched features may be defined in each domain. A similar approach may be used when selecting homologous points in multiple images in order to form a photogrammetric model.

Method is much, technique is much, but inspiration is even more.

Benjamin Cardozo

There are some enterprises in which a careful disorderliness is the true method.

Herman Melville

"Aelius Donatus (fourth century) is quoted by his student St. Jerome as saying 'Pereant qui ante nostra dixerunt,' freely translated as 'Damn the guys who published our stuff first.'"

As noted by mathematician Ralph P. Boas, Jr.

4 Approach

Although the current process for creating DIRSIG scenes has proven to yield excellent resultant images, a streamlining of the construction of large scenes would be beneficial to many users. To this end, a new process has been created where several of the scene design tasks have been replaced by semi-automated methods. The baseline approach is to initially extract enough feature information from the lidar data to perform a registration among the various data sets, then to refine and add to these features using all available imagery. These steps are depicted below in Figure 4.1, and will be explained in greater detail throughout the remainder of this chapter. This approach assumes that multiple image types are available, including lidar, hyperspectral, and high resolution frame-array imagery. If any of these image types is not available, the approach

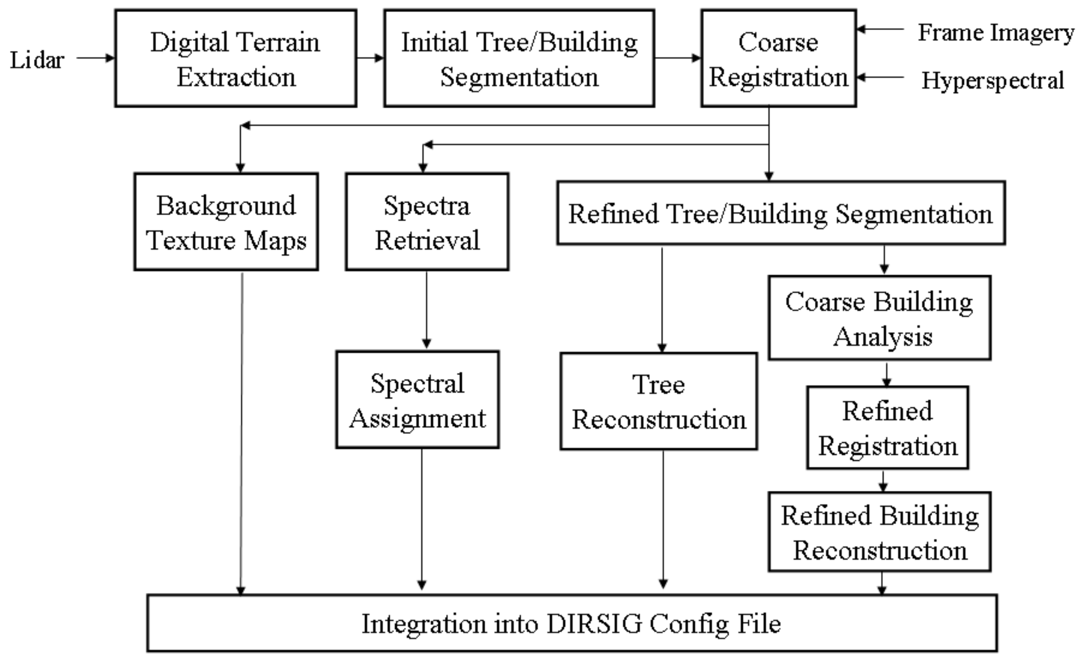


Figure 4.1: New process for scene reconstruction.

requires modification, as described in Section 4.8.

4.1 Extraction of the Terrain Model

For many years, models of the earth's terrain were obtained almost exclusively through the laborious use of traditional surveying techniques. This practice was augmented in the 1970's by the development of terrain-modeling photogrammetric techniques, which reduced the modeling time required in many cases. However, with the recent advances in laser sensors, digital versions of these terrain models are increasingly being collected with lidar systems. Lidar offers the advantages of rapid, high-density data collection that in most cases has a vertical accuracy significantly better than that of more traditional methods. Additionally, lidar models do not require the difficult task of identifying homologous points between images, thereby making lidar-derived terrain extraction algorithms much more amenable to automation. As such, the baseline approach to terrain extraction pursued for this research uses lidar point data. This also has the advantage that it will later help to isolate building structures, which aids in the registration lidar data to other imagery. If lidar data are not available, a digital elevation model and digital terrain model may be obtained from other imagery using the photogrammetric methods discussed in Section 4.8.

Before proceeding, it would be beneficial to clarify a few terms that will be relevant to the following discussion. In this dissertation, a digital elevation model (DEM) will be used as specification of the ground that includes objects such as buildings, vegetation, and vehicles. This contrasts with the digital terrain model (DTM)[1], which has removed such objects and retained only the geometry of the bare earth. A digital canopy model (DCM), which will be used to identify individual tree locations, retains only the ground and the outer-boundary points of trees. These definitions are illustrated in Figure 4.2. In most cases, each of these models may be specified as either a set of irregularly-spaced points that are connected via a triangular irregular network (TIN), or they may be in a rasterized form, where the model points have been interpolated to a regularized grid. Also possible, although significantly less common, is to use parametric functions to define these models.

4.1.1 DTM Extraction from Lidar Data

Due to the nature of the lidar imaging process, the data resulting from a collection are really just a listing of points in arbitrary locations, where each point contains a location (x, y, z) in 3-dimensional space, a return intensity value, a return number (1 meaning that the point represents the first echo received back from a given transmitted pulse, 2 being the second received echo from the same transmitted pulse, etc.), a time stamp, and potentially other information as well. This listing of arbitrarily-located points is termed a point-cloud, and it represents the laser data in the

[1]Many references also make use of the term *digital surface model* (DSM), but its use in somewhat inconsistent. In many applications it references to the DEM, but it is occasionally used to represent the DTM instead

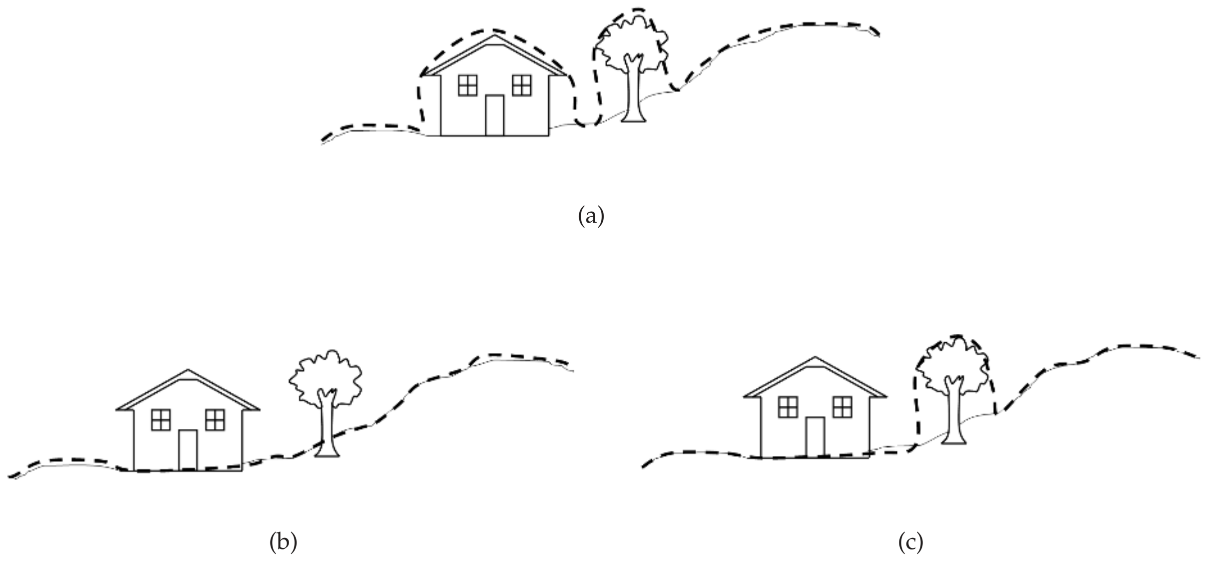


Figure 4.2: Digital models: (a) Digital Elevation Model (DEM), (b) Digital Terrain Model (DTM), and (c) Digital Canopy Model (DCM).

rawest form that will be considered here.

Although many algorithms make use of the point cloud directly, due to the non-uniform spacing of the data in 3-space, it is often advantageous to resample the data to a regularised 2-dimensional grid. In doing so, the data takes the form a basic 2D array of values, and may be processed using conventional image processing techniques. Such a resampled dataset will be referred to in this work as a rasterized image.

Many rasterized images may be produced from a given lidar point cloud. If the array values are range (z-coordinate height values), the raster image is termed a range image. In this research, we define five different range images and seven different intensity images. The R_1 image is obtained by performing a bilinear interpolation using only the first return values. Similarly, the R_L image uses only the last return of each pulse, while the R_A raster image makes use of all range data in performing the interpolation. If we assign a grid value based simply on the minimum range value in a region (where empty cells are filled in by interpolating neighboring values), the

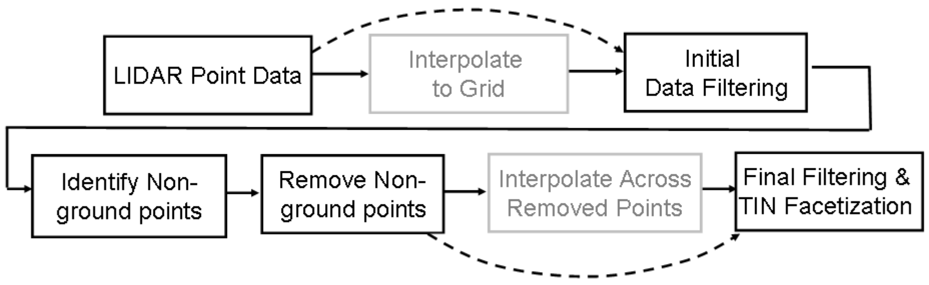

LIDAR Point Data
Interpolate to Grid
Initial Data Filtering
Identify Non-ground points
Remove Non-ground points
Interpolate Across Removed Points
Final Filtering & TIN Facetization

Figure 4.3: Block diagram for DTM extraction.

raster image is termed R_{MN}. The R_{MX} raster uses maximal local values to assign the grid value. A corresponding set of rasterized intensity images are also available. Note that the I_{MN} and I_{MX} images use the minimum or maximum intensity values within a grid region. If we wish to use the intensity value corresponding to the minimum or maximum range points in each cell, the corresponding raster images are termed I_{MNR} and I_{MXR}. Although many of these raster images are used throughout this dissertation, several are not. However, the code included in the bound copies of this work makes use of all of these, so for completeness, all are defined here.

When point locations are explicitly given, the raw lidar point cloud may be thought of as a type of DEM, as it is a straightforward process to produce a TIN representation from the 3D points. In producing a DTM from this DEM, the fundamental concept is to separate the ground points from the non-ground points, then to remove the non-ground points (and potentially interpolate across them). Due to the simplified nature of processing data in range image format, this will be the primary baseline for the methods discussed here. However, it should be emphasized that processing may be carried out in either a point-based or rasterized form. Even when the final model is to be specified via a list of points and a TIN connectivity, the processing may be performed on rasterized images, in order to reduce computational requirements. The generic process for converting a DEM to a DTM using lidar data is highlighted in Figure 4.3.

4.1.1.1 DTM Extraction in the Literature

Due to the many features that distinguish ground points from adjacent objects, a multitude of DTM extraction techniques have been proposed in the literature. As illustrated in Figure 4.4, these may be divided into three broad categories, based on the underlying assumptions about what ”ground” consists of, as well as the methods used to identify these ground points. The first approach, which is termed *region-based* here, makes the assumption that object points are locally at a higher elevation than are ground points. Such algorithms typically make use of local minimum points, height differences, or the angular relationship between points. The second class of DTM filters is termed *function-based*, and it assumes that the ground model may be described by a parametric function that best fits the data according to certain rules. The final class of filter, which includes those algorithms which are *feature-based* or *segmentation-based*, assigns properties to each data point. The points are then classified with standard pattern classification algorithms using the pre-defined properties as feature vectors. In the following paragraphs, a selection of techniques from the literature will be explored in slightly greater detail.

Using the assumption that object points are higher than the adjacent ground, morphology filters were among the first applied to distinguish terrain points from other features in the lidar data. As shown in Chapter 3, replacing a pixel’s value with the minimum value over a windowed area is equivalent to performing a gray-scale erosion with a uniform structuring element. Intuitively, such a technique makes sense for ground extraction, provided the window is sufficiently large to span the broadest object structures (see Figure 4.5. By applying a dilation (point replacement with a regional maximum) after this erosion (an opening), improved results may be obtained. This idea was first proposed in ([Lindenberger 1993], and other morphological approaches have been presented in [Weidner and Forstner 1995], [Petzold et al. 1999], and [Morgan and Habib 2002]). These techniques define a window that is larger than the largest building, and this window is used to specify a neighborhood of pixels used at each point during the processing. For each pixel in the DEM, the minimum value in it’s neighborhood (with the window centered on the pixel of

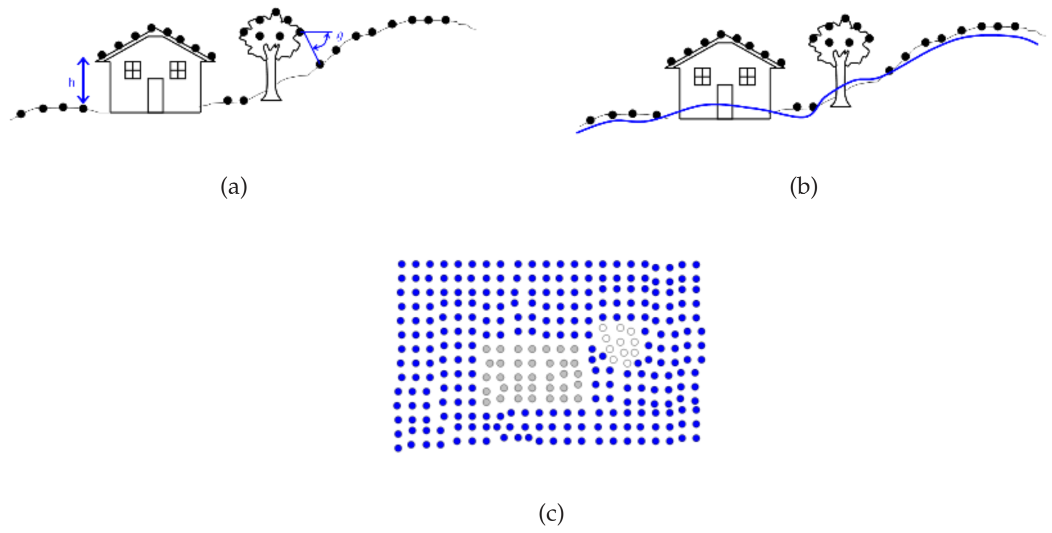


Figure 4.4: Methods of DTM extraction: (a) Local region-based, (b) Function-based, and (c) Feature/segmentation-based.

interest) is found and assigned to that pixel. After this *erosion* operation is completed, a *dilation* is performed, whereby each pixel in the processed image is examined relative to its neighbors, and the maximum pixel value in the neighborhood is assigned to the pixel of interest.

Many modifications to the standard morphological approach have also been proposed. In [Kilian et al. 1996], multiple window sizes are used, and the ground points identified at each iteration are assigned weights based on the current window size. A weighted calculation is then performed to create the final ground model. In [Lohmann 2000], a sequential operation (dual-rank filter) is used. Progressive filtering using different window sizes was also shown to be effective in [Zhang et al. 2003] and [Zhang and Whitman 2005]. These method of terrain extraction have proven to be successful in many applications, but they have also been found to be susceptible to noise in the DEM. Although a median filter may be applied to mitigate some of the noise effects, such an approach is unable to compensate for larger patches of noisy data [Ma 2004].

Many authors have also proposed functional techniques for extracting the terrain model. The

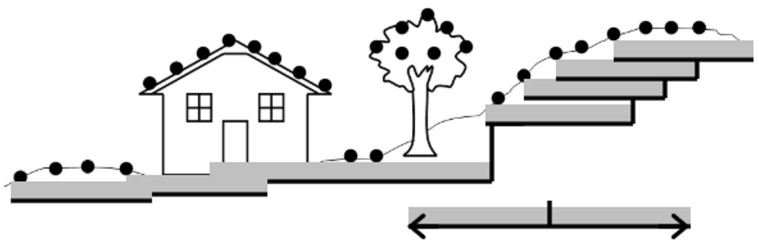


Figure 4.5: Illustration of morphological erosion (local minimum) terrain filtering.

authors of [Kraus and Pfeifer 1998] use a hierarchic approach, where at each level, a polynomial function is fit to the data. At each iteration, weights are assigned to the points based on residual values. Points below the surface are given larger weights, so that when the final surface is derived using a kriging technique based on the previously computed weights, the surface is attracted to the lower data. This technique was modified in [Pfeifer et al. 2001] to produce improved results in an urban setting. Active contours are used in [Elmqvist 2001], [Elmqvist et al. 2001], and [Elmqvist 2002]. A conformable membrane is brought up from below the data, and is allowed to stick to the data subject to energy constraints.

Iterative approaches where individual points are incrementally added or removed have also proven effective. The method described in [Haugerud and Harding 2001] analyzes the local curvature at each point. Ground points are then identified by removing tree points in an iterative algorithm. In [Axelsson 1999], a sparse TIN estimate of the ground is progressively densified, subject to certain constraints, until the full set of ground points has been identified.

4.1.1.2 DTM Extraction Techniques Used in This Research

In addition to the above methods, this research sought to investigate the feasibility of a few additional simple techniques. One potential method for determining non-ground pixels is based on fitting polynomial splines to each row and column of the range image. The height of each pixel above these curves is then determined, and through a simple thresholding points may be classified as being ground-level or not. Although this technique was successful in several cases, it was found to be difficult to determine parameters that were robust with respect to different sampling densities and different sized scenes. Additionally, in many cases this method failed to produce accurate results at the scene edges. As such, this technique was determined to be unsatisfactory.

An alternate approach, the modified minimum filter (MMF), was used in the early stages of this research and was first presented in [Lach et al. 2006]. This technique uses a variation of the erosion operator (spatial sliding window minimum filter) to identify points that are significantly higher than their neighbors. Although in many cases a standard erosion filter could be used, as noted above, care must be taken to ensure that the kernel is large enough to span the roof structures of the largest buildings in the scene. If it is not, the central points of large objects may not be flagged as being non-ground, and more complicated processing would be required. However, when the kernel is large, this technique may fail in regions where there is a low ratio of ground to non-ground points. We avoid these issues by computing the minimum value for a small region (typically 10m x 10m), and then flagging points in a larger region that are significantly higher than this median value. This modified minimum filter also has the advantage of being much more computationally efficient, and a similar technique may be employed directly on the point cloud, if desired. Figure 4.6 illustrates this concept for a single location of the filter's sliding windows. Note that the window still moves on a pixel-to-pixel basis, and not block-by-block.

Once the points that are significantly higher than their surroundings have been flagged as being non-ground locations, a second filtering operation must be performed in order to remove additional objects from the terrain. These include large tree canopies, isolated small trees, vehicles, and other man-made objects, which would all be ignored by median filter processing. In order to effectively remove these points, the lidar range image is high-pass filtered, thereby highlighting regions with rapid changes in elevation. Bright pixels in this second filtered image (those with

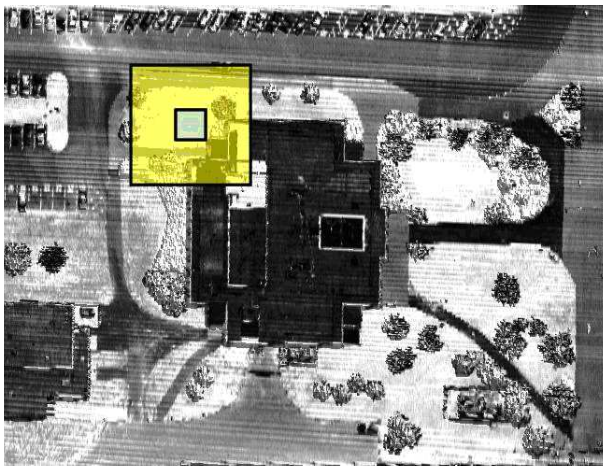


Figure 4.6: Illustration of the Modified Minimum Filter at a single location; minimum height value is calculated in smaller window, points 3m above minimum in larger window are flagged as non-ground points.

maximum rate of change) are also flagged as being non-ground. A simple sliding window filter based of the Laplacian of the form

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad \Longrightarrow \quad \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & -8 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \tag{4.1}$$

may be used for this purpose. We have also noted that first-derivative based approaches also work well in most applications for determining transition regions.

After the non-ground pixels have been identified, they are removed from the data set, an initial rasterized version of the terrain model may be obtained by interpolating across the removed points. If desired, a smoothing filter may be applied to remove remaining high-frequency content in the resultant data, and the output is then facetized to produce the final DTM. If the terrain model does not need to be in a rasterized form, this interpolation may not be needed, but a final smoothing may still prove to be desirable.

More recently, this research has replaced the MMF approach with a modified version of a

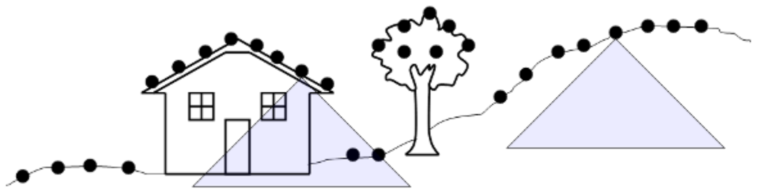


Figure 4.7: Slope-based terrain extraction.

different region-based algorithm from the literature. In [Vosselman 2000] (and updated in [Vosselman and Maas 2001]), the authors propose using a slope-based filter to identify object points. In this approach, a point is classified as being non-ground if the maximal slope of the vectors connecting a point to its neighbors is above a pre-defined threshold, where the threshold may be a function of the planimetric distance between the points. This is equivalent to placing a funnel-shaped structuring element representing the functional threshold at each point, and determining if any other data points fall beneath the funnel (see Figure 4.7). If a point falls below the funnel, the original point is labeled as an object point. When viewed in this way, the slope approach may be seen to have close ties to a grayscale erosion operation. In his original work, Vosselman demonstrated that this approach could be viewed as performing a grayscale erosion on the scene, and then checking the resulting values. A point was considered a ground point if its z-coordinate did not exceed the z-coordinate of the eroded surface. This technique was further modified in [Sithole 2001], where the slope function was made adaptive based on local terrain features.

This approach is extremely attractive for two reasons. First, the technique inherently accounts for the non-uniform nature of the data. When comparing points using only a height threshold, no measure was used to account for the potentially variable distance between the points. Points that are further apart should conceivably be permitted to also have larger height differences. This is inherently captured in Vosselman's approach. Second, although the original authors implemented the algorithm in point form, they also show that the slope-based approach is very similar to the morphological filters in a mathematical sense.

As such, an implementation of the slope-based approach may be implemented efficiently on a rasterized range image. If a TIN model is ultimately desired, the subset of the original data belonging to the ground may be found by finding all points within a given region of the filtered (rasterized) surface. It is this approach that has ultimately been used as the baseline approach for the initial terrain filtering.

4.1.2 Subsampling the DTM

The output of the above approaches provides either a raster image of the terrain or a TIN representation of the full set of ground points. For small regions or low spatial resolutions, these models may be used as-is, as their storage and processing requirements will not be prohibitively large. However, for broader scenes, the data may have to be sub-sampled in order to yield useable run times when producing simulated imagery from the scene models. In working with DIRSIG on current hardware, we have found that an upper limit to the number of ground facets is approximately 300,000, with 100,000 providing significantly faster results.

While raster models are inherently subsampled uniformly through standard techniques, several options are available for raw point DTMs. If a reduced point count is desired, a simple approach would be to simply remove a certain percentage of the ground points, either uniformly or randomly. In general, this produces acceptable results in most cases; however, if we know in advance that certain regions will be more important in the final imagery, it makes sense to provide more terrain detail in these regions and less elsewhere.

Many techniques are available in the literature for intelligently reducing point data in such a way as to retain the most shape information for a given percentage of retained points. Examples include [Ma and Saetzler 2008] and [Lafontaine 2000]. However, these techniques all assume that rapid shape transitions imply importance, so most of the final points are in regions where the sur-

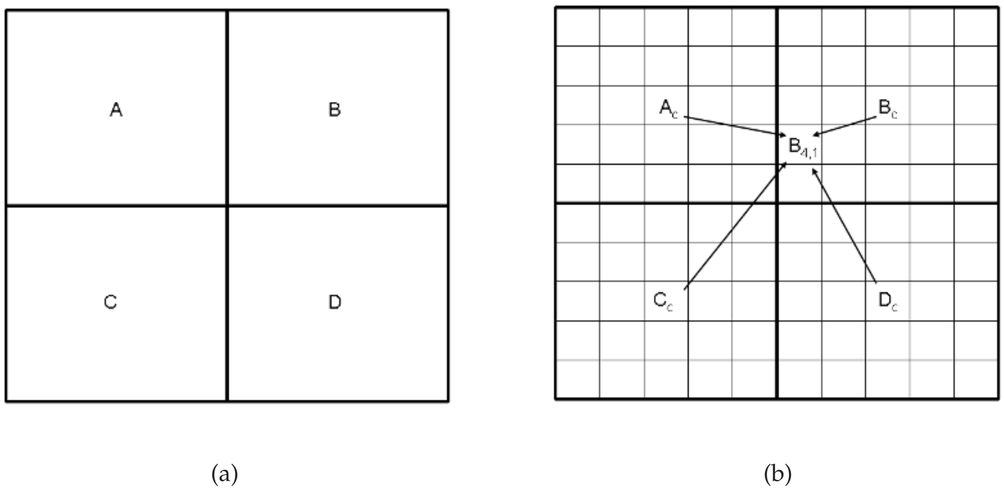


Figure 4.8: Subsampling the DTM: (a) Subsampled point densities are specified for each region A, B, C, D, (b) Each region is then divided into 25 smaller regions, each with a point density defined through a bilinear interpolation of the original regions A, B, C, and D.

face is varying rapidly. This contrasts with the typical DIRSIG requirement, where the important regions, which contain residential neighborhoods, factories, and other urban-type environments, are typically the flattest in the scene. Mountains, hills, cliffs and ravines, although required in many scenes, may often be represented at much lower resolutions. Therefore, a re-sampling approach that does exactly the opposite of those in the literature is required.

We begin by dividing the terrain model up into equally-sized square regions, each being approximately $2.25 \times 10^4 \mathrm{m}^2$. Each region is then assigned a reduction ratio, based on the data with that particular region. These ratios may be defined manually, they may be related to the number of houses, trees, or other features in the region (items that will be determined during later steps in the scene-building approach), or they may be based on the terrain model.

One useful method for assigning reduction ratios has been to base them on the average angle between normal vectors of a smoothed model of the terrain. In this approach, the TIN DTM is sampled to a raster grid at 2m point spacing. The image is then low-pass filtered using a 5×5 uniform convolution filter, and the normal vector of each point is then computed. The angle between each normal vector $\mathbf{n}_1$ and that of each of its neighbors, $\mathbf{n}_i$, is then computed according to $\theta = cos^{-1}(\frac{|\mathbf{n}_1 \cdot \mathbf{n}_i|}{\|\mathbf{n}_1\| \|\mathbf{n}_i\|})$, and an average angle for each region is determined. The average normal separation angle for each region θ_{avg_i} is divided by a scalar k to produce the initial reduction ratio $r_i = \theta_{avg\_i} k$ for each square. The value k is calculated according to

$$k = \frac{\sum_{i=1}^{M} \theta_{avg\_i}}{N}, \tag{4.2}$$

where M is the number of square regions and N is the desired number of points in the final model.

Although we may now use r_i to randomly remove that ratio of data from each square, if we do this directly, the different resolutions of each area will be visible in the resultant terrain model. To reduce this effect, we subdivide each square into 25 smaller squares, per Figure 4.8 (b). The initial r_i values are then assigned to the central small squares in each larger square. Additional reduction ratio values $r_{i\_(i,j)}$ are then assigned to the remaining small squares. A bilinear interpolation is used, where the distance to the 4 nearest central squares and their corresponding ratios are the relevant parameters.

4.2 Initial Segmentation of Trees from Buildings

Through the method proposed in the previous section, a digital model of the terrain was extracted. We may now subtract the DTM from the original data (in either point or raster form) to obtain a *normalized digital elevation model*, or NDEM. From the NDEM, by simply selecting data with a z-coordinate larger than a certain threshold, we may extract a set of points that are significantly higher than the terrain. If the threshold is set to approximately 2 meters, this effectively removes most shrubs and vehicles from consideration, while retaining larger vegetative structures (termed 'trees' here) and larger man-made objects (termed 'buildings'). A 'high-point' mask may be created from this thresholded image, where high points are assigned a value of one, while the points not exceeding the threshold are set to zero. The next logical task is to identify to which class each of these points belongs. With finely registered multi-modal imagery, many features are available for addressing this problem. However, at this stage, the other data sources are not yet co-registered with the lidar data so we must first perform an initial segmentation using only information available in the lidar data. This segmentation will permit us to handle trees and buildings differently (if desired) during the initial data registration process.

Previously documented approaches have exploited various lidar-related differences between these two classes. In [Ma and Meyer 2005], the author lists several approaches for building detection that, given certain assumptions, could also be used for the building-tree classification problem. These include techniques for finding buildings based on homogeneity of both height and lidar return intensity, using the average length of extracted surface edges as a distinguishing parameter, and identifying buildings through the detection of planar patches. Additionally, [Ma and Meyer 2005] notes that multiple returns are more likely in forested regions, and examining the number of secondary returns over a small neighborhood enables the classification of spatially clustered points. Taking a cue from the first of these methods, we were able to show that the local average magnitude of the high-pass filter output (from the DTM extraction stage) may be used in many cases to distinguish trees from buildings.

In the early stages of this research, we successfully implemented a simple segmentation using the assumption that building regions have larger areas without holes than do tree regions [Lach et al. 2006]. When considering the point cloud or R_A, it was noted that even in fairly densely forested regions, many points were able to penetrate the foliage. In cases where the building regions had few vertical transitions, their data structure had significantly fewer low points. A binary image was created in which all locations were initially set to 0. Then, cells where all of the points inside the cell were greater than 2m above the NDEM were set to 1. Each region of this image was then identified through a connected-components analysis as described in [Gonzalez and Woods 2002]. A binary morphological opening was then performed, using a kernel sized to preserve just a few pixels from the smallest building in the scene. This effectively removed all tree

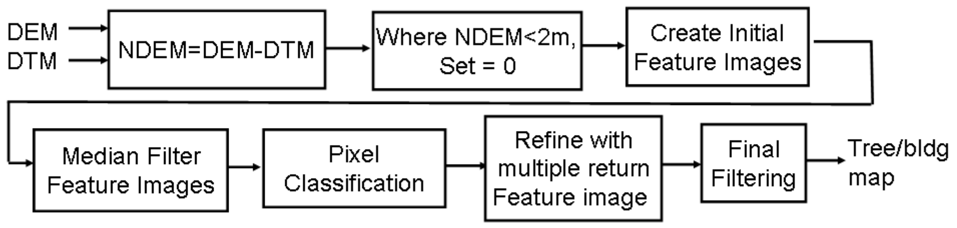

DEM
DTM
NDEM=DEM-DTM
Where NDEM<2m, Set = 0
Create Initial Feature Images
Median Filter Feature Images
Pixel Classification
Refine with multiple return Feature image
Final Filtering
Tree/bldg map

Figure 4.9: Approach for segmenting buildings from trees.

regions. Objects in the connected-components image that contained pixels in common with the opened image were then specified as buildings, and the remaining objects were classified as trees.

Although this method proved to be adequate in identifying building and tree locations in many scenes, trees adjacent to buildings tended to be classified as buildings as well. Additionally, in the event of actual low points inside building boundaries, or internal vertical transitions causing multiple returns, entire buildings had the potential to be omitted. To this end, a new approach was created that is considerably more robust. It uses a methodology similar to that first presented in [Persson 2001], although the features used and the specifics in implementation are somewhat different. This modified approach is depicted in Figure 4.9.

In the updated approach, we make use of several of the previously defined raster images to create a feature vector at each pixel location. These feature vectors are then segmented using standard binary-class pattern recognition techniques. Additional processing is then performed on the resultant classification image in order to improve the results.

The first few features make use of the fact that local height variations occur much more rapidly in vegetative regions than on building roof structures. Entropy, which may be defined as

$$H = -\sum_{i}(p_i)\ln(p_i), \tag{4.3}$$

where p is the probability of a certain pixel value i occurring, which may be estimated from a histogram of values. We may create a local entropy image by computing the entropy within a small window (5×5 was used in this work) around each pixel, and assigning that value to the corresponding pixel in the entropy image. Symmetric padding is used on the edges so that the final entropy image is of the same size as the original image. In a similar manner, additional texture images are created by considering the variance of pixel values, maximum pixel brightness difference, maximum slope difference, and Laplacian (see Equation 4.1). These features tend to be significantly higher in tree regions than in building regions, due to the rapid height fluctuations. However, the values around building edges and vertical transitions are also usually high.

Similar features are also created from the intensity images. Intensity-based entropy, maximum pixel brightness difference, variance, and Laplacian are used in this work. An additional feature image is also created at this time by counting number of non-first return points there are within the area defined by each cell. As noted above, this feature will also tend to be larger in vegetative regions than it is over building structures.

As noted in [Persson 2001], we may amplify the effect of each of the window-based measures by using multiple images to populate the local window when the features are being computed. By using the R_{MX} (or I_{MX} image value for the central window point and R_{MN} (or I_{MN} values for the surrounding points, we effectively enhance the local variations in height. Another idea borrowed from this source is to perform a median filtering on each texture image in order to reduce unwanted noise effects as well as to degrade the strong texture values due to height variations at building edges.

The feature images are then combined such that each pixel location has a corresponding feature vector associated with it, where all features except the multiple return feature are used. A binary classifier is then employed to separate the pixels into two classes, based on the feature vectors. Due to the nature of the data, this is an inherently difficult problem. Using the defined texture images, building and tree points are not fully separable in the feature space. Additionally, the spread of the data is somewhat large relative to the distance between class means, so simple techniques such as minimum distance to the class mean tend to work poorly.

Many other relevant classifiers exist in the literature, and may be found in any text on pattern recognition (see [Duda et al. 2000], for example), including support vector machines, neural networks, and genetic algorithms. The Bayesian approach, which was used by Persson in his work, is optimal in terms of yielding the lowest probability of classification error if certain statistics are known. If the distribution of the data within each class may be assumed to be Gaussian, the Bayesian approach takes on the form of the Gaussian Maximum Likelihood (GML) detector. Although the classes in this instance are not strictly Gaussian (since at a minimum the individual feature values are not negative), they are close enough for this approach to work effectively.

As developed in [Schott 2007], we may begin by defining the a posteriori probability of a pixel with feature vector $\mathbf{x}$ belonging to class ω_i as

$$p(\omega_i|\mathbf{x}) = \frac{p(\omega_i)p(\mathbf{x}|\omega_i)}{p(\mathbf{x})}, \tag{4.4}$$

If the conditional probabilities are normally distributed, we may express them as

$$p(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{\frac{k}{2}}|\mathrm{S}_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{m}_i)^T\mathrm{S}_i^{-1}(\mathbf{x}-\mathbf{m}_i)}, \tag{4.5}$$

where k is the number of features and $\mathbf{m}_i$ and S_i are the mean vector and covariance matrix for ω_i. Combining these equations yields

$$p(\omega_i|\mathbf{x}) = \frac{p(\omega_i)}{p(\mathbf{x})(2\pi)^{\frac{k}{2}}|\mathrm{S}_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{m}_i)^T\mathrm{S}_i^{-1}(\mathbf{x}-\mathbf{m}_i)}. \tag{4.6}$$

Since the decision as to which class produces a larger value of $p(i|\mathbf{x})$ does not depend on $p(\mathbf{x})$, this term may be ignored. Taking the natural logarithm of the remaining terms and removing constant terms does not change the rank ordering, so we may define the discriminant function as

$$d_i(\mathbf{x}) = \ln(P(\omega_i)) - \frac{1}{2}\ln|\mathrm{S}_i| - \frac{1}{2}(\mathbf{x}-\mathbf{m}_i)^T\mathrm{S}_i^{-1}(\mathbf{x}-\mathbf{m}_i). \tag{4.7}$$

The class producing the larger value of $d_i(\mathbf{x})$ is the class to which $\mathbf{x}$ is assigned.

It is apparent that this approach requires training data to be provided by the user. In this research, such a region was initially input manually by right- or left- clicking on range image points. However, we have also achieved success with a more automated approach in regions where there is an abundance of individual trees. Although all building points due not have an entropy value below a given threshold, if the maximum entropy value in a given region is below a certain threshold, such a region is very likely to represent a building. Additionally, regions with very small areas are likely to represent vegetation. A connected components analysis is first done to assign a unique label to each contiguous region. Small regions (under 100m^2) are then temporarily assigned to the tree class. Next, the regions are eroded by a 5×5 structuring element to remove the region boundaries. If a formerly-labelled tree region is completely removed by this process, then it is ignored. If the eroded tree region has an average entropy of greater than a given threshold, then the entire original region is kept as training data for the tree class. Each additional eroded region is then checked to determine the density of non-first return points present in that region. If the non-first return point density is below a second threshold, and the average entropy is below the first threshold, then the entire original region is kept as training data for the building class.

Of course, training data may also be provided in advance by analyzing other imagery containing similar features and with the same grid spacing. This has been shown to be an effective means of removing the man-in-the-loop requirement. In the simplest implementation of this concept, we were able to achieve useable results in many images by simply thresholding the local entropy value at 0.8 (grid spacing was 0.35m) and ignoring other feature values altogether.

Once an initial classification of the points has been made, refinements may still need to be made. These typically occur around the boundary of objects. To address this, another connected components analysis is performed to assign a unique label to each contiguous region. The regions are then eroded by a 5×5 structuring element to remove the region boundaries. These pixels are then re-filled using the the nearest object point for the assignment. After this is performed, the entire image is median filtered with a 3×3 structuring element, and the result is re-masked by the original high-point mask.

The output from this processing stage is either an indexed image or a classification value associated with each original non-rasterized data point.

4.3 Coarse Image Registration

When working with multi-modal imagery, the proper registration of all data sources is critical if we wish to fully exploit the complementary nature of the data. Although in certain fields, full registration also requires an adjustment of image intensity values, in this research registration's sole aim is to determine the best spatial mapping between two (or more) images. That is, the registration represents the transform which brings two images into alignment with each other in such a way that features in one image may be related to their corresponding features in the other.

In this research, the registration is performed in two separate stages. The initial registration is a coarse alignment of the spectral and nadir-viewing frame-array imagery with the lidar data. This registration does not take into account the full projective geometry, and uses only approximate values for sensor exterior orientation. It is used for multi-pixel tasks such as improved isolation of building structures, identifying tree-regions, defining coarse material maps and assigning texture. Once initial building models are retrieved, a finer registration is be performed using a full photogrammetric solution in order to refine the location of object models. This second registration process is discussed in Section 4.6.5.

It should be emphasized that while a technique for automating a portion of the registration process is included in this dissertation, there are currently no fully-automated techniques that work well in all situations. In this work, accuracy of the registration is deemed to be more important than autonomy; therefore manual intervention will be used as required in order to produce the best results possible. This includes a validation and adjustment of every registration result,

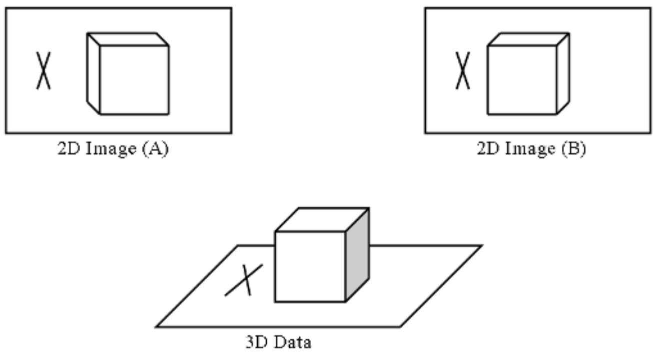


Figure 4.10: Two perspective images of a single 3D object.

even those that are performed autonomously.

4.3.1 Fundamental Concepts

Traditionally, images have been registered through a process that begins with a manual selection of matching points. The point correspondences are then used to determine the parameters of a mapping function which 'best' transforms the selected *sample* image points to those of the *reference* image. This same transform is subsequently used on all other sample image points in order to bring them into the geometric frame of the reference image. If a registered version of the sample image is needed, a re-sampling is typically required; however, in many cases all that is needed is knowledge of the transformation itself, and features such as lines or areas in the sample image may be transformed without need for an entirely new image.

The standard transform models that are usually assumed include the Euclidean, similarity, affine, and projective warpings as well as simple polynomial mapping functions of the form

$$\begin{aligned} x_t &= a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 \\ y_t &= b_0 + b_1 y + b_2 x + b_3 yx + b_4 x^2. \end{aligned} \tag{4.8}$$

Although many introductory courses covering image registration techniques end the discussion at this point, a simple example should illustrate that there are many images that can not be fully registered by this technique. Consider the example shown in Figure 4.10. If ground-level point correspondences are chosen, the top of the boxes will be out of alignment if we use any of the previously-mentioned transforms. Similarly, if the top four corners of the box are selected as control points, features along the ground plane will remain mis-registered. This effect is actually quite simple to explain, although it its often overlooked. Assuming a central projection camera model, images of a planar surface taken at two different locations and orientations may be perfectly registered via a projective transform. However, if the scene is not restricted to lying in a plane, the relationship between the images can not be described by a simple homography. In these cases, knowledge of the three dimensional scene geometry is required in order to fully map all pixels in one image to the other. Because of this requirement, in this research we use an estimate of the scene geometry to re-project the 3D scene content vertically onto a horizontal plane, thereby producing an orthographic view of each image. These orthoimages may then be further registered using one of the standard transform models.

An orthographic image (or orthoimage) is one in which all effects of image tilt and relief displacement have been removed, as is seen in Figure 4.11. In the most common usage of the term, relief displacement is removed by taking only a bare-earth DTM into account. This rectifies most of the ground coordinates, but fails to properly display buildings in an orthographic sense. When the full DEM (including building structures, trees, and other non-terrain objects) is used instead, the resultant projection is termed *truly* orthographic, and the tops of buildings now appear in their correct locations. Once all images are converted to orthoimages, relevant features are identified, and a final polynomial transform using control points is applied to fine-tune the registration. Specific methods for orthorectifying frame-array and linescanner imagery are detailed in the following sections.

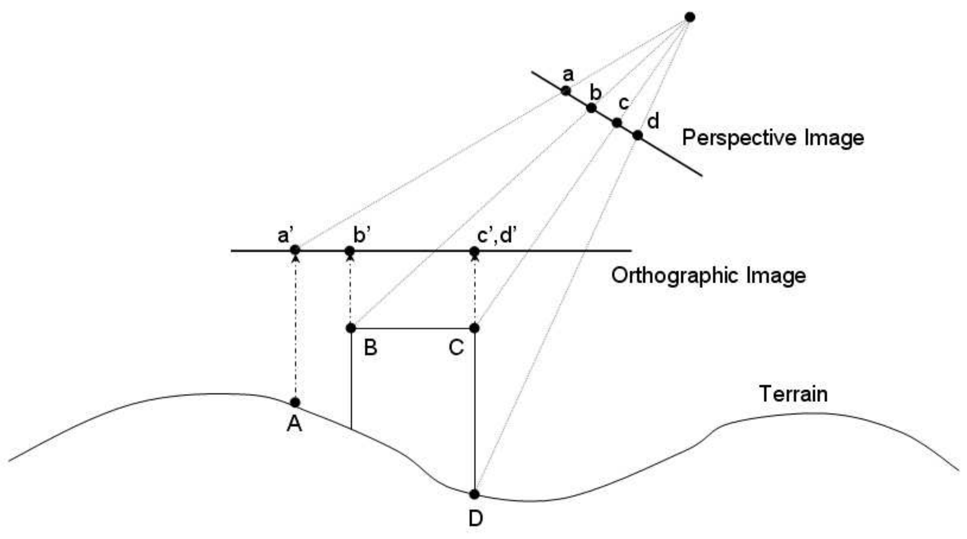


Figure 4.11: Illustration of an orthographic projection.

Thus far, we have only covered the case of registering two 2D images of the same scene. Provided we can produce orthographic views from our images and that we can also find an appropriate number of point correspondences, this technique usually works fairly well. However, the question remains as to what is desired when we wish to register a 3D point cloud obtained by a lidar sensor with a standard 2D image. Fortunately, the answer is actually quite simple. Given that by registration we mean the transformation that brings a sample image (or dataset) into the space of a reference image, we may see that it is the camera model that accomplishes this for 3D data. Stated another way, by specifying the EOP and IOP of the camera needed to image the 3D data such that it appears at the image coordinates of the reference image, we have fully identified the transform required to perform point-to-point correspondences. In compact notation, these parameters, and hence the registration, are fully captured in the camera matrix P (reference Section 3.1.3.2). Although the EOP need to be defined in relation to the lidar data, since many airborne-collected lidar point sets are stored using UTM coordinates, the relative orientation is often the same as the world-coordinate orientation. This concept is illustrated by Figure 4.12.

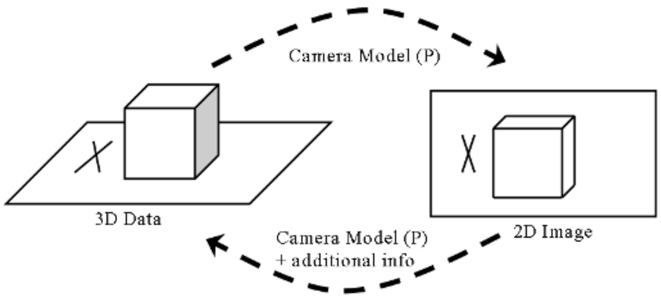


Figure 4.12: 3D to 2D registration.

One persisting difficulty is that this geometric mapping is only unambiguous in the forward direction. That is, given a lidar point cloud, we may uniquely determine the location of each point in the image plane given P. However, given the image and P, we can not uniquely specify the location of the corresponding points in 3D space. Each image point represents a ray from that point through the camera center. However, without any additional geometric information the actual position of the point along this ray is ambiguous. Only through the addition of constraints may we fully express the transformation in the reverse direction. This approach will be used in Section 4.6.6, where we determine the 3D feature (we use lines instead of points) location by finding the intersection of the back-projected feature with a previously-determined plane in object space.

4.3.2 Orthorectification of Frame-Array Imagery

Once the calibrated interior distorting effects have been compensated for, frame-array images depict a pinhole camera perspective view of the world to first order. In the general case, such an image exhibits displacements due to tilt of the camera as well as vertical relief in the object space. Even when the camera tilt is reduced to near-zero angles (as is often the case for aerial photography), objects placed at the same (x, y) coordinate but at different heights will be projected to different positions in the resultant image, unless the points lie along the camera's central axis. As a result, objects at higher elevations appear larger in the image, and the scale across the image is not constant. This effect is eliminated by converting the image to an orthographic projection; that is, one where the rays are parallel rather than passing through a projective center. Such an image has all perspective effects removed, and thereby maintains a constant scale throughout the image.

Orthorectification is done through a reprojection of the image points back onto a model of the object space terrain. As such, a detailed DEM is required in order to produce accurate results. This has typically been done using photogrammetric techniques, although recently, lidar-produced DEMs are becoming increasingly common (Section 4.1 of this report discusses DTMs and DEMs produced from lidar data). The reprojection of image space rays may be done in one of two ways; using either forward or backward projection techniques.

In the forward projection, the source image is directly projected back onto the model of the terrain and the (x, y) coordinates of intersection of each projected ray and the DEM are recorded. However, since these coordinates are irregularly spaced, they must be interpolated onto a rasterized array of points in order to produce an actual digital image. In backward projection, the spatial location of a desired output pixel is projected backwards towards the source image. This replaces the previously described interpolation with an interpolation in image space, which is typically easier to implement. See [Nielsen 2004] for additional information on these techniques.

There are many commercial and freeware applications available for performing this frame-array orthorectification process. Example utilities include OSSIM (the flagship freeware GIS utility) [RadiantBlue Technologies Inc. 2006], GeoPixel-Ortho [Ahn et al. 2001], and VirtuoZo [Supresoft Inc. 2006], and studies have been conducted comparing these and many other products (see [Baltsavias and Kaser 1998], for example). For the method implemented in this research, Leica Geosystems' ERDAS Imagine software [Leica Geosystems 2006] was selected to provide orthographic rectification of the frame-array imagery.

4.3.3 Georectification of Linescanner Imagery

While most airborne single-band and color imagery is collected using frame-array architectures, multi-spectral, hyperspectral and lidar datasets are often collected with linescanning sensors. The non-instantaneous image collection inherent in this design introduces additional difficulties when orthorectifying data collected by these imagers. Two primary methodologies exist for georeferencing linescanned data: non-parametric and parametric.

Non-parametric techniques, also known as image warping, often use a least-squares fitting polynomial or bi-variate mapping function to resample the airborne image from its collected array format to a more desirable raster. A common polynomial used by many authors is of the form given in Equation 4.8, but several sources have noted that additional terms may yield improved results when dealing with linescanned images. As such, a mapping of the form

$$\begin{aligned} x_t &= a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2 \\ y_t &= b_0 + b_1 y + b_2 x + b_3 yx + b_4 y^2 + b_5 x^2, \end{aligned} \tag{4.9}$$

is often preferable, especially when applying this to linescanner imagery that has already undergone an initial georectifying projection [Zhang et al. 1994]. In these models, x_t and y_t refer to pixel coordinates in the original (sample) image frame, while x and y represent the new location in the transformed (reference) image. The a_n and b_n parameters are the constants that define the transformation.

In order to use this model, homologous features must be identified in each image. For the standard case of corresponding points, the (x, y) and (x_t, y_t) values of the matching set are used to solve for the constant parameters using traditional least squares regression. Once the geometric relation between the images has been defined, we then determine a greyscale value for each (x, y) location in the transformed image. This is typically done through a backwards transform, where the (x, y) pixel is transformed to the original image space using the inverse of the defined model.

The correct greyscale value for this pixel is then determined through an appropriate interpolation using image data from the original image, and this interpolated greyscale value is inserted in the transformed image at pixel (x, y). Such a technique avoids interpolation in the transformed space, which is often difficult because neighborhood pixel values may not yet be defined. A more complete description of this polynomial modeling technique may be found in [Schott 2007] and [Goshtasby 1988].

Non-parametric mapping functions may be computed and applied globally to the entire image, or locally to particular subregions within the image. As noted in [Schott 2007], for each subregion, the input data should cover the entire solution space for which it will be applied. If the model is employed outside the point area used to define the transform, severe distortions may arise. This is particularly true when higher-order models are used.

Although such non-parametric techniques are well documented in the literature (see [Wiemker 1996] and [Gottesfeld-Brown 1992] for additional examples), it is generally agreed upon that they produce less than satisfying results in many instances [Novak 1992] and [Breuer and Albertz 1996]. Furthermore, in the cases where non-parametric techniques are employed, a large number of ground control points are typically required, and the image-to-image mapping usually requires manual intervention [Pope and Scarpace 2000].

Parametric techniques model the 3D geometry between the aircraft, the scanning instrument, and the terrain, and effectively perform a ray-tracing solution that maps image intensities to points on the ground. However, the resulting ground positions are still in a temporary coordinate frame. An additional re-gridding (interpolation) is required to convert these projected intensities into a rasterized digital image, and in many cases a final polynomial transform using control points is required to fine-tune the result ([Zhang et al. 1994] and [Breuer and Albertz 1996]). Although in most cases, parametric techniques provide a superior solution, use of these methods are based on the assumption that highly accurate sensor position and orientation data are available. Loss of GPS reception or INS equipment malfunctions may preclude the use of this approach in some cases.

To illustrate the use of a projection technique as derived for this research, we will briefly consider a new method for the orthorectification of MISI imagery. As noted in Section 2.1, MISI makes use of two fiber optics placed slightly off-axis in the focal plane. To take this effect into account, we shall treat the sensor model's contribution to the aircraft-leaving projection vector in our discussion.

The basic object-space (world) and camera-space (aircraft) coordinate frames used in our development are shown in Figure 4.13. The aircraft is located at position $\mathbf{p}_a$ relative to a defined (usually local) origin, and it is re-oriented by *relative* rotations along the three aircraft axes. In practice, the rotation angles are recorded from the chosen navigation hardware according to an active $RPY_{relative}$ standard. However, since many legacy datasets did not record aircraft yaw from the inertial sensors, aircraft heading derived from successive positions must be used as at least an initial approximation of the correct rotation about the $\mathbf{z}_a$ axis. In order to make use of this however, this rotation must be applied first, and the axis convention should be of a relative nature. As such, the technique derived in Section A.2 is used to convert the given $RPY_{relative}$ roll, pitch, and heading values into corresponding $YPR_{relative}$ angles. By doing the rotations in this manner, the final projection of the $\mathbf{x}_a$ axis onto the (x, y) plane will be in the direction of the aircraft heading.

It should also be noted that since the roll and pitch angles vary slowly with respect to most of the other parameters, noise may be removed from these measurements by using an averaged angle value for each rotation. Although an ideal way to do this would be to compute this average for the i^{th} pixel according to

$$angle(i) = \sum_{j=i-N/2}^{i+N/2} angle(j), \tag{4.10}$$

in practice we have found that we may simply average the roll or pitch data for each scan line and apply those values for the entire line.

Since the image of a linescanner is collected one pixel at a time, a description of the image

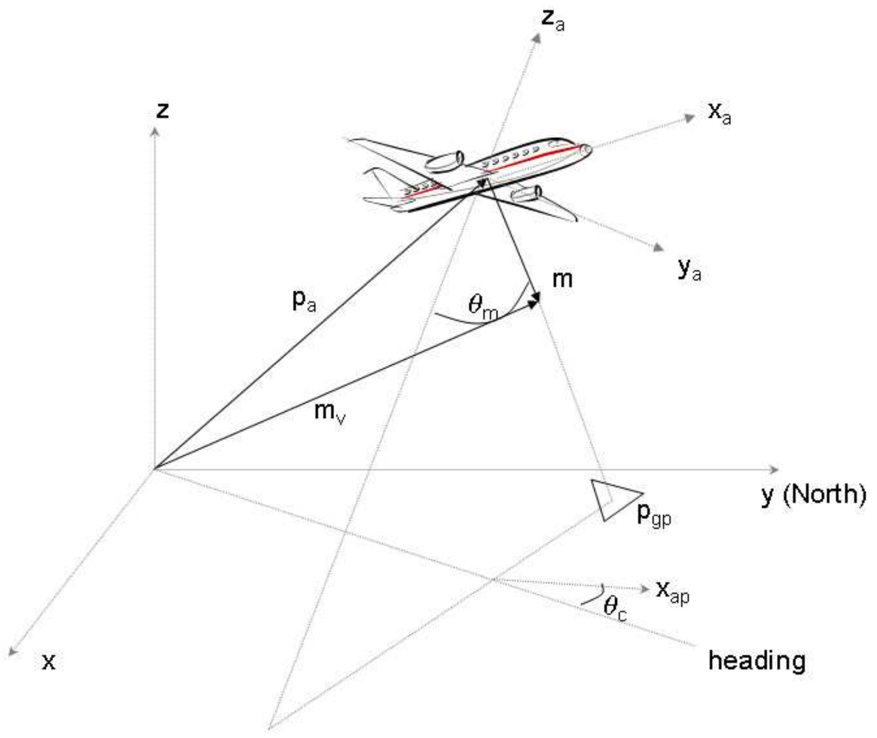


Figure 4.13: Object-space (world) and aircraft coordinate frames.

space in relation to the aircraft coordinate frame is not needed. However, an additional diagram describing the internal sensor geometry model in terms of the aircraft axes is warranted. This is depicted in Figure 4.14.

In this diagram, we see the internal components of the linescanner design. The center of the scan mirror is defined as the origin, and a fold mirror is located in the negative $\mathbf{x}_a$ direction. Inbound radiation reflects off these surfaces and through the telescope optics (reflective in reality, but modeled here as a single ideal refractive element) and into the fiber optics, which are located slightly off the optical axis at the focal plane, one of which is at the location $\mathbf{p}_f$. In order to arrive at the parametric solution, we will backproject rays from the fiber optics through the center of the optical system and calculate where they intersect the ground. Note that lever-arm effects are ignored in this discussion since the inertial sensors are in close proximity to this mirror and the

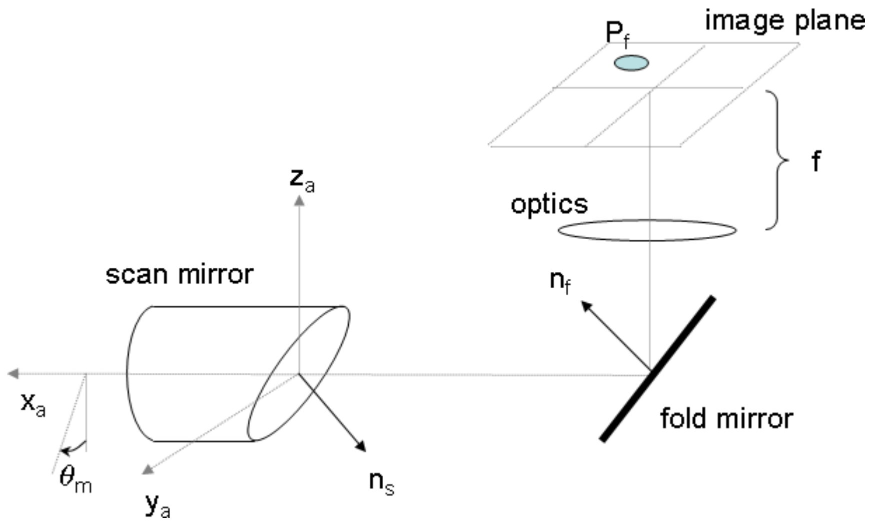


Figure 4.14: MISI optical path geometry model.

positional offset of the GPS antenna produces only a small error. However, these will be taken into account in the final georectification algorithm.

If we are willing to restrict our analysis to the angular effects of this projection (and ignore the slight offset induced by projected rays not reflecting off the aircraft frame's origin) we may replace this model by the simpler representation shown in Figure 4.15. This model assumes that the backprojected ray from each fiber optic intersects the scan mirror at its center, and that this ray may be completely specified by the angles at which it approaches this surface. Since all optics behind the scan mirror are designed to be stationary, this is a reasonable assumption, even in the general case where the two mirror sources are not properly aligned. To simplify the notation, the fiber optic from $\mathbf{p}_f$ is now assumed to be at $\mathbf{p}'_f$, such that the angles of the reflected ray $\mathbf{m}_a$ are the same as they were in the model of Figure 4.14.

Once the sensor has been calibrated, the position $\mathbf{p}'_f$ is known for each fiber. Therefore, at the time a particular pixel's digital count is recorded, the scan angle θ_m in combination with the aircraft's position and orientation define the object-space ray $\mathbf{m}$ which points to the location on the ground at which the sensor was looking when the inbound radiance was received. If we assume

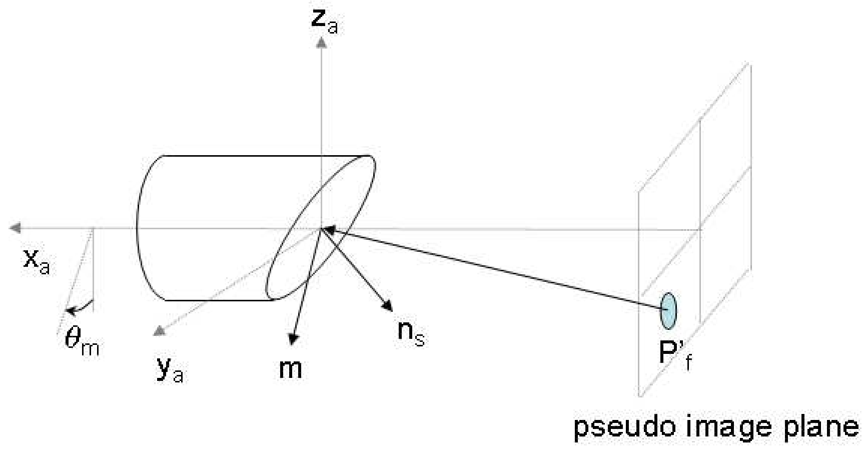


Figure 4.15: Simplified geometrical model of MISI's optical path.

the scan mirror was aligned properly, the point $\mathbf{m}_V$ (and hence the ray $\mathbf{m}$) may be computed as follows.

First, the normal vector of the scan mirror is seen to be

$$\mathbf{n}_s = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ \sin\theta_m \\ -\cos\theta_m \end{bmatrix}. \tag{4.11}$$

Next, the ray from $\mathbf{p}_f^{'}$ to the aircraft origin may be reflected off the scan mirror by recasting $\mathbf{p}_f^{'}$ in homogeneous coordinates, and applying Equation A.22 relative to $\mathbf{n}_s$ and using an angle of π radians. This basically computes the reflection by rotating the incident vector around the mirror's normal. This yields the outbound unit vector $\mathbf{m}_a$ in aircraft-based coordinates.

We may view the vector $\mathbf{m}_a$ as a point in 3D space, and specify a ray passing from the origin through this point. This ray may then be transformed to object space by taking the aircraft position and orientation into account. If the orientation is specified by the rotation matrix R, then the point $\mathbf{m}_a$ transforms to $\mathbf{m}_V$ via

$$\mathbf{m}_V = \mathrm{R}\mathbf{m}_a + \mathbf{p}_a, \tag{4.12}$$

where $\mathbf{m}_V$ is the endpoint of a ray from $\mathbf{p}_a$ in the direction of interest. Of course, this may also be computed linearly using homogeneous coordinates.

We next assume a planimetric model of the ground, and see if $\mathbf{m}$ is parallel with this plane. This is done by specifying the plane with a normal vector $\mathbf{g}_n$ and a point in the plane $\mathbf{g}_p$. Then, if and only if $\mathbf{g}_n \cdot \mathbf{m} = 0$ is the line containing $\mathbf{m}$ not going to intersect the planar ground model. In practice, this step may usually be omitted, since the aircraft roll angle is typically less than 10 degrees, and image data is not recorded at angles greater than $\pm\frac{\pi}{4}$ radians.

We next project the line containing $\mathbf{m}$ to the ground plane and determine the point of intersection, $\mathbf{p}_{GP}$. This may be done according to

$$\mathrm{S}_{gp} = \frac{\mathbf{g}_n \cdot (\mathbf{g}_p - \mathbf{p}_a)}{\mathbf{g}_n \cdot (\mathbf{m} - \mathbf{p}_a)} \tag{4.13}$$

and subsequently applying

$$\mathbf{p}_{gp} = \mathbf{p}_a + \mathrm{S}_{gp}(\mathbf{m} - \mathbf{p}_a). \tag{4.14}$$

It should be noted that if S_{gp} is negative, $\mathbf{m}$ points upward, and it is an extension of the backside of the ray of interest that intersects the ground. In many cases involving aerial imagery, this indicates an error, and the corresponding pixel should be flagged as such.

Once the point of intersection with the planar model of the ground has been determined, we need to refine our solution by taking the full DEM into account. Assuming the DEM is extracted per the discussion in Section 4.1, the ground model will ultimately be represented by a triangular facetization. Therefore, we need to determine which triangular facets are intersected by the projected $\mathbf{m}$ (see Section A.3) and then select the one that is intersected first from these candidate facets. Facets are examined based on their proximity to $\mathbf{p}_{gp}$, and currently an exhaustive search is being used. The point of intersection, $\mathbf{p}_g$ of $\mathbf{m}$ with the closest candidate facet is then computed using Equations 4.13 and 4.14.

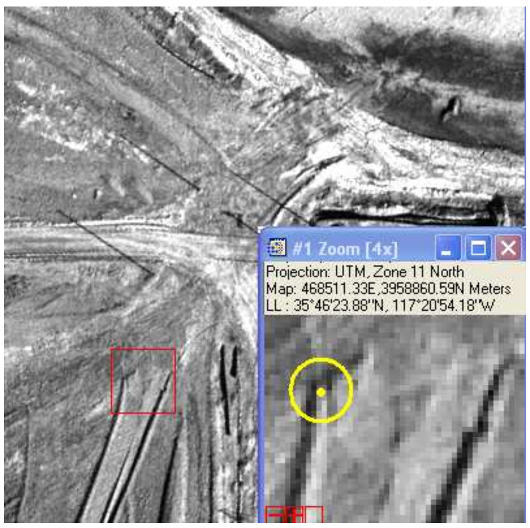
Once all image points have been projected to their ground locations, an interim orthoimage may be produced by defining a grid in object-space, and interpolating to this grid the greyscales associated with the projected locations. In many cases, a nearest neighbor interpolation is used, so as to not distort the spectral nature of the image. It should be noted that band data from each fiber optic will be projected separately, and a separate grid interpolation should be performed for each fiber. The initial orthoimage may be refined significantly in many cases if surveyed control points are available. By applying a polynomial transformation model in the form of Equation 4.9 the final orthoimage is obtained.

A similar parametric approach is often used to rectify linescanned lidar data as well. In this case, however, **m** is used in conjunction with range information, and the resultant points are computed in 3D space. As such, a DEM is not required. However, a final polynomial fit may indeed be employed by taking known ground points in 3D space (usually surveyed with a precise GPS receiver) and forcing a facetized version of the lidar point data to intersect these control points.

4.3.4 Automating the Final Transformation

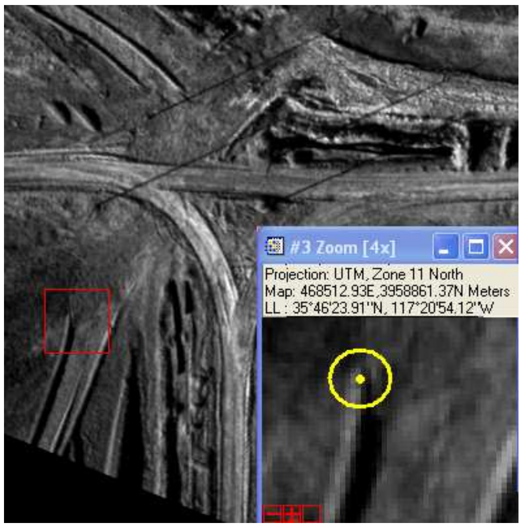
Once all of the images (frame-array, linescanned multi- and hyperspectral and lidar) have been remapped as 2D orthographic perspective projections, a final least squares polynomial or projective mapping may be required to remove any remaining geometrical discrepancies. An illustration of why this is needed is given below in Figure 4.16. These images are both from the red channel of WASP's RGB camera. The left image was taken at 1100h local, while the right image was flown later in the day at 1700h. Using the EOP provided by the Applanix navigational system as well as a USGS-provided DTM[1], ERDAS Imagine produced the orthoimages shown in the figure. By zooming in on a prominent terrain feature, we are clearly able to see a mis-registration of approximately 1.75m (15 pixels). In this work, we use an image warping technique to remove this

[1]USGS terms the product a DEM, but the data actually represents the bare earth



Projection: UTM, Zone 11 North
Map: 468511.33E,3958860.59N Meters
LL : 35°46'23.88"N, 117°20'54.18"W

(a)



Projection: UTM, Zone 11 North
Map: 468512.93E,3958861.37N Meters
LL : 35°46'23.91"N, 117°20'54.12"W

(b)

Figure 4.16: Mis-registration of two ortho-images: (a) 1100h Image, (b) 1700h Image. Both images are WASP red channel, orthorectified using ERDAS Imagine.

effect.

Before discussing the final stage in the coarse image registration process, it is worthwhile to note that in addition to the geometry underlying the transform used for the registration, we may classify individual registration techniques according to whether they are predominantly manual or automated. In manual approaches, feature correspondences (points or lines) are selected by hand, and the location of these correspondences permits a direct solution for the transform parameters. In automated systems, a different approach is required. One approach attempts to autonomously find correspondences, then proceed in a manner similar to the manual approach. A feature detector (a corner detector is preferable) may be used to detect features in both the sample and reference image. Next, an attempt at finding actual feature correspondences is made. This is done through a robust (RANSAC-based) approach, where a minimal number (in the sense of defining the transform) of matches are hypothesized through random matching, and the transform arising from that particular feature set is applied to the sample image. The number of feature points in the transformed sample at the same location as feature points in the reference image is then recorded for that particular set of hypothesized correspondences. A new set of correspondences is then randomly chosen and the process performed again. The transform yielding the largest number of correct feature-feature matches is the one selected to perform the registration of the entire image. This approach is very similar to the one presented in [Hartley and Zisserman 2003], except those authors use the point correspondences to define the fundamental matrix instead of image warping parameters.

A notably different approach for automation is also possible. In this alternate process, the sample image is first warped according to an initial transform, and the transformed image is compared to the reference image through the use of a merit function. The parameters of the transform are then updated, and a new warped image is obtained and compared to the reference. This process is iterated until a parameter set yielding the best value for the metric function is obtained. In some cases, the entire transformed image is used to define the merit function; in others, only pixels corresponding to autonomously-selected features (such as the output of an edge detector) are used.

In this research, we use three different approaches for performing the final registration used to bring the ortho-rectified images into alignment. The first approach is a simple manual point-selection technique, where homologous pixels are selected in a pair of images. We then use these point to define several transforms (typically affine, projective and polynomial), and the best one is used for the registration. Although this is the most labor-intensive technique, it usually provides accurate results.

The second approach uses the first paradigm defined above for automated registration. In [Yang et al. 2007], the authors present their Generalized, Dual-Bootstrap Iterative Closest Point (GDB-ICP) process. This technique is an end-to-end set of algorithms that extracts edge and corner features across both images. However, unlike most other approaches using this general structure, the GDB-ICP does not initially perform a global transformation. Rather, initial matches are used to define transforms that are only valid in small regions surrounding the feature points. These regions are then grown while the transform parameters are updated through multiple iterations. When needed, a higher-order transformational model may also be autonomously inserted in order to produce a better fit. After the final transformation is defined, the approach transitions to a decision algorithm whereby a decision is made as to whether or not the transform indeed provides correct registration of the two images.

This algorithm is available as a DOS executable directly from [Stewart 2008]. It performs very well on images where the phenomenology is similar and perspective effects are benign, even if the overlap between the images is very small (less than 10%) and there are large scale differences. As such, it is often the first choice for registering different bands of the WASP-lite sensor, or WASP-lite to WASP RGB images. This algorithm does fail in the presence of even moderate perspective variations, but in many cases these projective effects are reduced through the initial ortho-rectification process. A second difficulty arises when the images are obtained using sensors of differing modalities. That is, the algorithm fails when attempting to register a rasterized lidar intensity image to a single band from a nadir-viewing WASP-lite image set.

In order to autonomously handle the various phenomenologies encountered in multi-modal data sets, a third approach is required. Although a portion of what follows is novel and was created for this research, most of the technique was developed by Xiaofeng Fan for work on a complimentary project. His full documentation, [Fan 2009] will be available in the near future.

For the coarse lidar to 2D frame-image registration, we must first construct a composite rasterized lidar image (CRLI) from the lidar height and intensity data. This image is created by using the intensity image for locations within 2m of the DTM, and a scaled range image elsewhere. Rasterized range, intensity and CLRI images from a portion of the RIT campus are shown in Figure 4.17.

Once the CRLI is obtained, we may then register it to overhead imagery via a robust multimodal registration technique. Unfortunately, since the CLRI and frame array images recorded differing phenomenologies, using cross-correlation as the quality metric often produces sub-par results, as corresponding features may have differing radiometric properties across the bandwidths

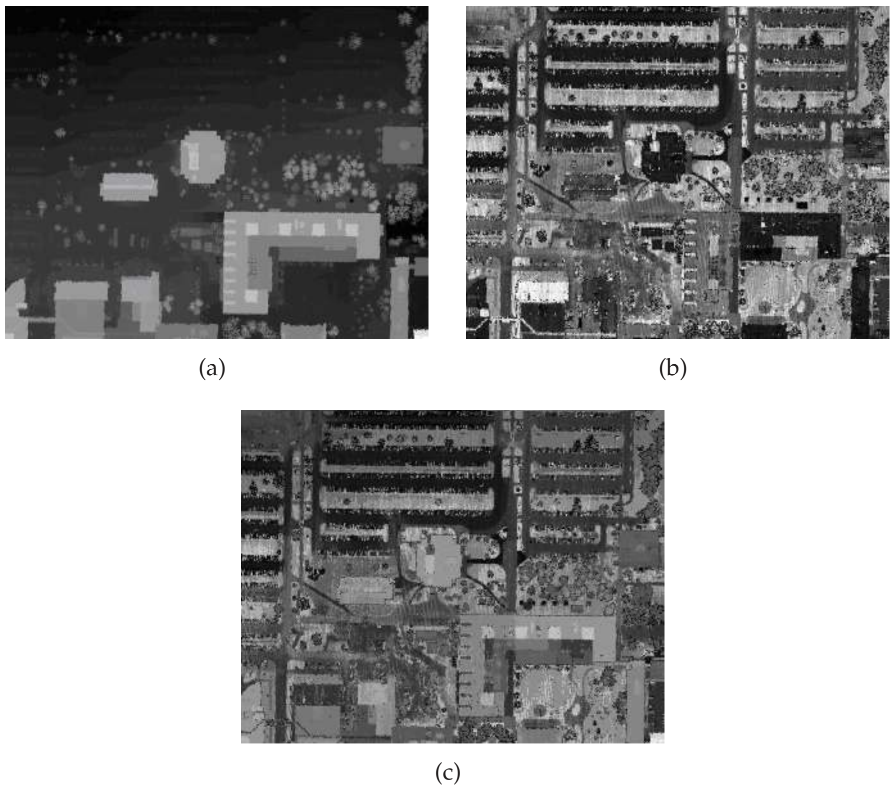


Figure 4.17: Constructing the CRLI : (a) Range image, (b) Intensity image, (c) CRLI image.

of the sensors being used. In these instances, a metrics such as maximization of mutual information (MMI) [Viola and Wells 1995] and phase correlation [Humblot et al. 2005] have been found to be preferable. We consider a modified MMI-based approach here.

The entropy of the random variable X with probability density $p(X)$ is defined as the expectation of the negative logarithm of the probability density. That is

$$H(X) = -E_X[\ln(p(X))] = -\sum_{x_i} p(x_i)\ln(p(x_i)). \tag{4.15}$$

We now describe a pixel in the reference image as $r(X)$ where r is the pixel value and X is the location. The pixel in the sample image at the same location is similarly denoted $s(X)$. The pixel at location X in the transformed sample image therefore has the value $s(\mathrm{T}(X))$, where T represents the transformation.

The similarity between $r(X)$ and $s(\mathrm{T}(X))$ may then be described according to their mutual information

$$\begin{aligned} I(r(X), s(\mathrm{T}(x))) &= H(r(X)) - H(r(X)|s(\mathrm{T}(X))) \\ &= H(r(X)) + H(s(\mathrm{T}(X))) - H(r(X), s(\mathrm{T}(X))). \end{aligned} \tag{4.16}$$

Here we see that this definition conveys the idea that mutual information is the sum of the entropy of each random variable, minus the joint entropy of the two variables. By selecting the transform that maximizing this metric, we effectively find the warping function that properly registers the images.

The key contribution in [Fan et al. 2005] is finding that this approach may be improved by first extracting edge and corner features in the imagery using a Harris-based [Harris and Stephens 1988] technique, then performing the MMI only on these featured pixels. Additionally, instead of using the actual pixel values in the algorithm, non-feature pixels are replaced with the value 0, edge pixels are assigned a value of 1, and corners are given a pixel value of 2. This modified MMI technique is termed Feature-Enhanced MMI (FE-MMI), and it has proven to be successful in registering lidar, visible and long-wave infrared frame array imagery, linescanned imagery and maps with each other. A detailed description of the FE-MMI technique may be found in [Fan et al. 2007].

It is worthwhile to note a final detail regarding the coarse registration process. Although the final transformation may be obtained and applied locally to sub-regions of the imagery, it has been found that when the initial ortho-rectification has been completed fairly accurately (aircraft orientation artifacts removed to within approximately 10-15 pixels), the final polynomial mapping may be applied globally with no noticeable degradation. Also, the preferred method being used in this research is to map all sources to the most accurate image source, which in general is the lidar data.

4.4 Refined Building/Tree Segmentation

The spectral reflectance of plants has a distinct signature, and therefore multispectral and hyperspectral classification techniques excel at differentiating vegetation from most man-made objects. Through the visible light region, vegetation has a relatively low reflectance on the order of 5% up through approximately 680nm. However, there is an abrupt increase of reflectance in the near-infrared, where a peak of 50% is often recorded at about 730nm. This rapid variation in reflectance is commonly termed the *red edge*, and this distinctive signature may be used to easily distinguish trees from most man-made objects.

Through the use of the popular normalized difference vegetation index (NDVI), we are able to use a simple function of the digital counts for two bands to determine which positions in our list of high points contain trees. NDVI was first presented in [Rouse, Haas, Schell, and Deering 1973], and it attempts to reduce the atmospheric and illumination effects by using both differences and ratios of the received radiance in the IR and red bands. [Schott et al. 1999] writes the formula for NDVI as

$$NDVI = \frac{DC_{IR} - DC_R}{DC_{IR} + DC_R}, \tag{4.17}$$

where DC_{IR} and DC_R represent the digital count values in the IR (approximately 0.86μm) and red (0.66μm) bands, respectively. This ratio is computed for each pixel we have specified as an object. High values of this index (above 0.25) at object locations are considered trees, and lower value are considered buildings.

4.5 Geometric Reconstruction of Trees

Once all of the lidar points depicting trees have been effectively identified, further processing of this data is required to determine the geometrical properties of each individual tree. Once these geometries have been defined, CAD objects representing these trees may be produced in one of two ways. The first is to use the extracted geometrical properties to define input parameters for software such as TreeProfessional [Onyx Computing Inc. 2006], and then to create each tree separately from scratch using these parameters. The second approach is to use the calculated tree geometry to select the best-matching member from a pre-defined object library, and then to scale this object to fit the height and width estimates derived from the data. The primary method employed in the preparation of this proposal was the latter.

In [Gray et al. 2000], the authors describe an approach to streamline the construction of large forested scenes using high-resolution optical imagery. They assert that most coniferous tree crowns may be identified and characterized using radially-symmetric correlation methods. By first blurring the imagery then using circle functions at various scales, features with high radial symmetry may be uncovered. A similar approach is also used on processed images where tree regions are first extracted through an analysis of the NIR/red spectral ratio. Although the authors' primary intent was merely to recreate similar spatial aspects for the forested regions in the synthesized scene, in many cases the results were an accurate representation of the real-world objects. However, in dense tree regions, individual tree locations and sizes were often in error, and it should be stated that this work also notes poorer results when attempting to isolate deciduous species. Additionally, such methods only provide information regarding tree location and canopy diameter, while tree hight must be inferred.

For slightly improved results, this approach may be augmented by a similar technique using lidar imagery. Through methods outlined above, lidar permits an accurate localization of tree data without the need for spectral ratios, and with the fusion of multiple image modalities even better results may often be obtained. By blurring the lidar data from the tree regions in order to remove much of the high frequency content, likely tree centers may be found by identifying local maxima in the resultant image. A radial analysis (using heights in place of grey-values) may then be applied to determine individual tree canopy sizes.

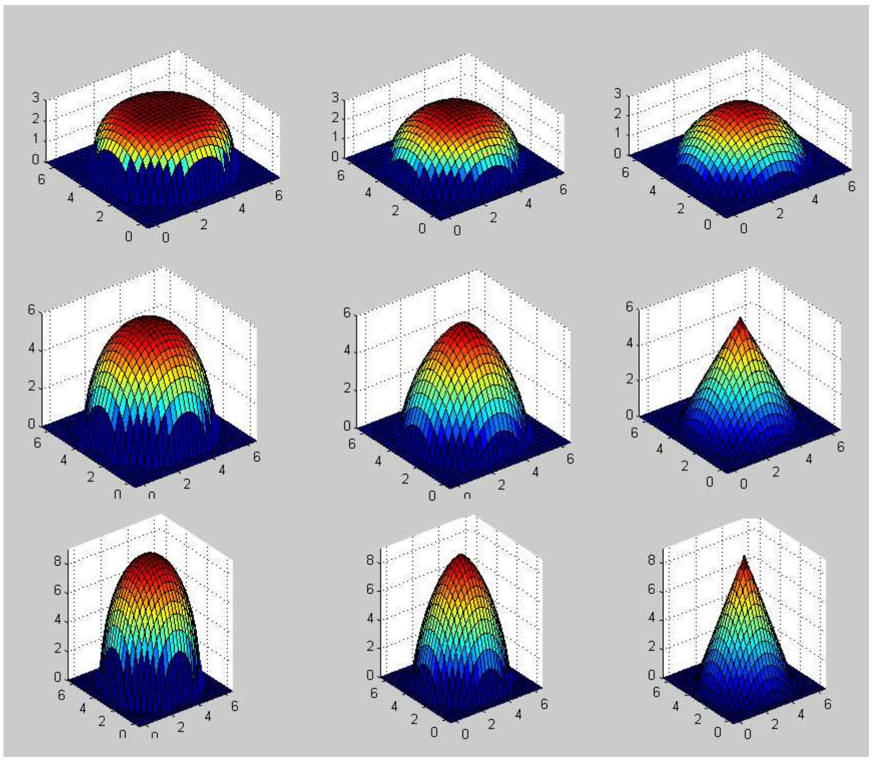
In order to determine whether a given tree is deciduous or coniferous, we use two approaches, depending on the density of trees in a given region. In stands where the trees overlap significantly, it is often difficult to use the lidar data to determine tree type, and we rely on a minimum distance classification using the hyperspectral data and supervised training. However, in regions where trees can be isolated, we may augment this approach with lidar-derived spatial information.

In [Pollock 1996], the author introduces a parametric model for tree geometry that he uses to predict the appearance of trees in reflectance images. We have taken this model and extended its use to determining whether trees are deciduous or coniferous. In a form slightly modified to that presented by Pollock, we note that tree shapes may be approximated by the outline defined by the generalized ellipsoid

$$\frac{(z - z_o)^n}{a^n} + \frac{(x - x_o)^2 + (y - y_o)^2}{b^n} = 1. \tag{4.18}$$

In this equation, (x_o, y_o) represents the center of the tree model in the $x - y$ plane, z_o is the ground elevation at the tree center, a represents the height of the tree above ground, b represents the radial spread of the tree, and n is a parameter that modifies the general shape of the model. For large value of n, the shape approximates a cylinder of height a. At $n = 2$, the shape become more round (it is actually a 3D ellipsoid of revolution), and at $n = 1$ the model is a cone. For values of $n < 1$, the model becomes increasingly concave. Figure 4.18 illustrates the form this model may take for various parameter combinations.

In order to solve for the proper tree model parameters, a Levenberg-Marquardt approach is used to determine the optimal values of x_o, y_o, z_o, a, b, and n so that the model fits the raster NDEM image in a least-squares sense. If we assume that conifers have a predominantly conical shape and that deciduous trees are more convex, we may use the n parameter to distinguish between the two types. In this work, we have set a simple threshold value of 1.2, but in general this threshold could be a function of a and b. For each isolated tree, these parameters are used to select the closest fitting tree from a previously defined library. The selected tree model is then scaled so that the height, width, and location match the extracted tree parameters.



(a)

Figure 4.18: Various shapes represented by the tree model. $b = 3$ in all caes. Top row: $a = 3, n = 4,3,2$. Middle row: $a = 6, n = 2, 1.5, 1$. Last row: $a = 8, n = 2, 1.5, 1$.

4.6 Geometric Reconstruction of Buildings

This section describes a novel method for constructing simplified building models from the available image sources. We use the term 'simplified' when discussing the desired model since the lidar data itself actually represent an overly-defined, noisy model of the objects we wish to define. By simply connecting each point via a 2D triangulation, we obtain a facetized object that may easily be represented in a CAD format. However, this model is not what we are after; we aim to reduce the data such that the dominant (and hopefully important) features are described with a minimal number of points, and unimportant features, or those smaller than the desired level of detail are ignored.

The literature describes this as *building reconstruction*, and the same terminology will be used here. Traditionally, this has been a predominantly manual process, requiring high-quality calibrated imagery, trained analysts and specialized photogrammetric equipment. However, with the tremendous rise in applications for 3D building models in recent years (many based on navigation and augmented reality, see [Brenner 2003]), a significant amount of work has been done in the hope of automating, or at least semi-automating, this task. An overview of a selection of these approaches is given below.

4.6.1 Recent Approaches to Building Reconstruction in the Literature

There are several ways to categorize the recent literature devoted to the field of building reconstruction. As noted in [Brenner 2003], a primary classification is based on the degree of process automation. Techniques range from fully manual, to interactive, to semi-autonomous, to fully-autonomous. These approaches may also be categorized according to the data they use. For years, photogrammetric processes using multiple images dominated the field. However, with the recent advances in laser scanning, many techniques relying on an analysis of lidar-derived DEMs have been developed over the last 10 years. Finally, building models may be classified according to the general framework in which models are defined. *Model-based* approaches attempt to determine the best-fitting member of a library of pre-constructed building models. These library models are typically parametric, so their geometry is not completely rigid, although exact building geometries are occasionally used as well. *Data-based* approaches, on the other hand, extract composite geometric features directly from the data and combine these features to complete the reconstruction. Typically, these features are the dominant planes in the building's roof structure, and by intersecting these planes, the model construction is achieved.

Photogrammetric processes using calibrated aerial images remain a primary means of deriving 3D scene content, and this includes objects such as building structures. The standard methods based on using image-to-image point matches and surveyed ground-to-image control-point matches are mature and produce accurate results. Unfortunately, in many cases it has proven difficult to adequately automate these techniques, especially when ground truth is unavailable. While a fair amount of success has been realized in autonomous DEM generation in suburban and rural environments (see [F. Ackermann and Krzystek 1991]), for example), difficulties persist in applying these approaches to more urban scenes. However, significant progress has been made in specific areas of autonomous image-based building reconstruction.

In [Haala 1996], the author first extracts a DEM and a disparity map from homologous points in an image pair. Regions marking the potential location of buildings are then produced using standard image processing techniques, and a line extraction is performed in these regions. Using the image pair and the disparity map, these lines are transformed to 3D segments, which are then filtered, grouped and processed in object space. The lines are then fit to simple geometric primitives, which then undergo a second filtering stage. The filtered primitives are then fit to a simple saddleback parametric building model, which is then re-filtered so as to minimize the distance between the model and the line segments in 3D. Multiple variations of this model are then built using various parameters. Finally, heuristics are employed to choose the best model of the set that was produced. The drawbacks of this approach are that it requires a good estimate of the DEM, that it is only viable in a suburban setting, and that it can only handle a single simple building geometry [Brenner 2003].

The approach described in [Henricsson and Baltsavias 1997], termed ”ARUBA”, starts with manually-selecting various regions of interest containing at most a single building Image color is then combined with a previously derived DEM to determine candidate building locations. Then, in a manner similar to that of Haala’s approach, line segments are extracted and transformed into object space. By analyzing these line segments, potential plane candidates are determined. The object boundary for each plane hypothesis is then determined based on additional geometric metrics. The most likely set of planes are then selected from the original hypothesis, and these are combined to produce an initial roof model. Vertical wall segments are added by extending the horizontal outer roof boundary vertically to the ground. Although good results were produced, like the previous approach, this methodology is unable to handle objects in an urban environment.

A hierarchical approach is presented in [Fischer et al. 1998], where both the 2D and 3D representation of geometric entities is used throughout the process. Image features including points, edges, and regions are first extracted from a stereo pair. These are then processed to yield hypothesis for corner points in object space. These corners are then analyzed and categorized as belonging to various roof parts, which are subsequently combined into more complex building geometries. The resulting building hypotheses are then re-projected into 2D image space to verify the resultant models.

In [Ameri 2000], the author extends the use of image pairs to a data-driven approach, where the only constraint on building geometry is that the roof structure is strictly composed of interconnecting polygons. Ameri uses both 2D image and 3D object spaces in a bottom-up approach that initially forms geometric primitives in the image domain, and subsequently combines these in object space. A top-down verification stage is then performed whereby the model hypothesis is back-projected into the images, and the resulting projections are then compared to the originally extracted 2D image features. Topological model information is also used as additional constraints in the verification process.

With the increased use of laser scanners starting in the early 1990's, it is not surprising that methods exploiting the inherent 3D nature of this data have received increasing attention in recent years. While 2D image data had to handle the problem of finding matched features in multiple images, lidar data was able to yield 3D information directly without any additional processing. These benefits were exploited early on in [Weidner and Forstner 1995], [Weidner 1996], and [Weidner 1997], where the author extracted DEMs using simple morphological techniques, detected buildings via comparison of the DEM and DTM, and recreated simple buildings with a parametric model-based approach. More complicated building outlines made use of a prismatic model, but building roof slopes were not derived for this case.

In [Maas and Vosselman 1999], the authors present a model-based approach using lidar data where invariant moments are applied to reconstruct simple rectangular shaped buildings with non-flat roof structures. This approach works well if the data fits the assumed building model, but does not degrade gracefully for more general geometries. In this same work, the authors also describe a data-based approach using a variant of the Hough Transform to detect dominant planar surfaces. They first used a Delaunay triangulation to create a TIN facetization of the point cloud, then extracted planar parameters from each resultant triangle. These parameters were then used to vote in parameter space, and the dominant parameter sets were identified as the planar faces of the buildings. Additional studies have been performed using region-growing approaches. These approaches use seed points that are then expanded based on the relative geometry of neighboring points. The reader is directed to [Chen, Teo, Rau, Liu, and Hsu 2005] for details regarding the implementation of this type of approach. In [Ma 2004], the author uses a different method for detecting planes, namely employing a clustering routine on local normal vectors. A variation of this approach is proposed as the primary method of building reconstruction for this research.

Schenk and Csatho note that due to the complimentary nature of lidar data and high-resolution imagery, models may be improved by fusing the information provided by the two modalities [Schenk and Csatho 2000]. Lidar data are inherently 3D in nature and they excel at defining planar surfaces. However, due to the comparatively low-point density, they may not precisely define object transition regions. Also, although lidar data may suffer from occlusions, they are not hindered by traditional solar shadows. This contrasts with high-resolution images, which excel at defining edges and other transition regions, but do not inherently convey 3D information. By fusing the two modalities additional information may be obtained than is possible with either sensor by itself.

An example of this fusion is given in [Rottensteiner and Briese 2003]. In this work, the author uses local normal approximations and a region-growing algorithm to define the dominant roof planes. An adjacency relationship is then created from the disjoint plane regions using a distance transform based approach first described in [Ameri 2000]. Aerial images are then used to help identify additional planar segments that may have been missed by the initial lidar segmentation (a variation of this concept is used in our approach). Rottensteiner also notes that imagery may be used to help refine building model edges, but no detail is provided in that area. In [Rottensteiner et al. 2005], the authors extend the use of multi-modal imagery in a Dempster-Shafer fusion approach for extracting building locations in an urban setting. Dempster-Shafer allows the combination of probability masses from several input variables to be used to determine a combined probability mass for each class. The authors use NDVI and their versions of the R_1 and R_L range images as inputs, and the output at each grid location is an index indicating class membership. Building, tree, grass, and soil classes are used.

A second approach for data fusion is found in [Ma 2004]. In this example, buildings are extracted through a thresholding of the lidar-derived NDEM followed by an area-based analysis used in combination with planar-fitting errors. Buildings are reconstructed using a polygon-based intersection-of-planes approach where dominant roof planes are determined by segmenting the local normal vectors. Once an initial model estimate is generated, it's exterior boundaries are refined through the use of high resolution aerial image pairs. Corresponding edge features are identified in each image using a Canny-based approach, and line segments are fit to these edges. The 3D location of these segments are then found through a photogrammetric projection into object space, and the building edge is adjusted to match the projected image edge if certain constraints are met.

The above approaches are by no means exhaustive. Additional surveys of the recent literature may be found in [Ameri 2000], [Brenner 2003], [Ma 2004], and [Rottensteiner 2001]. Also, a large selection of references is available via the topic bibliographies section of the data CD included in the bound copies of this document.

4.6.2 Overview of the Novel Building Reconstruction Approach

This dissertation makes use of a data-based, intersection of planes, building reconstruction approach that is fused with high-resolution aerial imagery. After the building regions are isolated per Section 4.4, an initial estimate of a building's outer boundary is obtained through a robust boundary extraction algorithm operating on the raw point data. Initial plane estimates are then found via a direct segmentation of feature vectors assigned to each raster image cell. These planes are then processed using a modified split and merge technique, and are then combined to produce an initial building model estimate. This estimate is then compared to the lidar data, and regions showing large errors are further analyzed (in both the lidar data and high-resolution image data) in an attempt to find additional building features. In regions where additional features are found, they are added to the model; in regions where the data still does not fit the model, the raw points are used to generate additional facets, and the user is flagged. Boundaries and internal vertical transitions are then potentially updated by backprojecting edges found in a single frame-array image to object space, and intersecting this projected edge with the building model.

In this section, it will be assumed that building locations have already been correctly identified, and that the described algorithms may work on a single region of raw data points (and the corresponding interpolated raster images) uncontaminated by trees or other objects. It will also be assumed that buildings are primarily composed of planar faces, and that their dominant internal and external edges are linear. Additionally, it is assumed that the lidar data were collected from a predominantly downward-looking sensor, and that few point samples are available from any building component other than its roof. This effect is amplified if the point cloud is interpolated to a rasterized grid before processing is performed. As such, we will be assuming that the roof boundary describes the location and orientation of the exterior vertical walls, and no effort is made to identify wall structure from actual wall data points.

4.6.3 Extraction of the Building's Outer Boundary

In the approach presented here, buildings are reconstructed primarily by identifying the dominant planes in their roof structure, then intersecting these planes to form edge lines. Due to the availability of many points per plane (as well as the use of robust algorithms), these planes, and subsequently the lines of intersection may be determined with high accuracy. However, since most airborne lidar datasets are collected from near-nadir orientations, there are very few data points that lie on vertical surfaces. As such, it is usually difficult to determine the planes corresponding to exterior walls using data points on these walls. Thus, the determination of building outer wall structure requires an alternative to the intersection of planes approach. If we assume that exterior walls are oriented directly under the outer boundary of the roof structure, we may identify the geometry of these walls by modeling the 2D shape of the building exterior.

4.6.3.1 Determining the Outer-most Data Points

In order to solve this problem, many researchers make the assumption that buildings have a strictly convex boundary geometry. An example of such an approach is given in [Chen et al. 2005], where the authors use the exterior triangles from a Delaunay facetization to determine outer boundary line segments. However, such techniques fail for more complex geometries. In Figure 4.19(a), we present a set of points representing the outer boundary of an 'L-shaped' object in the (x, y) plane. The connectivity shown in (b) represents the intuitive (and desired) outer-boundary points and their connectivity. In (c) we see the result of using a Delaunay-based convex hull approach. This method fails to highlight all of the desired outer points, and more importantly, it misses the interior angle. An alternative that was considered during this research was based on minimal-area polygons that enclose the full point set. This approach usually provides a better solution than the triangulation approaches, since it typically yielded additional (correct) boundary points. However, in practice we found that algorithms using this method had two residual issues. First, the approach still tended to miss points around interior angles in many cases, and second,

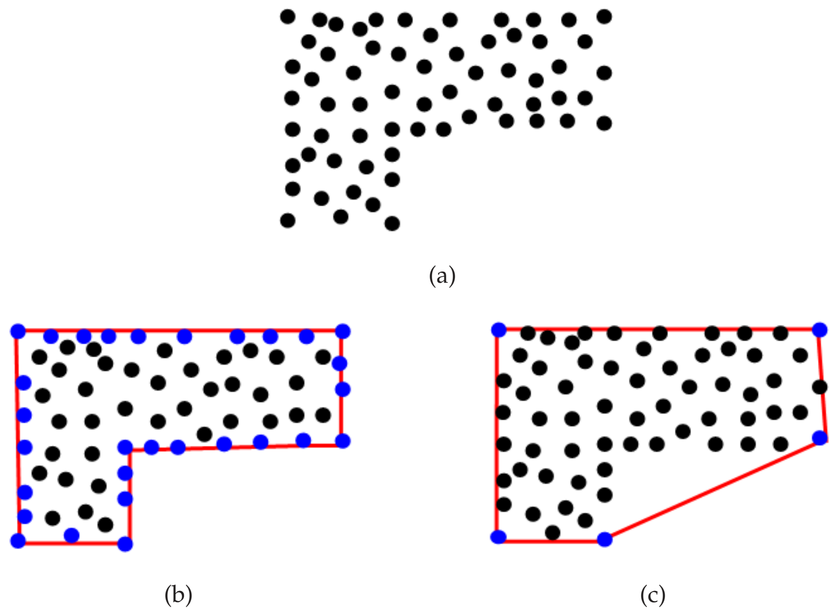
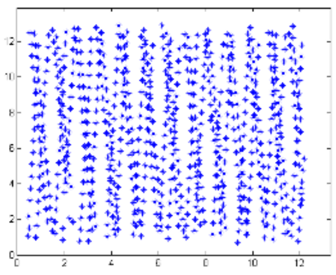


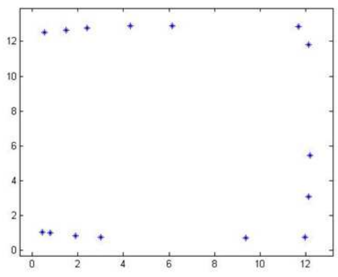
Figure 4.19: Determination of a group's exterior boundary: (a) 2D data points (b) Desired outer points (blue) and connectivity (red) (c) Convex hull points (blue) and connectivity (red) obtained from Delaunay triangulation.

the connectivity among the points was not easily determined.

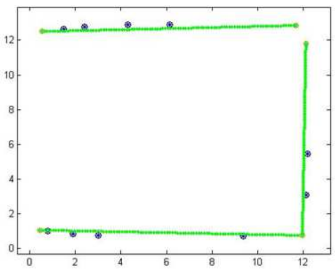
Even in cases when the building model fits the convex assumption, many of the published techniques may fail to determine the proper exterior boundary. This is especially true if the data is obtained by combining collections from two flightlines; in these cases, the criss-cross pattern on one edge may be entirely concave between the corner points. As a case in point, data from a square building on the RIT campus is considered in Figure 4.20. Plot (a) of this figure shows the building data projected into the (x, y) plane, and (b) shows the points representing the convex hull of the object. If we now use a Hough-based approach to try to find line segments representing the four building sides, we run into difficulties. Diagram (c) of this figure shows that the left side is not among the 10 most dominant line shapes in parameter space (which is shown in (d)). A maximal-area polygon does provide a better solution in this case, and heuristics may be used to fix the Hough solution. However, in general these techniques do not perform well with real data.



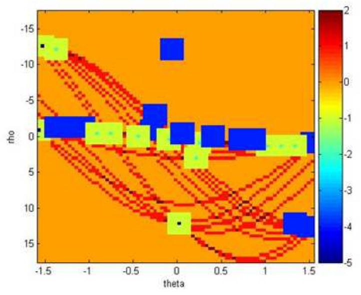
(a)



(b)



(c)



(d)

Figure 4.20: Determination of a group's exterior boundary via convex hull and Hough transform: (a) Initial object points (b) Convex hull boundary points, (c) Dominant lines as determined by Hough transform, (d) Hough parameter space representation.

To this end, we have opted to use *alpha shapes* for the determination of exterior object boundaries. Like the convex hull, alpha shapes are simply another approach to formally describe the 'shape' of a set of spatial point data. Unlike the convex hull, however, alpha shapes are not limited to convex geometries, and may even represent holes inside the geometry. Additionally, for a given point set, the resultant alpha shape is a function of a pre-determined structuring element. Therefore, the final 'shape' is actually an entire family of shapes.

Per [Edelsbrunner et al. 1983], alpha-shapes are actually a generalization of the convex hull that are based on an additional parameter α. In non-rigorous terms, we may define the alpha shape as follows. We begin by considering a set of points S, and a sphere of radius α (termed the α-ball). For any α between zero and infinity, the α-hull of S is the compliment of the union of all empty α-balls. From this, we can see that for $\alpha = \infty$, the alpha-shape is the convex hull of

S, and for α less than half the smallest distance between points, the alpha-shape is the full set S. It is worthwhile to note that in the paper introducing alpha shapes [Edelsbrunner, Kirkpatrick, and Seidel 1983], the authors use a different definition of alpha (which I term α_d where $\alpha_d = -\frac{1}{\alpha}$) than they used in later papers. This discrepancy, although not usually mentioned, has propagated throughout some of the source code and applications available on the internet.

In two dimensions, alpha-shapes may be computed directly from a 2D Delaunay triangulation of the data [Kavraki 2007]. In order to do this, we simply remove all edges and triangles that have circumscribing spheres with radius greater than α. The α-complex is the portion of the Delaunay triangulation that remains, and the alpha-shape is the boundary of the α-complex.

As described in [Edelsbrunner and Mucke 1992], we may conceptually think of alpha shapes in the following manner. Assume we have a volume of chocolate chip ice cream, where the chips represent a set of points. If we then used a spherical scoop to scoop away all the ice cream possible without bumping into the chips, and are somehow even able to scoop out holes in the same manner on the inside of the volume, we would eventually end up with a shape bounded by arcs and points. If we then replace these round facets with triangles by using line segments connecting the points, the result would be a representation of the alpha shape of the points. In this example, the parameter alpha is related to the radius of the scoop that was used.

As can be seen from the above description, a very small value of alpha will permit us to scoop away all of the ice cream between the chips, just leaving us with the point locations in the final shape. Similarly, a large scoop will be unable to scoop away concave regions between points, and the resultant shape will be identical to the convex hull of the points. For intermediate alpha values, there is a set of shapes that decrease and eventually develop cavities as alpha decreases. Figure 4.21 depicts the convex hull and one of the alpha shapes for a given set of 2D data points.

To use alpha shapes for boundary extraction of lidar data, we need to augment the above technique with a few additional steps. First, although the lidar data is in 3D, we will omit the height information, and simply perform the shape extraction in $\mathbb{R}^2$. Additionally, since the lidar

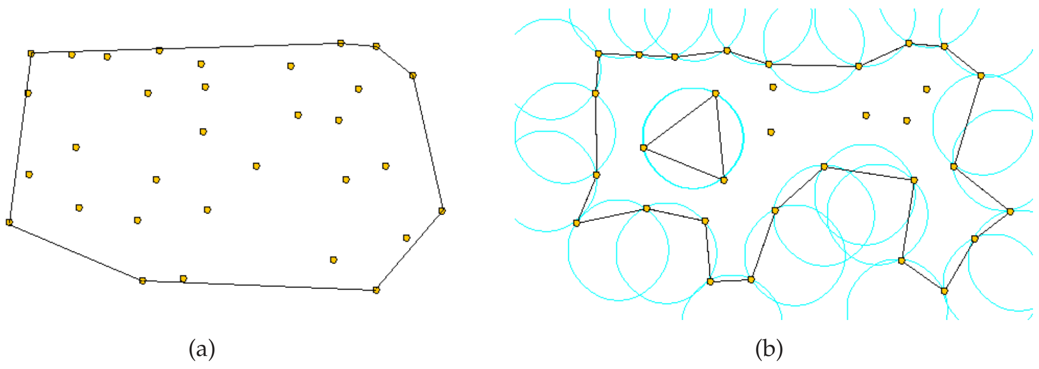


Figure 4.21: ’Shape’ of a collection of 2D points: (a) Delaunay triangulation convex hull (b) Alpha shapes with α-balls sized as shown.

sensor collects the data in strips, one dimension of the data may have a higher sampling density than the orthogonal dimension. In order to still effectively use alpha shapes, we may opt to use rotated ellipses as our structuring element. However, a simpler approach is to stretch the data along either the flightline, or the dimension orthogonal to the flightline, whichever is sampled at a greater frequency. Once this stretching has occurred, the sampling density will be approximately uniform, and circular structuring elements may be used. A standard 2D alpha shapes analysis is then be performed, with the α-ball sized to be slightly larger (roughly 20%) than the largest inter-data spacing. Once the initial alpha-shapes process is completed, we are left with a grouping of shape boundaries. Internal cavities should be discarded, and the remaining points represent the outermost data points that describe the building’s exterior boundary. It should be noted that in practice, we have also implemented alpha-shapes on real data without performing the initial data stretching. Although the actual alpha-shape is slightly different, in most cases this is compensated for by the boundary regularization procedure described later in this section.

After the points are extracted, they need to be arranged in the proper order, so that we may proceed point-by-point around the boundary. Since the alpha-shape is a derivative of the Delaunay triangulation connectivity, a by-product of the alpha-shape process is the specification of unstructured connectivity. That is, a listing of point pairs is produced, where each pair of points

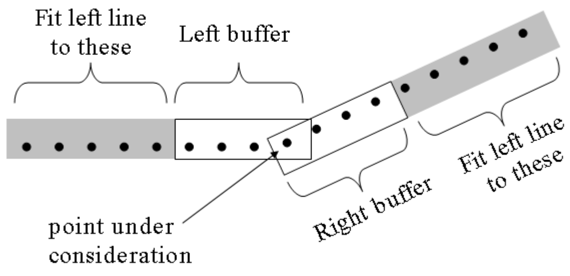


Figure 4.22: Estimation of local curvature; illustration of buffer regions and points used for slope estimates.

is connected in the final shape. By reordering these pairs so that the second point in a given pair is the first point in the next, we are able to proceed point-by-point around the shape.

4.6.3.2 Line Simplification

Once the alpha-shape points are extracted and ordered, we have an initial estimate of the building boundary. However, since this shape typically has an irregular geometry, it is usually undesirable to use this as the final building boundary. In general, a line simplification algorithm must be employed to produce less noisy results. In the early work on lidar-based building reconstruction, this was achieved through a point-merging algorithm where points are successively eliminated "where the corresponding triangle height d_i in the triangle formed by the points $i-1$ and $i+1$ is the minimum of the polygon until the minimum triangle height in an iteration is greater than a prefixed threshold, or the prefixed minimum number of points is reached" [Weidner and Forstner 1995]. In [Sampath and Shan 2007], the authors extract points that lie near single line segments by sequentially following the boundary and looking for positions where the slopes between adjacent point-pairs is significantly different. In this way, consecutive point pairs with similar slopes are grouped into longer segments.

In this research, we implemented two approaches for this line simplification. The first of these is a novel generalized slope-based approach, and the second is a modified sleeve-fitting algorithm. In the slope-based approach, at each point we generate an estimate of the curvature using local point data. For a given point being considered, two buffer regions are defined; one using points preceding the point of interest (termed the 'left' buffer) and one following it (the 'right' buffer), as shown in Figure 4.6.3.2. Lines are then fit to points within a window outside of each buffer region using a Deming orthogonal regression (see Appendix A.1). The window is usually sized based on absolute length, but windows based on a specific number of points have also been used. If the angle between the fitting lines exceeds a given threshold, the point under consideration is considered a *critical point*. Lines are then fit to all data points between (but not including) critical points, and the intersection of these lines form the simplified building vertices. Although this method works well for structures containing long boundary segments, it frequently fails if there are regions of rapid directional changes. In these cases, additional heuristics are used to bridge the gap between adjacent line segments that do not intersect near the original point data. In order to provide a more robust solution without the need for heuristics, a second line simplification method was adopted.

The second line simplification algorithm we implemented is based on the sleeve-fitting approach described in [Zhao and Saalfeld. 1997]. The sleeve method was originally published as a polyline simplification algorithm for efficient reduction of map scales. Traditionally this approach has been regarded as slightly less effective (and less common) than the Douglas-Peuker algorithm [Douglas and Peucker 1973], and recent work comparing the performance of several simplification methods has verified this for mapping applications [Shi and Cheung 2006]. However, in [Ma 2004] the author states that the Douglas Peuker algorithm may generate unexpected results when applied to building shapes (although Ma's concerns may be alleviated through a different implementation of the approach - see [Lee et al. 2008], for example). It was in this dissertation that the sleeve algorithm was first applied to the building boundary problem, but both Ma's work and the present research make significant modifications to the baseline approach.

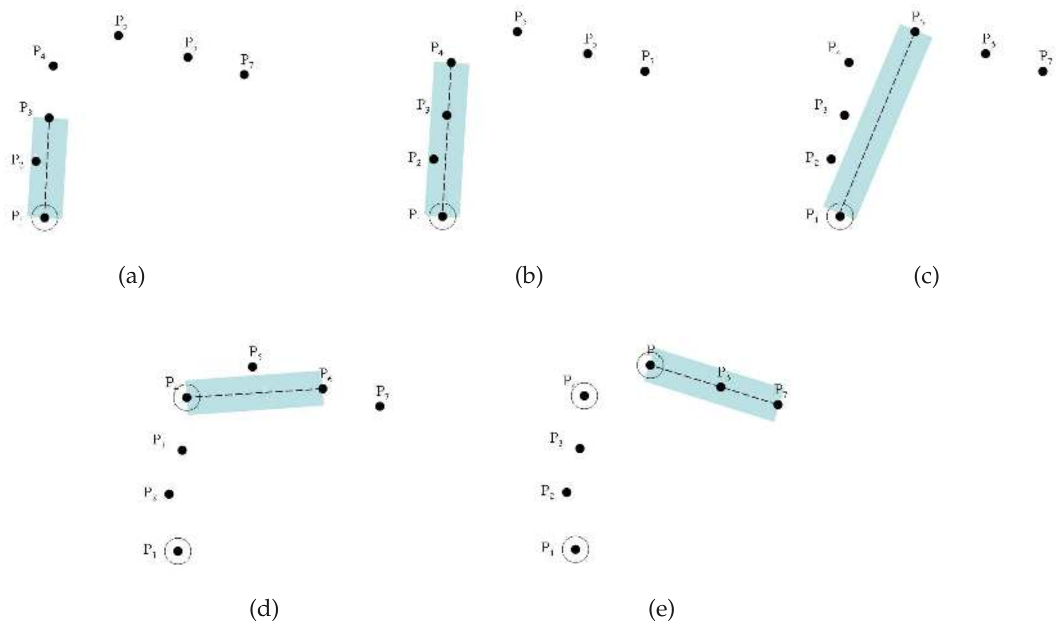


Figure 4.23: The sleeve method: (a) P_2 falls within the sleeve defined by P_1 and P_3, (b) P_2 and P_3 fall within the sleeve defined by P_1 and P_4, (c) Points fall outside the sleeve defined by P_1 and P_5; P_4 identified as a critical point, (d) P_5 falls outside the sleeve identified by P_4 and P_6; P_5 identified as a critical point, (e) The process continues around the set of boundary points.

The basic sleeve algorithm is better understood through the process illustrated in Figure 4.24. The width of the sleeve is selected at the outset, and this determines the maximum deviations that will be permitted before new line segments are defined. Starting at a chosen point P_1 (which is also labelled as a critical point), we then consider the distance of point P_2 from the line segment connecting P_1 and P_3 (a). If this distance is less than the sleeve width, P_2 lies within the sleeve and is considered a non-critical point. In the next iteration, we consider the segment P_1 to P_4 (b). If all the intermediate points fall within the sleeve (which is the case in the Figure), P_3 is deemed non-critical. Moving to the next iteration, we connect P_1 to P_5 and see that intermediate points now fall outside the sleeve (c). As such, the previous point, P_4 is declared a critical point. The process begins again starting at the location of this new critical point. Therefore, we now connect P_4 to P_6, and find that P_5 lies outside the sleeve (d). As such, P_5 is also labeled a critical point.

The process continues in (e), where P_6 is connected to P_8, and since P_7 lies within the sleeve, it is determined to be non-critical. This process is then continued until all points have been examined.

Algorithmically, the sleeve approach is implemented differently than would be indicated by the previous discussion. Since each intermediate point effectively provides an angular constraint on where the next point must be located in order to not produce a critical point, we may proceed considering only the angular constraint and the most recent point at any given iteration. This makes the algorithm both easy to code and quick to execute. The approach may be implemented as follows [Zhao and Saalfeld. 1997], [Ma 2004]:

```
• Define:  $w$ = the half-width of the 2D sleeve

1. Pick a starting point $P_1$ and consider its adjacent point $P_2$

2. Determine the direction $\theta_0$ and length $l_0$ of the line connecting these two points

3. Calculate the direction range $d_0$ such that the angle range for the next point is $\theta_0 \pm$
   $d_0$, where $d_0 = \tan^{-1}(w/2l_0)$

4. Connect the next point $P_{i+1}$ to $P_1$, and determine the direction $\theta$ and length $l$ of the
   resulting segment

5. Calculate a new direction range $d = \tan^{-1}(w/2l)$

   (a) If $\theta$ is within the direction range $d_0$, the current point $P_i$ is discarded.
       Calculate a new direction range at the point $P_{i+1}$ where the new $d$ is the
       intersection of the old $d$ and $d_0$.  Return to Step 3.

   (b) If $\theta$ is outside of the direction range $d_0$, the current point $P_i$ is considered
       a critical point.  $P_i$
       is taken as the first point in a new line segment and $P_{i+1}$ is the second point.
       Repeat steps 1 through 5.

6. when all points have been considered, END
```

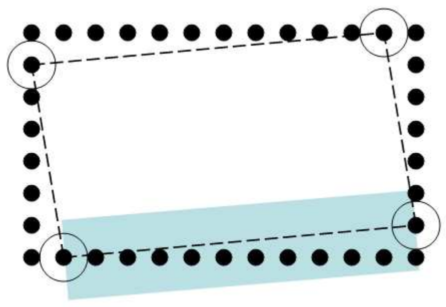


Figure 4.24: An illustration of the angular bias in the standard sleeve algorithm when applied to convex geometries.

We have modified the basic sleeve approach somewhat in order to select better critical points. In the original paper, Zhao delineates the starting point as a critical point, and proceeds without ever checking if this point truly represents a notable position of angular transition. In our work we improve on this by running the algorithm multiple times. First, the algorithm is run to find the initial set of critical points. We then run the algorithm again, starting at the third critical point, in order to get a better characterization of the original starting location. Also, since the sleeve algorithm on works in one direction, the critical points may have an angular bias to them that will result in a undesired rotation on the final building shape. This is illustrated in Figure 4.6.3.2. We typically need to have a wide enough sleeve to account for positional variations along each line segment. However, this means that we may not detect a critical point until slightly after the true corner point is reached. In convex shapes, this will always occur in a single direction, thereby producing an undesired angular offset in the final shape. Although the effect is reduced for non-convex shapes, a bias is still present in most cases. We correct this by running the sleeve algorithm both in the forward and reverse directions, thereby producing two sets of critical points. If the critical points for a given are not at consecutive points (or are not the same point), the center-most point between them is selected as the true critical point.

In the original reference, Zhao uses the segments connecting the critical points as the simplified boundary, discarding the points in between. However, these points contain useful information, so in way similar to that presented in [Ma and Meyer 2005] (who applied variation of the technique to raster images), we improve the accuracy of the simplified lines by using an orthogonal regression through the points between (and including) the critical points. These regressions produce lines, and consecutive lines are then intersected to produce a new set of critical points. In general, these points do not lie at locations covered by the original data set, but they will more accurately represent the location of building corners.

4.6.3.3 Boundary Regularization

In the vast majority of buildings, the outer boundary contains many edges that lie either parallel or perpendicular to the dominant building orientation. We may use this fact to our advantage in refining our simplified building edges through a process termed *boundary regularization*.

Most researchers have used variants of four main approaches to regularization in their work thus far. The first of these, the Minimum Description Length (MDL) was introduced in [Weidner and Forstner 1995], and it provides a statistics-based approach to perform a final adjustment of the building boundary. MDL imposes orthogonality constraints and seeks to either merge or remove the critical points while minimizing a shape complexity function. While this approach has the advantage of favoring orthogonality in adjacent segments while not requiring it, it is a complex, computationally intense approach that occasionally produces unexpected results. A hierarchical least squares approach is described in [Sampath and Shan 2007], where the authors use a least-squares regression to find initial slope estimates subject to the constraint that all segments must be either parallel or perpendicular. A second adjustment is then performed where the slopes of the long line segments are used as weight functions. Although this approach works well for many data sets, We have noted that approaches based on slope rather than angular orientation may be sensitive to noise for near vertical line orientations. Also, this approach is not fully robust when a long edge does not lie either parallel or perpendicular to the other edges (we describe these edges as being *rogue*, while edges that align with the dominant orientation of the building are *conforming*).

In [Mayer 2001], the author proposes the use of active contours to refine the building boundary. However, this approach has difficulty in properly handling the small line segments that frequently arise in the line simplification process. Also, as noted in [Ma 2004], active contours may not work well with boundaries defined by a lidar-based DEM. In order to improve his results, Ma created his own algorithm that used weighted averages of the angular orientation of each segment to determine the dominant building orientation. However, like the approach of Shan, this method relies on having two separate segment classes, and it does not adequately handle rogue edges.

In the present work, we have created an algorithm that forces boundary line segments to be either parallel or perpendicular to the dominant building orientation when appropriate, and to fit the data elsewhere. Additionally, the rogue segments do not influence the calculation of the primary building orientation. A final benefit of this new approach is that all conforming segments are initially brought into a common space where a segment and one perpendicular to it would have the same orientation. This allows the algorithm to work on all conforming segments simultaneously, and is not a weighted average of the solution defined for each of two orientation classes.

To begin our approach, we first collect all of the line segments defined in the line simplification process (we term the set S) , and re-align them so that their azimuthal orientation angle is in the range $0^\circ \leq \theta < 90^\circ$. For the segments shown in Figure 4.25, this would mean that A would be rotated counterclockwise by 90^o to C, and D would be rotated clockwise by 90^o to B. The entire set of (potentially rotated) segments is termed S_r. Note that a record is kept for each vector as to whether or not it has been rotated, so later in the process this rotation may be undone.

The next step is to identify all line segments in S_r with a length l greater than a given threshold (typically 3m) to use for the initial processing. We term this set of segments S_{long}. If at least four segments of the required length are not available, all segments are assigned to S_{long}. We then step

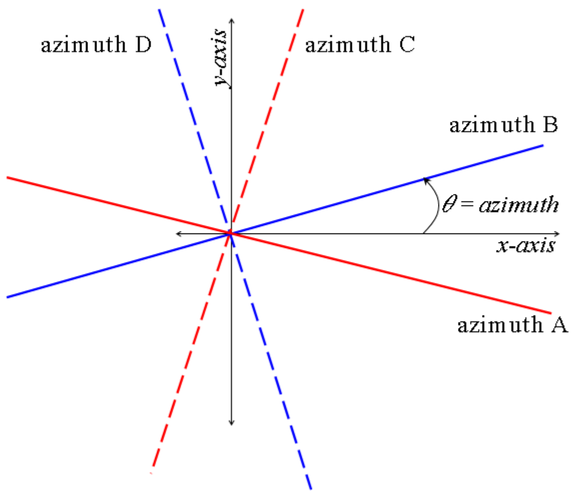


Figure 4.25: Line-segment re-alignment

through each segment of S_{long} in turn and determine the set of S_{long} segments that are within 15° of current segment. The set of segments for the i^{th} segments is termed S_i, and the S_i with the largest cumulative length is selected and redefined as S_a.

A weighted average angle is then obtained from the segments of S_a according to

$$\theta_a = \frac{\sum\limits_{i \in S_a} l_i \theta_i}{\sum\limits_{i \in S_a} l_i}, \tag{4.19}$$

where the l_i and θ_i values are the lengths and azimuthal angles of the line segments in S_a. This is the dominant orientation of the set S_{long}.

We now define the primary orthogonal azimuth as $\theta_b = \theta_a \pm 90°$, using whichever sign keeps the orientation within the range $-90° \leq \theta_b < 90°$.

Next, we return to the initial (unrotated) set S and evaluate an updated azimuth for each segment. Thresholds are defined for both angular matching (typically 15°) and for minimum length (typically 4m). Segments below the length threshold are assigned to the closest angular class (that is, they are given the slope θ_a or θ_b while keeping the location of the centroid point unchanged), regardless of the degree of angular similarity. This assignment is made since it is fairly common for the line simplification algorithm to detect multiple critical points at a single corner if the boundary data is severely contaminated with spatial noise. By forcing all small segments to align with the dominant building orientation, this effect is mitigated. Segments that are longer than the length threshold have their slope modified to θ_a or θ_b if their original slope is within the angular threshold of the potentially assigned slope value. Segments longer than the length threshold that have an angular difference greater than the angular threshold are left unchanged.

Finally, adjacent segments whose slopes are the same are combined into a single segment if the orthogonal distance between the segments is less than 3m. If the orthogonal distance is greater than this threshold, a perpendicular connecting segment is added, and the user is flagged that there may be an issue with the boundary regularization process at that location. In order to combine adjacent segments having the same slope, the centroid of the combined segment is the weighted average of the centroids of the two segments, where the segment lengths are used as weight values and the slope is kept unchanged. That is

$$\begin{aligned} x_n &= \frac{l_1 x_1 + l_2 x_2}{l_1 + l_2} \\ y_n &= \frac{l_1 y_1 + l_2 y_2}{l_1 + l_2} \end{aligned} \tag{4.20}$$

and $\theta_n = \theta_1 = \theta_2$.

4.6.3.4 Illustration of the Boundary Extraction Process

Figure 4.26 presents an illustration of the entire boundary extraction process as applied to real data from a typical suburban residence. In (a) we see the original point data projected to the (x, y) plane. At this point, the data also includes points related to another house and two trees. In general, these points would be removed by the building/tree segmentation process, but in the case

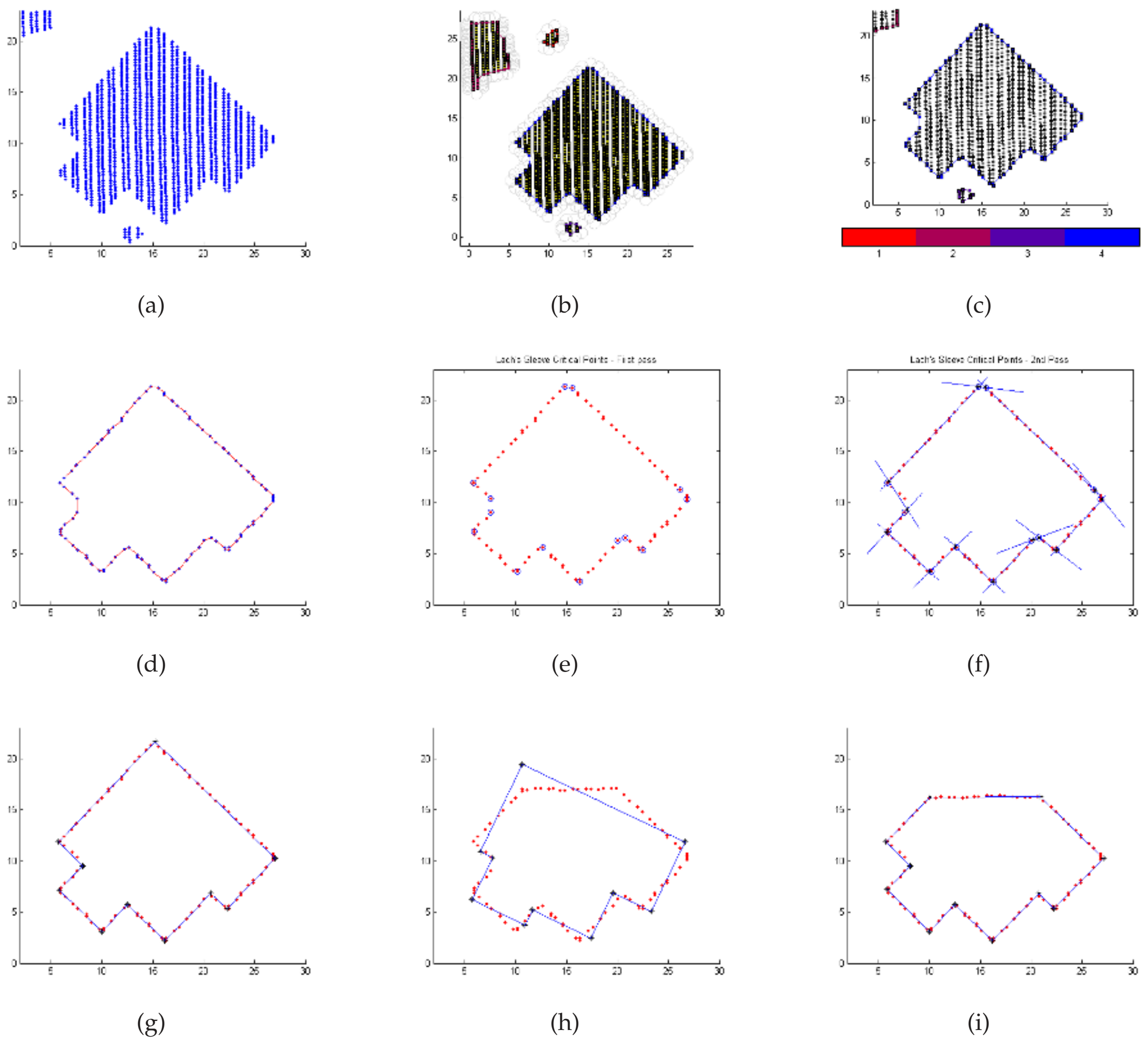


Figure 4.26: Illustration of the boundary extraction process.

at hand, they will be effectively identified and removed when the alpha-shapes are determined. Diagram (b) shows the location of the α-balls applied to this data, and (c) shows the points selected to represent the alpha-shape. Note that each object is assigned a unique alpha-shape index, and only the alpha-shape of interest is retained. In (d) the points have been properly ordered, such that we may trace the building outline by following a point-to-point path. The sleeve algorithm is then run a total of four times (two forward and two reverse) and the critical point shown in

(e) are identified. Orthogonal regression lines are then fit through all data between the critical points, and an initial simplified building model is produced (f). In (g) we see the outcome of the robust regularization algorithm. Note that all lines are either parallel or perpendicular, and the boundary fits the data well. If the original data is modified to represent a building with a long edge that is not at 90 degree increments with the other edges, other approaches not only miss the uniquely oriented edge, but the orientation of other edges are also biased by the presence of the non-orthogonal edge (h). Diagram (i) shows the result of applying our approach to this modified data. Note that the uniquely oriented edge is fit appropriately, and the other edges remain perpendicular while fitting the data without an angular offset.

4.6.4 Extracting the Internal Building Structure

After the outer boundary has been extracted, internal points are used to determine the remaining portions of the exterior building geometry. Like many of the papers listed earlier, this work assumes that the roof structure is primarily polyhedral in nature. That is, it is composed mostly of planar facets whose edges may be described using a group of straight line segments. Although many authors use a data-based approach in which initial seed regions are selected and grown to define the planes [Rottensteiner and Briese 2003], [Vosselman 1999], this research proceeds more along the lines of that given in [Ma 2004]. That is, instead of performing a point-based region growing, local feature vectors are created for each pixel of a raster image, and these features are then clustered using a segmentation algorithm. This segmentation is then processed in order to more accurately represent the dominant planes and their spatial relationships. These planes are then intersected to form edge lines, and edge lines are intersected to form the initial set of roof model vertices. This model is then compared to the raw point data, and regions of potential error are further processed to produce improved results. If the building has internal vertical transitions (step edges where adjacent planes should not be intersected), additional techniques are required. For the initial discussion of the approach used in this work, we will assume that the roof structure

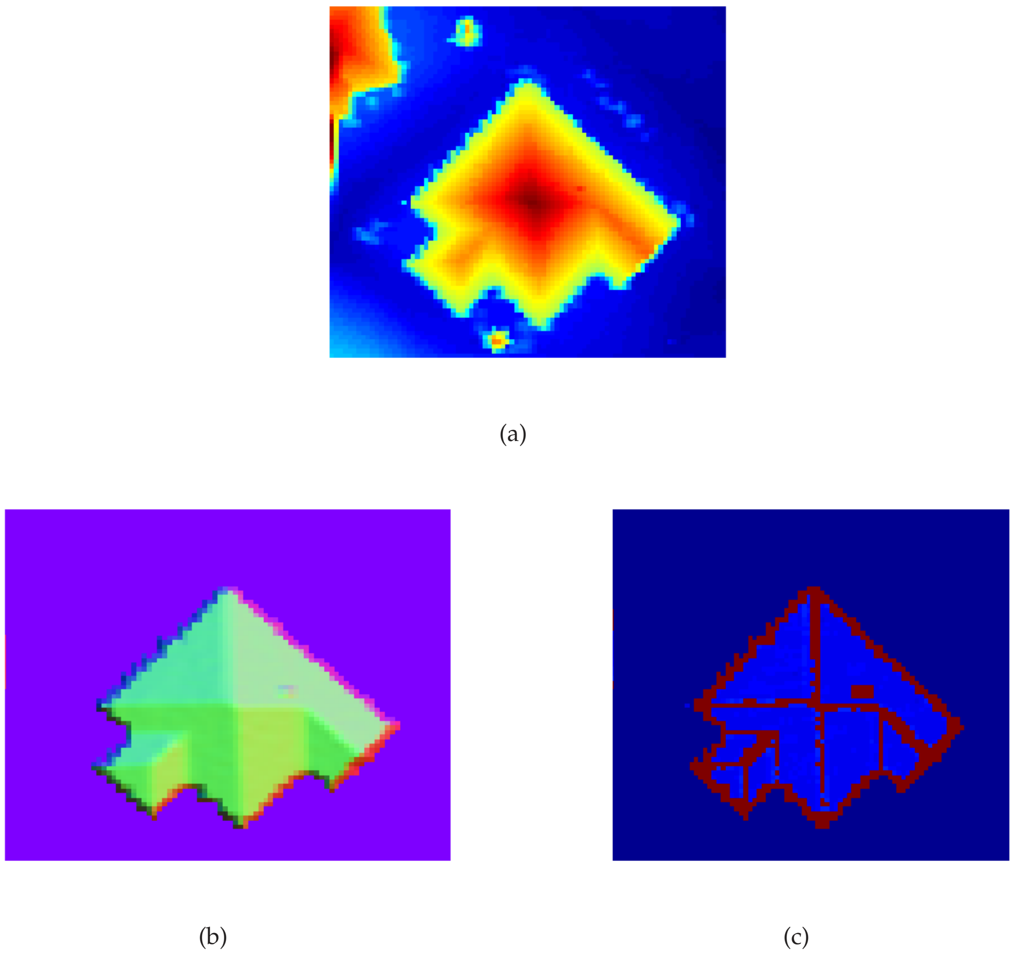


Figure 4.27: (a) R_1 Range image (b) N_3 normal image, and (c) E_3 plane-fitting error image.

does not contain any vertical transitions. We will then show how the approach is modified to account for these steep edges. It is worthwhile to note that the majority of building reconstruction methods in the literature are unable to adequately handle internal roof vertical transitions (including recent comprehensive works such as [Ma 2004]). As such the approach presented here is applicable to a wider range of building geometries than most.

4.6.4.1 Finding the Dominant Roof Planes

We begin the process of finding the dominant roof planes by building two different images of local normal vectors from the R_1 range image. Note that this version of the range image should produce accurate estimates of a planar roof height at the pixel center, whereas the other range images would introduce additional noise effects via their interpolation processes. In order to produce the first normal image, we consider all the points (z-value for a given pixel (x, y) location) within a small spherical regions (typically of radius 1m) around each point. We then do a singular value decomposition (SVD) on a matrix containing the point coordinates. The singular vectors corresponding to the two largest singular values represent the two dominant directions of variance in the point set, which is simply the plane that best fits this data in an orthogonal, least squares sense. The final singular vector is orthogonal to the first two, and therefore represents the direction normal to this plane. It is this normal vector that is stored for each pixel location. The normal image created via this approach is termed the N_3 image. We produce a similar normal image using points within a small circular region in the (x, y) dimension about each point location (that is, we ignore the vertical distance in selecting 'close' points. This normal image may contain distant points at vertical transitions, and is termed the N_2 image here. For each location (and for each variant of normal image) we also compute the average error describing how well the local points fit the plane defined by the first two singular vectors. In general, the error will be higher in transition regions (where planes intersect or at steep edges) than in the central part of each plane. These errors are combined into the 'error images' E_3 and E_2, and will later be used to highlight regions where the normal images are likely to be mis-classified. Figure 4.27 shows the R_1, N_3 and E_3 raster images for the same house that was considered in the boundary extraction process. In the image of N_3, the x-component of each normal vector was set to the red image channel, y-component to green, and the z-component to the blue channel. In the E_3 image, all errors above one standard deviation above the mean are set to red, and the rest are blue.

We then build up a feature vector at each pixel location, where the first three elements of each vector are the x, y, and z components of that particular N_2 normal vector, and the final two elements are the x and y locations of the pixel being considered. The set of feature vectors are then segmented via the mean-shift algorithm. The mean-shift is a nonparametric mode-seeking algorithm that uses a kernel for density gradient estimation. It was first introduced in [Fukunaga and Hostetler 1975], and has subsequently been refined by many researchers. The discussion that follows is based on [Comaniciu 2000], who proved the algorithm converges, provided certain kernel criteria are met. By taking a subset of the data available and allowing these points to migrate to points of local maximal density, density modes for the data may be identified. Once these modes are refined, the space may be delineated based on a nearest-neighbor approach, so each data point is identified with a particular mode, thereby producing the segmentation.

In order to perform a segmentation using mean-shift, we first define the multivariate kernel density estimate as

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \tag{4.21}$$

where $K(x)$ is the kernel and h is the radius of the window. $K(x)$ must be selected such that it satisfies

$$\sup_{x \in R^d} |K(x)| < \infty, \qquad \int_{R^d} |K(x)| dx < \infty, \qquad \lim_{x \to \infty} \|x\| K(x) = 0, \qquad \int_{R^d} K(x) dx = 1. \tag{4.22}$$

Comaniciu discusses two typical kernels in his dissertation. The first of these, the multivariate Gaussian, is defined as

$$K_N(x) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{1}{2}\|x\|^2}, \tag{4.23}$$

and the second, the Epanechnikov is defined as

$$K_E(x) = f(z) = \begin{cases} \frac{(d+2)(1-\|x\|^2}{2c_d} & \text{for} \quad \|x\| \leq 1 \\ 0 & \text{for} \quad \|x\| > 1 \end{cases} \tag{4.24}$$

where c_d is the volume of a $d-$dimensional unit sphere. This kernel arises when the mean integrated squared error of $\hat{f}$ is minimized.

By using a differentiable kernel (such as the two above), we may compute an estimate of the density gradient by determining the gradient of the kernel density estimate. That is

$$\hat{\nabla} f(x) \equiv \nabla \hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} \nabla K \left(\frac{x - x_i}{h} \right). \tag{4.25}$$

For the Epanechnikov kernel, this is given by

$$\hat{\nabla} f_E(x) = \frac{C}{n_x} \sum_{x_i \in S_h(x)} x_i - x. \tag{4.26}$$

where C is any constant and $S_h(x)$ is a hypersphere of radius h, centered on x, having the volume $h^d c_d$, which contains n_x data points. The last portion of this equation,

$$M_h(x) = \frac{1}{n_x} \sum_{x_i \in S_h(x)} x_i - x \tag{4.27}$$

is called the sample mean shift. Since this mean shift vector always points towards the direction of the maximum density gradient, it may be used to navigate to a local maximal density point; that is, a mode of the density. Additionally, it should be noted that the mean shift step size is inherently large when the originating position is of low density, and it is smaller as x approaches a local maximum. This has the effect of allowing the algorithm to quickly advance in an appropriate direction when the density is low, yet still hone in on the solution without significantly overshooting when the desired modal point is nearly reached.

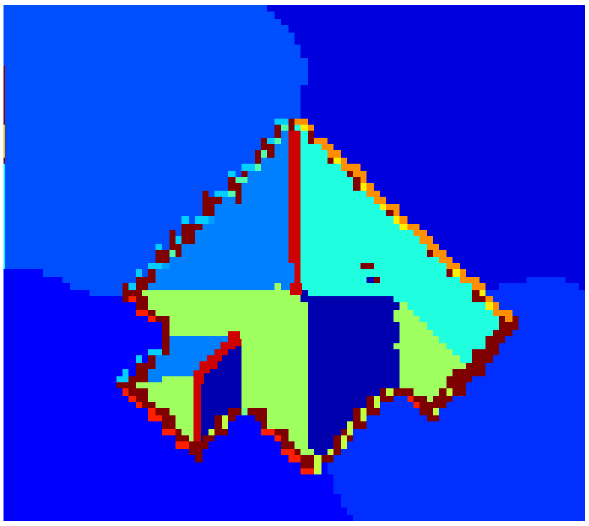


Figure 4.28: Result of segmenting the normal and spatial data via mean-shift.

In order to use this approach to perform data clustering, the mean shift vector $M_h(x)$ is computed for a given initial position x, then the window $S_h(x)$ is shifted by $M_h(x)$. The new mean shift vector is then computed, and the window shifted again. This process is performed iteratively until convergence. In practice, this is done with m starting values for x, where $m << n$, the number of data points to be clustered. The m points should be selected such that the distance between two neighboring points should never be smaller than h, and the points should not lie in sparsely populated regions. Additionally, after the modes are found from the initial iterations on the m points, the starting positions should be perturbed, and the procedure performed a second time. This helps eliminate problems associated with local plateaus in the space being considered. Cluster centers (the local density maxima) are then validated by ensuring the presence of a significant density valley between it and its neighboring modes. Once the final cluster centers are determined, all n data points are associated with a cluster center. Typically, a k-nearest neighbor technique is used.

Comaniciu discusses applying the mean-shift clustering to joint spatial-range data, but we have found that this technique also works well with our spatial-normal orientation feature vectors. Figure 4.28 shows the result of segmenting the feature vectors associated with the N_2 image. As can be seen, for this particular example, all of the major roof faces have been detected, but many extraneous classes are also produced. These are typically in the transition regions, where the points used to estimate the normal vectors were truly co-planar. Also, in many cases roof faces with a similar orientation are segmented into the same class, even if these faces lie on different planes in $\mathbb{R}^3$, and are spatially separated in the (x, y) plane. Although using the x and y pixel coordinates help to have spatially separated segments assigned to separate classes, it is still fairly common for facets lying on parallel planes to be classified together.

In order to improve this initial segmentation (termed SI_1) and provide a single class for each roof face, a refinement procedure is employed for the non-ground pixels. We begin this process by performing a binary erosion on the building shape in order to remove the outer pixels along the inner building boundary of the segmented image, producing the segmented image SI_2. The depth of this data removal should be at least as large as the radius used in selecting points for the N_2 image. These eroded pixels will be replaced with pixels of the proper class at a later stage in the refinement.

Next, we identify pixels in the E_2 image where the value (that is, the plane-fitting error) is greater than a given threshold. We use a threshold value of greater than one standard deviation above the mean error value over the image, although this tends to be constant across building for a given resolution of the R_1 image. The corresponding pixels in the SI_2 image are then assigned to a temporary class T. The segmented image with the high-error points assigned to T is termed SI_3

We next use a split-and-merge approach to modify the SI_3 segmentation. A binary image is created for each non-T class, where pixels representing that class are set to 1, and the remaining pixels are assigned a 0 value. A morphological region-counting routine is then performed on each class image, and when multiple regions are found within a single class, unique class indices are assigned to each region. This process resolves the issue of multiple roof surfaces being classified to the same class, so long as there is a spatial separation between the classes. It is for this reason that we used the N_2 image (instead of N_3) in the clustering process. If two adjacent regions have the same slope but lie on different planes, the N_2 image will typically have a high-error region between the facets, but the N_3 image may not. This high-error region serves to separate the surfaces, enabling the simple splitting technique.

A more complex class splitting technique is also employed. This approach is able to split classes representing data on two or more different planes, even if the classes are adjacent and the planes are parallel. For each class, the data representing that class's pixels are extracted from the R_1 image. A RANSAC-based plane-fitting routine is then performed on those pixels. That is, three pixels are selected randomly, and the plane defined by those three pixels is identified. The remaining pixels are then classified as inliers or outliers, depending on their proximity to this plane. This is repeated with a new set of three pixels over many iterations, after which, the largest set of inliers is used to define the true dominant plane. Note that since only three points are used, the plane parameters may be solved for exactly, and an SVD of the data is not required. The inlier points are then assigned a unique class index, are removed from the data being considered, and the approach is re-run on the outlier data set. If the outliers do not represent errors, but are actually from a second plane, a new set of inliers is found. This process of checking the outlier data for additional planes is repeated until no new planes are found. This splitting process is similar to the one described in [Khoshelham, Li, and King 2005], although this reference does not use the RANSAC algorithm for the identification of outliers. The segmentation image after class splitting operations has been completed is termed the S_4 image.

Once the splitting operations have been completed, we perform a merging process to combine adjacent regions that likely represent a common facet. Two neighboring regions are merged if their slopes fall within an angular threshold and their distance from a common point (we use the origin) falls within a distance threshold. That is,

$$\begin{aligned}\theta_{1\_2} &= \cos^{-1}(n_1 \cdot n_2) < \theta_{thresh} \\ d_{1\_2} &= |n_1 \cdot x_1| - |n_2 \cdot x| < d_{thresh}, \end{aligned} \tag{4.28}$$

where n_i is the unit-length normal for class i, and x_i represents a point in the i^{th} class. Note that many authors just use an angular threshold for the merging requirement; however, we have found many house geometries where this incorrectly merges roof facets that are adjacent and parallel, but not co-planar. Once two regions are merged, the index associated with each class is updated, and the new segmented image is termed S_5. As noted in [Khoshelham, Li, and King 2005], in certain cases it may be advantageous to iterate through the split-and-merge process several times, until no regions are changed in the process. However, in our work, this is a seldom-used option that the user has to enable.

The S_{final} (or simply 'final') segmentation image is obtained by re-assigning the class T pixels and boundary points that were removed during the initial refinement stage to the class of the 'closest' non-ground pixels that are not also a part of either of these sets. Classes with membership below a pre-defined area threshold (we use 4m^2) are also re-assigned. We have used two variants of the distance metric used in defining 'close.' The first is a Euclidean distance in the $x - y$ plane, and the second is the angular separation of the normal vectors as obtained from the N_3 (not N_2) image. Both techniques work reasonable well in practice, although the normal-vector matching appears to perform slightly better in this class re-assignment. However, in producing the final building models, the robustness built into the rest of the reconstruction process makes the choice somewhat less important; we have yet to realize a notable difference in the resulting geometry as a result of the distance metric used. The final segmentation image for the house data we have been using is given in Figure 4.29.

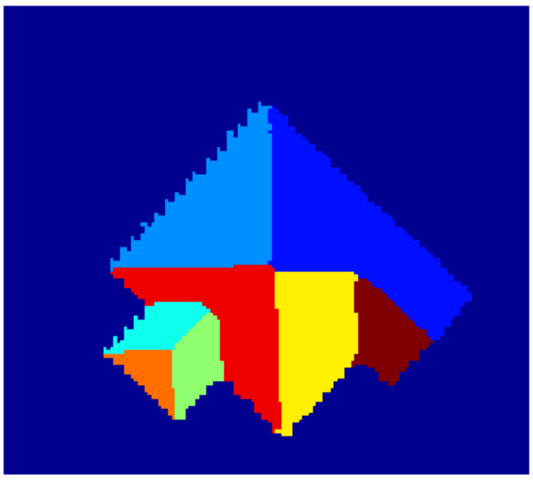
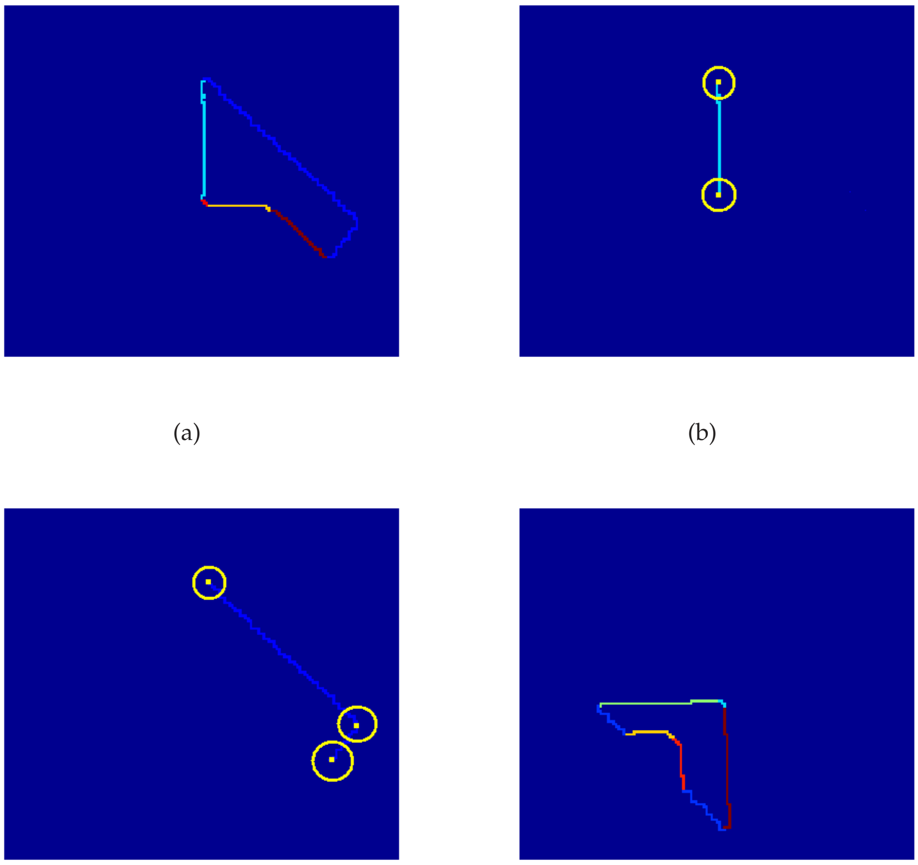


Figure 4.29: Final segmentation image (S_{final}).

4.6.4.2 Initial Roof Reconstruction

Once the segmentation process is complete, we are in a position to begin reconstructing individual roof facets. Each class of the segmentation is processed in turn, such that the facets are built in 3D space one at a time. However, before beginning individual facet analysis, we will find the plane best fitting each segmented region. We use a RANSAC-based approach for this process, using only the R_1 data contained in a given region for defining the plane representing that region. Once the region representing each class has a plane associated with it, we proceed with the individual facet reconstruction.

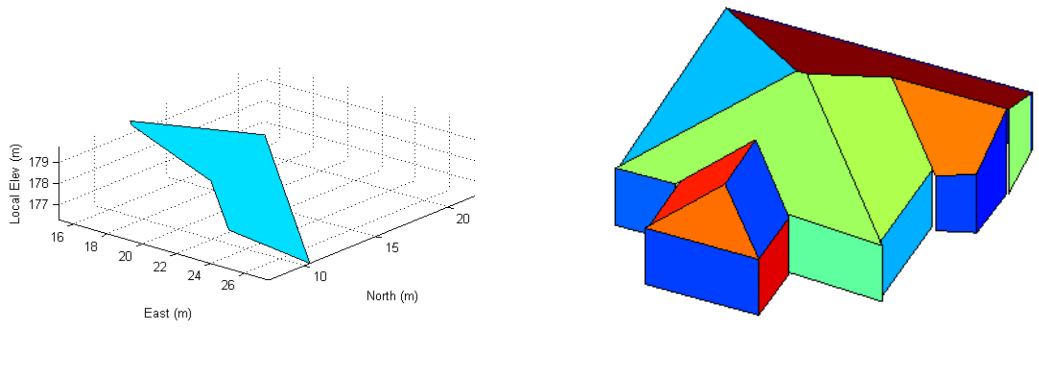
We start constructing a single facet by taking the binary image of a single class, dilating it, and then subtracting the original binary image. This leaves an image that is all zeros except for the pixels immediately outside the boundary for that class. We then use this image as a mask over S_{final} to produce the image shown in Figure 4.30 (a). This image contains all of the adjacency relationships for that facet, and also gives a rough estimate as to the location of the facet edges in the (x, y) plane.



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

Figure 4.30: Reconstructing a single facet: (a) Exterior facet boundary showing adjacent facets, (b) Split segment with a single DLS, (c) Split segment with 2 DLS's, (d) A separate facet where the ground-adjacent segment is composed of 2 split segments, each with a single DLS.

We term each colored region in this figure a *segment*, and each segment shows how the adjacent class planes intersect the plane being considered. In general, a segment may be a single line segment, several line segments, or even two or more non-connected line segments. Within a given facet, we now work on each segment individually. We first split any disjoint segments into multiple *split segments*, each with an individual index. Each split segment is then broken down further into disjoint linear segments (DLS), where the points separating each DLS are identified using the



(a) (b)

Figure 4.31: 3D facet and initial building model.

sleeve algorithm. For our facet of interest, this is illustrated in Figure 4.30 (b) and (c). Diagram (b) shows a segment containing a single split segment with only a single DLS. The sleeve algorithm has identified the beginning and end points, but no intermediate critical points are present. In (c), we see where the facet is adjacent to the ground. In this case, the single segment also has a single split segment, but has two DLS's and three critical points (circled). For the building model being considered, the only facet with a segment containing more than one split segment is shown diagram (d). This is the red 'rotated L' facet shown in Figure 4.29, and the two split segments are where the facet is adjacent to the ground. If the building model contains no internal vertical transitions, the only places we will see a split segment having multiple DLS's is for the segments indicating adjacency with the ground. However, multiple DLS's are possible for any segment if interior vertical transitions are present (a case that will be considered later). For each segment that does not represent an adjacency with the ground, we now replace the pixel-based line approximations representing each DLS with the vectorized (explicitly defined) 2D line obtained by intersecting two planes, projected into the (x, y) plane. The planes needed are the ones we have previously computed for the current class and the adjacent class represented by the DLS. Each pixel-based DLS line is replaced by an intersection-determined 2D line in-turn, and these lines are then intersected to produce vertices in the (x, y) plane. Vertices for the ground DLS's are obtained by replacing the critical points found when working with the DLS's with the vertices obtained from the building boundary extraction process done earlier.

Since we know the equation of the plane for the facet of interest, we may now compute the facet 2D vertices to 3D in-the-plane vertices by simply augmenting the (x, y) coordinates with the proper z-coordinate at that position. A resultant facet created using this approach is given in Figure 4.31 (a).

By reconstructing each facet in-turn, an initial 3D model of the roof structure is obtained. This initial roof model is then refined by merging vertices that are within 1m of each other (through a positional average), and an initial building model is created by adding vertical wall facets below the external boundary segments. An example of an initial building model is shown in Figure 4.31 (b).

4.6.4.3 Dealing with Internal Vertical Transitions

Unfortunately, not all polyhedral roof types can be completely modeled using a set of intersecting non-vertical planes. In many roof geometries, there are step edges or *vertical transitions*, where a vertical planar segment is required to connect two planar facets that are adjacent in the (x, y) plane but do not intersect. In general, a single split segment (representing a 2D adjacency) may be composed of multiple DLS's, and some of these DLS's may or may not represent planar intersections while others may or may not represent vertical transitions. Due to the potential presence of these regions, the above approach requires a slight modification.

For each DLS that is detected, we fit a 2D line to the pixels composing that DLS. We then intersect the two planes whose adjacency is represented by that particular DLS, and project this line of intersection into the (x, y) plane. If the line fitting the pixels is within a given distance and angular threshold (we use 2.5m and 20°), the DLS is deemed an actual intersection of planes, and the boundary is defined by the projected line of intersection. If the lines do not match up,

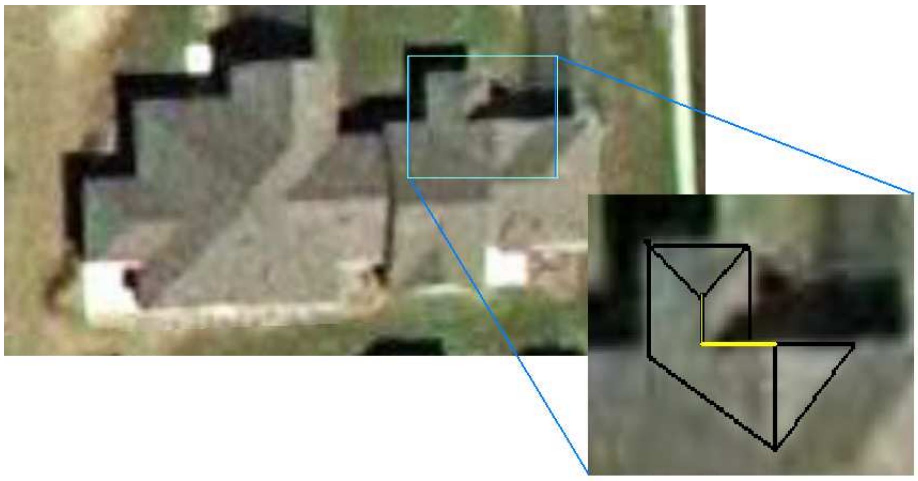


Figure 4.32: Single segment containing both a shared edge and a vertical transition.

the DLS is considered a vertical transition, and the line fitting the pixels is used to define the step edge. In [Rottensteiner and Briese 2003] the pixels on the DLS are directly compared to the projected line of planar intersection without first fitting a 2D line to them. This may represent a potential improvement, but the investigation of which technique is preferable is reserved for future research.

We use the location of vertical transitions to truncate the affected facet polygons at the location of the step edge. This edge is then projected into the planes of the two classes bordering the vertical transitions, and a vertical facet representing the height transition is created using these projected edge lines in each plane.

If all DLS's are found, the above approach works quite well. However, in cases where multiple DLS's are present in a single split segment but are not distinguished by the sleeve algorithm, the process fails according to one of two failure modes. Consider the house shown in Figure 4.32. The L-shaped split segment highlighted in yellow is composed of two DLS's, a vertical segment (representing the intersection of planes) and a horizontal one (the vertical transition). If both of the DCS's are detected, the reconstruction proceeds smoothly. However, if either the data density

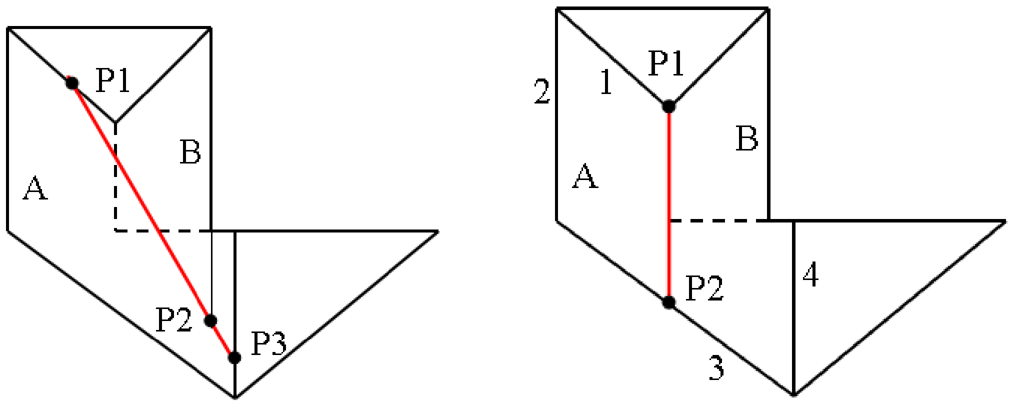


Figure 4.33: Two potential outcomes of an undetected DLS.

of the pixel density of the R_1 image is small relative to the size of the DLS's, we may only detect a single DLS. In this case, that DLS is determined to either represent a vertical transition or an actual intersection of planes. In the former case, we get the result shown in Figure 4.33 (a). Here, the split segment is thought to be a vertical transition, so the line best fitting the edge pixels is used to define the edge. When working with facet A, this edge is extended such that it spans $P1$ to $P3$. Since the nearest intersecting DLS for facet B is different, when working with this facet the extension only covers the length $P1$ to $P2$. The A and B facets will then be defined in such a way that they do not fit together properly. Additionally, $P1$ is higher when projected onto the B-facet than it is when projected to facet A, but $P2$ and $P3$ are lower when projected to facet B. This translates to an instantly-recognizable error in the final reconstruction.

The second failure mode is if the DLS is determined to represent an intersection of planes, as illustrated in Figure 4.33 (b). In this case, neither facet A or facet B will be able to be reconstructed. Consider the case for facet A. If we start at $P1$ we may start computing edge intersection points working counterclockwise through segments 1 through 4, then proceeding to the vertical transition edge. However, the vertical transition edge and segment 4 do not intersect, so the process breaks down. A similar situation occurs for facet B.

In order to rectify either failure mode, we perform a check on the line fitting through the edge pixels. If the location that this line intersects the adjacent segments is far from the actual edge pixels (we use a 1m threshold), the user is flagged and given the opportunity to split every segment of the problem facets into multiple DLS's. Although have had some success with automatically rectifying this situation by using additional heuristics, these fixes are not yet robust. As such, manual intervention is still preferable.

4.6.5 Refining Building Models Using High Resolution Imagery

Once a building has been reconstructed using lidar data, refinements may be made by incorporating information from other image modalities. Several references (see [Schenk and Csatho 2000], for example) recommend the use of high resolution frame array imagery, as the information contained in such images is often complimentary to the lidar data. As a case in point, 3D position information is easily obtained from the lidar data, while definitive features such as edges and vertical transitions are usually better represented in more conventional image types. Unfortunately, the research in fusing these modalities usually concentrates on the theoretical benefits of such a fusion, while there is comparatively little work detailing implementation methods. However, before we can effectively fuse information from the two modalities, we require a more precise registration between the datasets than was achieved during the coarse registration done in Section 4.3.

4.6.5.1 Registering 3D Lidar Data to 2D Frame-Array Imagery

As noted above, the first step in fusing the two modalities is to perform a precise geometric registration of the data sets. That is, we need to find a pair of transformations that permit us to take a 2D image feature and determine it's position in 3D space (relative to the lidar data), as well as transforming 3D lidar information into the 2D image space. The first transform may be found by projecting rays from the camera center through the feature of interest out into 3D space. If

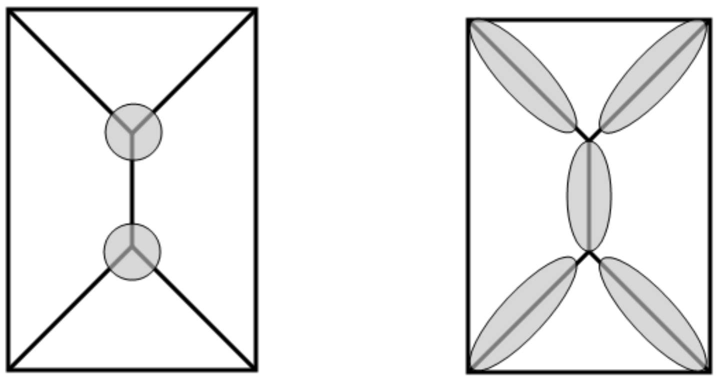


Figure 4.34: Features arising from the intersection of planes: (a) Points and (b) Line segments.

we then have additional information telling us what location along this projection is needed, we may uniquely determine the 3D location of a 2D image feature. The additional information may come from several sources. In certain cases we may have two or more images, each containing the feature of interest. By back-projecting this feature from all images, the intersection of the surfaces represents the location of the feature in 3D space. When using point features, this is the standard photogrammetric solution. If we only have one image, but can constrain the location of the feature, we may also be able to uniquely determine its position. If either the range of a given point from the camera center or the plane in 3D space upon which the feature lies is known, we may also derive a unique solution.

Going the other way, it is easy to see that the transformation projecting the 3D lidar space into the 2D image space is simply the camera matrix P defined in Section 2.2. That is, assuming a central projection-type frame array camera, we may determine the corresponding image location for any 3D feature by using the collinearity constraint given that we know the camera's IOP and EOP. Therefore, in order to precisely register the lidar data to the frame array imagery, we need to determine P.

If we assume that the internal camera parameters are known, the traditional approach to determining the EOP (and hence the registration) would be to manually select homologous image and lidar data points, then perform a standard resection. Since the initial building model is derived in the 3D lidar space, we may opt to use point locations on this model instead of from the actual lidar data. If the model points correspond to locations where planes intersect, and the planes were robustly fit to the data using RANSAC, these points will be very accurate. Therefore, model points would be selected from internal points of intersection, and not from external boundary corners or vertical transitions.

This idea was extended by [Habib et al. 2005] and [Ma 2004] to use linear features in place of point features. As can be seen in Figure 4.34, in many cases there are more of these internal linear features than point features. They also have the added benefit of being easier to extract in the image sets. Both of these references use a traditional Euclidean development to perform the resection using linear features. However, as is discussed in [Hartley and Zisserman 2003], such a solution may also be done using homogeneous coordinates.

Following Hartley's development, if we are given a set of point correspondences $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$, by the collinearity constraint we have the following system of equations:

$$\begin{bmatrix} \mathbf{0}^T & -w_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ w_i\mathbf{X}_i^T & \mathbf{0}^T & -x_i\mathbf{X}_i^T \\ -y_i\mathbf{X}_i^T & x_i\mathbf{X}_i^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{bmatrix} = \mathbf{0}, \tag{4.29}$$

where $\mathbf{x}_i = (x_i, y_i, z_i)$ is the homogeneous representation of the image point, $\mathbf{X}_i$ is the point in $\mathbb{P}^3$, and $\mathbf{P}^{iT}$ is the i^{th} row of P. Since the third equation is linearly dependent on the other two, we may shorten this to

$$\begin{bmatrix} \mathbf{0}^T & -w_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ w_i\mathbf{X}_i^T & \mathbf{0}^T & -x_i\mathbf{X}_i^T \end{bmatrix} \begin{bmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{bmatrix} = \mathbf{0}, \tag{4.30}$$

Next, we obtain a $2n \times 12$ matrix A by stacking up the above equations for n point correspondences ($n \geq 6$). P is then obtained by finding the right nullspace A$\mathbf{p} = \mathbf{0}$, where $\mathbf{p}$ contains the elements of P rearranged into a 12×1 vector. In the over-determined case ($n = 6$), a least-squares solution may be found through the standard SVD-based approach.

In addition to resectioning via point correspondences, we may also use this approach to determine P from a set of 3 or more line correspondences. The plane formed by back-projecting an image line $\mathbf{l}$ is $\pi = \mathrm{P}^T\mathbf{l}$. Therefore, the condition that a point $\mathbf{X}_j$ lies on this plane is given by $\mathbf{l}^T\mathrm{P}\mathbf{X}_j$. If we do this for two points $\mathbf{X}_0$ and $\mathbf{X}_1$, those two points define a line in $\mathbb{P}^3$. Therefore, two equations are obtained for each line correspondence, so stacking 3 equation pairs permits a solution for P.

Given a single point $\mathbf{X_j} = [x, y, z, w]^T$ on a line in the 3D world coordinates and corresponding line $\mathbf{l} = [a, b, c]^T$ in the image, we thus may build a single equation to stack into A according to

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \\ P_5 & P_6 & P_7 & P_8 \\ P_9 & P_{10} & P_{11} & P_{12} \\ P_{13} & P_{14} & P_{15} & P_{16} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = 0 \tag{4.31}$$

which yields

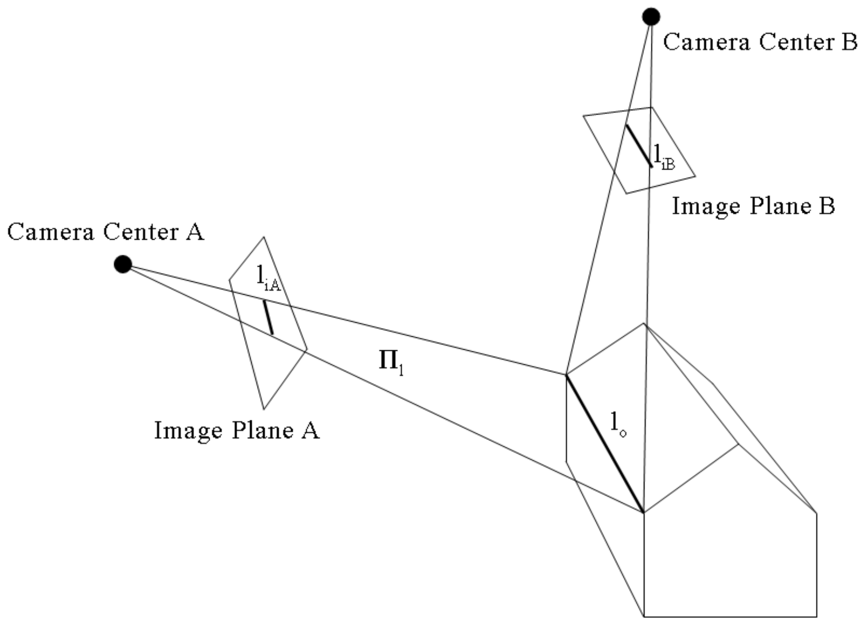
$$\begin{bmatrix} ax & ay & az & aw & bx & by & bz & bw & cx & cy & cz & cx \end{bmatrix} \begin{bmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{bmatrix} = 0. \tag{4.32}$$

This linear method of solving for P is termed the *direct linear transform* (DLT). However, rather than applying this technique as written, a few additional points should be highlighted. First, as noted in [Hartley 1997], data normalization (pre-conditioning) is crucial when working with such an SVD-based algorithm. This normalization may be carried out according to the methods discussed in [Hartley and Zisserman 2003]. Second, the DLT seeks to minimize the norm $||\mathrm{A}\mathbf{p}||$, an algebraic error metric that does not necessarily minimize the residual error in a geometric or statistical sense. In the previous reference, Hartley also discusses several additional (non-linear) algorithms that address this issue. Many researchers have concluded that it is often best to use the DLT to provide an initial estimate of the solution, then use a nonlinear technique to refine this solution. However, for the initial work done in this research, only the linear solution was implemented.

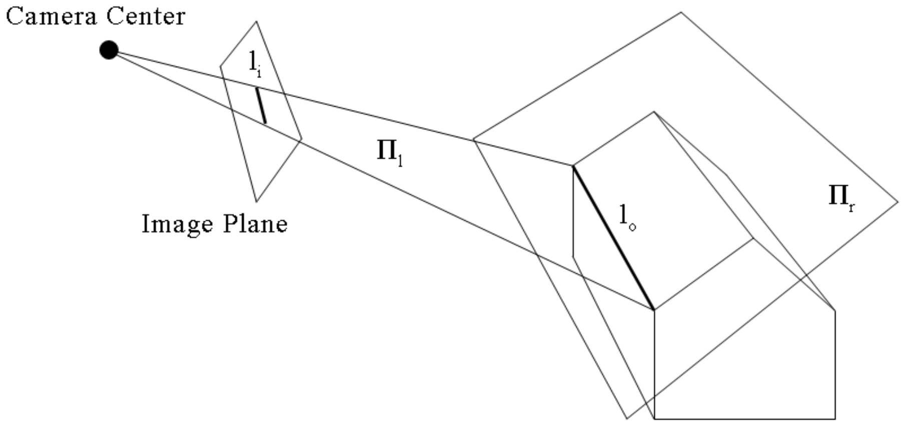
4.6.6 Building Edge Refinement

Once we obtain P in the method defined above for each frame image, we may project line segments detected in these images in the lidar space. If two images are present, a unique solution may be obtained using the geometry illustrated in Figure 4.35 (a). This is the approach taken by [Ma 2004]. By using two images, an explicit definition of the edge line may be found in 3D space without any reliance on the building geometry. However, since we do know the equation for the planes in which each roof facet lies, in this research we obtain the modified edge location using the geometry of Figure 4.35 (b). This permits us to adjust our edges and vertical transitions when only a single high resolution image is available.

In order to find the linear equation for the new edge, we simply take the plane representing the back-projected line $\boldsymbol{\pi} = \mathrm{P}^T\mathbf{l}$ and intersect it with the plane containing the roof facet of interest. The endpoints for the 3D line segment are then obtained by intersecting this new edge with other edge lines, which themselves may be back-projected lines from images.



(a) Traditional dual image approach



(b) Single image approach

Figure 4.35: Improving edge locations with high-resolution imagery.

4.7 Spectral Assignment

In order to assign material properties to the objects in the scene, we need to build a set of spectral reflectance curves for each material. There are two primary ways that this may be accomplished. When quality hyperspectral data are not available, or the spatial resolution is such that the materials of interest are significantly sub-pixel, it is usually preferable to use the available image data to select the best reflectance spectra from a previously collected spectral library. However, when high SNR hyperspectral imagery is available with sufficient resolution to have fully-resolved pixels on each material type, the reflectance spectra may be estimated directly from the imagery.

It should be emphasized that the spectral radiance received by the sensor is not solely based on the reflectivity of the material being imaged. Rather, the received radiance is dependent on the radiation incident to the target, which is then reflected and modified by the atmosphere. These effects must be compensated for before reflectance values may be obtained from the received radiance values. There are many methods of performing atmospheric compensation in the literature.

Physics-based models of the atmosphere such as the Moderate Resolution Atmospheric Transmission (MODTRAN) [Berk et al. 1998] can be used to predict its radiative transfer properties, thereby permitting a ground-based reflectance to be converted to a sensor-reaching radiance or calibrated radiance values to be converted into reflectance units. The Fast Line-of-sight Atmospheric Analysis of Spectral Hypercubes (FLAASH) [Adler-Golden et al. 1999] and the University of Colorado's Atmospheric Removal (ATREM) [CSES 1999] use an approach along these lines.

However, as noted in [Schott 2007], if large (approximately 3 times the ground instantaneous field of view), near-Lambertian ground panels of known reflectance are available, one of the most attractive techniques for recovering reflectance values from the sensed spectral radiance is through the Empirical Line Method (ELM).

To perform the basic ELM per [Schott 2007], we assume a received radiance model of

$$L_{sensor} = \left(\frac{E_s' \cos\theta \tau_1}{\pi} + F \cdot L_d \right) \tau_2 r + L_u, \tag{4.33}$$

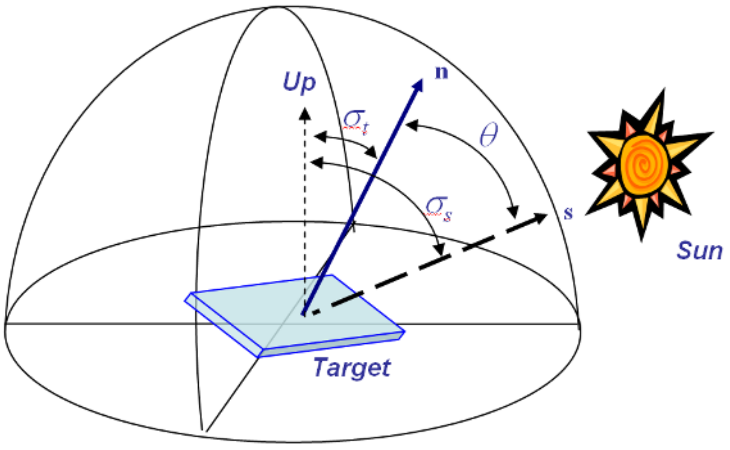


Figure 4.36: Definition of angles.

at each wavelength, where E_s' is the direct exo-atmospheric solar irradiance, L_d is the downwelled radiance, F is a shape factor between 0 and 1 used to scale L_d based on the percentage of sky seen by the target, τ_1 is the sun-target path transmission, τ_2 is the target-sensor path transmission, r is the target reflectance, and L_u is the upwelled radiance. θ is the angle between the target-to-sun vector (**s**) and the target's normal vector (**n**), and is effectively the angle at which direct solar irradiance is incident upon the target. As such, this angle is the same as the solar zenith angle (σ_s) for a horizontal target. This geometry is illustrated in Figure 4.36.

The atmospheric calibration may then be achieved by noting this model is linear with respect to reflectance,

$$L_{sensor} = m \cdot r + b. \tag{4.34}$$

Thus, the slope (m) and intercept (b) may be determined through a regression with known reflectance values for each band. Frequently, these known reflectance values are obtained via ground measurement of reflectance panels, and in the traditional implementation, both b and m are assumed to be constant throughout the scene.

This concept is illustrated for a single spectral band in Figure 4.37 (a). In this figure, the received radiance for a highly reflective target (L_{light}) and the radiance for a less reflective target

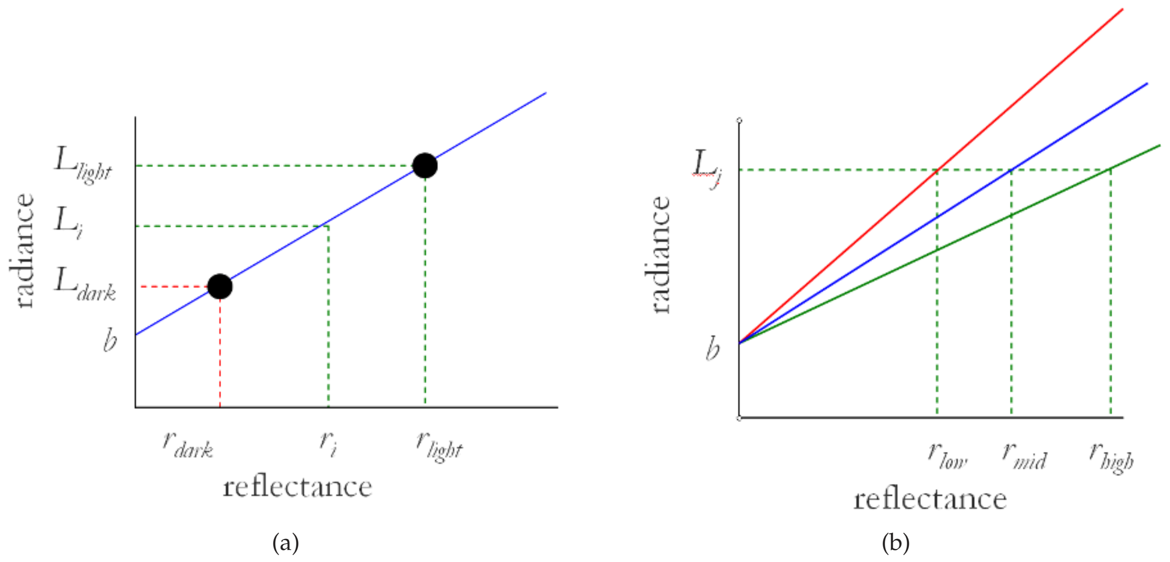


Figure 4.37: (a) Illustration of the ELM concept for a single spectral band , and (b) modifying the ELM slope based on target orientation.

(L_{dark}) are indicated, as are the known reflectance values for each of these materials. By fitting a line through these two points, we have effectively created a function that allows us to convert any other received radiance value L_i (in the specified spectral band) to its reflectance value r_i.

4.7.1 Using Geometric Information to Improve the ELM

In the traditional ELM solution, all ground points are typically assumed to lie in the horizontal plane. However, when lidar data or other three-dimensional information is available, improved results may be obtained by using proper angular values (θ_t) and shape factors (F_t) at each target point being considered. To illustrate how we may take these parameters into account, consider Figure 4.37 (b). Assume we have arrived at the ELM solution represented by the blue line by considering points on light and dark calibration panels, each lying flat in a horizontal orientation. If we then consider target of interest with a received radiance of L_j, using the standard ELM solution would yield a reflectance value of r_{mid} for that material. However, if the target were oriented such that its normal-to-sun angle (θ_t) is larger than the calibration panel's normal-to-sun angle (θ_c), we would actually expect the target to have a higher reflectance than is yielded by the standard ELM. This is easily understood, as the reflecting material would have its received radiance lowered (to L_j) due to the increased area effect caused by the larger θ_t value. In order to compensate for this effect, we would need to reduce the slope of the ELM regression line when processing the pixel related to this target. This is illustrated by the green line in Figure 4.37. Similarly, for targets that face the sun more than the calibration panels do, we would need to increase the ELM slope in order to reduce the resultant reflectance value. This is illustrated by the red line in the same figure. It is also desirable to modify the ELM slope to account for various target shape factors F_t, which may differ from the shape factor for the calibration panels, F_c.

In a manner analogous to that given in [Schott 2007], we may compute the updated ELM slope value m_t for a given target if we know the original ELM slope m, θ_t, θ_c, F_t, F_c and the ratio of downwelled radiance to total received radiance at the target, l. This is accomplished by applying

$$m_t = \left((m - l \cdot m) \frac{\cos\theta_t}{\cos\theta_c} + \frac{F_t}{F_c} \cdot l \cdot m \right), \tag{4.35}$$

where

$$l = \frac{L_d}{E_s' \cos\theta_c \tau_1 \pi^{-1} + L_d}, \tag{4.36}$$

and then completing the ELM solution according to

$$r = \frac{L_{sensor} - b}{m_t}. \tag{4.37}$$

If true values of F are not available, [Piech and Walker 1971] give the following approximation for a point on a sloped plane whose normal vector is an angle σ_p away from zenith:

$$F_p \approx 1 - \frac{1}{2} \cos(\frac{\pi}{2} - \sigma_p). \tag{4.38}$$

In practice, l is ideally obtained through a field measurement at the time of the collection.

However, if such a measurement is not feasible, this ratio may be estimated either through atmospheric propagation models such as MODTRAN or through in-scene techniques such as that presented in [Piech and Walker 1974]. An alternate method for estimating this ratio is also available if fully-resolved pixels of a Lambertian material are available on two planar faces with known orientations (such as a residential roof structure). In this case, the ratio l may be determined at each wavelength by setting the reflectance values of the two planar faces equal to each other. That is, we establish

$$[(m - lm)k_1 + F_1 lm][L_2 - L_u] = [(m - lm)k_2 + F_2 lm][L_1 - L_u] \tag{4.39}$$

at each wavelength, which yields

$$l = \frac{k_1(L_2 - L_u) + k_2(L_u - L_1)}{(k_1 - F_1)(L_2 - L_u) + (k_2 - F_2)(L_u - L_1)} \tag{4.40}$$

where

$$k_1 = \frac{\cos\theta_1}{\cos\theta_c}, k_2 = \frac{\cos\theta_2}{\cos\theta_c}, F_1 = \frac{F_{target\_1}}{F_c}, F_2 = \frac{F_{target\_2}}{F_c}. \tag{4.41}$$

In this development, θ_i is the angle at which the solar irradiance is incident upon the i^{th} facet ($i = 1, 2$), and L_i is the received radiance from a pixel on the image of the i^{th} facet. When applying Equations 4.35, 4.37 and 4.40, we have termed the solution the geometrically-compensated empirical line method (GC-ELM).

4.7.2 Brightness-Derived Orientation Mapping

If the orientation angles needed for the GC-ELM are unknown, this geometry may be estimated from image brightness values if multiple images are available and certain constraints are met. This approach is termed Brightness-Derived Orientation Mapping (BDOM), and it is essentially a shape-from-shading approach to estimating a plane's azimuth and elevation angles. Two cases of this approach are presented below. The first, termed *known/unknown*, assumes that the orientation of one of the surfaces is known, and we solve for the orientation of the second surface. In the

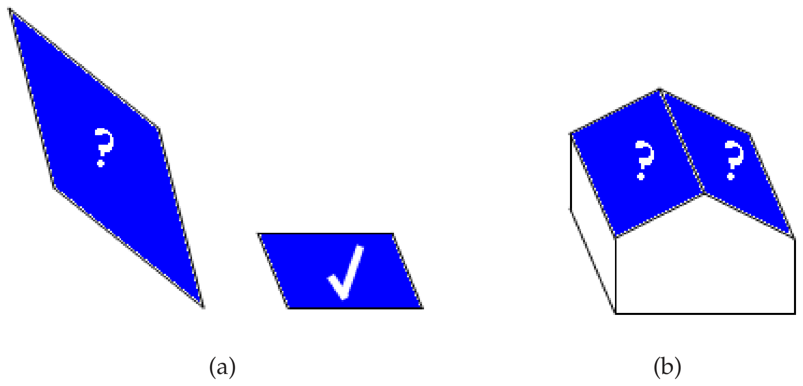


Figure 4.38: BDOM cases: (a) known/unknown, and (b) Rooftop. Check indicates the orientation of the surface is known, while '?' indicates that it is not.

second case, termed *rooftop*, we do not know the absolute orientation of either surface, but we establish a constraint on the relative geometry of the two surfaces. These two cases are illustrated in Figure 4.38.

In order to complete the BDOM derivation, a simplified radiance propagation model is required. Typically, at higher wavelengths (above 1 micron), both the L_d and L_u terms from Equation 4.33 become negligible compared to the direct solar term. This is illustrated in Figure 4.39, where we plot $E_s' \tau_1 \pi^{-1}/(L_d + L_u)$ versus wavelength. The data for this figure was obtained from a DIRSIG simulation, where a mid-latitude summer/rural atmosphere was specified.

With the contributions of the downwelled and upwelled radiances removed, Equation 4.33 is simplified to

$$L_{sensor} = \frac{E_s' \cos\theta \tau_1 \tau_2 r}{\pi} = A \cos\theta r. \tag{4.42}$$

We now need to assume that we have a Lambertian material with reflectance r at two locations a and b, and each location is on a separate planar surface. These locations are imaged at times $t1$ and $t2$ to provide four brightness values, L_{a1}, L_{a2}, L_{b1} and L_{b2}. Since the reflectance of each point

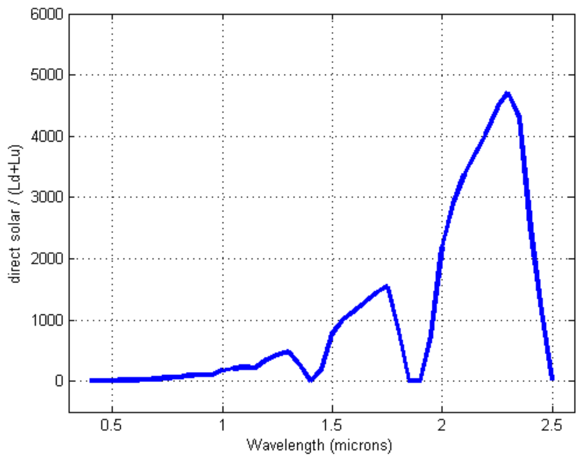


Figure 4.39: The ration of direct solar to downwelled plus upwelled radiances as a function of wavelength.

is the same, we are able to write two equations

$$\cos\theta_{ai} = \frac{L_{ai}\cos\theta_{b_i}}{L_{bi}}. \tag{4.43}$$

where i represents either the image at $t1$ or $t2$.

For the first BDOM case (known/unknown), we assume the orientation of one of the planar faces (assume location b) is known (often this surface is assumed to be horizontal). Therefore, if we know the solar geometry at both $t1$ and $t2$, we may easily determine θ_{bi}. This information is then used to solve for θ_a at each imaging time, which may subsequently be used to determine two azimuth/elevation angle pairs, each of which is a valid solution to the equations. The actual orientation of the unknown facet will be one of these solutions.

We may conceptualize the geometry according to Figure 4.40. The image at time $t1$ reveals that the normal vector of our unknown surface is a given angular offset θ_{a1} away from a vector pointing from the unknown facet towards the sun. That is, the potential orientations of the unknown normal vector sweep out a cone of angular width $2\theta_{a1}$ centered on this solar vector, as shown in (a). The second image sweeps out a cone of width $2\theta_{a2}$ about the vector pointing to the sun at time

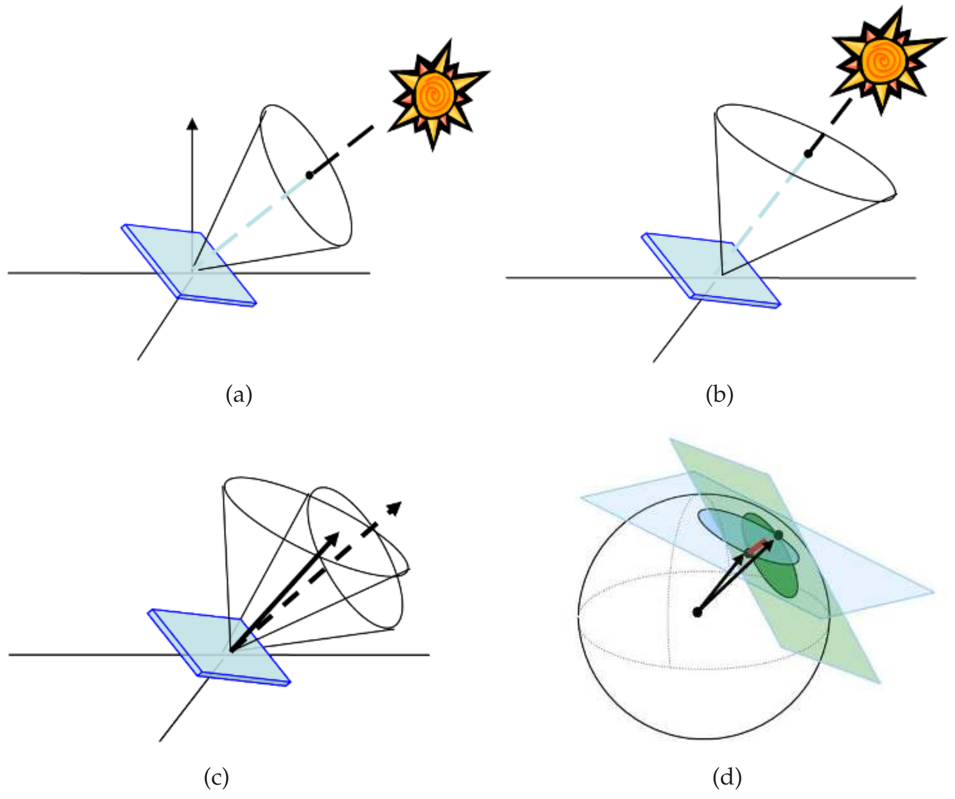


Figure 4.40: Illustration of the BDOM known/unknown case solution: (a) Possible **n** at $t1$, (b) possible **n** at $t2$, (c) two solutions for **n**, (d) two planes intersecting the unit sphere representation .

$t2$, as shown in (b). In general, these two cones intersect along two lines, and these lines define the potential normal vectors for the unknown surface, shown in (c). One line represents the actual orientation of the surface, while the other is an artifact of the ambiguity in the solution. It is fairly easy to see that if three imaging times are used, this ambiguity is effectively removed.

The actual equations used in isolating the target's normal vector $\mathbf{n} = [n_1, n_2, n_3]^T$ given θ_{a1}, θ_{a2}, and the two target-sun vectors $\mathbf{s_i} = [s_{i1}, s_{i2}, s_{i3}]^T$ for $i = (1, 2)$ are

$$\begin{aligned} s_{11}n_1 + s_{12}n_2 + s_{13}n_3 &= \cos(\theta_{a1}) \\ s_{21}n_1 + s_{22}n_2 + s_{23}n_3 &= \cos(\theta_{a2}) \\ n_1 + n_2 + n_3 &= 1, \end{aligned} \tag{4.44}$$

which actually represent two planes and the unit sphere. As illustrated by Figure 4.40 (d), each plane intersects the unit sphere along a circle, that when considered with the origin effectively describes one of the cones highlighted in (a) and (b). The two solution points are where these two circles intersect, and the vectors described by these two points' coordinates are the same as those shown in (c).

The full derivation of the BDOM known/unknown case is straightforward but lengthy, and the closed form solution is not included here. In practice we have also found that this closed-form solution may be ill-conditioned for certain orientations. As such, we usually simply cycle through many combinations of azimuth and elevation candidates until the closest equality for Equation 4.43 is obtained.

For the second BDOM case, we do not require knowledge of either surface's orientation. However, we do assume that the two surfaces have equal elevation angles and opposite azimuths (an offset of π radians). This is the common configuration found on most residential roof structures. With this constraint, we are able to determine the azimuth and elevation angles of each surface without ambiguity, provided the Lambertian and other previously noted assumptions are met.

In this case, we again assume four instances of Equation 4.42, one each for brightness values, L_{a1}, L_{a2}, L_{b1} and L_{b2}. Assuming unit length vectors, we may therefore write each equation in the form

$$L_{j,i} = A_i(\mathbf{s_i} \cdot \mathbf{n_{j,i}})r, \tag{4.45}$$

where $\mathbf{s_i}$ represents the vector from the target to the sun at time i, and $\mathbf{n_{j,i}}$ is the unit normal vector for surface location j at time i. A_i is a constant for each time i. These equations may be re-written in terms of the unknown azimuth (α) and elevation (δ) angles according to

$$\frac{L_{j,i}}{A_i} = s_{i1}\cos(\alpha_{j,i})\sin(\delta_{j,i}) + s_{i2}\sin(\alpha_{j,i})\sin(\delta_{j,i}) + s_{i3})\cos(\delta_{j,i}), \tag{4.46}$$

where s_{ik} represents the k^{th} vector component of s at time i. Equating the reflectances of the two roof surfaces at a given time yields the equations

$$L_{bi}[s_{i1}\cos\alpha_i\sin\delta_i + s_{i2}\sin\alpha_i\sin\delta_i + s_{i3}\cos\delta_i] = \tag{4.47}$$
$$L_{ai}[s_{i1}\cos(\alpha_i+\pi)\sin\delta_i + s_{i2}\sin(\alpha_i+\pi)\sin\delta_i + s_{i3}\cos\delta_i].$$

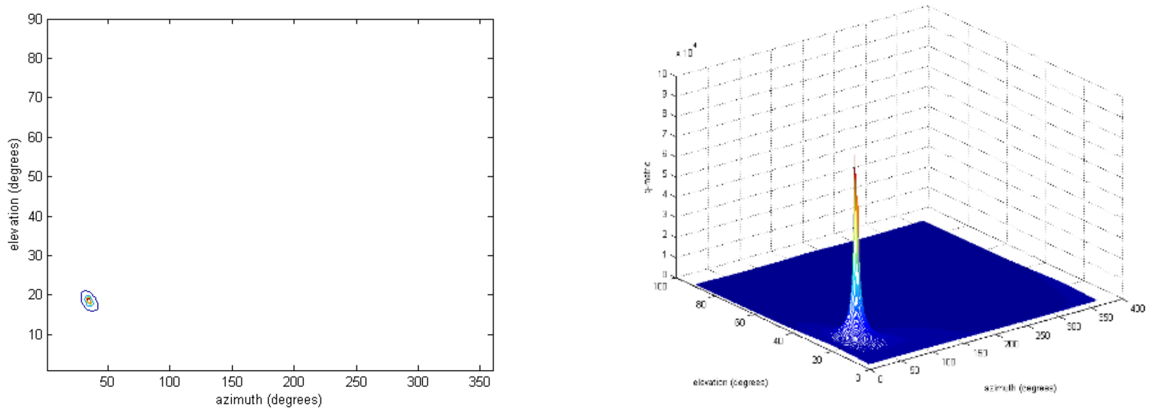
Again, the analytical solution of the azimuth and elevation angles from these equations is straightforward but lengthy, and it will not be included here. Also as before, in practice a useable solution may be found by cycling through combinations of azimuth and elevation pairs until an approximate solution is found. We have used Equation 4.47 to define the quality metric $q = 1/\epsilon$, where

$$\epsilon = |L_{b1}Q_1 - L_{a1}Q_2| + |L_{b2}Q_3 - L_{a2}Q_4| \tag{4.48}$$

and

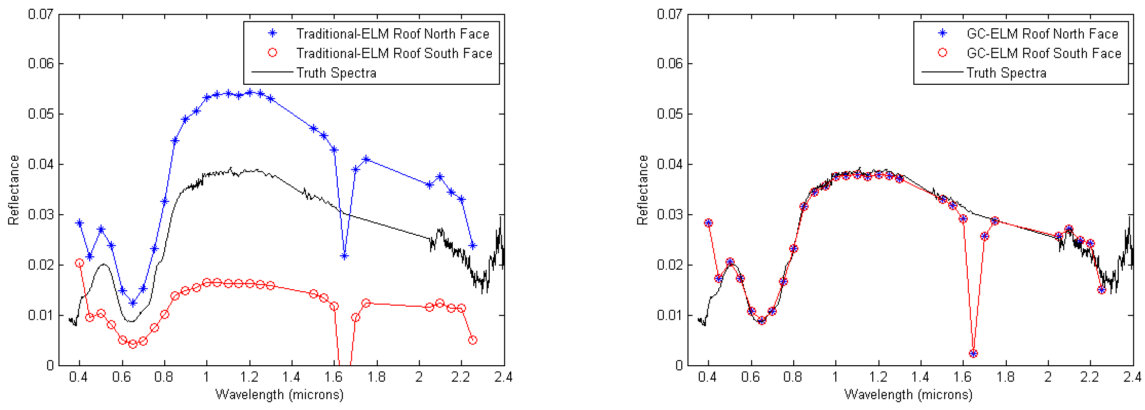
$$Q_1 = s_{11}\cos\alpha\sin\delta + s_{12}\sin\alpha\sin\delta + s_{13}\cos\delta \tag{4.49}$$
$$Q_2 = s_{11}\cos(\alpha+\pi)\sin\delta + s_{12}\sin(\alpha+\pi)\sin\delta + s_{13}\cos\delta \tag{4.50}$$
$$Q_3 = s_{21}\cos\alpha\sin\delta + s_{22}\sin\alpha\sin\delta + s_{23}\cos\delta \tag{4.51}$$
$$Q_4 = s_{21}\cos(\alpha+\pi)\sin\delta + s_{22}\sin(\alpha+\pi)\sin\delta + s_{23}\cos\delta. \tag{4.52}$$

Figures 4.41 and 4.42 illustrate the utility of these techniques by example. After creating a simple synthetic scene consisting of a shed structure and its surroundings (the 'SAS' scene discussed later in Section 5.4), a synthetic hyperspectral image of the scene was produced using DIRSIG. This synthetic image was then processed using the methods discussed above. Figure 4.41 shows the result of applying the BDOM rooftop approach to two of the roof pixels in the image. As can be seen in (a), the unknown orientation was determined to have an azimuthal angle of approximately 40 degrees and an elevation of 20 degrees, each of which was accurate to within 1 degree of truth. The second plot of this figure shows the relative sharpness of the quality metric's peak.



(a) (b)

Figure 4.41: Quality metric q for BDOM rooftop example: (a) Contour plot and (b) 3D surface plot.



(a) (b)

Figure 4.42: ELM vs. GC-ELM results (DIRSIG simulation): (a) True roof spectrum and traditional ELM-derived reflectances for north and south roof face, and (b) GC-ELM results.

Once the roof orientations were obtained, atmospheric compensations based on the ELM were performed. Figure 4.42 (a) shows the true roof reflectance spectra, as well as the results obtained from a standard ELM process applied to the simulated image. Note that the northern roof face, whose normal pointed more to the sun, had its reflectance estimates too large, while the southern face values were under-estimated. Compare this with (b), where the truth spectra is now compared with reflectance spectra obtained through the GC-ELM approach. Note that at nearly all frequencies the result shows a marked improvement, and the offset between the northern and southern roof spectra has been removed.

It should be emphasized that when using the baseline ELM algorithm, the raw digital counts from the sensor may be used as the effective received radiance values, as the sensed values do not have to be calibrated for this technique to work [Schott 2007]. This insensitivity to sensor calibration is also applicable for the GC-ELM and BDOM techniques discussed above.

4.7.3 Additional Considerations

Note that Equation 4.40 assumes that all parameters, including L_d, come from a homogeneous sky. In practice, the atmosphere is not uniform for all pixels in an image, and at a given location, the reflected skylight is not uniform in all directions. Also, this model does not account for the BRDF effects of non-Lambertian surfaces. As such the solution provided is merely an approximation to the actual material reflectance spectrum. As noted before, we may also use a tool such as FLAASH to perform our atmospheric compensation. When the input parameters to FLAASH are available and the hyperspectral imagery is well calibrated, this approach may be preferable. However, with many of the datasets used for this work the spectral calibration was suspect. As such we often had difficulty in getting FLAASH to yield reasonable results.

Instead of using the atmospherically-compensated spectra directly, we also have the option of using the sensed data to select spectral curves from a previously defined spectral library. To do this, we first atmospherically compensate the data. We then separate the scene into multiple class types using a supervised minimum Mahalanobis distance classification. Typically, the ENVI image processing tool is used for this purpose. Each material type is then matched to a library material type according to which library spectral curve best fits (in a Euclidean distance sense) a locally-averaged spectrum (within a 5×5 pixel window) for that material. This approach produces spectral results that are more in line with the current scene production process. However, a drawback is that the scene being considered has a limited spectral set with which to work.

DIRSIG assigns spectra to spatial regions in the scene using two complimentary methodologies. The first of these is on a single material per-facet basis, where material types and corresponding reflectance spectra are directly assigned to object facets. This approach is used to assign spectra to building roof facets and other objects in the scene.

The preferred method of assigning ground spectra to the DIRSIG terrain model is through the use of spectral material and texture maps. In order to specify spectra using this approach, a spectral classification routine is used to assign a material identification index to each location of a chosen facet or group of facets. A texture image is then used to select an appropriate spectral curve from a library of spectra of that particular material type for each location on the facet. As such, two adjacent roof shingles may both be classified as ”type 1 asphalt roof shingles”, but their assigned spectra may be different due to differing gray levels in the texture image.

In order to fit a material map based on a frame-array image to a vertical building facet, perspective effects inherent to the imaging process must first be removed. Each point correspondence matching a specific image location to a CAD vertex produces two constraints on the projection needed to orthorectify the image of the facet. As such, four point correspondences allows the recovery of the projection matrix defining the perspective effects, as well as the inverse transform which can remove them. Applying this inverse transform to all pixels belonging to the facet of interest yields the desired orthographic view of the planar feature.

Typically, we generate material maps for the terrain via a supervised classification of a nearly orthonormal RGB or spectral image. An unprocessed image of the same type is often used for the terrain texture image. For building surfaces, oblique views of RGB images are reprojected such that the facet of interest matches the geometry of the CAD model. Currently, the features used to determine the appropriate projective transform are derived manually, but research to automate this process is ongoing.

4.8 Alternate Methods: Missing Image Modalities

Up to this point, we have assumed the availability of high resolution lidar data, nadir looking frame-array imagery, obliquely orientated frame-images, and well-registered hyperspectral data with a ground sample distance (GSD) smaller than the model facets. However, in most practical cases, these data are not usually all available. Frequently, the hyperspectral information will have a GSD larger than the largest building in the scene, or it may have serious registration artifacts. In other cases, hyperspectral information may not be available at all, and we may have to do the best we can with limited-band multispectral images or no spectral information at all. Perhaps even more significant to the presented process, even with the recent proliferation of lidar sensors, many data collects are restricted to passive imaging sensors. In order be able to build scenes when confronted with these reduced data sets, this section presents some strategies for modifying the previously-discussed process. It should be noted when we consider what image types we might have available for a given data collection, the number of potential combinations is extremely large. To this end, the following does not attempt to cover every possible case; rather, general strategies are presented, and it is up to the user to intelligently decide how to best use the available data in any specific scenario.

4.8.1 Degraded or Missing Hyperspectral Imagery

If high resolution, well-calibrated, hyperspectral radiance images are available, we may assign spectra as described earlier, where the sensor is used as a field spectrometer. In this case, the received radiance values are converted to reflectance curves using an approach such as that used by FLAASH [Adler-Golden et al. 1999], and these reflectance curves are assigned to the various objects or locations in the scene. If the hyperspectral imagery is not well calibrated, we may still use the sensor as a spectrometer if we know (or can estimate) two or more spectra in the scene. This is when techniques such as the standard or geometrically-compensated ELM become attractive.

If the hyperspectral imagery is not well calibrated and has poor noise performance, we will not usually want to use the data to directly define reflectance values. Rather, we use the compensated imagery to pick appropriate reflectance spectra from a library of previously measured materials. In this research, this was done when using MISI imagery by first mitigating atmospheric effects via an ELM, then choosing a material type by minimizing the Euclidean distance (or spectral angle in some cases) between the compensated imagery and a set of library spectra. The library spectra are grouped into various material types, where each material type is populated by multiple spectral curves. Each image pixel is matched to a particular library curve, but only the material type for that particular curve is used. For a given material class (as defined by a material map), facet or object in the scene, the material type is then chosen according to which material type is most frequently chosen in the matching process. Individual spectra are still assigned through the use of texture images and are not dependent on the hyperspectral imagery.

If the hyperspectral imagery has a large GSD (larger than most building roof facets, but smaller than the local regions defined by the material map), it may still be used to define reflectance curves. In regions where the scene does not contain any objects (such as large grassy patches), the spectral imagery may be used as though it had a small GSD. In cases where subpixel objects are present in a larger background region, a few options are available. First, a standard spectral unmixing process could be performed at these locations using a selected few library spectra as endmembers. An alternative approach is to use a physics-based, radiance space approach such as that presented in [Healey and Slater 1999], [Ientilucci 2005], and [Foster 2007]. This concept will be further explored in Chapter 7.

In addition to the use of hyperspectral data for the definition of spectral reflectance properties, the baseline process also advocates the use of this data for creating the material map. In many cases this is possible even if this data is noisy or uncalibrated, as we may often be able to perform a simple supervised classification without concern for the absolute nature of the underlying data. However, if the hyperspectral data is not well registered to the other data sources, we will often have to use other image sources (such a multispectral, or even RGB imagery) to produce this map.

In many instances where we need to use RGB imagery for the material map, a simple supervised classifier using distance metrics on the spectral feature vectors will not produce adequate results. In these cases, using a standard color segmentation routine (such as found in [Hill et al. 2003] or [Santos et al. 2007]) should be applied to the imagery. The user is then required to manually combine classes as appropriate. Although this process loses much in terms of the automation, in practice, little time is lost, and an accurate material map may usually be created in a matter of several minutes.

4.8.2 Only Lidar Data is Available

In many cases, most of the scene reconstruction may still be performed semi-autonomously if the only data available is the lidar point cloud. In this scenario, the DTM extraction process is unchanged. The initial tree/building may still be performed using the GML classifier. However, without the subsequent NDVI refinement, additional steps may need to be performed to properly isolate trees that are adjacent to buildings. The fundamental problem is that most of the features used by the GML classifier are derived using a local window, so the inherent blurring may cause tree and building regions to mix at the building boundaries. This is especially true since the building boundaries already have significant 'tree-like' properties than would be desired due to the rapid height variation when transitioning from the building roof to the ground. In this work, we have opted to refine building/tree boundaries via a manual region-of-interest (ROI) selection process, although the use of intensity data to refine edge features has also shown promise. The intensity value is not averaged over a small region, and therefore fine detail may be extracted if the building roof and adjacent tree points have significantly different reflectance properties at the wavelength used by the laser scanner.

Buildings are reconstructed using the described method, although the photogrammetric edge refinement stage is omitted. Individual trees are selected from a library using parameters extracted via the generalized ellipsoid model. The tree models are then scaled so that they match the ellipsoid height and width parameters. For clustered trees, a standard watershed analysis is performed on a blurred version of the rasterized DCM to identify tree center and boundary estimates, and the boundaries are adjusted so that the individual tree areas fall within predefined limits.

A material map may be created directly from the lidar data by performing a color segmentation on a color image where height and intensity values are mapped to the red and green channels. We have used many different items to populate the blue channel, including zero values, height, intensity and a ratio of height to intensity, and they have all shown varying degrees of success depending on the data set being considered. The fundamental goal is to use features than enable an over-segmentation of the image, then have the user manually combine the segmented classes as appropriate. The texture image is a finely sampled (0.25m) version of the rasterized I_A image. Without any real spectral information (other than the returned pulse intensity), spectra should be manually assigned.

4.8.3 Only Passive Imagery is Available (No Lidar Data)

Considering the wealth of geometric information provided by the lidar point cloud, the unavailability of lidar data poses perhaps the greatest hindrance on performing semi-autonomous scene construction. For the past ten years or so, great progress has been made in extracting DEM products from multiple-view frame images of significant resolution. However, current state-of-the-art systems are generally still unable to handle full scene generation in an urban environment. In [Kim 2001] and [Kim and Nevatia 2004], the authors present an approach to semi-autonomous segmentation of building and tree regions from multiple perspective images using a Bayesian network, but the general applicability of this technique is still undetermined. As was noted in Section 4.6.1, several techniques have also been proposed for autonomously creating building models from stereo image pairs, and a method for producing similar results from standard video sequences is describe in [Gurram et al. 2007]. Gurram's approach is to first build a pair (or triplet) of parallax-preserving stereo mosaics from the video sequences, then process these mosaics for 3D scene information. Rather than using a standard edge detector for identifying linear features corresponding to facet edges, he first segments each mosaic based on the approach presented in [Hill et al. 2003], then attempts to build segment correspondences between the mosaics. For segments present in both mosaics, the 3D positioning of the segment may be determined, and coplanar, adjacent segments are merged. A research effort being conducted in parallel with the one described in this dissertation employs a similar technique applied to standard frame-array image pairs. Unfortunately, these approaches all suffer from occlusion effects, and may perform erroneously in the presence of solar shadows, incorrect camera calibration, or mis-segmentation. These shortcomings are actually some of the primary drivers for the recent push for fusing lidar data with more traditional photogrammetric techniques.

Since autonomous, purely-photogrammetric techniques are still several years away from working across generalized scenes, most vendors still perform the majority of their building reconstruction tasks using photogrammetric software tools that require significant user interaction. It is expected that as the computer vision and photogrammetric processes advance of the upcoming years, this will change. However, until these advancements occur, it is recommended that scene geometric content be derived either through the current DIRSIG procedures or through the use of the digital photogrammetric tools previously mentioned. Once the geometry has been specified, the additional scene content may be created using the methods presented in this work.

"No effect that requires more than 10 percent accuracy in measurement is worth investigating."

Walther Nernst (1864-1941) German physicist, chemist. Nobel prize, 1920

"They've done studies, you know. 60 percent of the time, it works every time...."

Brian Fantana, from the movie *Anchorman*

5 Results and Discussion

In this chapter, we demonstrate the approach by applying the newly developed techniques to several real-world scenes. The first scene we consider is a 26 acre portion of the RIT campus. This region has a sloping terrain and contains both isolated and grouped trees. It is used to illustrate a basic implementation of the entire synthetic scene-generation process as well as to analyze the terrain extraction and tree modeling processes in greater detail. Although there are two buildings in this scene, they do not have sloped roof surfaces. Therefore, in order to fully demonstrate the building reconstruction techniques implemented in this work, we consider a second region which contains two geometrically-complex residential structures. The third scene represents the only region where we had high-resolution, low-noise hyperspectral imagery in conjunction with lidar

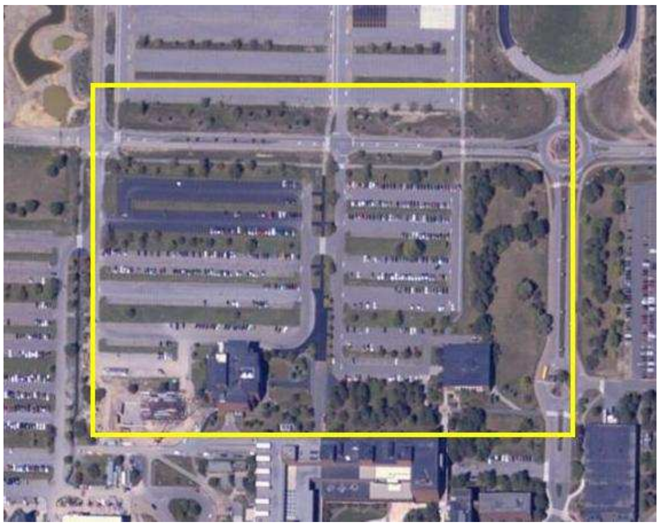


Figure 5.1: RIT scene.

and RGB frame array images. Data collected over this region is used to create a small (1.9 acre) full-fidelity synthetic scene which is subsequently compared to a synthetic scene produced using current DIRSIG procedures and tools.

5.1 Illustration of the Process: RIT Scene

The first location we consider is from a portion of the RIT campus, as highlighted in Figure 5.1. This area covers approximately 390 meters in the east-west direction and 270 meters north-to-south, and it contains two full buildings, three tree clusters, numerous isolated trees, grass, asphalt and concrete. The buildings contained in this scene include the Carlson Center for Imaging Science at the lower left and the Bausch and Lomb Building at the lower right. A third, newly-constructed building is partially contained at the south-western corner of the scene and is present in the lidar data, but this building had not been constructed at the time the frame imagery was acquired. There is more than a 10 meter change in ground elevation throughout the region, and a significant amount of truth data for this locale has been collected. As such, this has proven to be an excellent test scene for the techniques implemented in this research.

The data used in modeling this scene came from the following sources:

- Lidar point data were collected by Leica Geosystems flying a commercial Optech ALS-50 linescanning sensor. Multiple data sets were provided, of which two were selected for this analysis. The first contains approximately 5 points/m^2, but sampling is significant more dense in the cross-track direction since the east-west flightline was approximately 300m north of the scene being considered. The second data set contains approximately 13 points/m^2, and is roughly uniform in both the in- and cross-track dimensions. Multiple-return range and intensity data were provided.

- Hyperspectral imagery is from RIT's Modular Imaging Spectrometer Instrument (MISI), the 70-band VNIR linescanning spectrometer with a 3m ground resolution described in Chapter 2. Low temporal resolution GPS information (one update per second) was available for this image, but the complementary inertial measurement data describing the aircraft orientation contains only roll and pitch data. At the time of this collection, MISI's navigation system did not record yaw data.

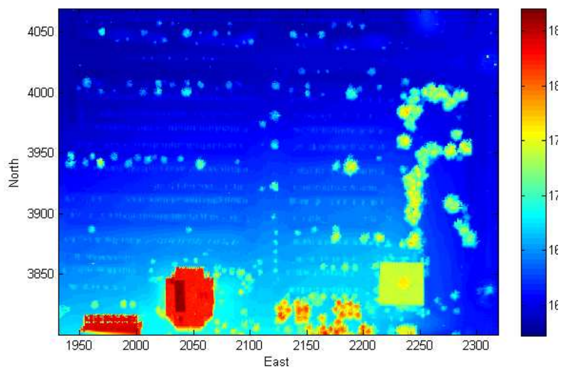
- Color imagery was provided by the Wildfire Airborne Sensor Program (WASP) high-resolution (half-meter) RGB frame-array sensor. Although this camera was co-mounted with three lower resolution (3m) IR cameras operating in the short-wave, mid-wave, and long-wave regions, only the RGB imagery was used for this scene. This sensor was introduced in Chapter 2.

The primary reason for selecting this scene for our initial modeling effort was to investigate the performance of the slope-based DTM extraction technique described in the approach. However, due to the rich features in the area, this region has also proven to be an excellent candidate for demonstrating a basic implementation of the entire synthetic scene construction process. Therefore, the remainder of this section will show how the presented techniques may be used to rapidly construct a simple, yet physically-realistic model of the scene. For this initial demonstration, we will not attempt to use high fidelity models at each stage. Rather, the focus will be on showing the general framework of the approach and a basic output of each step. Construction of a higher fidelity scene will be addressed later in Section 5.4.

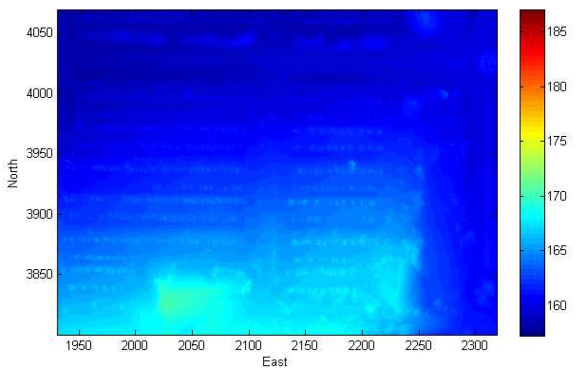
The baseline result considers all points from the lower density lidar data, but points with identical (x, y) coordinates were filtered so that only the highest of these points was retained. The filtered point set was subsequently rasterized by performing a bilinear interpolation to a grid with 0.5m x 0.5m pixels. This resulted in the R_A image shown in Figure 5.2 (a). In this figure, the x- and y-axes represent the pixel coordinates in the Universal Transverse Mercator (UTM) Zone 18 projection (after subtracting 280,000 from the easting coordinate and 4,770,000 from the northing coordinate to ease the notation), a system where the coordinates represent the offset in meters from a given origin for each zone. The elevation (height) values (z-axis) are indicated by color, where dark blue represents lower points and red higher points. These elevation values are in meters above the geoid.

We then applied the slope-based terrain extraction algorithm. This was done by defining a 35 meter radius, conical structuring element with a slope of 20 degrees, then eroding the R_A image using a grayscale morphological operation. Pixels in the R_A image more than 0.5 meters above the corresponding pixels in the eroded image were then flagged as non-ground pixels. The pixels were then re-organized as points at the grid locations, and the non-ground points were removed from the point list. The remaining (ground) point were then re-interpolated to the 0.5 m raster to produce the rasterized DTM. The result of this process is given in Figure 5.2 (b), and the pixels removed in the process are given in (c).

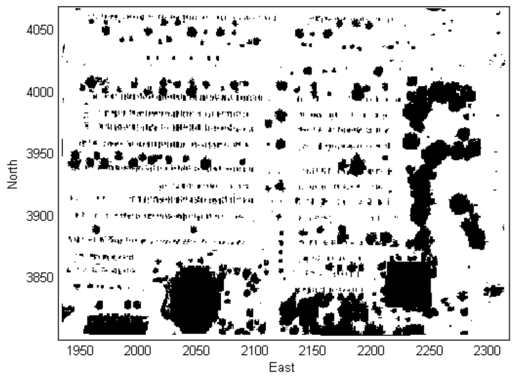
Note that all building and tree points have effectively been removed and replaced with interpolated values. Additionally, many of the smaller structures such as vehicles and shrubbery have also been eliminated. Since the removed points were filled in with a bilinear interpolation of the surrounding points, the outline of these features is still discernible to the human observer. This is because the local height variations in these regions is lower than that of the terrain in general,



(a)



(b)

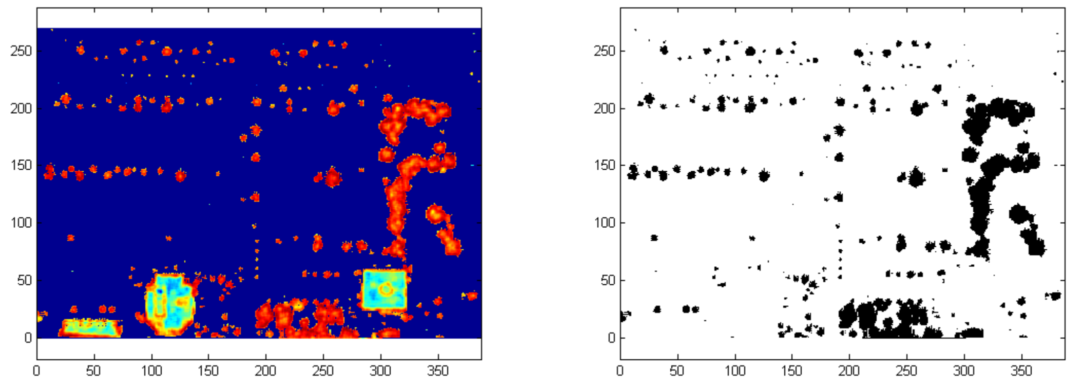


(c)

Figure 5.2: Terrain extraction: (a) Original R_A image (b) Extracted DTM using slope-based approach (c) Location of pixels removed in creating the DTM.

and the texture difference allows us to perceive the object locations. This effect may be reduced by passing the resultant DTM through a low-pass filtering operation, but this comes at the expense of reduced fidelity in regions of relatively rapid terrain transitions. Also, in many cases this is not needed, as many of the objects will be modeled and inserted at these locations in the final simulated scene.

In order to use this extracted terrain model, it has to be converted to a DIRSIG-compatible format. DIRSIG accepts object in either the wavefront '.obj' format or the DIRSIG specific '.gdb' format, although in each of these cases, a material identification index number has to be associated



(a) (b)

Figure 5.3: Initial tree/building segmentation: (a) Entropy image (b) Extracted tree regions using only the entropy texture metric.

with each facet. In this work, material ID '100' was assigned to each facet of the terrain using DIRSIG's Bulldozer tool. Bulldozer accepts .obj, .gdb, and autoCAD .dxf files as input, and allows the user to specify material IDs to these files at either the object or facet level. The file is then saved in the .gdb format which will be referenced in the .odb file that consolidates the scene geometry. It should be noted that in some cases, the Matlab code generated for this work created .obj files that technically fit the wavefront specification, but were unable to be opened by Bulldozer. In these cases, the .obj files were first opened in the Rhinoceros CAD tool, and immediately re-saved as an .obj file. This process effectively converts all of the resultant .obj files into a format compatible with Bulldozer. In future work, the terrain model (and other CAD objects) will be stored directly as .gdb files without the need to convert them using other software utilities.

Once the terrain model has been obtained, the next step is to subtract the DTM from the R_1 or R_{max} image and produce a NDEM. NDEM points higher than 2 meters are then flagged as 'objects', and need to be classified as either building or tree. For the simple case of using a single texture feature, the GML classification process discussed in the approach is reduced to a simple thresholding operation. In many cases, this is enough to provide a decent separation of the buildings from the trees, with the exception of those trees which are adjacent to buildings. For the RIT scene we used the entropy image (Figure 5.3 (a)) for this purpose, and the resulting initial tree locations are shown in Figure 5.3 (b). Note that all trees have been identified, with the exception of those that are adjacent to buildings. In more complicated scenes, additional texture measures are often required, and the full (supervised) GML approach must then be used.

Once the tree regions have been identified, individual trees are located via a watershed analysis. Isolated trees are subsequently analyzed using the generalized ellipsoid model of Equation 4.18. This model is used to help define height, width, shape and location parameters, which both enable selection of the proper tree type from a library of tree models as well a a fine-tuning of these models so that their geometry fits the data as close as possible. In general, as noted in Chapter 4, a second supervised GML classification may also be performed where the direct shapes parameters and secondary parameters (such as height to width ratio) are feature vectors used to classify trees as deciduous or evergreen. However, for this scene, a single tree model (a red maple CAD object with associated spectral information stored in .gdb format) was used, and the GML-based classification was deferred the higher-fidelity scene.

Once the trees were specified, the original data points in the 'building regions' are processed to extract simple building models. Instead of performing the full intersection of planes building analysis, for this scene the raw points were used to extract the building boundary, and a simplified flat roof building reconstruction was employed. This approach is suitable to scenes where precise building roof models are not required, and the roof facets predominantly lie in planes parallel to the (x, y) plane. In these cases, the manual intervention required to verify the correct handling of each vertical transition in the roof structure is not warranted, and a single height value for the entire roof is used instead. In most cases, we use the median roof height when building these simplified models.

Although the roof models are simplified to reduce the required manual intervention, the full robust boundary extraction technique was used. The resultant models were then output in .obj

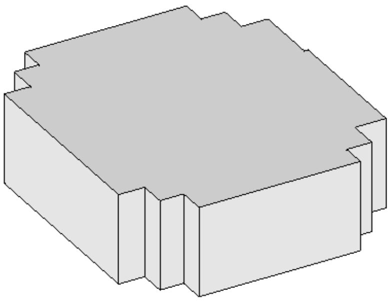


Figure 5.4: Simple CAD model of the Carlson building using exterior boundary information and the median height value.

format, re-saved as .obj files using Rhinoceros, and assigned material indices using Bulldozer. An example CAD model produced using this streamlined approach is given in Figure 5.4.

In order to obtain spectral reflectance curves associated with each location in the scene, we next attempted to register the MISI hyperspectral imagery with the lidar data. By using either a standard ELM approach (GC-ELM was not required since the scene contained few sloped surfaces), this imagery was converted to ground reflectance measures. By co-registering the data, these reflectance spectra could then be used to pick proper materials from a previously-collected materials database. For the case of low signal-to-noise hyperspectral data such as that provided by the MISI sensor, this library selection process is usually preferable to directly assigning the ELM-derived spectra as the reflectance curves for each material type.

To this end, we used the projection approach of Section 4.3.3 to geo-rectify the MISI hyperspectral imagery. The result of this rectification is shown in Figure 5.5. The left image depicts the raw hyperspectral image as collected, and the image on the right shows the result of projection and re-sampling one of the data bands. Although the geo-rectified image shows a marked improvement, a detailed inspection revealed spatial artifacts in excess of 25m throughout much of the image. It is believed that these artifacts are the result of two main factors. First, the inertial measurement

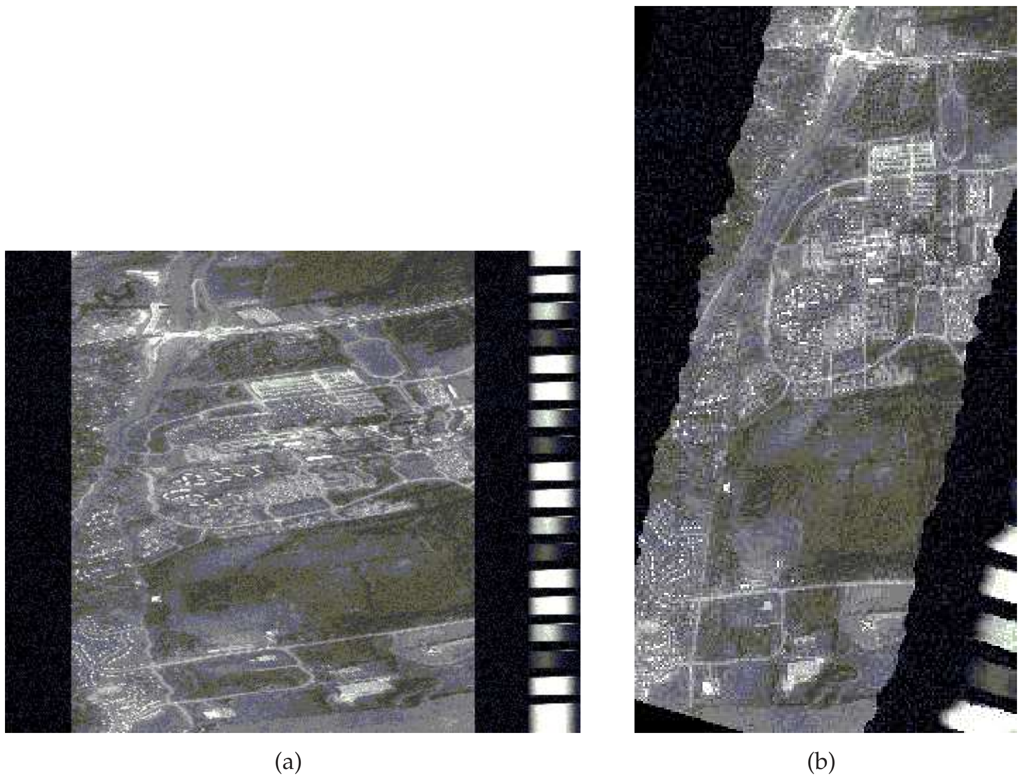


Figure 5.5: Un-registered MISI image (left) and geo-rectified image (right).

unit used for the MISI sensor usually has marginal performance at best, and the sensor drift is significantly higher than for the inertial sensors used on the WASP imager. Additionally, at the time the RIT imagery was collected, the yaw information was not recorded. To compensate for this, a yaw angle was selected such that final aircraft orientation projected into the (x, y) plane was along the aircraft flight direction (see Appendix A.2). However, this approximation is rarely valid, as the aircraft must usually *crab* (face away from the direction of flight) somewhat to counter the effects of crosswinds. These angular errors ultimately translate to large positional errors on the ground, and in the case of the RIT imagery, precluded the use of this hyperspectral data as a co-registered image for the RIT scene.

Once it was determined that the MISI data could not be easily registered to the other images, we had two primary options regarding how to proceed. First, we could manually associate

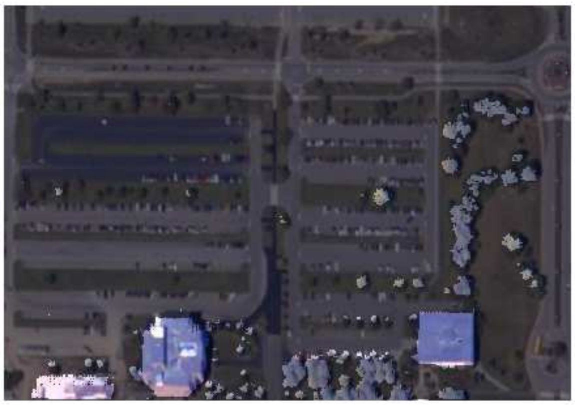


Figure 5.6: Illustration of lidar/WASP registration.

selected pixels from the MISI imagery to the other data sources, and proceed with the spectral library selection process. However, given the limited number of materials in the scene, we chose to pursue a second, quicker option. We decided to simply use the WASP color imagery to create a material map for the scene, then manually assign material types to each material type present in the material map. Building materials were attributed manually as well, assigning a single material (dark-grey roof material) to the roof facets, and a separate material (red brick) to the wall facets. The building material were attributed using Bulldozer, while the ground-based material map assignments were specified directly in the DIRSIG configuration file.

In order to achieve this, the WASP color image had to first be coarsely registered to the lidar data. For the RIT scene, this was accomplished using a simple point matching approach where homologous features were identified manually through an interactive GUI. Once the matching points were defined, we solved for the projective transform that best aligned the points. This transform was then applied to the entire WASP image. Figure 5.6 depicts the quality of the registration where the passive imagery has been converted to HSV space, with the lidar height values scaling the value channel. This registration was accurate to within 4 pixels (2 meters) throughout most of the scene.

The original WASP image was then processed to obtain texture and material maps. The tex-

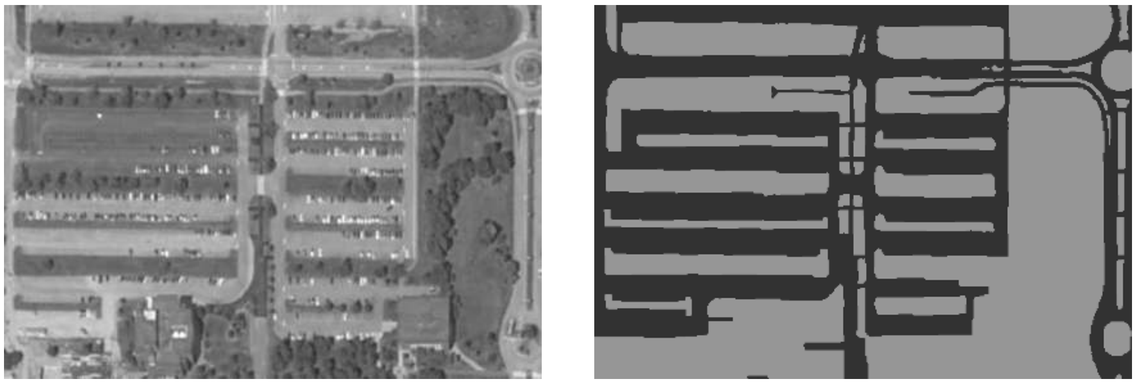


Figure 5.7: RIT Scene: Texture and material maps.

ture map was quickly obtained by averaging the red, green and blue pixel values at each location, thereby producing a grayscale image. This image was then registered to the scene using the transform parameters derived above. For the material map, we opted to use a minimum-distance supervised classification to divide the scene into two classes, vegetation (grass) and asphalt. This classification was performed using the ENVI image analysis software. After the initial classification was performed, isolated pixels that were misclassified were adjusted by median-filtering the image with a 5×5 kernel filter. Additionally, the regions previously identified as containing either buildings or trees were re-classified as grass. This map was also registered to the scene data using the previously derived transform. Both the texture and material maps used for the RIT scene are shown in Figure 5.7

Once all of the basic geometry and material files needed to specify the scene had been created, the geometry was consolidated by creating a DIRSIG '.odb' file. This process, while requiring some manual input, requires little time. Since insertion points for each object were determined from the geometry, the actual text blocks were autonomously produced during the extraction process. These text blocks are therefore simply copied to a common file to produce the .odb.

It should be noted that the entire scene construction process, which proceeds from the consolidation of image data to creation of the .odb file, took just over 45 minutes. Approximately

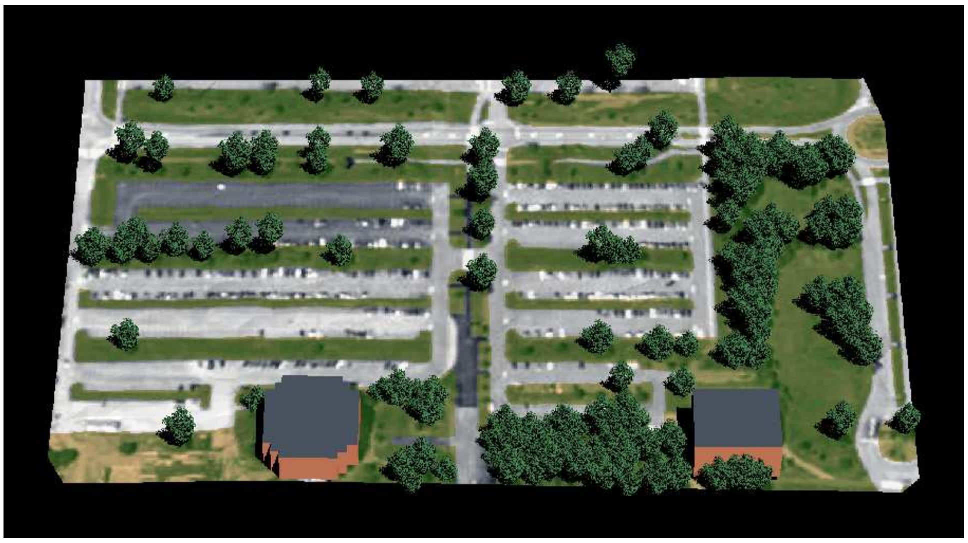


Figure 5.8: RIT Scene: Simulated image of the scene model.

two-thirds of this time was spent on manual processes such as transferring data from one software tool to another, and verifying the object geometries in Rhinoceros and Bulldozer. This represents a significant time savings over current DIRSIG scene modeling techniques, which would typically require several man-days to complete a scene of this size. However, it should be noted that a significant time savings and quality improvement was made by already having a library of tree models and reflectance spectra in hand before beginning the process. Had these not been available it would have taken an additional 4 hours (or more) to generate a tree model using the TreeProfessional software, and the manually assigned reflectance spectra obtained from the MISI imagery would have taken approximately one hour to process, it would have been significantly noisier, and its range of spectral coverage would have been significantly reduced.

In order to visualize the scene model, slightly more effort is required. The location (filepath) of all pertinent scene files must be specified manually, and a simulated sensor used to produce the simulated image must be configured. Material reflectance files must also be associated with each material ID, and parameters related to the mapping images must also be specified. These are all performed in the DIRSIG configuration file. Once this file has been created, DIRSIG is executed, and a simulated image of the model scene is produced. An example simulated color image of the RIT scene model is given in Figure 5.8.

In this image, we see that the scene model appears as expected, and that all of the geometries are proportioned correctly and appear in their proper location. Regarding the spectral characteristics, the grass and asphalt ground regions have appropriate texture, but no texture is apparent in the building structures. This is due to the fact the grass and asphalt material types contain 300 spectral reflectance curves each (that is, their associated emissivity files contain 300 emissivity curves), while the roof and brick material types contain a single reflectance spectra. Therefore, the texture image is able to drive selection of different spectra at different locations for the grass and asphalt regions, but the same spectra must be used across all building roof facets. Additional spectral curves may be simulated even if only a single instance of a reflectance spectra is available if we can define statistical properties (specifically mean and covariance) of the class. However, this was not done for the spectra used in this scene. It should be noted when viewing the simulated image that while trees and buildings have been re-created in true 3D, smaller objects such as vehicles have not. The reason that some of these smaller objects seem to appear in the DIRSIG image is that they were present in the texture map used for spectral assignment, and the spectral variations give the appearance of additional objects.

5.2 RIT Scene: Analyzing the Geometric Quality of the Process

5.2.1 DTM Analysis

Although the above discussion yields credence that the proposed approach may actually produce DIRSIG scenes with little manual intervention, there has not yet been a discussion of how accurate the process is compared to current techniques. To fill this gap, the following sections aim to analyze the quality and speed of the semi-automated results, as compared to fully-manual approaches using the software tools currently available for DIRSIG users. The case will be made that the new process produces results that are at least as accurate (and in many cases even more so) than the current methods, and that the new approach is typically an order of magnitude or more faster than the traditional techniques.

To start this analysis, we consider the approach used to derive the DTM. With the lidar-derived DTM shown to produce visually-pleasing results in the above example, we next sought to compare this DTM to one obtained from the USGS database since most DIRSIG scenes have been built using their 30 meter or 10 meter rasterized DTM products. To this end, we downloaded the most recent 10 meter DTM product for the greater Rochester region from their website [Survey 2008] as a GeoTIFF elevation image. We then used the ENVI image processing software to convert this GeoTIFF to UTM coordinates and extract the local region of interest. While doing this, we discovered an approximately 185 meter bias toward the south in the USGS product, relative to the lidar data. Coordinates for features found in Microsoft's Google Earth imagery [Microsoft 2008] coincided with those for the lidar data, so the USGS latitude values were increased by 185 meters so that the two DTMs were registered. The native pixel spacing in the USGS product was 7.54 meters in the x (eastern) direction and 10.29 meters in the y (northern) direction.

In order to compare the USGS product with the lidar-derived DTM, we downsampled the lidar-derived DTM to the grid locations of the USGS product using a bilinear interpolation. This permitted us to compare elevation values between the two models at points in which the USGS product was unaffected by an interpolation. Statistics related to this comparison would represent those points actually specified by the USGS DTM, and would not be as influenced by small features missed due to the USGS DTM's relative low sampling density. We also sought to compare the lidar-derived DTM with a version of the USGS product upsampled to 0.5 meter resolution. This comparison would not only highlight errors where the USGS DEM was defined, but would also conceivably show the effect of missing smaller spatial features due to undersampling. The original lidar-derived DTM, re-colored to highlight elevation changes with no buildings present, is given in Figure 5.10 (a). The upsampled USGS product, the downsampled lidar-derived DTM

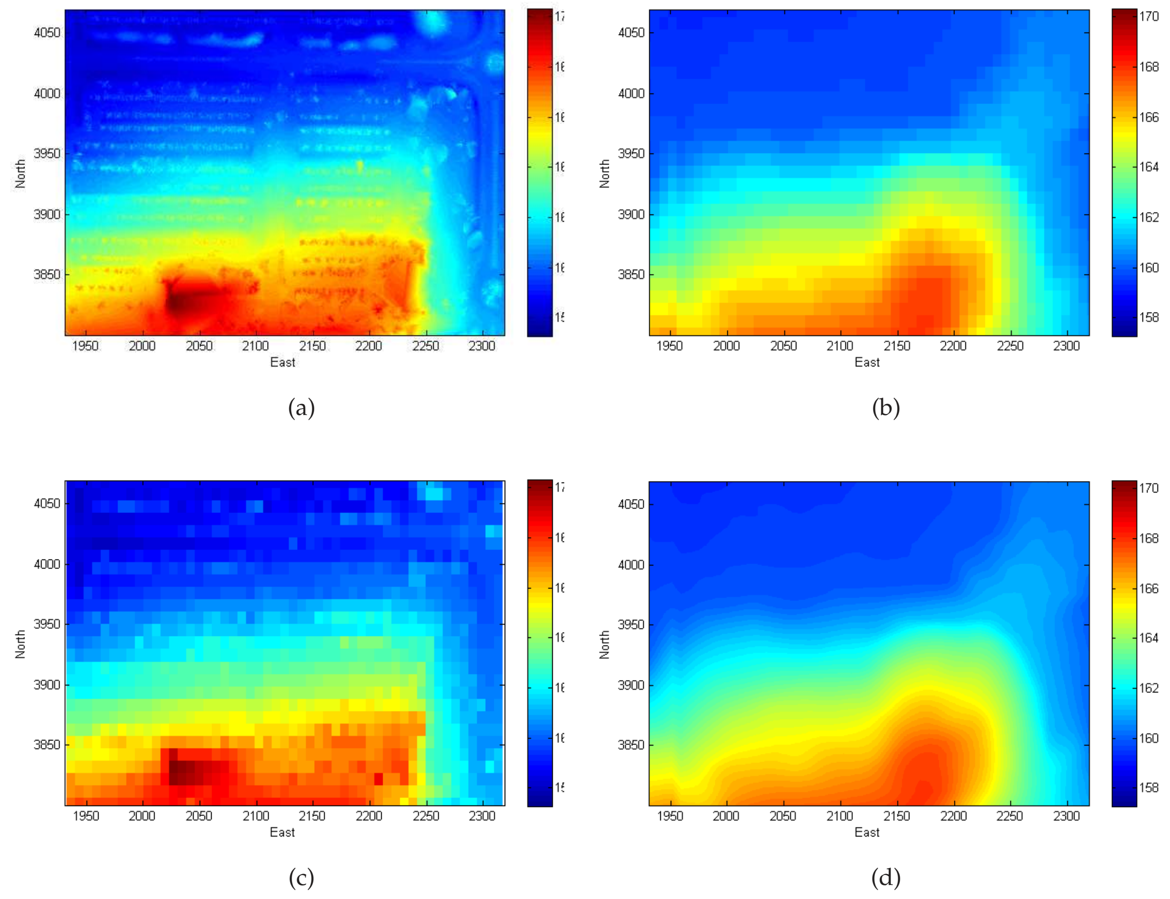
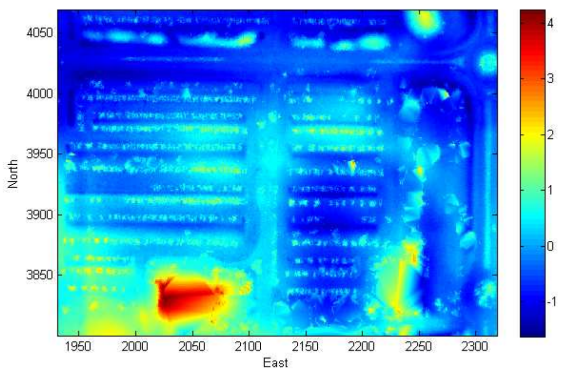


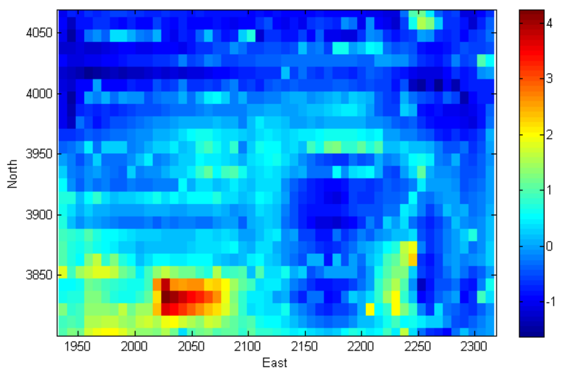
Figure 5.9: Comparison of DTMs: (a) DTM extracted from 0.5m raster range image with updated color scaling, (b) USGS 10m DTM upsampled to 0.5m pixel spacings (c) lidar-derived DTM downsampled to USGS 10m DTM spacing, (d) USGS 10m DTM at native resolution.

and the original USGS DTM are given in (b), (c), and (d) of this same figure.

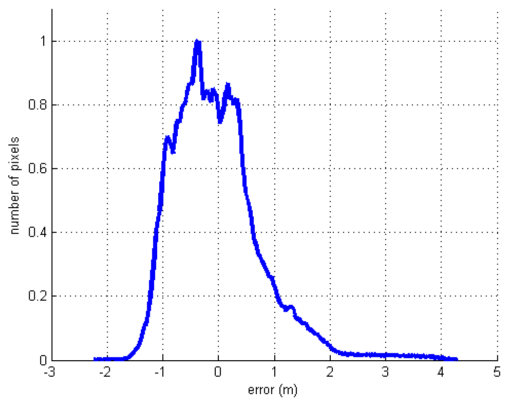
Subtracting the USGS product from the lidar-derived DTM gives the error images shown in Figure 5.10. Image (a) shows the 0.5 meter (lidar-baseline) case, and (b) shows the comparison at the native USGS resolution. A qualitative comparison of these images shows that largest height discrepancy is at the location of the Carlson building. The positive value in the error images at this location indicates that the lidar-derived DTM has a higher value at this position than does the USGS DTM. It was initially thought that this may be due to the lidar-processing not omitting



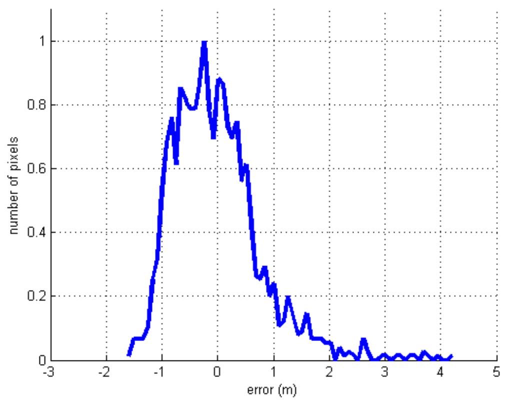
(a)



(b)



(c)



(d)

Figure 5.10: Comparison of lidar-derived DTM with USGS product: (a) Error image using baseline lidar DTM and upsampled USGS DTM (b) Error image using downsampled lidar DTM and baseline USGS DTM, (c) histogram of *(a)*, (d) histogram of *(b)*.

all building points when the DTM was produced, but further analysis revealed that this is not likely the case. What we believe is occurring is that in the lidar-derived DTM, the points used in interpolating across the building gap are immediately adjacent to the building, which is at the highest ground location in the scene. In the USGS product, the building was likely removed and points some distance away (at lower elevations) were used in the interpolation. This would translate into lower USGS values across the region being interpolated.

In the higher resolution difference image, we also see that many of the small, rapid terrain

Table 5.1: Statistics on the difference images: lidar-derived DTM minus USGS 10m DTM (Error values are in meters)

Metric	USGS DTM upsampled	lidar DTM downsampled
Mean	-0.0243	-0.0270
Variance	0.6398	0.6275
Mean squared error	0.6404	0.6278
Root mean squared error	0.8002	0.7923
Mean absolute error	0.6098	0.5992

changes are missed in the USGS DTM, a fact that is also clearly visible from the smooth nature of the USGS DTM itself. This effect is not present to the same extent in the lower-sampled difference image. For a more qualitative comparison, histograms of the difference images were also created, and are included in Figure 5.10 (c) and (d). The long histogram tail extending beyond 3m represents most of the error around the Carlson building interpolation noted above. Error in the 2m range is seen in on a hill in the northeast, as well as a steep slope to the East of the location of the Bausch and Lomb building. Small hill features that are captured by the lidar-DTM and are missed by the USGS product are captured as yellowish horizontal lines in the error images, and the lower parking lot locations are a dark shade of blue. Although it is not surprising to see these features in the high resolution error image, it is interesting to note their statistically-similar presence in the lower resolution error image as well. That is, the downsampling of the terrain did not filter out all of the rapid terrain variations. Statistical metrics related to these difference images are also given in Table 5.3.

The approximately -2.5 cm mean absolute error (for both pixel sizes) indicates that there is very little bias present between the DTM products, and that the north-south registration of the models (with the 185 meter bias) is likely adequate. Also, despite the obvious differences in the terrain models, the two DTM products are much more similar than they are different. Most pixel locations are within approximately 60cm from model to model, and the maximum deviation in regions that will not be masked by building points are around 2m.

Specifying which product is better for use as a terrain model when building a DIRSIG scene is a bit more difficult to answer. In some cases, Bulldozer has had difficulty assigning a material index to a 0.5m resolution terrain model of this size due to the large number of facets, so the lower density products may be beneficial from than standpoint. Additionally, in many cases, the scene designer is not interested in capturing every terrain feature, and instead prefers a smoothly varying terrain to avoid the appearance of facetization in the resultant simulated scene images. In these cases, the lidar-derived DTM may be smoothed with a low-pass filter, or the USGS terrain product may be substituted with little hesitation. However, in cases where precise specification of the terrain variations is important, such as trafficability studies and local floodplane analyses, the resolution of the lidar-derived DTM becomes significant. The ability to create high-quality terrain models from remotely-sensed airborne data is also critical when accurate DTM products are unavailable and ground surveys would be difficult.

Next, we decided to investigate how the rasterized implementation of the slope-based DTM extraction compares with a similar filter operating directly on the lidar point cloud. Although working with the raw points means that we no longer may use the highly efficient image erosion routine for determining non-ground points, we still investigate the point-to-point slopes could by effectively placing a conic function at each point and determining if any other points fall under this cone. Figure 5.11 shows the result of this process. Diagram (a) shows an oblique view of the 0.5 meter R_A raster image for the square building in the south-east corner of the scene, and (b) shows the DTM extracted using the rasterized process discussed above. Diagram (c) shows the (non-interpolated) point data for this same building, connected via a Delaunay triangulation, and (d) shows the result of DTM extraction without interpolation. Of note is the obvious gap present at the object locations in (d). While the raster implementation requires interpolated values across the location of the non-ground points in order to maintain its grid-based (matrix) form, the point-based approach is able to remove non-ground points from the list without inserting additional values. It is also able to eliminate any potential errors caused by the original interpolation of the

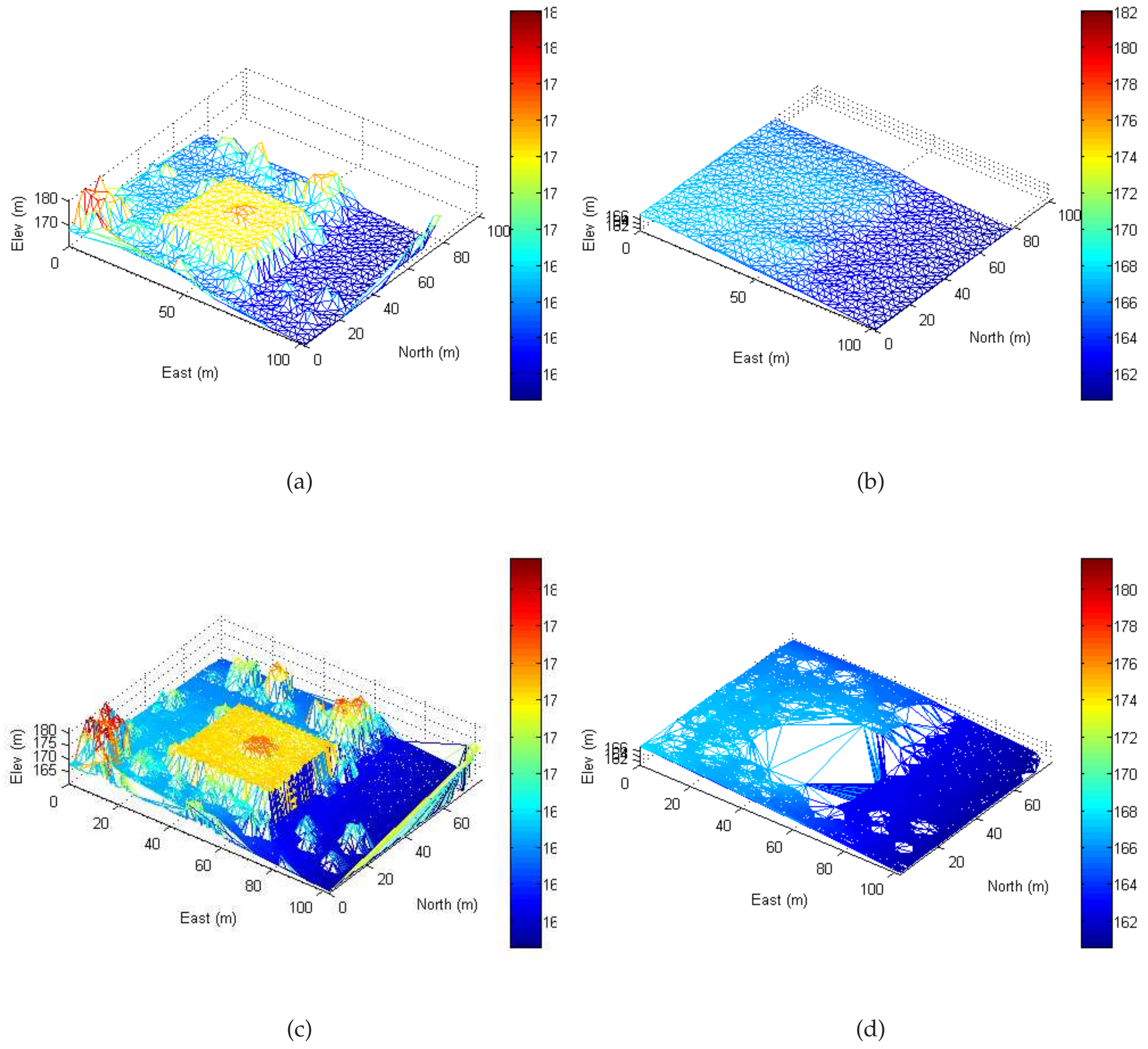


Figure 5.11: Raster versus point-based terrain extraction: (a) original raster range data (b) DTM obtained via slope approach applied to raster image and interpolating values to the non-ground pixels, (c) original point data, and (d) DTM obtained via slope approach applied to point data.

data. However, it remains to be seen how these potential errors affect the final terrain model.

In order to answer this question, the raster DTM elevation values were interpolated to the locations of the raw terrain points. The point elevation values were then subtracted from the re-interpolated raster values in order to produce difference statistics. A histogram of these difference

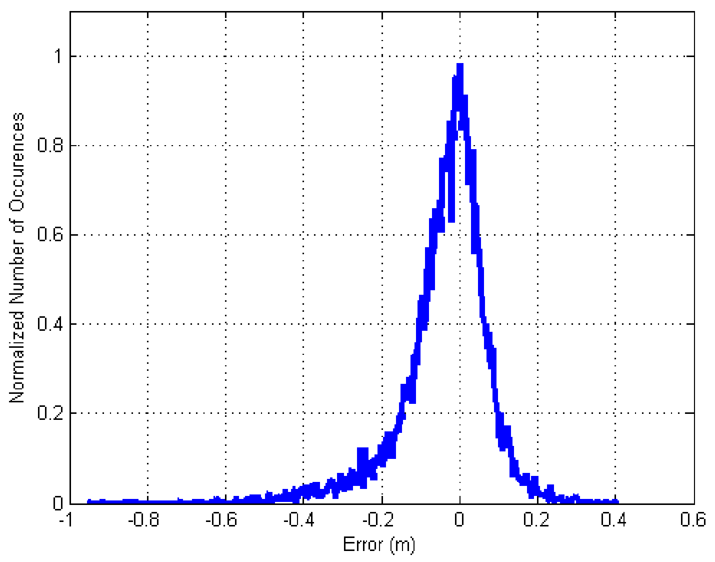


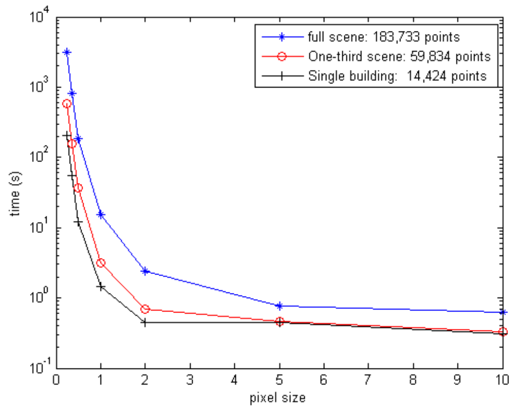
Figure 5.12: Comparison of raster and point-based DTMs: Histogram of difference between raster DTM and ground points at ground point locations.

values is included in Figure 5.12 (a). As can be seen in this figure, the vast majority of raster DTM points fall within 0.5 meters of the raw point values, but there are a small number of outliers in which the difference between the terrain models approaches one meter. Differences between the images arise for numerous reasons. Interpolation may cause rapid height variations in the raster image, such that points may be removed from this image but not in the corresponding locations of the point set. The undersampling in these regions may then lead to offsets. On the other hand, the interpolation may also remove some of the point-to-point transitions that are significant enough to identify object points. In these cases, the raster version would not flag an object as being present at that location, but the raw points would. In either case, baseline truth should be taken as the raw point values. However, this increased accuracy comes at a price. When using a 35 meter radius conical structuring element on this small (14,424 point) data set, the 0.5 meter pixel raster-based approach takes 5 seconds to perform the interpolation and 10 seconds to perform the filtering, while the point based approach takes over 665 seconds.

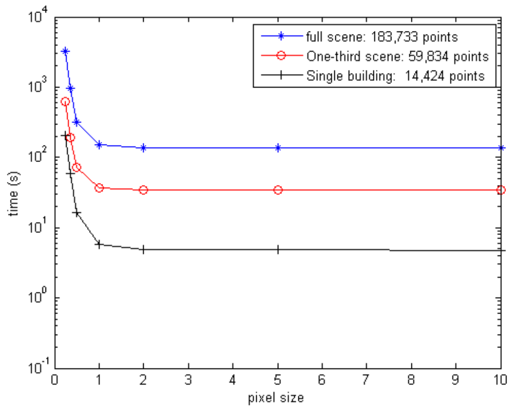
As was shown above, even with small scenes, the process of extracting the DTM can take a significant amount of time. For larger scale scene geometries, such a process could potentially dominate the computational requirements. To this end, we decided to investigate the time required to extract terrain models given multiple scene sizes and multiple structuring element sizes. The baseline would be based on the raster approach, but results would also be run for the point-based filtering.

To determine the computational requirements, three subsets of the lower point density RIT lidar dataset were processed. The first set includes the entire RIT region of interest that we have used this far in our discussion, which is roughly 390 meters east-to-west and 270 meters north-to-south, and contains 183,733 raw data points. The second set includes a strip spanning the full width of the scene, but only spanning approximately 85 meters in the north-south direction. This set includes both buildings, and has 59,834 points. The final set that was used is the small example scene containing the 14,424 points shown in Figure 5.11 (c). Results for the raster DTM extractions are given in Figure 5.13. Part (a) gives the processing time as a function of pixel size (sampling density for the interpolation) for the operations that are not dominated by the pixel size. These include building the conical structuring element, performing the erosion of the range image, and interpolating across the removed pixels. A 35 meter structuring element is assumed. Part (b) yield similar information, although this plot includes the total processing time for all operations, including reading in the data and removing duplicate (x, y) points. As can be seen, for all three scene sizes, the processing starts to become significant for pixel sizes below 1 meter, and below 0.5 meters, they become increasingly prohibitive. Not only are the processing times significantly longer with increasing pixel density, but the memory requirements are increased as well. For the larger scene case, with a structuring element radius of 35 meters, the final interpolation failed due to memory issues in the 0.25 meter pixel case (3GB RAM using Matlab).

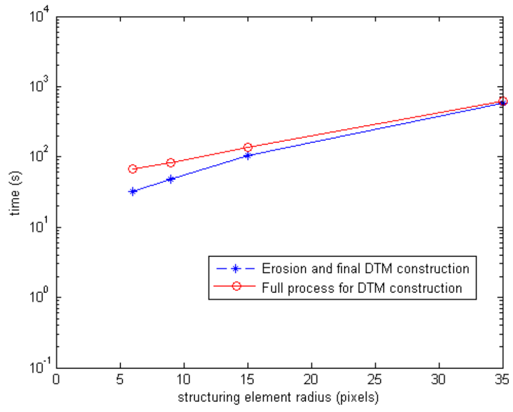
Figure 5.13 (c) shows the increase in computation time as a result of varying the structuring element radius for the mid-sized dataset with a pixel size of 0.25 meters. We see from this plot



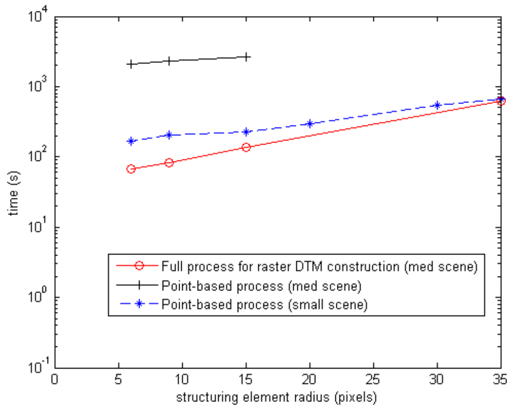
(a)



(b)



(c)



(d)

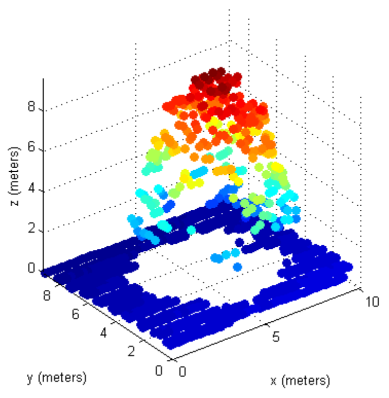
Figure 5.13: DTM extraction processing times: (a) Raster-based as a function of pixel size, only includes times for generating the S.E., eroding the image, and interpolation (b)Raster-based as a function of pixel size, total time, (c) Raster-based as a function of structuring element radius, (d) Raster and point-based as a function of structuring element radius (two scene sizes used).

that as the structuring element radius is reduced, significant time savings may be realized. This comes at a small price, however. Without additional processing, the structuring element radius must be at least half the size of the largest building, or else points towards the center of the building may not be flagged as being non-ground. However, in many cases, these internal points may be removed by applying additional heuristics that compare the height of 'ground' points that are completely surrounded by non-ground points with the height of the non-ground points themselves. If they are of comparable height, the internal points are reclassified as object points and are removed from the terrain model. An alternative is to permit the user to interactively select any object regions that may have been missed by the initial slope-based filtering. We have implemented both of these approaches in our work, and with them, we are able to obtain excellent results with structuring element radii down to approximately 10 meters, even with much larger buildings in the scene. In most cases with grid sizes below 0.3m, the extra work required by the manual intervention or implementation of heuristics is a small price to pay for the significant savings in computation time. However, for lower resolution range images, or in cases where the process must run with full automation, using a larger structuring element is still warranted.

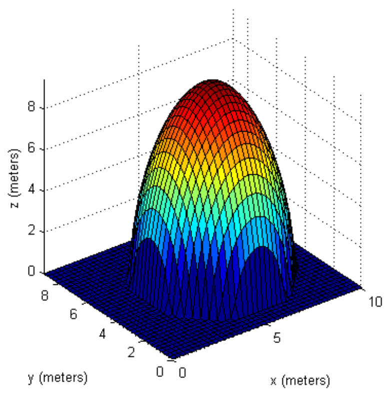
Two final time comparisons should still be made, and these are for the cases where the raw points are processed directly. Figure 5.13 (d) contains a plot of execution time as a function of structuring element radius for the raster case (medium scene) and for the point-based case (small and medium scenes). We see here that the point-based approach operating on the small dataset requires roughly the same time to execute as the rasterized approach operating on the mid-sized dataset. The point-based approach operating on the mid-sized set takes an order of magnitude longer for the 15 m radius case, and this time increases rapidly as the structuring element size increases. With the small relative errors between the point-based and raster-based implementations combined with the significant execution time demanded by the point-based approach, the raster-based approach is usually preferable.

5.2.2 Tree Parameter Estimation Using the Generalized Ellipsoidal Model

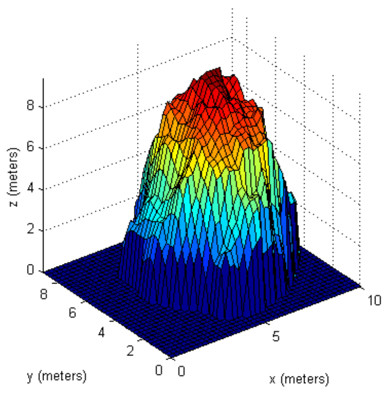
In the previous section, one of the things we considered was whether the terrain extraction should be performed on a raster image or on the raw point cloud. A similar question arises for the tree reconstruction uusing the generalized ellipsoid model of Equation 4.18. This model is used to help define height, width, shape and location parameters, and if needed, these may also be used



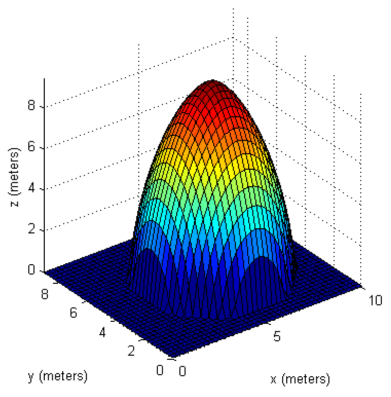
(a)



(b)



(c)

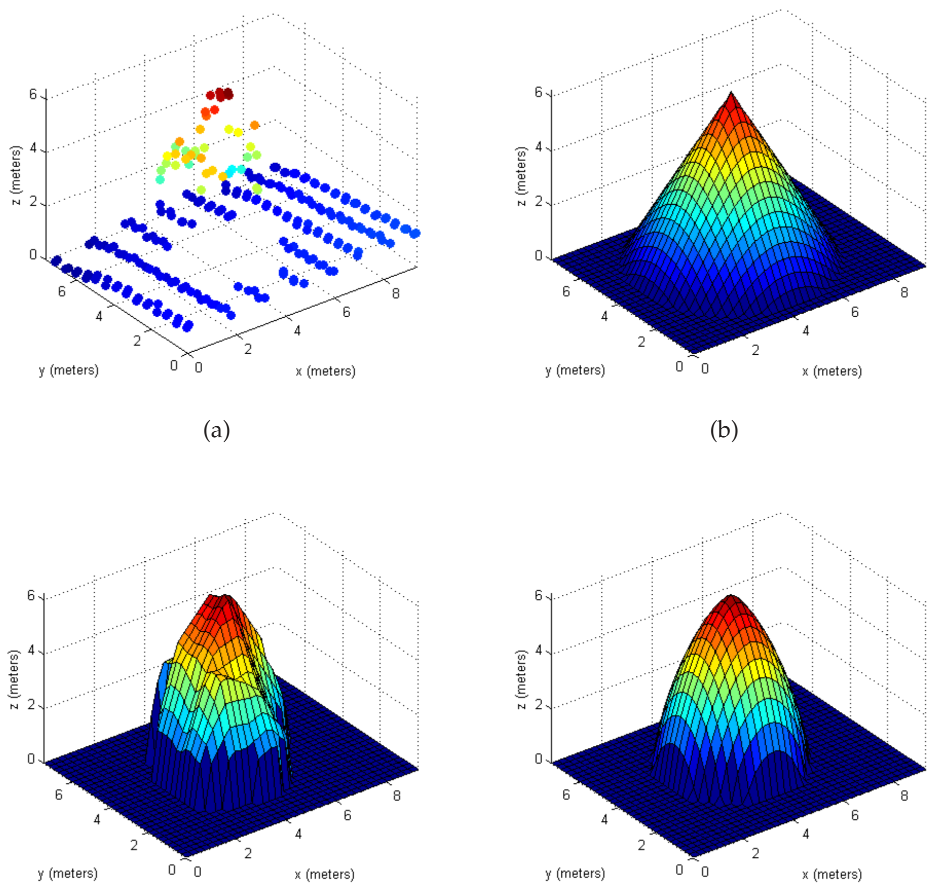


(d)

Figure 5.14: Tree model fitting: high data density (a) Original point cloud (b) model fit to point cloud (c) Raster image (d) model fit to raster image.

to help classify the tree as deciduous or evergreen. Even though the terrain is usually processed in raster form, after identifying tree regions in the NDEM, we may choose to either process these tree regions using the pixel values or by processing the raw points contained in this region.

Figure 5.14 illustrates both cases for data from the high-density point cloud. Image (a) shows the point data, consisting of approximately 270 tree points. Image (b) shows the best-fit model when the point data is used in the fitting process. Image (c) shows the tree data interpolated to a 0.25 m range image, and (d) illustrates the best-fit model for the rasterized data. It is clear that the



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

Figure 5.15: Tree model fitting: low data density (a) Original point cloud (b) Model fit to point cloud (c) Raster image (d) Model fit to raster image.

two models are basically the same, and aside from a slight deviation in the shape parameter (in (b) it is 1.55 while in (c) it is 1.45), the models are identical. The small difference in shape is due to interpolation artifacts inherent in generating the range image. As such, the point-based model is slightly more accurate in the high-density data case.

This contrasts with Figure 5.15, which shows the same illustrations for a similarly shaped tree, but this time using the lower density point data. In this case, only 45 tree points are available, and they are arranged in such a way that the best-fitting model actually has a significantly larger spread than the data would indicate. However, by re-sampling the data to a high-resolution grid before fitting the tree model, a more accurate result is obtained. In this case, we achieve a result almost as good as for the higher density data case.

As a result, we have found that in most cases, as long as the pixel spacing is sufficiently small, the parameter fitting may be done in raster space with little penalty. However, as the pixel sizes grow to be above approximately 1 meter, high density data sets should switch to using the points directly. For low-resolution raster images where the underlying data is also low density, inaccurate tree models will frequently arise. Although not ideal by any means, in certain low-density cases the best performance is obtained by resampling the point data in the vicinity of the tree to a higher-resolution (approximately 0.25m) grid before processing.

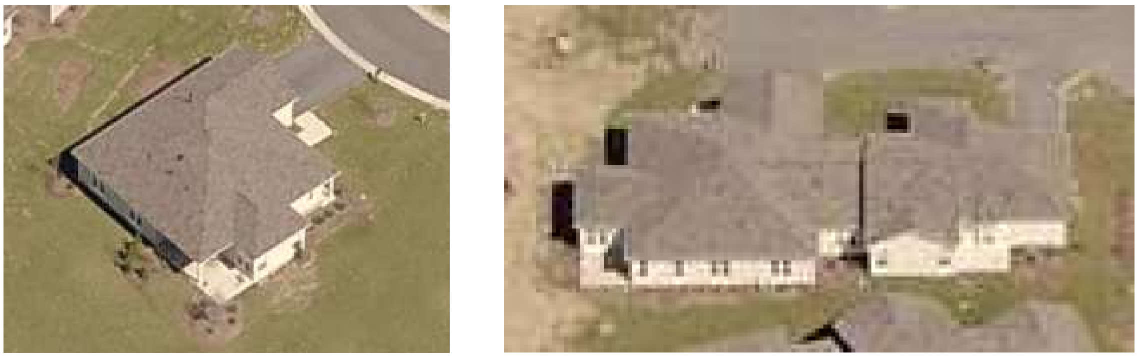

(a) (b)

Figure 5.16: Residential building reconstruction: (a) House A (no vertical transitions) and (b) House B (multiple vertical transitions and several small features).

5.3 Residential Scene: Building Reconstruction

The second location we will focus on is from a residential neighborhood in a town just east of RIT's campus. This area contains multiple buildings with complicated roof structures, two of which we will use to illustrate some of the benefits of the automated building reconstruction process presented earlier. The selected buildings are illustrated in Figure 5.16. These structures may seem somewhat familiar; the first (Building A) was used to illustrate the boundary extraction and facet reconstruction techniques in the previous chapter, and the second (Building B) was used to illustrate the processing of internal vertical transitions.

While a description of the extracted building models may prove interesting in their own right, of greater importance is how these derived models compare to those created by hand, both in terms of geometric accuracy and the time required to produce the model. In order to perform this comparison, two users experienced in creating CAD models of buildings were recruited to create their own CAD models of Building A, and one of them also created a model of Building B. They were instructed to use whatever tools they would if they needed to produce such a structure for their own research. The only constraints were that the final models were to be specified in a DIRSIG-compatible format, and that CAD drafters log the time it took to perform each task.

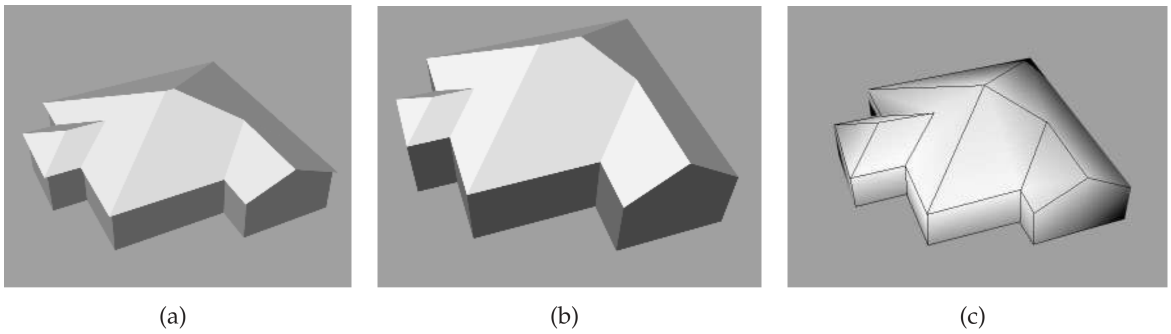
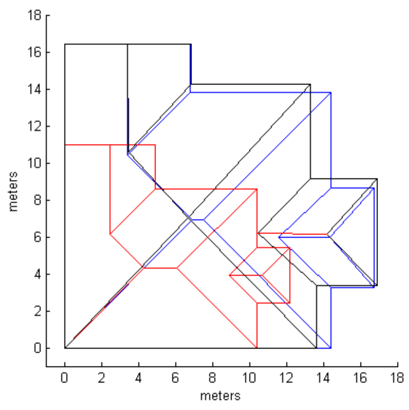


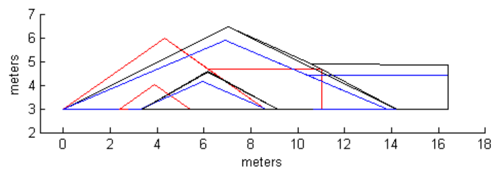
Figure 5.17: Building A: (a) A_1 manually-produced model, (b) A_2 manually-produced model, and A_L semi-autonomous lidar-derived model.

The resultant building models for Building A are presented in Figure 5.17. As can be seen in this figure, each model has the correct approximate shape, and no facets were missed in any of the models. The A_1 model (created by the first CAD drafter) is seen to have a somewhat shallow sloping roof structure, and the ridge line at the top of the roof is fairly small. This contrasts with the A_2 model, where the roof structure shows significantly higher slopes, and the ridge line is comparatively large. In the semi-autonomously lidar-derived structure A_L, we see a model very similar to the A_1 model. The roof slopes appear to be about the same, and the overall ratio of dimensions appear to be similar as well. One item of note is that in the building refinement stage of creating the A_L model, the two vertices representing the ridge line were merged into a single vertex, as the reconstruction algorithm incorrectly assumed that two distinct roof vertices would lie at a distance of more than 1.5 m apart. A better comparison of these structures may be made by rotating and translating the models so that they share a common origin and primary orientation. This was done in Figure 5.18. It is clear from Image (a) that the A_2 model is of a significantly different size than the other two. The drafter of this model made the initial assumption that the garage was 16 feet wide, and scaled the rest of the house accordingly. The first drafter took a more accurate (and more time consuming) approach, and 'measured' the building sides using coordinate values obtained from Google Earth [Microsoft 2008].

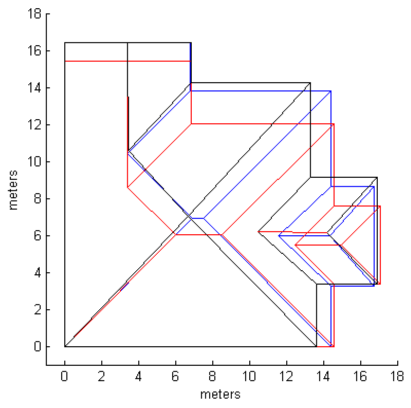
A summary of the vertex offset relative to the A_L model is provided in Tables 5.2 through



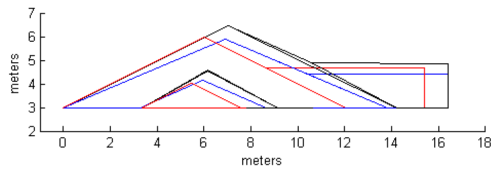
(a)



(b)



(c)



(d)

Figure 5.18: CAD models for Building A: Nadir and front views: (a) A_2 model not scaled, (b) A_2 model scaled in the (x, y) dimension.

5.4. Both the A_1 and A_2 models are considered, as well as a scaled version of the A_2 such that the (x, y) values were stretched by a factor of 1.4, while the height was kept unchanged. This rescaling brought the A_2 model into significantly better alignment with the other two models. The A and B columns in these tables represent 2D (in the (x, y) plane) and 3D distance measures in the rows where such a distinction is appropriate. Other than the initial offset values, all parameters were adjusted so that the mean offset value was forced to zero.

There are several observations that may be made from a comparison of these building models. Among the most prominent is the large size discrepancy between the two manually-derived representations. In this example, two experienced CAD users arrived at two very different geometries (in an absolute sense), based on different assumptions on how best to scale the model. This results in average vertex errors that vary by more than an order of magnitude from model to model. Although potential gains in accuracy and speed were expected, we had not fully considered the aspect of model consistency. By using a pre-defined, semi-automated approach, there are significantly fewer subjective decisions left to the scene creator.

Also included in these tables are the times taken to produce the DIRSIG compatible object file. While each manually-produced model took approximately 40 minutes to complete, the lidar-extracted model took less than 6 minutes to create. While accuracy and consistency of results are certainly the primary drivers for most applications, an 85% reduction in the time needed to produce each building model has potentially huge manpower impacts when considering urban scenes on the order of several square miles. While the scene designer would likely need to review each model even when using an automated approach, the time savings in not having to measure and create each facet by hand is certainly quite significant. Although in general some manual intervention may be required to ensure proper building segmentation and the correct handling of vertical transition structures, Building A was extracted without any manual intervention.

Although the time savings was apparent in the previous example, we thought it would be beneficial to show how the semi-automated approach compares to manual-derived models for a significantly more complex building structure. To this end, we selected Building B as a second residential structure to investigate. This building contains significantly more facets, multiple vertical transitions, and several small features in the roof structure. Some of these features, such as a small fan-shaped roof overhang over a bay window will not cause inconsistencies if they are missed by the semi-automated algorithm. However, other features, such as split segments that must be divided into multiple DLS's will cause major artifacts in the resultant model if they are not identified.

Table 5.2: Statistics on building point differences: lidar-derived model minus A_1 model (Errors are in meters)

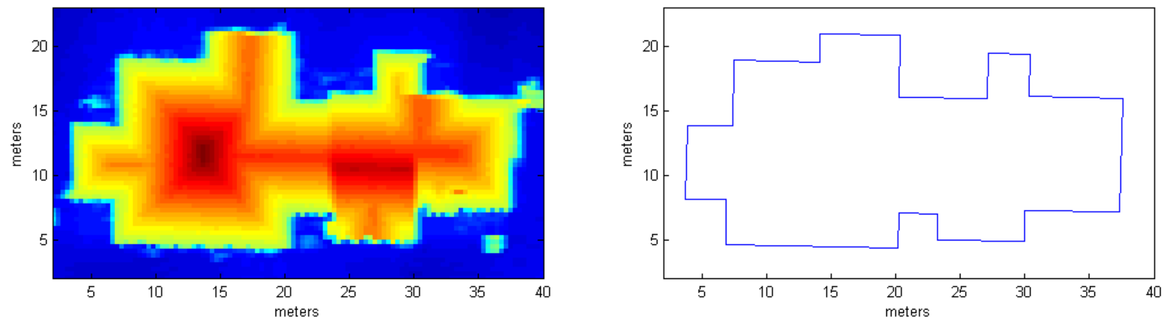
Metric	A	B
Mean x offset	-0.3434	
Mean y offset	0.1803	
Mean z offset	0.1807	
Mean distance (mean offset removed)	0.2632	0.3201
Variance (mean offset removed)	0.0435	0.0409
Mean squared error (mean offset removed)	0.1101	0.1408
Root mean squared error (mean offset removed)	0.3318	0.3753
Time to estimate vertex locations	25 mins	
Time to create .obj file	30 mins	
Total Time	**55 mins**	

Table 5.3: Statistics on building point differences: lidar-derived model minus A_2 model (Errors are in meters)

Metric	A	B
Mean x offset	2.2331	
Mean y offset	3.2021	
Mean z offset	0.1535	
Mean distance (mean offset removed)	5.9214	5.9703
Variance (mean offset removed)	18.2629	18.0179
Mean squared error (mean offset removed)	52.1843	52.5360
Root mean squared error (mean offset removed)	7.2239	7.2482
Time to estimate vertex locations	20 mins	
Time to create .obj file	25 mins	
Total Time	**45 mins**	

Table 5.4: Statistics on building point differences: lidar-derived model minus scaled A_2 model (Errors are in meters)

Metric	A	B
Mean x offset	-0.5344	
Mean y offset	0.9933	
Mean z offset	0.1535	
Mean distance (mean offset removed)	1.0916	1.1405
Variance (mean offset removed)	0.4180	0.4477
Mean squared error (mean offset removed)	1.5834	1.7204
Root mean squared error (mean offset removed)	1.2583	1.3116



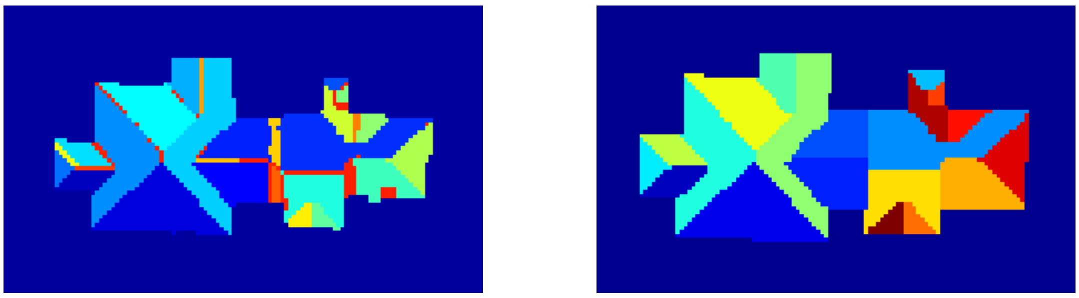
(a) (b)

Figure 5.19: Building B: (a) range image and (b) boundary extracted from point data.

Figure 5.19 shows the range image and the boundary extracted from the raw point data. It should be noted that the tree present in the upper-right corner of the range image was not removed by the texture-based tree analysis and had to be removed via the included ROI tree removal tool built into the code used to implement the semi-automated building reconstruction process. This tool served to removed the tree data in both the raster image and raw point cloud data.

After local normal vectors were calculated at each pixel location on the roof structure, the mean-shift algorithm was used to achieve the initial facet segmentation shown in Figure 5.20 (a). As expected, this segmented image contains many additional (unwanted) classes, and the outer boundary consists of somewhat jagged edges. By autonomously reclassifying those segments that fall where at large values in the E_2 image, all but two of these segments were removed. By clicking on these remaining segments, they were also merged with the adjacent regions to produce the segmented image of Figure 5.20 (b).

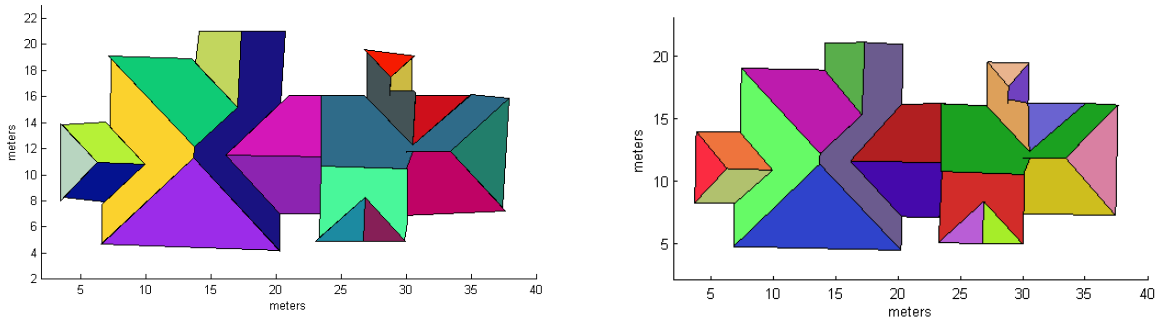
Each facet was then reconstructed in turn, and facets with potential vertical transitions or small edge segments with high curvature were flagged to the user for additional information. By clicking on vertical transition DLSs and edge segments that need splitting, the reconstructed model



(a) (b)

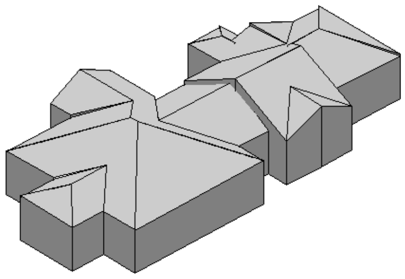
Figure 5.20: Building B segmentations: (a) initial facet segmentation obtained via applying mean-shift to the N_3 image, and (b) refined segmentation.

shown in Figure 5.21 (a) was produced. Although some manual intervention was required at this step as well, the total time in proceeding from the segmented image to this initial building model was under 3 minutes. In this nadir view of the resultant building model, it is apparent that several of the exterior corners are not properly defined, and that the outer boundary shape (as derived from the pixel analysis) is somewhat irregular. By forcing the boundary to fit the one described by Figure 5.19 (b), the model is refined, and the B_L geometry shown in Figure 5.21 (b) is achieved.



(a) (b)

Figure 5.21: Building B CAD models: (a) Initial CAD model produced by processing each facet independently, and (b) Refined model using previously-derived outer edge information.


(a)

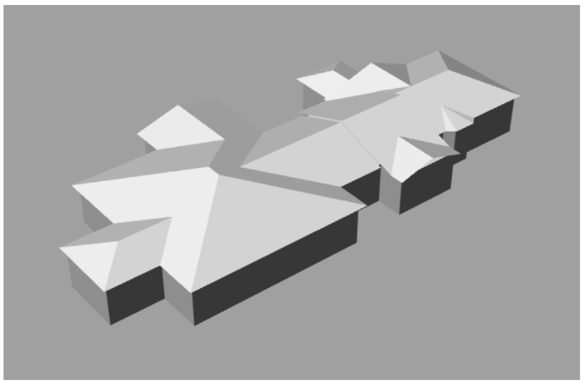
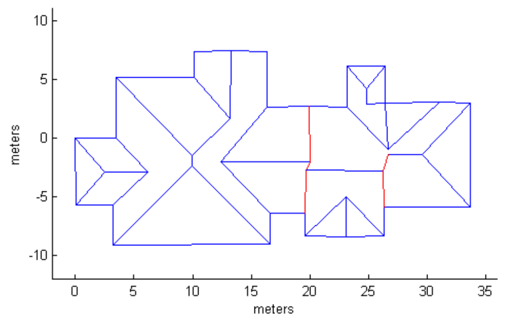

(b)

Figure 5.22: Building B CAD models: (a) semi-autonomously created and (b) manually-produced.

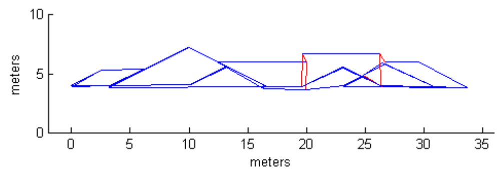
Figure 5.22 illustrates a perspective view of this lidar-derived CAD model in (a) and also shows a manually produced CAD model (B_1) in (b). As was the case with Building A, the two models appear very similar. However, a few minor differences in the models reveal themselves when they are inspected closer. First, the lidar-derived model is completely missing the fan-shaped feature over the bay window (shown in black), and the facets forming one of the dormer structures do not align perfectly. The fan structure was missed in the initial segmentation, and the dormer junction was erroneously classified as a vertical transition.

However, this model did correctly handle all dormer structures aside from this vertical transition issue, while the manually-produced model has two dormer structures with slightly erroneous shapes. The dormer by the fan structure is extended from the main roof structure in B_1, and the dormer opposite this one does not contain a proper vertical face.

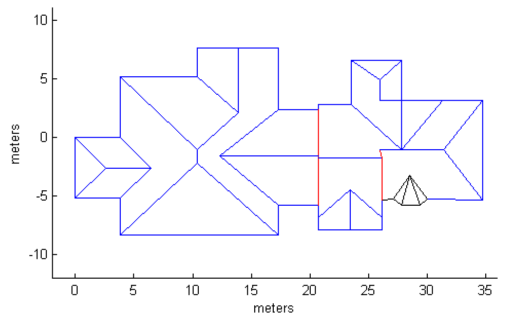
Figure 5.23 shows nadir and front views of the roof structures of these building models. Images (a) and (b) depict the B_L model, (c) and (d) show the B_1 model, and (e) and (f) plot the two models on the same axis. Although the roof slope on the B_1 model is a little shallower than the actual value, the models are very similar, and both would be suitable for use in a simulated scene.



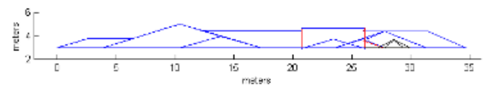
(a)



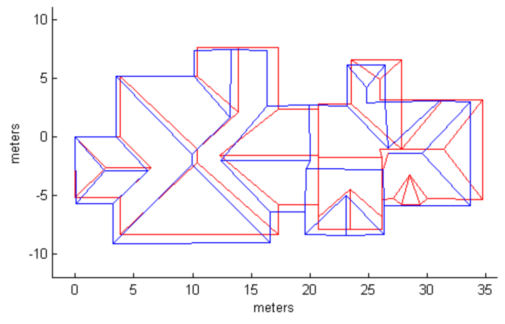
(b)



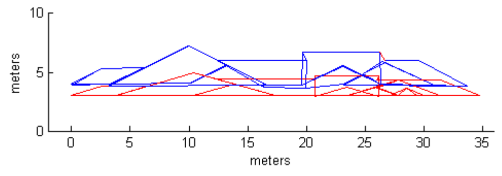
(c)



(d)



(e)



(f)

Figure 5.23: CAD models for Building B: nadir and front views: (a-b) B_L model, (c-d) B_1 model, and (e-f) both model plotted together.

Statistics related to the vertex offsets are included in Table 5.5. Although the average vertex error is slightly larger than it was in the A_1 case, the average error is still on the order of a half meter. However, most notable is the time it took to create the B_1 model. It took an expert CAD drafter nearly 2.5 hours to produce this model, while the B_L model was extracted in under 6 minutes. This represents a 97% reduction in time to create this model.

Table 5.5: Statistics on building point differences: lidar-derived model minus B_1 model (Errors are in meters)

Metric	A	B
Mean x offset	-0.5949	
Mean y offset	-0.3988	
Mean z offset	1.2247	
Mean distance (mean offset removed)	0.4123	0.5895
Variance (mean offset removed)	0.2273	0.2566
Mean squared error (mean offset removed)	0.3922	0.5984
Root mean squared error (mean offset removed)	0.6263	0.7736
Time to estimate vertex locations	25 mins	
Time to create .obj file	120 mins	
Total Time	**145 mins**	

5.4 Shed Scene: Comparing the Semi-Automated and Manual Processes

The final scene considered for this dissertation consists of three simple aluminum shed structures located in the middle of a dense grouping of trees. This region is significant since it is the only one for which we have the full set of desired image modalities. In this scene, the high-resolution lidar data and WASP frame array imagery is complemented by 1m GSD data from the COMPASS hyperspectral sensor discussed in Chapter 2. Since all the desired image modalities are present for this scene, this region was selected to perform a high-fidelity reconstruction using the methods described in Chapter 4. This semi-autonomously generated scene is termed *SAS* in the following discussion. Additionally, a manually produced synthetic scene (termed the *MPS*) of this same

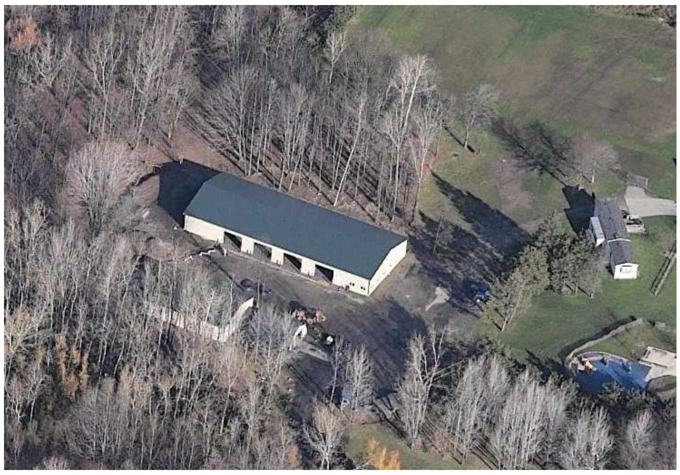


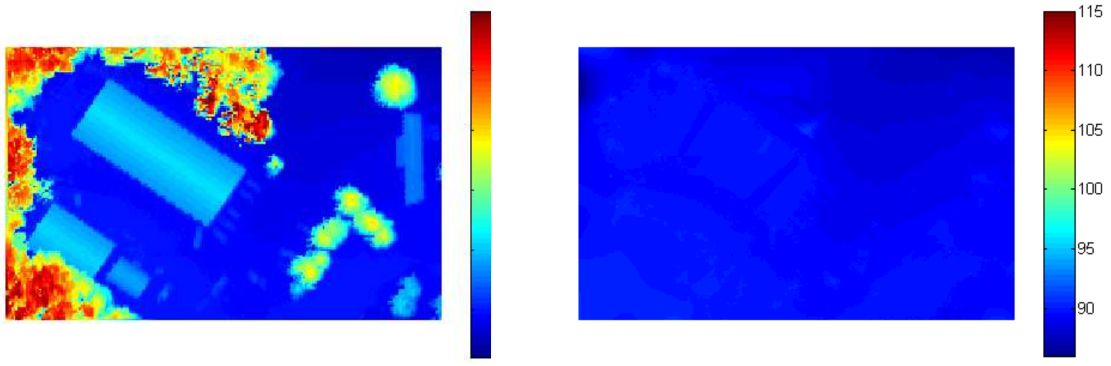
Figure 5.24: Aerial image of the shed scene region.

region has been prepared by an experienced DIRSIG scene builder. This enables a side-by-side comparison of the methods and time required to produce DIRSIG input files using both semi-automated and manual approaches.

Figure 5.24 presents an aerial view of the region used for the shed scene reconstruction. It consists of a large shed structure with a roof that is not occluded by any trees and a smaller structure of similar shape that is partially obstructed. A smaller, half-cylindrical storage shed lies nearby, and a long, thin structure is present on the eastern edge of the region. The scene contains both deciduous and evergreen trees, and the tree spacing is fairly dense around the immediate shed region.

In order to reconstruct this scene using the semi-automated approach, we began with an extraction of the DTM. The lidar data was interpolated to a 0.35m grid via the methods described in Chapter 4 in order to produce the various range images. The R_{all} image was then processed to extract the DTM using the slope-based approach with subsequent heuristics applied to remove residual tree points. The R_{all} range image and the resultant DTM are shown in Figure 5.25, and the colors in these images represent absolute elevations.

The NDEM was then created by subtracting the DTM from the R_1 image, and the non-ground



(a) (b)

Figure 5.25: Shed scene: (a) R_{all} image and (b) extracted DTM.

pixels higher than 1.5m above the NDEM were flagged as object points. Tree regions were then identified using a local texture metric on the object points. As was the case in the RIT scene, entropy proved to be adequate for distinguishing the tree and building regions, so the full GML classifier using multiple texture metrics was not employed. Once the tree regions were identified, individual tree crown locations were estimated using a watershed analysis on the smoothed CHM. The normalized entropy image (values range from zero to one) and the resultant tree locations are illustrated in Figure 5.26.

Once these tasks were completed, the main shed was reconstructed using the intersection of planes approach. Although the reconstruction of both partially occluded and cylindrical/ ellipsoidal objects have been addressed during this research effort, these techniques have not yet been fully developed, and the corresponding algorithms were not applied to this scene. As such, since the second largest shed was significantly occluded by trees, its reconstruction failed. The cylindrical shed also was not reconstructed, since it did not meet the polygonal assumption.

COMPASS hyperspectral imagery was used to build a library of spectral curves for each material in the scene. This imagery was atmospherically compensated using FLAASH, and 20 instances

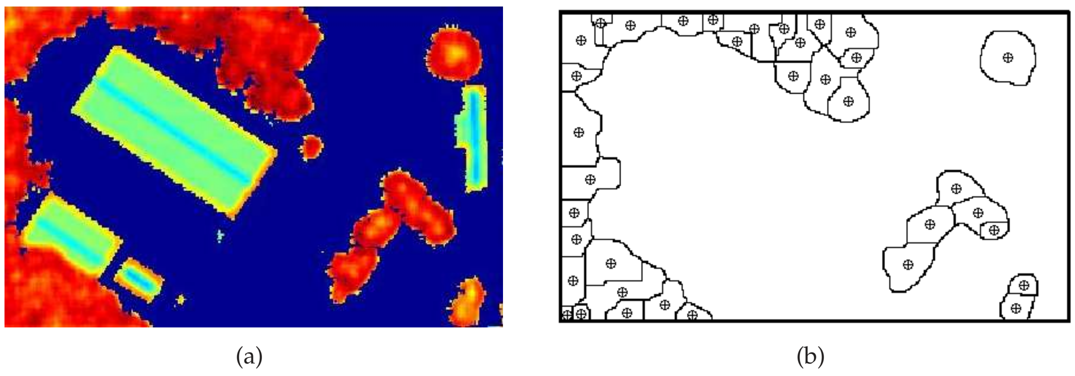


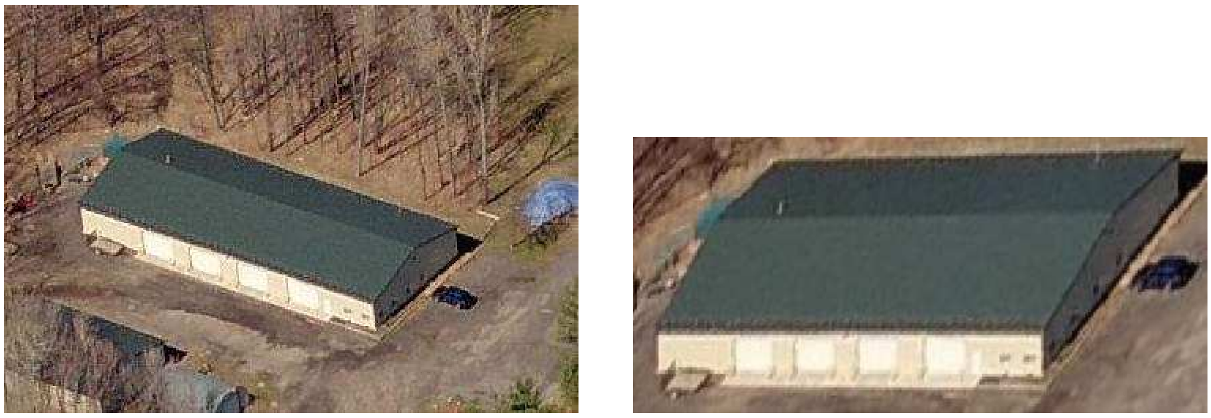
Figure 5.26: (a) Entropy image and (b) extracted tree locations.

of shed roof, grass, dirt, and gravel material types were extracted using ENVI. These spectral curves are included at the end of this chapter in Figures 5.31 and 5.32

Spectral assignment for terrain features was accomplished through the use of a material map overlaid on the terrain model. This map was produced by performing a supervised minimum-distance classification on the WASP RGB imagery (which was better registered to the other data sources than was the hyperspectral COMPASS data), then performing spatial median filtering using a 5×5 kernel to produce a more uniform result. Classes were related to actual material types through manual specification. A grayscale version of the same WASP image was used as the texture map, which was applied to the ground and shed roof classes.

Painted aluminum spectra (tan and black) were then assigned to the vertical surfaces of the shed using DIRSIG's UV mapping capability [Brown 2005]. These materials were obtained from a previously collected spectral library, since the COMPASS imagery did not contain any spectra related to the shed walls. Black was selected to be the material for the shed doors (in lieu of white) since it would show up better in the resultant imagery. In this case, we were hoping to clearly demonstrate the UV material mapping capability, and sacrificed 'spectral correctness' as a result.

An initial UV material map came from re-projecting a perspective view of the shed's wall surface to the building model facet geometries, as shown in Figure 5.27. The region relating to the facet of interest was then segmented into two classes (siding and door) using a minimum



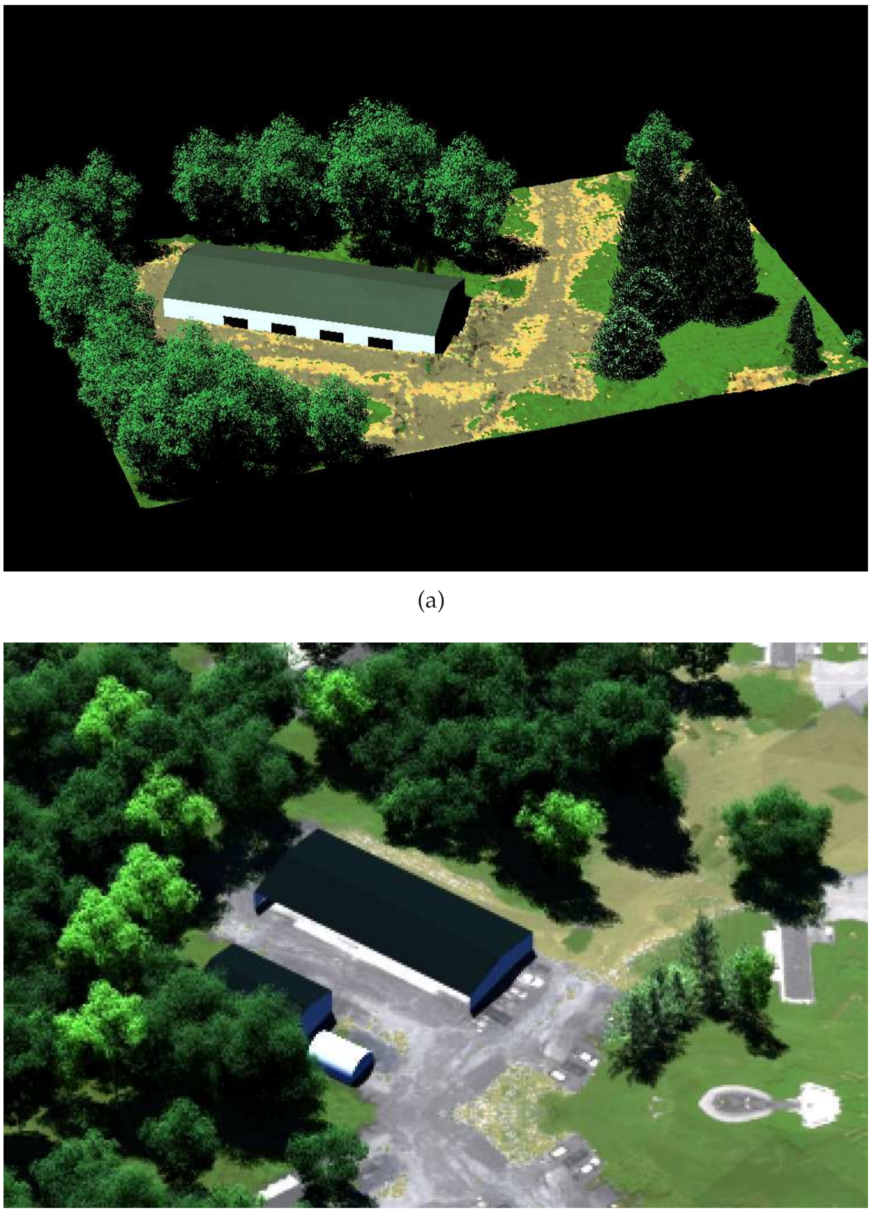
(a) (b)

Figure 5.27: Material map for shed model. (a) original frame array image (b) re-projected view used to generate material map for one facet (building side).

distance classifier. Using the first material map as a baseline, a second UV material map was then generated with standard drawing software. Small artifacts were removed in this second product, which became the baseline UV map for the scene.

Concurrent with the semi-automated scene reconstruction, an experienced DIRSIG scene drafter was asked to synthetically recreate the same region using standard scene construction techniques. He obtained a DTM model from the USGS website, built the shed models using Rhinoceros, assigned spectral signatures using Bulldozer, and also used Bulldozer's internal tree-planting tool to position tree models. The shed and several tree locations were then refined manually so that they lined up with the material and texture maps, which were obtained using WASP imagery processed in ENVI.

The two resultant DIRSIG scenes were then used to create the simulated imagery shown in Figure 5.28. The top image in this figure is a DIRSIG product created from the semi-autonomous approach, while the lower image was obtained from the manual scene. Although the manual scene appears to be of a higher resolution, this is merely due to the simulated sensor used in producing the simulated image, and is not an inherent quality of the scenes themselves. Also, the white balance used in displaying these two images was not held constant, thereby producing a

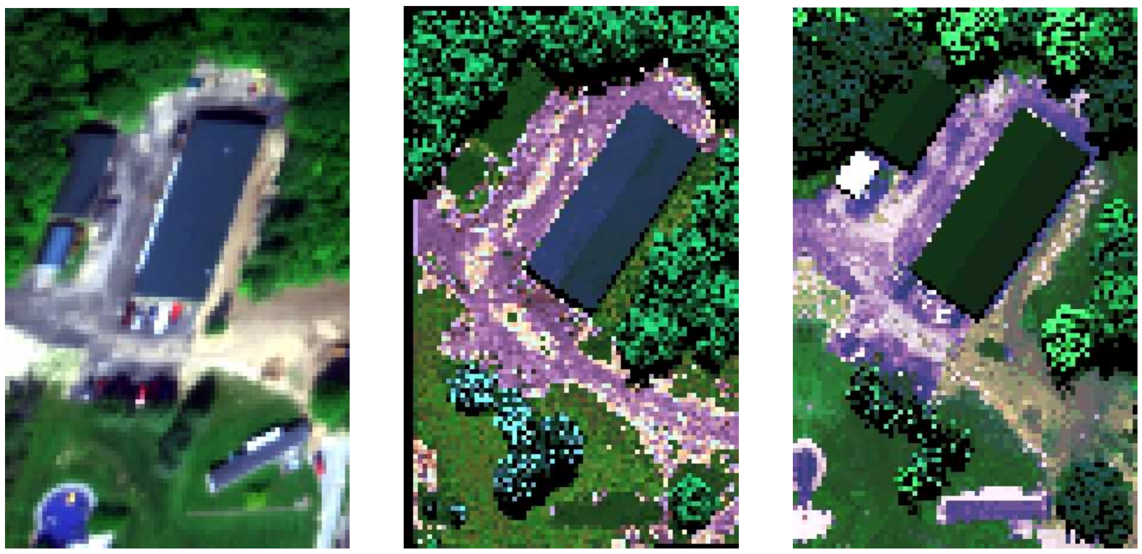


(a)

(b)

Figure 5.28: Synthetic images of the shed scene models: (a) Semi-automated approach and (b) Traditional manual approach.

noticeable color difference in the images. Figure 5.29 provides a more direct comparison of these two scenes from a nadir perspective, as the simulated GSD and color display was set to mimic



(a) (b) (c)

Figure 5.29: Shed scene images: (a) Real-world COMPASS, (b) SAS, (c) MPS. (Note: north is to the right.)

that of a portion of the COMPASS image, which is also included in the figure.

In comparing the scenes, it is worthwhile to note several interesting items. First, it is apparent that the creator of the manual scene had little difficulty recreating all three sheds in Rhinoceros, while the semi-automated approach failed in cases where the structures were either non-planar or heavily obstructed. Second, both scenes inadvertently omitted the thin structure contained on the eastern edge of the scene.

However, the manual scene is not more accurate in all details. In investigating the location of trees, it was found that the manually-derived scene had its trees planted somewhat arbitrarily, with the exception of the evergreens in front of the main shed. After talking with the scene designer, it was determined that while a few of the larger, isolated evergreen trees were located manually using Google Earth [Microsoft 2008] as a reference, most of the trees were planted in Bulldozer. While Bulldozer permits the rapid planting of tree models on the terrain using simple clicks of the mouse, when this process is being accomplished, only the 3D models are visible. Since

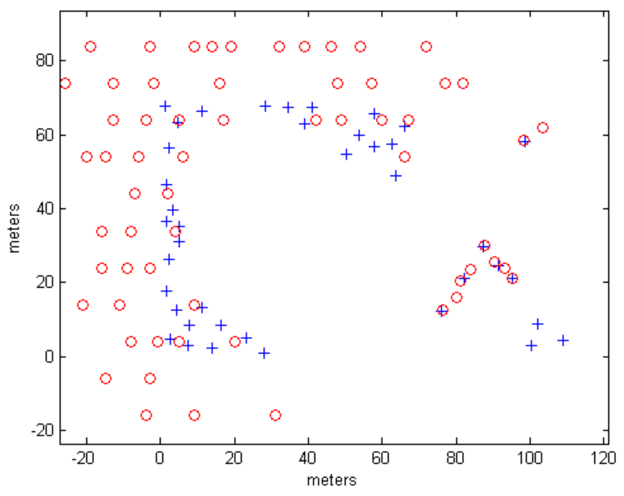
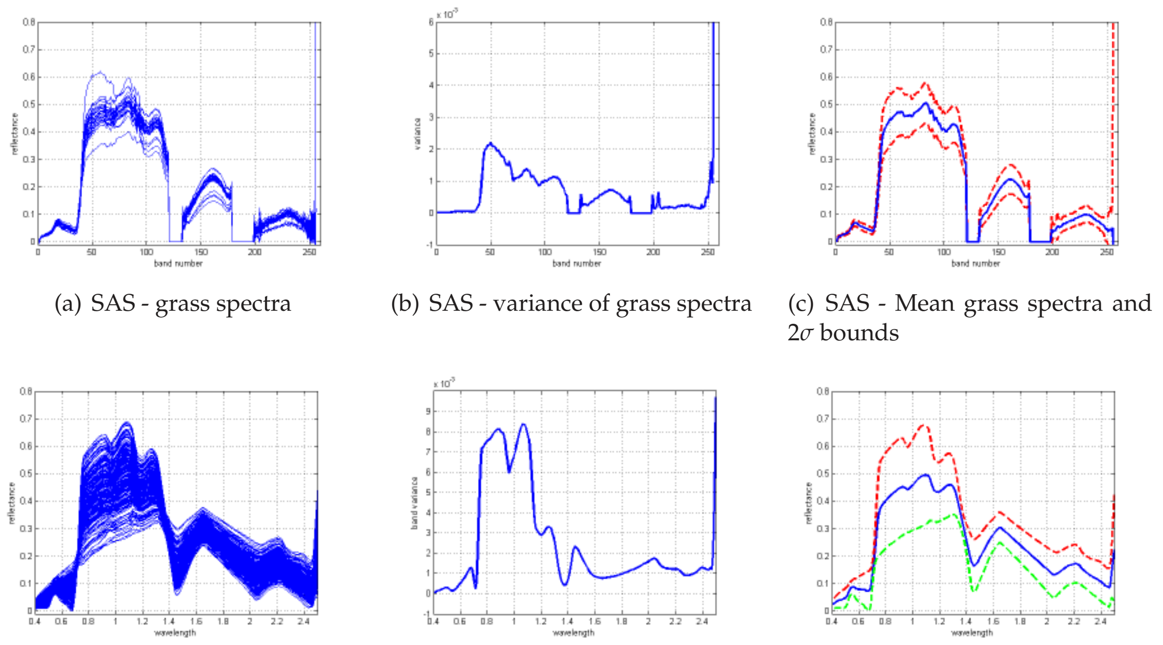


Figure 5.30: Comparison of tree center locations; blue crosses represent tree location in the SAS, and red circles represent tree locations in the MPS.

the texture and material maps are not present during this process, many trees were planted at incorrect locations, some of which had to be manually removed (those that ended up on the road, for example). Figure 5.30 highlights the difference in tree locations between the two methods. The blue crosses represent tree locations as determined by the semi-automated approach, while the red circles are the locations selected during the manual process.

Figures 5.31 and 5.32 compare the spectra used in specifying the SAS and MPS scenes. As noted above, the SAS spectra were obtained from FLAASH-compensated COMPASS imagery, while the MPS spectra were obtained from a previously-collected spectral library. Figure 5.31(a) and (d) show the full set of 'grass' spectra used to define each scene. Of note is that while the average spectral shape is nearly the same in both cases, the MPS spectra show a significantly larger range of values. This fact is more clearly shown by plotting the band variances, as was done in (b) and (e). These variances were subsequently used to place bounds of two standard deviations above and below the mean spectra, as shown in (c) and (f). Although the MPS spectra vary by more than was expected, the large number of individual spectra do provide a useful



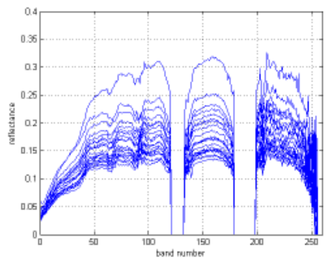
(a) SAS - grass spectra (b) SAS - variance of grass spectra (c) SAS - Mean grass spectra and 2σ bounds

(d) MPS - grass spectra (e) MPS - variance of grass spectra (f) MPS - Mean grass spectra and 2σ bounds

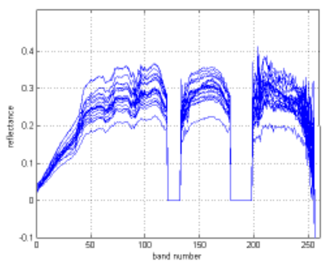
Figure 5.31: Grass Spectra: SAS (FLAASH-derived) and MPS

property; that is, the simulated imagery generated from the MPS scene will likely have better spectral texture characteristics, at least from a correlation-covariance standpoint. However this gain could potentially be more than offset by the problem of having unrealistic spectra in the scene. The SAS spectra appear to be quite similar, nicely adequately represent the desired material. However, since the number of SAS grass library is limited in size, the quality of the texturing process could suffer.

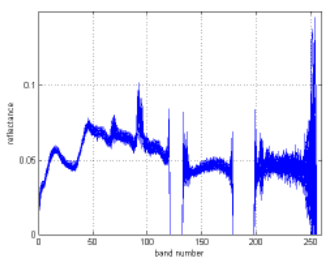
It is believed that the increased variance in the MPS spectra are due to the method in which the library was generated. It was initially thought that several grass spectra (on the order of 10-20) were obtained manually (that is, with an ASD spectrometer), and the first and second-order statistics of this set were computed. These parameters were then used to generate many additional spectral samples by modifying a large number of white noise samples so that they had first and second-order statistics that matched the collected data. However, if the original data con-



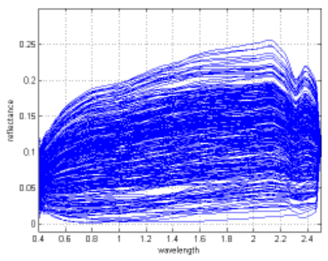
(a) SAS - gravel spectra



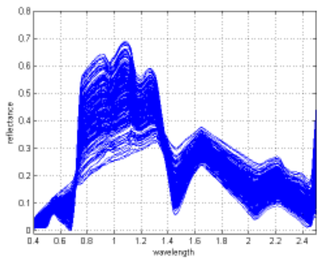
(b) SAS - dirt spectra



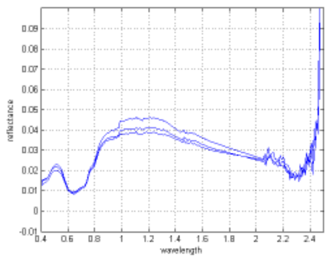
(c) SAS - green metal roof spectra



(d) MPS - gravel spectra



(e) MPS - dirt spectra



(f) MPS - green metal roof spectra

Figure 5.32: Reflectance Spectra: SAS (FLAASH-derived) and MPS

tained samples from two or three distinct grass classes (healthy and non-healthy, for example), this spectra-generation procedure would significantly blur the class distinction. The newly-generated spectra would be forced to have Gaussian characteristics, and many intermediate spectra would be produced that would not accurately represent either class in the physically-collected data.

Figure 5.32 sheds additional light on this matter. These plots show the spectra used in defining the other material types used in the scenes, and contain several items of note. Most significantly, it was discovered that the library selected to represent MPS 'dirt' actually pointed to the same file as the MPS 'grass' library. This lent further credence to the discussion above regarding large spectral variances within a single class. Although the final library had notably Gaussian characteristics, it is now believed that the originally collected data came from a mix of grass and grass-plus-dirt classes. Also apparent in this figure is the large difference between the two gravel libraries. While both libraries have a fairly large variance, the spectral character of the MPS gravel does not indicate the presence of any vegetative spectral features. However, the SAS spectra for both the gravel and dirt have noticeable features between bands 45 and 120. Although the sharp red-edge is not present to the extent it was in the grass spectra, both of these SAS libraries indicate at least some level of vegetative spectral contamination (mixing).

However, as was the case with the building reconstruction process, the scenes are more similar than they are different. Even with the noted spectral differences, the spectra still share many similarities. Also, the scene geometries, while not exactly matching, would both provide realistic and functional DIRSIG imagery. However, a very interesting comparison comes when we consider the time it took to produce the two results. Table 5.4 presents the times required to complete each task performed in the scene reconstruction process. While the manually-derived scene took over 18 hours to produce, the semi-automated approach achieve comparable results in just under 3 hours.

Table 5.6: Times to create simulated shed scene components (times in minutes)

Task	Manual Approach (Using Standard DIRSIG Tools)	Semi-Automated Approach
Generate terrain model	30	8
Generate shed CAD model(s)	60	35
Create texture image(s)	5	3
Create material map(s)	45	10
Insert material map(s)	120	1
Assign shed roof and wall materials	20	15
Assign wall texture & material maps	N/A	20
Assign ground materials	120	20
Insert terrain object	30	1
Insert shed objects	180	1
Insert tree instances	240	6
Additional worktime	240	55
Total time	**1085 minutes (18h 05m)**	**175 minutes (2h 55m)**

6 Summary and Conclusions

This research demonstrates an approach to reducing man-in-the-loop requirements for several aspects of synthetic hyperspectral scene construction. Through a fusion of 3D lidar data and passive imagery, we were able to partially automate several of the required tasks in the DIRSIG scene generation process. These included extraction of a bare-earth digital terrain model, identification of buildings and trees, object reconstruction, and the generation of background maps. Through the proper application of these techniques, we are also able to create synthetic scenes where truth data is not available, as well as to significantly reduce the time required by current scene-building methods.

The basic process first used lidar data to determine a digital model of the terrain. This terrain model was then subtracted from the original data points to yield a normalized data set. Points in this normalized data were then thresholded to yield object points, which were then segmented into building and tree regions using a GML-based approach applied to feature vectors built from texture metrics. Individual trees were then isolated and matched to previously constructed models stored in a tree library. Building geometries were extracted from the data using an intersection-of-planes approach with additional techniques applied to refine the boundary and vertical transition features. Finally, spectral assignment was done through assigning library spectra based on image features, or through a direct atmospheric compensation of hyperspectral data.

After describing the new process for scene construction, we demonstrated the feasibility of the techniques by applying them to various scenes. The DTM extraction, identification of buildings and trees, geometric reconstruction of buildings and final production of DIRSIG scenes was shown to be successful. However, due to the small number of scenes modeled, general trends related to the process may not be readily inferred from these results alone. As such, in this section we will discuss some additional aspects of the results that include trends observed from a larger number of applications of the process.

Recall that the primary goal of the terrain modelling is to produce an accurate model of the terrain, where all object points have effectively been removed. In this process, the goal is not to minimize the error associated with classifying points as either object or terrain, but rather to remove all object points, potentially at the expense of also removing some ground points. In the baseline slope and MMF processes, we achieve an inclusion of object points at a rate of approximately 2 to 4 percent. However, in the final implementation of the technique, the user is shown the resultant terrain model and is given the opportunity to remove the remaining object points, a process that nearly always removes all remaining error points. This manual intervention typically requires under one minute for scenes sized comparably to those discussed in this dissertation.

Regarding the determination of tree individual locations, it has been determined that the watershed-based approach presented here often underestimates the number of individual trees while overestimating their widths. This is due to the fact that such an algorithm is not inherently designed to trace boundaries around neighboring objects separated by small non-object regions, but instead seeks too identify a single boundary curve that separates the individual objects into unique regions. Such an algorithm may be improved upon by invoking energy-minimization constraints. However, in practice the tree placement process is adequate for most DIRSIG users, as the trees are located in vegetative regions, and in most cases, an exact forest structure is not required.

In reconstructing simple building structures without vertical transition features, we have found that the correct segmentation of planar patches is the key to producing accurate results. In more complex models, the proper identification of steep edges is equally critical. When both of these tasks are performed well, useable building models with consistent geometries are nearly always produced. Although these models may have some internal features that are not as precise as those drawn manually, in general these models have slopes and internal edge geometries that are significantly more accurate than those obtained via other means.

In general the time savings achieved by using the semi-automated approach in lieu of the manual baseline methods depends heavily on the scene being modeled. In regions with many trees and few building structures, the time savings may be limited to around 75%, since the current DIRSIG tools allow an experienced user to complete the scene with a fair amount of automation. Most of the time spent when using the manual approach on such a scene is used to fine tune the placement of object geometries. However, when the scene being considered is more complex and contains complicated building geometries, the time savings may approach 95 to 98%. In these cases, the effort required to manually understand and define the building structures using conventional CAD software makes the semi-automated approach extremely attractive.

7
Future Work

Although this research has laid the foundation for producing spectrally accurate scenes in an efficient manner, there are still many areas where the techniques could be improved. While some of these have already been addressed in part while performing this work, most of theses areas have realized very little progress.

A primary focus area for future research addresses the problem of how to extract of geometric features without the need for lidar imagery. Although this problem has received a significant amount of attention in recent years due to a host of new applications for 3D city data as well as recent advances in computer vision theory, the current state-of-the-art systems are still somewhat lacking. Occluded surfaces, illumination effects, and difficulty in performing precise feature extraction all contribute to the difficulty of this task. In a parallel research effort, we have begun to address this issue by considering new approaches to semi-autonomous digital photogrammetry. Rather than attempting to combine corner features across multiple images using robust matching algorithms, we are looking to define larger features in each image source, to include surface patches and linear structures. However, despite the progress that has been made, significant work is still required before we arrive at a system that can accurately handle arbitrary scene reconstruction with minimal user intervention.

A second topic for future research relates to the definition of vertical building walls. In the present work, walls were assumed to lie directly below the buildings' exterior roof boundary. However, in many building structures, this is not the case. This is especially true in residential environments, where porches and similar structures actually highlight walls that may be located several meters into the roof structure's interior. By considering ground-level or oblique aerial imagery, it is conceivable that the geometry of these walls could be more accurately described.

An in-depth study of the quality/errors related to this work would also be worthwhile. While an effort was made in this dissertation to demonstrate the overall quality of the process, there is currently no method to adequately describe the quality of a simulated scene. One potential method of approaching this would be through the use of simulated imagery. By using the constructed scenes to generate synthetic images similar (in terms of location, resolution, atmospheric effects, spectral sampling, etc) to the original imagery, we could do utility-based comparisons of the synthetic imagery with the original data. Examples of this include such things as target detection performance (where we would implant fractional targets in each image type and generate receiver operating characteristic curves for each image) and classification performance (where each image would be spectrally classified, and the classes compared statistically through a spatial/spectral analysis). Assuming the synthetic image generation is done well, this would permit an indirect comparison of the synthetic scene model to the real-world scene.

However, to date we have been restricted to merely describing the errors associated with individual stages in the process. And even in this, more work would be beneficial. As an example, consider the case of describing the quality of a building model. Although one may talk about the statistics of the errors of individual vertices, or the slopes of facets, we currently have no way of quantifying more significant errors, such as facets that are either missed or inadvertently added. Creation of suitable metrics for a better analysis of the results would also be extremely useful in comparing the algorithms of multiple researchers.

A more sophisticated method for creating texture images represents another area that would be a good source of future research. The current approach of using single images for texture often means that artifacts present in these images appear to be present in the resultant scenes. For example, if a tree casts a shadow that is visible in the texture image, to a certain extent that shadow will manifest itself through selection of darker spectra in the simulated imagery. Additionally, when specifying texture on vertical surfaces, if a tree is blocking a portion of the surface in the texture image, artifacts will persist throughout the scene generation process. However, by using multiple images to understand what pixels do not lie on the facets of interest and potentially inpainting new pixels, improved results could conceivably be obtained.

Another exciting front in the field of scene reconstruction deals with finding new ways to incorporate additional information into the models. Presently, we use the available imagery to extract the scene geometric and spectral content. However, if we were able to combine additional sources of information such as GIS data and models obtained by other sources, improved results could be obtained. We have briefly worked with combining multiple models, as the current research also contains two additional approaches to building reconstruction not described in this dissertation. These models may then be combined using a hypothesis and verify based approach, where for each feature, the definition of that feature from each model is compared to the image data. The best representation of this feature across all models is then retained. An additional possibility for fusing multiple models into a refined solution would be to use a Bayesian-based statistical technique.

Further research is also warranted in the area of selecting the best spectra from a previously collected library when the facets of interest are sub-pixel. As noted in Section 4.8.1, if the hyperspectral imagery has a large GSD, it may still be used to define reflectance curves. In regions where the scene does not contain any objects (such as large grassy patches), the spectral imagery may be used as though it had a small GSD. In cases where subpixel objects are present in a larger background region, a few options are available. First, a standard spectral unmixing process could be performed at these locations using a selected few library spectra as endmembers. An alternative approach is to use a physics-based, radiance space approach such as that presented in [Healey and Slater 1999], [Ientilucci 2005], and [Foster 2007].

The basic idea of these approaches is to build up a space representing all possible ways that a known target reflectance spectrum could appear in a radiance image, given a range of unknown ground geometries and atmospheric conditions. In the first two approaches cited, individual target T and background B spaces are created, and a given image pixel is then compared to $aB + n$ and $aB + cT + n$, where a and c are scalars representing faction values and n is a noise term. If the image pixel is 'closer' to $aB + cT + n$ than it is to $aB + n$ in some sense, the pixel is deemed to contain the target material. In the approach introduced by Foster, the two spaces propagated into the radiance domain are a background space B and a target plus background space TB. That is, the target and background are combined in reflectance space rather than in the hypothesis test.

Although not implemented for this dissertation, we propose that a modified version of the model introduced by Foster could potentially be used for the subpixel spectral library selection for object facets, assuming that all facets are made of the same material. In this case, a separate TB space is computed for each of the chosen library spectra, which are in turn treated as 'target' vectors. The background is assumed to be known from an analysis of the surrounding pixels. The material whose TB space is 'closest' to the hyperspectral radiance pixel is then selected as the material of the object facets. The propagation model we propose is an extension of that introduced by Foster, and it permits multiple object facets at known orientations and pixel fill ratios, so long as they contain identical materials. The assumption of diffuse facet materials is not made, but it is assumed that BRDF effects are known.

A
Tutorials

This appendix provides additional mathematical insights into some of the tasks performed in the main text. Although it will not shy away from presenting these details rigorously, in many cases the tone is intentially somewhat informal, as such a style often lends itself to easier reading. Since these sections are meant to be viewed as mini-tutorials (and are not ground breaking research concepts), it was determined that clarity and ease of comprehension would be a higher priority than formalism. That being said, much care has been taken to ensure that the topics are accurately and thoroughly covered. Through the inclusion of this material, it is hoped that the readers will be able to gain a better understanding of those topics with which they are not fully familiar, as well as be able to use the included formulas as a reference.

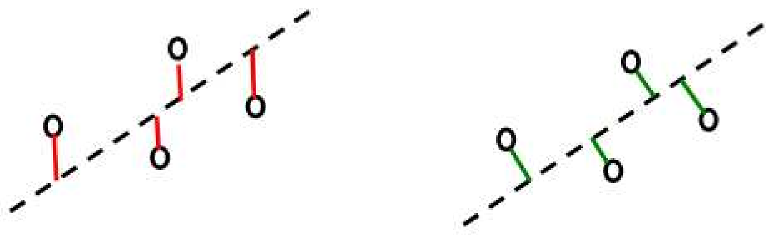


Figure A.1: Traditional (left) and orthogonal (right) least squares residuals.

A.1 Orthogonal Least Squares: Fitting Lines and Planes to Data

The method of least squares is often used to determine the line of best fit through a set of two-dimensional data samples. In traditional least squares regression, it is assumed that the independent variable (x, for the purpose presented here) is known exactly, and that the dependent variable contains all of the error. In order to obtain a line of best fit, the goal is therefore to minimize the sum of the squares of the vertical residuals (the traditional residuals are colored in red in the left portion of Figure A.1). Such an analysis is appropriate when data measurements are taken at precise intervals or known conditions, but the functional response values may be measured with error. However, for the case of finding the best line through data where errors may exist in both the x and y dimensions, and alternative solution is required. This is especially true if the data is oriented so that small changes in the x value represent large changes in y (i.e. the data is spread vertically).

‘ In these cases, it is better to minimize the sum of the squares of the orthogonal residuals, as depicted in the right side of Figure A.1. In order to do this, we must take an alternate approach. This method is referred to as Orthogonal Regression, Total Least Squares, or Deming Regression. In this approach, there is no real distinction between the various axes (at least for the methods presented here), and all variables are assumed to be measured with error. A standard method of solving this problem is given in [Li 1984], and relies on defining the residual in terms of its x and

y components, then doing a partial differentiation to determine the parameters of the line of best fit. Without derivation, this approach may be implemented as follows:

Assume the line of best fit is of the form

$$y = mx + b \tag{A.1}$$

We then let n be the number of data points, $\bar{x}$ and $\bar{y}$ are the arithmetic means of the x and y values, and

$$\begin{aligned} A &= \sum_i x_i y_i - n\bar{x}\bar{y} \\ B &= \sum_i y_i^2 - n\bar{y}^2 - \sum_i x_i^2 - n\bar{x}^2 \end{aligned} \tag{A.2}$$

Then our estimate of m and b are given as follows:

$$\begin{aligned} m &= (B + \sqrt{B^2 + 4A^2})/2A \\ b &= \bar{y} - m\bar{x} \end{aligned} \tag{A.3}$$

As shown in [Shuchat 1985], this may be shown to be equivalent to doing a principal components analysis (SVD) on the data. Per standard principal components theory, the first principal component defines the direction of maximum variation in the data, and so intutively it makes sense that this could at least be a possibility. Shuchat's proof that this must be true is given below.

For generality, let us consider the case of a line in $\mathbb{R}^n$. Let L be the line and x be the coordinates of a point, and $d(x, L)$ be the orthogonal distance from x to L. We therefore wish to minimize the sum $\sum_k d(x_k, L)^2$. By defining, v as a unit vector such that $x = w + tv$ is an arbitrary line in $\mathbb{R}^n$, by the Pythagorean theorem, we see that $d(x, L)^2 = \|x\|^2 - (x \cdot v)^2$. Therefore, we need to minimize $\sum_k \|x_k\|^2 - \sum_k (x_k \cdot v)^2$ over $\|v\| = 1$. This is the same as maximizing $f(v) = \sum_k (x_k \cdot v)^2$, since $\sum_k \|x_k\|^2$ is a constant relative to v. We now define an $m \times n$ matrix X such that its rows are the x_k data points, and an $n \times n$ matrix A as the symmetric correlation matrix $X^T X$. Now, we diagonalize A via the spectral theorem for symmetric matrices. A is orthogonally similar to the diagonal matrix D of it eigenvalues (λ_i), so $D = P^T A P$ for an orthogonal matrix P. The quadratic form corresponding to D is then

$$g(u) = Du \cdot u = \sum_i \lambda_i u_i^2 \tag{A.4}$$

Since $f(v) = g(P^T v)$ and P is an isometry (preserves lengths and dot products),

$$max\{f(v) : \|v\| = 1\} = max\{g(u) : \|u\| = 1\}. \tag{A.5}$$

However, since $g(u)$ lies in the interval spanned by the eigenvalues, its maximum value is the largest eigenvalue, and occurs when $u = e_1$ the first standard basis vector. Therefore, the maximum value of f occurs when $v = Pe_1$, which is the first column of P and the first principal component value. Therefore, this vector yields the minimum squares solution in the orthogonal sense.

This reasoning may be extended to fitting a plane to data in $\mathbb{R}^3$. In order to minimize the sum of the squared orthogonal residuals, we should choose the plane as that defined by the first two principal components. The third principal axis will be normal to this plane, so we may equivalently use this normal vector and a point in the plane to define our least squares solution.

For an extended tutorial on PCA, the reader is referred to the excellent tutorial [Shlens 2005], which is available on the author's homepage.

A.2 Rotation Matrices

Rotation matrices may be used to re-orient a vector in space (i.e. change its direction) without changing its magnitude. In the fields of computer vision and photogrammetry, they are frequently used as part of the process (along with scaling and translation) to transform points in two- or three-dimensional space from one frame of reference to another. The fundamental ideas behind the application of rotation matrices are quite simple, and are often introduced at the undergraduate level. However, due to the numerous conventions, combined with a lack of resources describing how these conventions are related, many students experience difficulty when forced to implement rotations according to a different standard than what they were taught. The goal of this section is to provide a quick tutorial on several of the ways rotation matrices may be specified, and to serve as a reference for the different matrices underlying these conventions.

The primary reason students often run into difficulty when working with rotations is the many standards that are available. Rotations may be either active or passive; they may be done using roll, pitch and yaw, other Euler Angle conventions, direction cosines, or quaternions. Multiple rotations may be defined to be relative to the original coordinate frame or relative to a previously rotated coordinate frame. A rotation may also be simply specified by a single angle and an appropriate axis of rotation. Different coordinate frame definitions are also quite common.

This section will focus on explaining the relationship between the order of rotations and the relative coordinate frame of these rotations. In practice, these are often the most difficult parameters to understand, and the remaining variations may be produced simply by following similar derivations. To be consistent with the instruments used in this research, a roll, pitch and yaw convention will be used to define the transform, and we will arbitrarily use a nose, left-wing, up (x, y, z, respectively; see Figure A.2) local right-handed coordinate system and an east, north, up global coordinate system. Additionally, for ease of discussion, we will often represent the local coordinate frame as the frame representing the orientation of an aircraft. Under these conventions, a positive 'relative' (to the aircraft) roll is defined as a rotation about the local (aircraft) x axis such that the left wing rises and the right wing dips down. Similarly, a positive relative pitch is about the local y axis, and causes the nose to dip in the $-z$ direction. A positive local yaw is therefore about the z axis, and is defined as the nose moving towards the $+y$ direction.

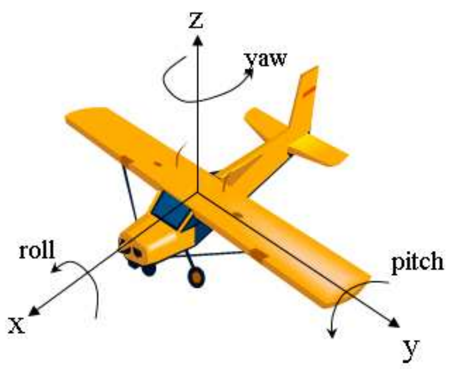


Figure A.2: Nose, left wing, up coordinate frame

Let us first address the difference between an active and a passive rotation. Although this tutorial will primarily use active rotations, many texts choose to use the passive notation, and the reader should be aware of both conventions. An active rotation rotates a vector (or equivalently, a point) by a specified angle relative to an unchanging coordinate system. An equivalent rotation may also be realized passively by keeping the point fixed and rotating the coordinate system in the opposite direction by the same angle. In two dimensions, this concept is illustrated in Figure A.3. Here, the active rotation (left) shows a vector being rotated counterclockwise by an angle θ. The passive equivalent (right) is realized by keeping the vector fixed and rotating the axes clockwise by θ. In both cases, the rotated point will be in the same position relative to the final coordinate frame, but the reader will note the change in what was being rotated, as well as the direction of this rotation. As noted above, unless otherwise specified, we will be defining rotations in the active sense.

Continuing the discussion in 2-dimensions, it is easy to see that for a point $\mathbf{V_0} = (x_0, y_0)$, we

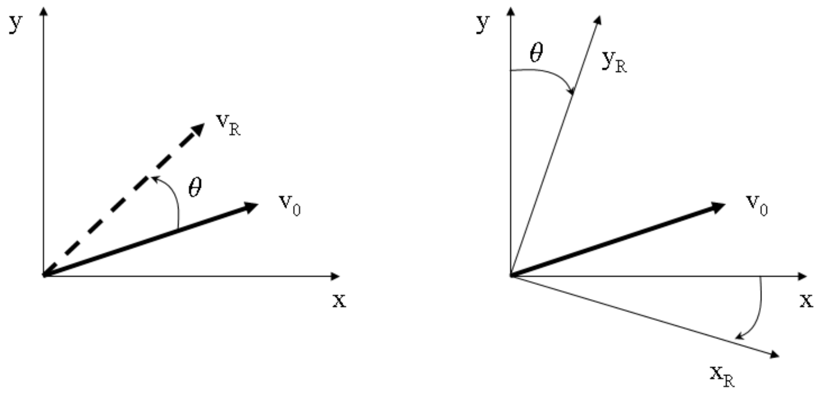


Figure A.3: Active (Left) vs. Passive (Right) Rotation

may define $\mathbf{V_R} = (x_R, y_R)$ as being the same point rotated by an angle θ, where

$$\begin{aligned} x_R &= x_0 \cos\theta - y_0 \sin\theta \\ y_R &= x_0 \sin\theta + y_0 \cos\theta. \end{aligned} \tag{A.6}$$

This can be re-written in matrix notation as

$$\begin{bmatrix} x_R \\ y_R \end{bmatrix} = \mathrm{R} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}, \tag{A.7}$$

where

$$\mathrm{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{A.8}$$

Similarly, the standard rotations about the principal axes may also be specified in three dimensions. A rotation of θ_R about the $x-$axis (roll) is defined as

$$\mathrm{R}_x(\theta_R) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_R & -\sin\theta_R \\ 0 & \sin\theta_R & \cos\theta_R \end{bmatrix}. \tag{A.9}$$

.

Pitch, or a rotation of θ_P about the $y-$axis is defined as

$$\mathrm{R}_y(\theta_P) = \begin{bmatrix} \cos\theta_P & 0 & \sin\theta_P \\ 0 & 1 & 0 \\ -\sin\theta_P & 0 & \cos\theta_P \end{bmatrix}, \tag{A.10}$$

and finally, yaw, which is a rotation of θ_Y about the $z-$axis is defined as

$$\mathrm{R}_z(\theta_Y) = \begin{bmatrix} \cos\theta_Y & -\sin\theta_Y & 0 \\ \sin\theta_Y & \cos\theta_Y & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.11}$$

In general, rotations are not limited to being about a principal direction, but may be defined relative to any arbitrary axis. However, a rotation can always be decomposed into three sequential rotations about specified axes, so long as these axes are chosen properly. If the chosen axes are the three basis vectors defining the world coordinate frame, the rotation is said to be in absolute terms. A relative rotation is one in which each of the three rotations is about the to the local coordinate frame basis vectors. Given these definitions, the reader should note that an absolute roll then pitch would first roll the aircraft about the world $x-$axis, then pitch it about the world $y-$axis. However, a relative roll then pitch would roll the the aircraft about the local $x-$axis, then pitch it about the newly produced (from the previous roll) local $y-$axis.

If we are given the appropriate rotation axes and angles for three rotations, it is quite simple to determine the generalized rotation matrix R, if the absolute convention is assumed. If we choose to roll, pitch, and yaw (in that order) about the world coordinate axes, it is easy to see that we should first multiply the point vector by $\mathrm{R}_x(\theta_R)$. This result is in turn multiplied (on the left) by $\mathrm{R}_y(\theta_P)$, and that result is multiplied by $\mathrm{R}_z(\theta_Y)$. Therefore, the generalized rotation matrix is seen to be the product of three individual rotations according to

$$\begin{aligned} \mathrm{R}_{RPY_{abs}} &= \mathrm{R}_z(\theta_Y)\mathrm{R}_y(\theta_P)\mathrm{R}_x(\theta_R) \\ &= \begin{bmatrix} \cos\theta_Y\cos\theta_P & \cos\theta_Y\sin\theta_P\sin\theta_R - \sin\theta_Y\cos\theta_R & \cos\theta_Y\sin\theta_P\cos\theta_R - \sin\theta_Y\sin\theta_R \\ \sin\theta_Y\cos\theta_P & \sin\theta_Y\sin\theta_P\sin\theta_R - \cos\theta_Y\cos\theta_R & \sin\theta_Y\sin\theta_P\cos\theta_R - \cos\theta_Y\sin\theta_R \\ -\sin\theta_P & \cos\theta_P\sin\theta_R & \cos\theta_P\cos\theta_R \end{bmatrix}. \end{aligned} \tag{A.12}$$

A fact that may not be entirely obvious, however, is that if the world and local coordinate frames are aligned before rotation (that is, the aircraft is at the world origin, and is upright facing in the $+x$ direction), then the following relationship holds,

$$\mathrm{R}_{RPY_{abs}} = \mathrm{R}_{YPR_{rel}}. \tag{A.13}$$

where $\mathrm{R}_{YPR_{rel}}$ represents starting with the yawing first, and proceeding though the pitch and roll rotations in a relative sense.

Through a similar construction, it may be seen that by deriving the equation specifying absolute yaw, pitch, and roll, the equation also specifies a relative roll, pitch yaw. The single generalized rotation matrix defined according to this convention is given below in Equation A.14.

$$\begin{aligned} \mathrm{R}_{YPR_{abs}} &= \mathrm{R}_{RPY_{rel}} = \mathrm{R}_x(\theta_R)\mathrm{R}_y(\theta_P)\mathrm{R}_z(\theta_Y) \\ &= \begin{bmatrix} \cos\theta_P\cos\theta_Y & -\cos\theta_P\sin\theta_Y & \sin\theta_P \\ \sin\theta_R\sin\theta_P\cos\theta_Y + \cos\theta_R\sin\theta_Y & -\sin\theta_R\sin\theta_P\sin\theta_Y + \cos\theta_R\cos\theta_Y & -\sin\theta_R\cos\theta_P \\ -\cos\theta_R\sin\theta_P\cos\theta_Y + \sin\theta_R\sin\theta_Y & \cos\theta_R\sin\theta_P\sin\theta_Y + \sin\theta_R\cos\theta_Y & \cos\theta_R\cos\theta_P \end{bmatrix}. \end{aligned} \tag{A.14}$$

Now, if we are able to take a given generalized rotation matrix of the form

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}. \tag{A.15}$$

and find the appropriate Euler angles according to any of the above conventions, it becomes a simple matter to convert a set of angles in one convention to the analogous angle set in the other convention. In order to do this conversion, we simply take the three angles defined using the first convention and use either Equation A.12 or A.14 to determine the single transformation R. We then use R to solve for the angles in the desired convention. However, although many authors claim the recovery of Euler angles from a rotation matrix is trivial (see [Trucco and Verri 1998], for example), the non-uniqueness of the solution and potential degenerate cases present the potential for error.

A good discussion of how one may find all possible Euler angles from a rotation matrix is given in [Slabaugh 1999]. Table A.2 summarizes the results Slabaugh gives for the RPY_{abs} case, and extends the analysis to include the RPY_{rel} case. Note that For the RPY_{abs} convention, CV (the critical matrix value) is the R_{31} entry, while for the RPY_{rel}, it is the R_{13} value.

As stated before, R may be described not only in terms of three Euler angles, but may be completely specified by a single angle of rotation (θ) about an appropriate axis (specified by the vector $\mathbf{v}$). If the components of $\mathbf{v}$ are given by

$$\mathbf{v} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{A.16}$$

then the rotation matrix may be found according to Equation A.17.

Table A.1: Determining Euler Angles from a General Rotation Matrix
(CV for RPY_{abs} is the R_{31} entry, and the R_{13} entry for RPY_{rel})

	RPY_{abs}	RPY_{rel}
if CV $\neq \pm 1$		
θ_{P1}	$-\sin^{-1}(R_{31})$	$\sin^{-1}(R_{13})$
θ_{P2}	$\pi - \theta_{P1}$	$\pi - \theta_{P1}$
θ_{R1}	$\text{atan2}(\frac{R_{32}}{\cos\theta_{P1}}, \frac{R_{33}}{\cos\theta_{P1}})$	$\text{atan2}(-\frac{R_{23}}{\cos\theta_{P1}}, \frac{R_{33}}{\cos\theta_{P1}})$
θ_{R2}	$\text{atan2}(\frac{R_{32}}{\cos\theta_{P2}}, \frac{R_{33}}{\cos\theta_{P2}})$	$\text{atan2}(-\frac{R_{23}}{\cos\theta_{P2}}, \frac{R_{33}}{\cos\theta_{P2}})$
θ_{Y1}	$\text{atan2}(\frac{R_{21}}{\cos\theta_{P1}}, \frac{R_{11}}{\cos\theta_{P1}})$	$\text{atan2}(-\frac{R_{12}}{\cos\theta_{P1}}, \frac{R_{11}}{\cos\theta_{P1}})$
θ_{Y2}	$\text{atan2}(\frac{R_{21}}{\cos\theta_{P2}}, \frac{R_{11}}{\cos\theta_{P2}})$	$\text{atan2}(-\frac{R_{12}}{\cos\theta_{P2}}, \frac{R_{11}}{\cos\theta_{P2}})$
if CV $= 1$		
θ_Y	anything (set to 0)	anything (set to 0)
θ_P	$\pi/2$	$\pi/2$
θ_R	$\text{atan2}(R_{12}, R_{13}) + \theta_Y$	$\text{atan2}(R_{21}, -R_{31}) - \theta_Y$
if CV $= -1$		
θ_Y	anything (set to 0)	anything (set to 0)
θ_P	$-\pi/2$	$\pi/2$
θ_R	$\text{atan2}(-R_{12}, -R_{13}) - \theta_Y$	$\text{atan2}(-R_{21}, R_{31}) + \theta_Y$

$$\mathrm{R}(\mathbf{v},\theta) = \begin{bmatrix} \cos\theta + (1-\cos\theta)x^2 & (1-\cos\theta)xy - (\sin\theta)z & (1-\cos\theta)xz + (\sin\theta)y \\ (1-\cos\theta)yx + (\sin\theta)z & \cos\theta + (1-\cos\theta)y^2 & (1-\cos\theta)yz - (\sin\theta)x \\ (1-\cos\theta)xz - (\sin\theta)y & (1-\cos\theta)yz + (\sin\theta)x & \cos\theta + (1-\cos\theta)z^2 \end{bmatrix}. \tag{A.17}$$

A similar construction using homogeneous coordinates is also frequently used [Bourke 2002]. This method translates the space so that the rotation axis goes through the origin, then rotates the space so that the rotation axis lies in the $x - z$ plane. The space is rotated again so that the rotation axis is the $z-$axis, and then the desired rotation is performed. The space is then unrotated about the $y-$ and $x-$ axes to return to the original coordinate frame. To implement this technique, we specify the axis vector as $\mathbf{v} = [v_x, v_y, v_z, 1]^T$ and the point to be rotated as $\mathbf{P} = [P_x, P_y, P_z, 1]^T$. We then define $d = \sqrt{v_y^2 + v_z^2}$,

$$\mathrm{T} = \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.18}$$

$$\mathrm{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{v_z}{d} & \frac{-v_y}{d} & 0 \\ 0 & \frac{v_y}{d} & \frac{v_z}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.19}$$

$$\mathrm{R}_y = \begin{bmatrix} d & 0 & -v_x & 0 \\ 0 & 1 & 0 & 0 \\ -v_x & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.20}$$

and

$$\mathrm{R}_z = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{A.21}$$

Then, the rotated point is seen to be

$$\mathbf{P_R} = \mathrm{T}^{-1}\mathrm{R}_X^{-1}\mathrm{R}_Y^{-1}\mathrm{R}_Z\mathrm{R}_Y\mathrm{R}_X\mathrm{T}\mathbf{P}. \tag{A.22}$$

Determining $\mathbf{v}$ and θ from a given rotation matrix R is also straightforward. The eigenvalues of all orthonormal matrices lie on the unit circle, and for the particular case of 3×3 rotation matrices, the eigenvalues will always be of the form 1, $e^{i\theta}$, and $e^{-i\theta}$, where θ represents the angle of rotation in a counterclockwise direction around the axis specified by **v**. This axis may in turn be found by noting that **v** is simply the eigenvector associated with the eigenvalue whose value was equal to 1.

A.3 Determining Whether or Not a Ray Intersects a Triangle

As noted in [Scott 2006], A common way to check if a point is in a triangle is to find the vectors connecting the point to each of the triangle's three vertices and sum the angles between those vectors. If (and only if) the sum of the angles is 2π, then the point is inside the triangle. This technique works, but in practice it is quite slow. Scott's methodology, as detailed below, is much more efficient.

Starting with a triangle with vertices **A**, **B**, and **C**, we note that a point **P** inside the triangle must be on the proper side of the segments $\overline{\mathbf{AB}}$, $\overline{\mathbf{BC}}$, and $\overline{\mathbf{AC}}$. To determine if it is, we take the cross product of $[\mathbf{B}-\mathbf{A}]$ and $[\mathbf{P}-\mathbf{A}]$, and see if it points in the same direction as $[\mathbf{B}-\mathbf{A}]$ cross $[\mathbf{C}-\mathbf{A}]$ (that is, **P** is on the same side of the line segment $\overline{\mathbf{AB}}$ as is **C**). If this test holds, we test **P** with the other line segments as well. If **P** is seen to be on the same side of $\overline{\mathbf{AB}}$ as **C** and is also on the same side of $\overline{\mathbf{BC}}$ as **A** and is on the same side of $\overline{\mathbf{CA}}$ as **B**, then **P** is inside the triangle.

Per the code sample in the reference, this may be implemented in Matlab as follows:

```
function return_val = point_in_tri(p,a,b,c);
return_val=0;
if sameside(p,a,b,c) & sameside(p,b,a,c) & sameside(p,c,a,b)
    return_val=1;
end


function return_val = sameside(p1,p2,a,b);
return_val=0;
cp1=cross(b-a,p1-a);
cp2=cross(b-a,p2-a);
if dot(cp1,cp2) >=0
    return_val=1;
end
```

A.4 Line of Intersection of Two Planes

Given two planes

$$\begin{aligned} Z &= a_1 X + b_1 Y + c_1 \\ Z &= a_2 X + b_2 Y + c_2, \end{aligned} \tag{A.23}$$

we may solve each equation for c_n, then subtract the second from the first to remove Z and produce

$$(a_1 - a_2)X + (b_1 - b_2)Y = -c_1 + c_2. \tag{A.24}$$

Therefore, we find the line of intersection of the two planes to be defined by

$$\begin{aligned} X &= X \quad \text{(given X, solve for the other parameters)} \\ Y &= \frac{(c_2 - c_1) + (a_2 - a_1)X}{b_1 - b_2} \\ Z &= a_1 X + b_1 \frac{(c_2 - c_1) + (a_2 - a_1)X}{b_1 - b_2} + c_1 \end{aligned} \tag{A.25}$$

B

DIRSIG Input Files

Although some of the required DIRSIG input files were introduced in Section 2.5, specifics regarding the proper formating of these files were not discussed. However, a brief description of these files is relevant, since without the ability to convert the extracted models to DIRSIG compatible input files, they are of reduced utility. This appendix will briefly cover the required file types and their basic formatting. Examples of each file are provided, and code used to convert the extracted models into the proper format is also given. Additional details regarding file formats may be found in [Brown 2005].

B.1 Conversion of Models to DIRSIG Compatible Files

Most of the discussion in this dissertation has focused on obtaining the parameters necessary for specifying a DIRSIG scene. However, there has been little information on how to take the input parameters and create actual DIRSIG input files. The code included in this section should help those who are interested convert their geometry and spectral files into a DIRSIG compatible format.

B.1.1 DTM Raster or Points to DTM .obj File

The following code may be used to convert the terrain raster image to a DIRSIG compatible .obj file. It is assumed the x-coordinate of each pixel is specified in the matrix XI, the y-coordinate in YI and the height values are in Zgnd_a_all. XI and YI are in a form obtained via the *meshgrid* command. If the terrain data is stored as point data, the first few lines that convert the pixels to point values may be omitted, and the point data may be processed directly. Once the .obj file is created, we take this file into Rhinoceros, open it and re-save it in order to easily create a new .obj file with additional geometry included (such as normal vector orientations). This new .obj file is the attributed with a singe ground class in the Bulldozer tool, and the result is saved as a .gdb file that is accessed by the DIRSIG configuration file.

```
h=rand(size(XI)); %may be used to subsample the terrain, if desired...
k=find(h>.7);
Xc=XI(:); Yc=YI(:); Zc=Zgnd_a_all(:);
Xc=Xc(k); Yc=Yc(k); Zc=Zc(k);
pts=[Xc,Yc,Zc];
tri=delaunay(Xc,Yc);
[m_tri,n_tri]=size(tri);

[filename, pathname] = uiputfile('*.obj', 'Save terrain .obj as...');

fid=fopen([pathname,filename],'wt');
fprintf(fid,'%s\n','# Rhino');
fprintf(fid,'%s\n','');
```

```
for i=1:length(k)
    fprintf(fid,'%s','v ');
    fprintf(fid,'%6.3f %6.3f %6.3f\n',pts(i,:));
end
fprintf(fid,'%s\n','');
for i=1:m_tri
    fprintf(fid,'%s','f ');
    fprintf(fid,'%1.0f %1.0f %1.0f\n',tri(i,:));
end
disp('Done creating terrain .obj')
```

B.1.2 Spectral Curve to Emissivity File

The following code may be used to convert a spectral curve (as obtained from an ENVI .sli library, from ASD measurements,or from atmospherically-compensated hyperspectral imagery) into a DIRSIG compatible emissivity file. It is assumed that the spectral curve is in units of reflectance. For converting an ENVI library into a DIRSIG emissivity file, we use:

```
[spectrum_name, pathname1] = uigetfile( {'*.TXT'},'Pick a file to convert ');
fid=fopen([pathname1,spectrum_name],'rt');
strings = textscan(fid,'%s%*[^\n]');
q=strings{1};
fclose(fid);
flag=0;
indx=2;
while flag==0;
    letters=char(q(indx));
    if letters(1)~='C'
        outval=indx-3;
        flag=1;
    end
    indx=indx+1;
end
n=outval;
spec_read=dlmread([pathname1,spectrum_name],'',n+2,0);
spec_read(:,1)=spec_read(:,1)./1000;
spec_read(:,2:end)=1-spec_read(:,2:end);

spectrum=spec_read';
```

```
[filename2, pathname2] = uiputfile('*.txt', 'Save emissivity file as...');

fid=fopen([pathname2,filename2],'wt');
fprintf(fid,'%1.0f\n',n);
dir_hem_weight_coeff=ones(91,1);
fprintf(fid,'%2.1f \n',dir_hem_weight_coeff);

for i=1:n
    a=spectrum(1:2,:);
    a(2,:)=spectrum(i+1,:);
    fprintf(fid,'%s\n','CURVE_BEGIN');
    fprintf(fid,'%8.7f %8.7f\n',a);
end
fclose(fid);
```

If we instead wish to write out a reflectance curve as obtained from compensated hyperspectral imagery, we may merely ignore the lines related to reading in the file. If we wish to read in data from an ASD file, we replace the first set of commands with the following:

```
[spectrum_name, pathname1] = uigetfile( ...
       {'*.TXT'}, ...
        'Pick a file for the image to be mapped');

spectrum=dlmread([pathname1,spectrum_name]);
spectrum(:,2)=1-spectrum(:,2);
spectrum=spectrum';
```

B.2 Example DIRSIG Input Files

B.2.1 Example DIRSIG Configuration File (.cfg)

```
DIRSIG_CFG

PATHS {
    GDB_PATH = /cis/phd/sfl1194/dirsig_shed/gdb
    ODB_PATH = /cis/phd/sfl1194/dirsig_shed
    EXTINCTION_PATH = /dirs/home/dirsig/megascene1/extinction
    ABSORPTION_PATH = /dirs/pkg/dirsig/lib/data/absorption
    MATERIAL_PATH = /cis/phd/sfl1194/dirsig_shed
    EMISSIVITY_PATH = /cis/phd/sfl1194/dirsig_shed/emissivity
    WEATHER_PATH = /dirs/pkg/dirsig/lib/data/weather
    TAPE5_PATH = /dirs/pkg/dirsig/lib/data/tape5
    MAPS_PATH = /cis/phd/sfl1194/dirsig_shed
    SOURCES_PATH = /dirs/pkg/dirsig/data/sources
    RESPONSE_PATH = /dirs/pkg/dirsig/lib/data/responses
    PROFILE_PATH = /dirs/pkg/dirsig/lib/data/data/responses

}

SCENE {
    GDB_UNITS = METERS
    ODB_FILENAME = shed.odb
    MATERIAL_FILENAME = shed.mat
    GROUND_ALTITUDE = 0.300
    DATE = 6 7 2004
   #GMT_TIME = 12.000
    GMT_OFFSET = 04.000
    LOCAL_TIME = 08.000
    LATITUDE = 43.000
    LONGITUDE = 77.000
}

ENVIRONMENT {
    TAPE5_FILENAME = mls.tp5
    ADB_FILENAME = shed.adb
    WEATHER_FILENAME = mls.wth
}

PLATFORM {
     NAME =
     INSTRUMENT {
         NAME = hydice
         TYPE = FRAMING_ARRAY
         FOCAL_LENGTH = 1208
         BAND_LIST {
             BAND {
                 NAME = VNIR spectral
                 X_PIXELS = 400
                 Y_PIXELS = 400
                 OVERSAMPLE = 2
                 MINIMUM_WAVELENGTH = 0.4000
                 MAXIMUM_WAVELENGTH = 2.500
                 DELTA_WAVELENGTH = 0.0500
                 RESPONSE_FILENAME = SPECTRAL
                 IMAGE_FILENAME = output_pics/shed_hydice_0800local.img
             }
         }
     }



    POSITION {
        #TARGET_LOCATION = 0.00,0.000,0
        #DECLINATION_ANGLE = 90.0
        #AZIMUTH_ANGLE = -90.000000
        #DISTANCE = 2000.0
        PLATFORM_LOCATION = 50,40,460
        TARGET_LOCATION =   50,40,0
    }

}

OPTIONS {
    ENABLE_TRUTH_IMAGES = TRUE
    ENABLE_THERMAL_MODEL = FALSE
    ENABLE_BRDF = FALSE
    ENABLE_MAPS = TRUE
    ENABLE_SOURCES = FALSE
    ENABLE_PLUME = FALSE
    #REMOVE_SENSOR_PATH = TRUE
    ENABLE_LINE_SKEW = FALSE
    ENABLE_EARTH_ROTATION = FALSE
    REGISTER_BANDS = FALSE
    REGISTER_DETECTORS = FALSE
    GENERATE_IMAGE_PER_SCAN = FALSE
    GENERATE_TRUTH_PER_SCAN = FALSE
    USE_SCENE_PLATFORM_ANGLES = FALSE
    USE_ALT_EPHEMERIS_GEOMETRY= TRUE
    ENABLE_APODIZATION = FALSE
    ENABLE_OFF_AXIS_ERROR = FALSE
    ENABLE_OPD_ERROR = FALSE
    #ZENITH_SAMPLES = 6
    #AZIMUTH_SAMPLES = 12
    USE_STEPWISE_PLUME = FALSE
}

TRUTH_IMAGES {
    IMAGE_FILENAME = output_pics/shed_truth.T.img
    MATERIAL_MAPS = TRUE
    HIT_MAPS = TRUE

}


 MAPS {
     MATERIAL_MAP {
         IMAGE_FILENAME = shed_material_map_nadir.pgm
         MATERIAL_ID = 100
         INSERT_POINT = 0.000, 0.000, 0.0000
         GSD = 0.3500
         LUT {
             0 = 8000
             10 = 8048
             20 = 3008

  }
    }

     MATERIAL_MAP {
         IMAGE_FILENAME = matmap2.pgm
         MATERIAL_ID = 101
         INSERT_POINT = 0.000, 0.000, 0.0000
         GSD = 0.3500
         LUT {
             50 = 1012
             51 = 1012
             52 = 1012
     53 = 1010
     54 = 7076
     #55 = 1021
     55 = 11
  }
```

```
    }
     TEXTURE_MAP {
        IMAGE_FILENAME = shed_texture_map_nadir.pgm
         MATERIAL_ID = 8000, 8048, 3008
         INSERT_POINT = 0.000, 0.000, 0.0000
         MINIMUM_WAVELENGTH = 0.4000
         MAXIMUM_WAVELENGTH = 0.7000
        GSD = 0.3500
    }

 }
```

B.2.2 Example DIRSIG Object Database File (.odb)

```
DIRSIG_ODB = 1.0

OBJECT {
    GDB_FILENAME = /cis/phd/sfl1194/dirsig_shed/gdb/shed_terrain.gdb
    UNITS = METERS
    INSTANCES {
        INFO =      0.000,      0.000,    0.000,  1.000,  1.000,  1.000,  0.000,  0.000,  0.0
    }
}
OBJECT {
    OBJ_FILENAME = /cis/phd/sfl1194/dirsig_shed/gdb/shed_uv.obj
    UNITS = METERS
    INSTANCES {
        INFO =      0.000,      0.000,    0.000, 1.000, 1.000, 1.000,  0.000,  0.000,  0.000
    }
}

OBJECT {
    GDB_FILENAME = /cis/phd/sfl1194/dirsig_shed/gdb/Red_Maple_2.gdb
    UNITS = METERS
    INSTANCES {
         INFO =  75.95 , 12.25 , -0.41 ,0.751 ,0.751 ,0.751 ,0.0 ,0.0 ,0.0
         INFO =  82.25 , 21.00 , -0.67 ,0.576 ,0.576 ,0.576 ,0.0 ,0.0 ,0.0
         INFO =  87.15 , 29.75 , -1.23 ,0.851 ,0.851 ,0.851 ,0.0 ,0.0 ,0.0
         INFO =  91.35 , 24.50 , -0.86 ,0.790 ,0.790 ,0.790 ,0.0 ,0.0 ,0.0
         }
}
```

B.2.3 Example CAD Object Files (.obj)

```
# Rhino

g object_1
v 20.64999961853027 52.84999847412109 6
v 25.20000076293945 59.84999847412109 3
v 25.20000076293945 59.84999847412109 -5
v 20.64999961853027 52.84999847412109 -5
vn -0.8384435772895813 0.5449884533882141 0
vn -0.8384435772895813 0.5449884533882141 0
vn -0.8384435772895813 0.5449884533882141 0
vn -0.8384435772895813 0.5449884533882141 0
f 1//1 2//2 3//3 4//4
g object_2
v 25.20000076293945 59.84999847412109 3
v 59.84999847412109 37.09999847412109 3
v 59.84999847412109 37.09999847412109 -5
v 25.20000076293945 59.84999847412109 -5
vn 0.5488408803939819 0.8359268307685852 0
vn 0.5488408803939819 0.8359268307685852 0
vn 0.5488408803939819 0.8359268307685852 0
vn 0.5488408803939819 0.8359268307685852 0
f 5//5 6//6 7//7 8//8
g object_3
v 54.95000076293945 29.75 6
v 54.95000076293945 29.75 -5
v 59.84999847412109 37.09999847412109 -5
v 59.84999847412109 37.09999847412109 3
vn 0.8320503234863281 -0.5547001361846924 0
vn 0.8320503234863281 -0.5547001361846924 0
vn 0.8320503234863281 -0.5547001361846924 0
vn 0.8320503234863281 -0.5547001361846924 0
f 9//9 10//10 11//11 12//12
g object_4
v 54.95000076293945 29.75 6
v 59.84999847412109 37.09999847412109 3
v 25.20000076293945 59.84999847412109 3
v 20.64999961853027 52.84999847412109 6
vn 0.1825522780418396 0.2745251953601837 0.9440924525260925
vn 0.1825522780418396 0.2745251953601837 0.9440924525260925
vn 0.1825522780418396 0.2745251953601837 0.9440924525260925
vn 0.1825522780418396 0.2745251953601837 0.9440924525260925
f 13//13 14//14 15//15 16//16
g object_5
v 15.75 46.20000076293945 3
v 20.64999961853027 52.84999847412109 6
v 20.64999961853027 52.84999847412109 -5
v 15.75 46.20000076293945 -5
vn -0.8050557971000671 0.5931991338729858 0
vn -0.8050557971000671 0.5931991338729858 0
vn -0.8050557971000671 0.5931991338729858 0
vn -0.8050557971000671 0.5931991338729858 0
f 17//17 18//18 19//19 20//20
g object_6
v 54.95000076293945 29.75 6
v 50.40000152587891 23.10000038146973 3
v 50.40000152587891 23.10000038146973 -5
v 54.95000076293945 29.75 -5
vn 0.825307309627533 -0.5646838545799255 0
vn 0.825307309627533 -0.5646838545799255 0
vn 0.825307309627533 -0.5646838545799255 0
vn 0.825307309627533 -0.5646838545799255 0
f 21//21 22//22 23//23 24//24
g object_7
v 50.40000152587891 23.10000038146973 3
v 15.75 46.20000076293945 3
v 15.75 46.20000076293945 -5
v 50.40000152587891 23.10000038146973 -5
vn -0.5547001957893372 -0.8320503234863281 0
vn -0.5547001957893372 -0.8320503234863281 0
vn -0.5547001957893372 -0.8320503234863281 0
vn -0.5547001957893372 -0.8320503234863281 0
f 25//25 26//26 27//27 28//28
g object_8
v 50.40000152587891 23.10000038146973 3
v 54.95000076293945 29.75 6
v 20.64999961853027 52.84999847412109 6
v 15.75 46.20000076293945 3
vn -0.1921903192996979 -0.2868295311927795 0.9385050535202026
vn -0.1921903192996979 -0.2868295311927795 0.9385050535202026
vn -0.1921903192996979 -0.2868295311927795 0.9385050535202026
vn -0.1921903192996979 -0.2868295311927795 0.9385050535202026
f 29//29 30//30 31//31 32//32
```

```
# Rhino
mtllib temp.mtl
g object_1
usemtl 101
v 59.84999847412109 37.09999847412109 3
v 59.84999847412109 37.09999847412109 -5
v 54.95000076293945 29.75 6
v 54.95000076293945 29.75 -5
v 50.40000152587891 23.10000038146973 3
v 50.40000152587891 23.10000038146973 -5
v 25.20000076293945 59.84999847412109 3
v 25.20000076293945 59.84999847412109 -5
v 20.64999961853027 52.84999847412109 6
v 20.64999961853027 52.84999847412109 -5
v 15.75 46.20000076293945 3
v 15.75 46.20000076293945 -5
vt .4655 .40533
vt .4655 .93
vt .2513 .17899
vt .2513 .93
vt .036974 .40533
vt .036974 .93

vt .039496 .6864
vt .9874 .6864
vt .9874 .999
vt .039496 .999

vt .5882 .4438
vt .84034 .4438
vt .84034 .5917
vt .5882 .5917

vn 0.818958044052124 0.2911122143268585 0.4945315420627594
vn 0.9798856973648071 0.1995595395565033 0
vn 0.6000692248344421 -0.4121401011943817 0.685607373714447
vn 0.8286938667297363 -0.5597021579742432 0
vn 0.04064996168017387 -0.8727313876152039 0.486505389213562
vn 0.1902058124542236 -0.9817442297935486 0
vn -0.05608460679650307 0.8672990798950195 0.4946179091930389
vn -0.2052528262138367 0.9787089824676514 0
vn -0.6018515825271606 0.4098961651325226 0.6853902935981751
vn -0.8221031427383423 0.5693386197090149 0
vn -0.8218757510185242 -0.278388649225235 0.4970110952854157
vn -0.9849203228950501 -0.173008531332016 0

f 5/13/5 3/12/3 9/11/9 11/14/11
f 5/8/5 11/7/11 12/10/12 6/9/6
f 3/3/3 5/5/5 6/6/6 4/4/4
f 11/5/11 9/3/9 10/4/10 12/6/12
f 3/11/3 1/14/1 7/13/7 9/12/9
f 3/3/3 4/4/4 2/2/2 1/1/1
f 7/8/7 1/7/1 2/10/2 8/9/8
f 9/3/9 7/1/7 8/2/8 10/4/10
```

B.2.4 Example DIRSIG Material File (.mat)

```
MATERIAL_ENTRY {
    NAME = Black Tarp
    ID = 1
    SPECIFIC_HEAT = 0
    THERMAL_CONDUCTIVITY = 0
    MASS_DENSITY = 0
    SPECULARITY = 0
    EXPOSED_AREA = 0.5
    THICKNESS = 0
    OPTICAL_DESCRIPTION = OPAQUE
    EMISSIVITY_FILE = mine/black_tarp.ems
    EDITOR_COLOR = 1.0000, 1.0000, 1.0000
    DOUBLE_SIDED = TRUE
}
MATERIAL_ENTRY {
    NAME = Grass
    ID = 2
    SPECIFIC_HEAT = 0
    THERMAL_CONDUCTIVITY = 0
    MASS_DENSITY = 0
    SPECULARITY = 0
    EXPOSED_AREA = 0.5
    THICKNESS = 0
    OPTICAL_DESCRIPTION = OPAQUE
    EMISSIVITY_FILE = megascene1/grass.ems
    EDITOR_COLOR = 1.0000, 1.0000, 1.0000
    DOUBLE_SIDED = TRUE
}
```

B.2.5 Example DIRSIG Emissivity File (.ems)

```
2
1.0
1.0
1.0
1.0
.
.
.
1.0
1.0
1.0
1.0
1.0
1.0
1.0
CURVE_BEGIN
0.3500000 0.9679560
0.3510000 0.9774490
0.3520000 0.9755070
0.3530000 0.9454230
.
.
.
2.4970000 0.7849070
2.4980000 0.7816680
2.4990000 0.7806690
2.5000000 0.7814130

CURVE_BEGIN
0.3500000 0.9579560
0.3510000 0.9874490
0.3520000 0.9455070
0.3530000 0.9554230
.
.
.
```

```
2.4970000 0.7449070
2.4980000 0.7616680
2.4990000 0.7206690
2.5000000 0.7314130
```

Works Cited

Adler-Golden, Bernstein, Levine, Berk, Richtsmeier, Acharya, Anderson, Felde, Gardner, Hike, Jeong, Pukall, Mello, Ratkowski, and Burke. "Atmospheric correction for shortwave spectral imagery based on MODTRAN4." *Proc. of SPIE, Imaging Spectrometry*. Vol 3753, SPIE, 1999, 61–69.

Ahn, C. H., S. I. Cho, and J. C. Jeon. "Ortho-Rectification Software applicable for IKONOS high resolution images : GeoPixel-Ortho." Sydney, Australia, 9-13 July 2001, 555–557.

Ameri, B. *Automatic Recognition and 3D Reconstruction of Buildings through Computer Vision and Digital Photogrammetry*. PhD thesis, Deutsche Geodtische Kommission, Reihe C, Nr. 526, Mnchen, 2000.

Argall, P. S. and R. J. Sica. *Lidar (Laser Radar) in The Encyclopedia of Optics*. Ed. T. G. Brown et al., ISBN 3-527-40320-5. Volume 2 . Wiley-VCH Verlag GmbH and Co., Dec 2003.

Axelsson, P. "Processing of Laser Scanner Data - Algorithm and Applications." *ISPRS Journal of Photogrammetry and Remote Sensing* 54 (1999): 138–147.

Baltsavias and Kaser. "DTM and Orthoimage Generation - A Thorough Analysis and Comparison of Four Digital Photogrammetric Systems." *ISPRS Commission IV*. Ed. Fritsh, Englich, and Sester `http://www.ifp.uni-stuttgart.de/publications/commIV/baltsavias202neu.pdf`, Germany: , 1998.

Behan, A., H. G. Maas, and G Vosselman. "Steps Towards Quality Improvement of Airborne Laser Scanner Data." 2000.

Berk, A., L. S. Bernstein, G. P. Anderson, P. K. Acharya, D. C. Robertson, J. H. Chetwynd, and S. M. Adler-Golden. "MODTRAN cloud and multiple scattering upgrades with application to AVIRIS." *Remote Sensing of the Environment* 65 (1998): 367–375.

Blevins, D. *Modelling Scattering and Absorption for a Differential Absorption LIDAR System*. PhD thesis, Rochester Institute of Technology, 2005.

Bourke, P. August. 2002 "Rotate a Point About an Arbitrary Axis.". `http://local.wasp.uwa.edu.au/~pbourke/geometry/rotate/`.

Brenner, C. "Building Reconstruction from Laser Scanning and Images." *Proc. ITC Workshop on Data Quality in Earth Observation Techniques*. Enschede, The Netherlands: , Nov 2003.

Breuer, M. and J. Albertz. "Geometric Correction of Airborne Line-Scanner Imagery." *ISPRS Commission III, Working Group I, Vienna* XXXI (1996): 19–23.

Brown, S. D. 2005 "DIRSIG User's Manual Release 4.". Digital Imaging and Remote Sensing Laboratory.

Brown, S. D. DIRSIG Homepage. `http://dirsig.cis.rit.edu/`, 2006.

Brown, S. D. and J. R. Schott. "Characterization techniques for incorporating backgrounds into DIRSIG." Orlando, FL, USA: SPIE, 2000, 205–216.

Burton, R. *Elastic LIDAR Modeling for Synthetic Imaging Applications*. PhD thesis, Rochester Institute of Technology, 2002.

Casey, J. T., S. R. Lach, and J. P. Kerekes. "Processing Misregistered Hyperspectral Data." Orlando, FL: SPIE Defense and Security Symposium, April 2007.

Casey, Jason. A Comparative Analysis of Hyperspectral Target Detection Algorithms in the Presence of Mis-registered Data. Master's thesis, Rochester Institute of Technology, Rochester, NY, 2008.

Chen, Teo, Rau, Liu, and Hsu. "Building Reconstruction from LIDAR Data and Aerial Imagery." Proc of IEEE, IGRSS '05, Vol 4, 2005, 2846– 2849.

Comaniciu, D. I. *Nonparametric Robust Methods for Computer Vision*. PhD thesis, Rutgers University, 2000.

CSES. 1999 "Atmosphere Removal Program (ATREM), Version 3.1, Users Guide.". University of Colorado, Boulder, Center for the Study of Earth from Space.

DIRS Laboratory. 2006 "MISI Homepage.". `http://dirs.cis.rit.edu/research/misi.html`.

Douglas, D. and T. Peucker. "Algorithms for the reduction of the number of points required for represent a digitzed line or its caricature." *Canadian Cartographer* 10 (1973): 112–122.

Duda, R. O., P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

Edelsbrunner, H., D. Kirkpatrick, and R. Seidel. "On the shapes of a set of points in the plane." *IEEE Transactions on Information Theory* IT29 (April 1983).

Edelsbrunner, H. and E. P. Mucke. 1992 "Three-dimensional Alpha Shapes.". Dept. Comp Sci, U. of Illinois Urbana-Champaign.

Elmqvist, M. "Ground estimation of laser radar data using active shape models." *Proceedings of OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Detailed Digital Elevation Models*. Official Publication No. 40, Stockholm, Sweden, 01-03 March 2001, 8p.

Elmqvist, M. "Ground surface estimation from airborne laser scanner data using active shape models." Graz, Austria. *IAPRS* 34 (September 2002): 114–118.

Elmqvist, M., E. Jungert, F Lantz, A. Persson, and U. Soderman. "Terrain modelling and analysis using laser scanner data." Annapolis, MD. *IAPRS* 34 (October 2001): 219–227.

F. Ackermann, F. and P. Krzystek. 1991 "MATCH-T: Automatic mensuration of digital elevation models.". 3rd Technical Seminar of the Sociedad Espanola de Cartografia Fotogrametria y Teledeteccion.

Fan, X. *Automatic Registration of Multi-Sensor Airborne Imagery*. PhD thesis, Rochester Institute of Technology, Rochester, NY, to be published: 2009.

Fan, Xiaofeng, Harvey Rhody, and Eli Saber. “Automatic Registration of Multi-Sensor Airborne Imagery.” AIPR, 2005, 81–86.

Fan, Xiaofeng, Harvey E. Rhody, and Eli Saber. “A harris corner label enhanced MMI algorithm for multi-modal airborne image registration.” VISAPP, 2007, 420–424.

Fischer, A., T. Kolbe, F. Lang, A. Cremers, W. Forstner, L. Plumer, and V. Steinhage. “Extracting buildings from aerial images using hierarchical aggregation in 2D and 3D.” *Computer Vision and Image Understanding* 72 (1998): 195–203.

Fischler, M. A. and R. C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.” *Communications of the ACM* 24 (1981): 381–395.

Foster, M. *Using lidar to geometrically-constrain signature spaces for physics-based target detection*. PhD thesis, Rochester Institute of Technology, Rochester, NY, 2007.

Fukunaga, K. and L. Hostetler. “The estimation of the gradient of a density function, with applications in pattern recognition.” *IEEE Transactions on Information Theory* 21 (January 1975): 32–40.

Fusiello, A. Elements of Geometric Computer Vision. `http://profs.sci.univr.it/~fusiello/teaching/visione/elementsCV.pdf`, April 2007.

Gonzalez, R. C. and R. E. Woods. *Digital Image Processing*. Second edition. New Jersey: Prentice-Hall Inc., 2002.

Goshtasby, A. “Registration of Images with Geometric Distortions.” *IEEE Transactions on Geoscience and Remote Sensing* 26 (January 1988): 60–64.

Gottesfeld-Brown, L. “A survey of image registration techniques.” *ACM Comput. Surv.* 24 (1992): 325–376.

Gray, R., S. D. Brown, and J. R. Schott. August. 2000 “Scene construction methodologies and techniques for simulating forest areas.”. http://dirsig.cis.rit.edu/doc/trees.pdf.

Gurram, Prudhvi, Stephen Lach, Eli Saber, Harvey E. Rhody, and John P. Kerekes. “3D Scene Reconstruction through a Fusion of Passive Video and Lidar Imagery.” AIPR, 2007, 133–138.

Haala, N. *Gebauderekonstruktion durch Kombination von Bild- und Hohendaten*. PhD thesis, Universitat Stuttgart, Institut fur Photogrammetrie, 1996. Deutsche Geodatische Kommission, C 460.

Habib, A., M. Ghanma, M. Morgan, and R. Al-Ruzouq. “Photogrammetric and Lidar Data Registration Using Linear Features.” *Photogrammetric Engineering and Remote Sensing* 71 (June 2005): 699–707.

Harris, C. and M.J. Stephens. “A combined corner and edge detector.” Proceedings of The Fourth Alvey Vision Conference, 1988, 147–151.

Hartley, R. and A. Zisserman. *Multiple View Geometry, Second Edition*. Cambridge University Press, 2003.

Hartley, R. I. “In defense of the eight-point algorithm.” *IEEE Trans. on Pattern Analysis and Machine Intelligence* 19 (October 1997): 580–593.

Hartley, R. I. and J. L. Mundy. “Relationship between photogrammmetry and computer vision.” SPIE, 1993, 92–105.

Haugerud, R. and D. Harding. “Some algorithms for virtual deforestation (vdf) of lidar topographic survey data.” Annapolis, MD. *IAPRS* 34 (October 2001): 211–218.

Healey, G. and D. Slater. “Models and Methods for Automated Material Identification in Hyperspectral Imagery Acquired under Unknown Illumination and Atmospheric Conditions.” *IEEE Transactions on Geo Science and Remote Sensing,* 37 (1999).

Henricsson, O. and E. Baltsavias. “3-D building reconstruction with ARUBA: A qualitative and quantitative evaluation." *Automatic Extraction of Man-Made Objects from Aerial and Space Images*. Ed. E. Baltsavias A. Grun and O. Henricsson 1997, 65–76.

Hill, P. R., C.N. Canagarajah, and D. R. Bull. "Image Segmentation Using a Texture Gradient Based Watershed Transform." *IEEE Transactions on Image Processing* 12 (Dec 2003): 1618–1633.

Humblot, Fabrice, Bertrand Collin, and Ali Mohammad-Djafari. "Evaluation and Practical Issues of Subpixel Image Registration Using Phase Correlation Methods." *Inter. Conf. on Physics in Signal and Image Processing (PSIP 05)*. Jan 2005, 115–120.

Ientilucci, E. *Hyperspectral sub-pixel target detection using hybrid algorithms and physics based modeling*. PhD thesis, Rochester Institute of Technology, Rochester, NY, 2005.

Ientilucci, E., K. Ewald, J. Marcin, and A. Spivey. A Guide to Building Large Scale DIRSIG Scenes. Digital Imaging and Remote Sensing Laboratoty, RIT, 2003.

Ientilucci, E. J. and S. D. Brown. "Advances in wide-area hyperspectral image simulation." Orlando, FL, USA: SPIE, 2003, 110–121.

Kavraki, L. Jun. 2007 "Molecular Shapes and Surfaces.". `http://cnx.org/content/m11616/latest/`.

Khoshelham, K., Z. Li, and B. King. "A split-and-merge technique for automated reconstruction of roof planes." *PE&RS* 71 (July 2005): 855–862.

Kilian, J., N. Haala, and M. Englich. "Capture and evaluation of airborne laser scanner data." B3, Vienna, Austria. *IAPRS* 31 (1996): 383–388.

Kim, Z. *Multi-view 3D object description with uncertain reasoning and machine learning*. PhD thesis, University of Southern California, 2001.

Kim, Z. and R. Nevatia. "Automatic description of complex buildings from multiple images." *Computer Vision and Image Understanding 96* (2004): 60–95.

Kraus, K. and N. Pfeifer. "Determination of terrain models in wooded areas with airborne laser scanner data." *ISPRS JPRS* 53 (1998): 193–203.

Lach, S. R., S. D. Brown, and J. P. Kerekes. "Semi-Automated DIRSIG Scene Modeling from 3D LIDAR and Passive Imaging Sources." Orlando, FL: SPIE Laser Radar Technology and Applications XI, Defense and Security Symposium, April 2006.

Lafontaine, P. A data density reduction algorithm for post-processed airborne lidar bathymetric survey data. Master's thesis, University of Florida, 2000.

Lee, D.H., K.M. Lee, and S.U. Lee. "Fusion of Lidar and Imagery for Reliable Building Extraction." *PE&RS* 74 (February 2008): 215–225.

Leica Geosystems. November. 2006 "ERDAS IMAGINE v9.1 Homepage.". `http://gi.leica-geosystems.com/LGISub1x33x0.aspx`.

Li, H. C. "A Generalized Problem of Least Squares." *The American Mathematical Monthly* 91 (Feb 1984): 135–137.

Lindenberger, J. *Laser-Profilmenssungen zur topographischen Gelandeaufnahme*. PhD thesis, Deutsche Geodatische Kommission, Munchen, 1993.

Lohmann, P. "Segmentation and filtering of laser scanner digital surface models." WGII/2, Xi'an, China. *IAPRS* 34 (August 2000): 311–315.

Ma, R. *Building Model Reconstruction from Lidar Data and Aerial Photographs*. PhD thesis, The Ohio State University, Columbus, OH, 2004.

Ma, R. and W. Meyer. "DTM Generation and Building Detection from Lidar Data." *PE&RS* 71 (July 2005): 847–854.

Ma, Y. and K. Saetzler. "A parallelized surface extraction algorithm for large binary image data sets based on an adaptive 3D Delaunay subdivision strategy." *IEEE Transactions on Visualization and Computer Graphics* 14 (January 2008): 160–172.

Maas, H. and G. Vosselman. "Two Algorithms for Extracting Building Models From Raw Laser Altimetry Data." *ISPRS Journal of Photogrammetry and Remote Sensing* 54 (1999): 153–163.

Mayer, S. "Constrained optimization of building contours from high-resolution ortho-images." Thessaloniki, Greece: ICIP, 2001.

McGlone, J. C. *Manual of Photogrammetry*. Fifth edition. American Society for Photogrammetry and Remote Sensing, 2004.

McKeown, D. M. May. 2003 "Wildfire Airborne Sensor Program: Project Overview.". Powerpoint Presentation, Industrial Associates Meeting.

Microsoft. 2008 "Google Earth Application.". `http://earth.google.com/`.

Morgan, M. and A. Habib. "Interpolation of LIDAR Data and Automatic Building Extraction." *Proceeding of ACSM-ASPRS 2002 Annual Conference*. (CD-ROM), Washington D.C.: , 2002.

NASA JPL. 2006 "AVIRIS Homepage.". `http://aviris.jpl.nasa.gov/html/aviris.concept.html`.

Nielsen, M. True Orthophoto Generation. Master's thesis, The Technical University of Denmark, August 2004.

Novak, K. "Rectification of Digital Imagery." *Photogrammetric Engineering and Remote Sensing* 58 (March 1992): 334–339.

Onyx Computing Inc. 2006 "Tree Professional Online Documentation.". `http://www.onyxtree.com`.

Persson, Asa. Extraction of Individual Trees Using Laser Radar Data. Technical Report FOI-R-0236-SE, FOI - Swedish Defense Research Agency, October 2001. ISSN: 1650-1942.

Petzold, B., P. Reiss, and W. Stossel. "Laser scanning - surveying and mapping agencies are using a new technique for the derivation of digital terrain models." *ISPRS JRPS* 54 (1999): 95–104.

Pfeifer, N., P. Sadler, and C. Briese. "Derivation of digital terrain models in the SCOP++ environment." *Proceedings of OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Detailed Digital Elevation Models*. Official Publication No. 40, Stockholm, Sweden, 01-03 March 2001, 13p.

Piech, K.R. and J.E. Walker. "Aerial Color Analysis of Water Quality." *Journal of Survey and Mapping Division, ASCE* 97 (1971): 185–197.

Piech, K.R. and J.E. Walker. "Interpretation of Soils." *Photogrammetric Engineering & Remote Sensing* 40 (1974): 87–94.

Pollefeys, M. Visual 3D Modeling from Images: Tutorial Notes. Technical report, University of North Carolina, Chapel Hill, 2002. `http://www.cs.unc.edu/%7Emarc/tutorial.pdf`.

Pollock, R. J. "A model-based approach to automatically locating tree crowns in high spatial resolution images." Proc of SPIE, Image and Signal Processing for Remote Sensing, 1996, 526–537.

Pope, P. A. and F. L. Scarpace. "Development of a Method to Geographically Register Airborne Scanner Imagery Through Parametric Modeling with Image-to-Image Matching." Washington DC: ASPRS, 2000, 11 pp.

RadiantBlue Technologies Inc. November. 2006 "OSSIM Homepage.". `http://www.ossim.org/OSSIM/Welcome.html`.

Raqueno, N. G., L. E. Smith, D. W. Messinger, C. Salvaggio, R. V. Raqueno, and J. R. Schott. "Megacollect 2004: hyperspectral collection experiment of terrestrial targets and backgrounds of the RIT Megascene and surrounding area (Rochester, New York)." *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XI. Edited by Shen, Sylvia S.; Lewis, Paul E. Proceedings of the SPIE, Volume 5806, pp. 554-565 (2005).*. Ed. S. S. Shen and P. E. Lewis June 2005, 554–565.

Robert McNeel and Associates. 2005 "Rhinoceros: NURBS Modeling for Windows.". `http:`

`//www.rhino3D.com`.

Rottensteiner, F. *Semi-automatic extraction of buildings based on hybrid adjustment using 3D surface models and management of building data in a TIS*. PhD thesis, Vienna University of Technology, 2001. Vol 56 of Geowissenschaftliche Mitteilungen der TU Wien.

Rottensteiner, F. and Ch. Briese. "Automatic Generation of Building Models from Lidar Data and the Integration of Aerial Images." Vol 34, Dresden: ISPRS, 2003.

Rottensteiner, F., J. Trinder, S. Clode, and K. Kubik. "Using the Dempster-Shafer method for the fusion of LIDAR data and multi-spectral images for building detection." *Information Fusion* 6 (2005): 283–300.

Rouse, J. W., R. H. Haas, J. A. Schell, and D. W. Deering. "Monitoring vegetation systems in the Great Plains with Third ERTS." *ERTS Symposium*. NASA No. SP-315, 1973, 309–317.

Roweis, S. Levenberg-Marquardt Optimization. `http://www.cs.toronto.edu/~roweis/notes.html`, November 1996.

Sampath, A. and J. Shan. "Building Boundary Tracing and Regularization from Airborne Lidar Point Clouds." *PE&RS* 73 (July 2007): 805–812.

Santos, T., C. Morimoto, and R. Chellappa. "Jmin-image based color-texture segmentation using watershed and hierarchical clustering." Rio de Janeiro, Brazil: International Symposium on Mathematical Morphology, Oct 2007, 35–36.

Schenk, Toni and Bea Csatho. "Fusion of LIDAR Data and Aerial Imagery for a More Complete Surface Description." ISPRS Commission III, Working Group 5, 2000.

Schott, J. R. *Remote Sensing: The Image Chain Approach*. 2nd edition. New York: Oxford University Press, 2007.

Schott, J. R., S. D. Brown, R. V. Raque no, H. N. Gross, and G. Robinson. "An Advanced Synthetic Image Generation Model and Its Application to Multi/Hyperspectral Algorithm Development." *Canadian Journal of Remote Sensing* 25 (June 1999).

Scott, J. 2006 “Point In Triangle Test.”. `http://www.blackpawn.com/texts/pointinpoly/default.html`.

Shi, W. and C. Cheung. “Performance Evaluation of Line Simplification Algorithms for Vector Generalization.” *The Cartographic Journal* 43 (March 2006): 27–44.

Shlens, J. “A Tutorial on Principal Component Analysis.” `http://www.snl.salk.edu/~shlens/notes.html`. *Unpublished* (March 2005).

Shuchat, A. “Generalized Least Squares and Eigenvalues.” *The American Mathematical Monthly* 92 (Nov 1985): 656–659.

Shufelt, J. A. “Performance Evaluation and Analysis of Monocular Building Extraction From Aerial Imagery.” *IEEE Trans on Pattern Analysis and Machine Intelligence* 21 (April 1999): 311–326.

Simi, C., E. Winter, and R. Dixon. “New Procedures for the Calibration of COMPASS.” Proc of SPIE, Vol 5425, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery X, 2004, 189–196.

SIMoN. 2006 “Sanctuary Monitoring Integrated Network.”. `http://www.mbnms-simon.org/other/moreLinks/whats_new_lidar.php`.

Sithole, G. “Filtering of Laser Altimetry Data Using a Slope Adaptive Filter.” Vol 34 - 3W4, IAPRS, 2001.

Slabaugh, G. G. August. 1999 “Computing Euler Angles From a Rotation Matrix.”. `home.comcast.net/greg\~slabaugh/publications`.

Slabaugh, G. G., R. Schafer, and M. Livingston. October. 2001 “Optimal Ray Intersection for Computing 3D Points From N-View Correspondences.”. `home.comcast.net/greg\~slabaugh/publications`.

Stewart, Charles. 2008 “Generalized Dual Bootstrap-ICP Software Guide.”. `http://www.vision.cs.rpi.edu/gdbicp/exec/`.

Supresoft Inc. November. 2006 "VirtuoZo Homepage.". http://www.supresoft.com.cn/english/products/virtuozo/virtuzo.htm.

Survey, U.S. Geological. 2008 "The National Map Seamless Server.". http://seamless.usgs.gov/.

Trucco, E. and A. Verri. *Introductory Techniques for 3-D Computer Vision*. New Jersey: Prentice Hall, 1998.

Viola, P. and W. Wells. "Alignment by maximization of Mutual Information." *Proceedings of the fifth International Conference on Computer Vision*. 1995, 15–25.

Vosselman, G. "Building Reconstruction Using Planar Faces In Very High Density Height Data." *IAPRS* 32/3-2W5 (1999).

Vosselman, G. "Slope Based Filtering of Laser Altimetry Data." Vol 33, Amsterdam: IAPRS, 2000.

Vosselman, G. and H. G. Maas. "Adjustment and Filtering of Raw Laser Altimetry Data." *Proceedings of OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Detailed Digital Elevation Models*. Official Publication No. 40, Stockholm, Sweden, 01-03 March 2001, 11p.

Weidner, U. "An approach to building extraction from digital surface models." *International Archives of Photogrammetry and Remote Sensing* 31 (1996): 924–929.

Weidner, U. *Gebaudeerfassung aus digitalen Oberflachenmodellen*. PhD thesis, University of Bonn, 1997. DGK Volume 474.

Weidner, U. and W. Forstner. "Towards Automatic Building Extraction from High Resolution Digital Elevation Models." *ISPRS Journal of Photogrammetry and Remote Sensing* 50 (1995): 38–49.

Wiemker, R. "Registration of Airborne Scanner Imagery Using AKIMA Local Quintic Polynomial Interpolation." San Francisco, CA: Second International Airborne Remote Sensing

Conference and Exhibition, June 1996.

Wolf, P. R. and B. A. Dewitt. *Elements of Photogrammetry with Applications in GIS*. Third edition. McGraw Hill, 2000.

Yang, Stewart, Sofka, and Tsai. “Registration of Challenging Image Pairs: Initialization, Estimation, and Decision.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (Nov 2007): 1973–1989.

Zadnik, J., D. Guerin, R. Moss, A. Orbeta, R. Dixon, C. Simi, S. Dunbar, and A. Hill. “Calibration Procedures and Measurements for the COMPASS Hyperspectral Imager.” Proc of SPIE, Vol 5425, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery X, Vol 5425, 2004, 182–188.

Zhang, K., S. Chen, D. Whitman, M. Shyu, J. Yan, and C. Zhang. “A progressive morphological filter for removing non-ground measurements from airborne lidar data.” *IEEE Transactions on Geoscience and Remote Sensing* 41 (April 2003): 872–882.

Zhang, K. and D. Whitman. “Comparison of three algorithms for filtering airborne lidar data.” *PE&RS* 71 (March 2005): 313–324.

Zhang, W., J. Albertz, and Z. Li. “Rectification of Airborne Line-Scanner Imagery Utilizing Flight Parameters.” First International Airborne Remote Sensing Conference and Exhibition, Strasbourg, France: , 11-15 September 1994.

Zhang, Z. Determining the Epipolar Geometry and its Uncertainty: A Review. Tech Report Theme 3, 2927, Institut National De Recherche en Informatique et en Automatique, July 1996.

Zhao, Z. and A. Saalfeld. “Linear-Time Sleeve-Fitting Polyline Simplification Algorithms.” AutoCarto 13, Seattle, Washington: , 1997, 214–223.