Broadband World Modeling and Scene Reconstruction

Benjamin Joseph Goldman

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Masters of Science

in

Mechanical Engineering

Alfred L. Wicks

Associate Professor of Mechanical Engineering

Kathleen Meehan

Associate Professor of Electrical and Computer Engineering

Kevin B. Kochersberger

Research Associate Professor of Mechanical Engineering

May 2, 2013

Blacksburg, Virginia

Keywords:

Unmanned Ground Vehicles, Robotic perception, Broadband cameras, 3D Scene Reconstruction

Broadband World Modeling and Scene Reconstruction

Benjamin Joseph Goldman

ABSTRACT

Perception is a key feature in how any creature or autonomous system relates to its environment. While there are many types of perception, this thesis focuses on the improvement of the visual robotics perception systems. By implementing a broadband passive sensing system in conjunction with current perception algorithms, this thesis explores scene reconstruction and world modeling.

The process involves two main steps. The first is stereo correspondence using block matching algorithms with filtering to improve the quality of this matching process. The disparity maps are then transformed into 3D point clouds. These point clouds are filtered again before the registration process is done. The registration uses a SAC-IA matching technique to align the point clouds with minimum error. The registered final cloud is then filtered again to smooth and down sample the large amount of data. This process was implemented through software architecture that utilizes Qt, OpenCV, and Point Cloud Library. It was tested using a variety of experiments on each of the components of the process. It shows promise for being able to replace or augment existing UGV perception systems in the future.

ACKNOWLEDGMENTS

I would like to first thank God for his provision in a allowing me to complete this phase of my life. Secondly I'd like to thank my parents for supporting my through my education from the very start and seeing me thus far. Thirdly, I want to thank my fiancé, Anna, for putting up with my stress, and neglect while I worked on this and for encouraging and supporting me through the whole process.

I would also like to thank my advisors at Virginia Tech: Dr. Wicks, Dr. Meehan, and Dr. Kochersberger. Without Dr. Kochersberger's encouragement in senior design, I would not even be studying this topic in graduate school. Dr. Wicks and Dr. Meehan have both be a great source of guidance and encouragement in my graduate studies. They have all played a significant part in my academic progression and have taught me many important academic and life lessons.

Finally, acknowledgements and thanks are also due to many of my fellow Mechatronics Lab members. Dr. Jonathan Gaines and Dr. John Bird both have been great sources of knowledge and helped to guide my early research. My lab mates and friends, Mark Umansky, Michael Teresi, and William French have made sure that I am working hard and thinking well. I have really enjoyed the years we've been together in the lab. I'd also like to recognize the help of Daniel Moodie, James Burns, Ryan Brown Chase Kees, and David Uliana who have been unbelievably great assets when I struggled with the coding portions of this project.

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

Acronym	Definition
2D	Two Dimensional
3D	Three Dimensional
BM	Block Matching
CCD	Charge-coupled Device
DARPA	Defense Advanced Research Projects Agency
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
EO	Electro-optical
FOV	Field of View
FPFH	Fast Point Feature Histograms
GPU	Graphical Processing Unit
GUI	Graphical User Interface
GUSS	Ground Unmanned Support Surrogate
I/O	Input and Output
ICP	Iterative Closest Point
INS	Inertial Navigation System
IR	Infrared
JPL	Jet Propulsion Lab
LIDAR	Light Detection and Ranging
LWIR	Long Wave Infrared light
MLS	Moving Least Squares
MLVT	Mechatronics Lab at Virginia Tech
MWIR	Mid Wave Infrared light
NIR	Near Infrared light
PCL	Point Cloud Library
PCL	Point Cloud Library
pdf	Probability Distribution Function
PFH	Point Feature Histogram
RANSAC	Random Sample Consensus
SAC-IA	Sample Consensus – Initial Alignment
SAD	Sum of Absolute Differences

SDK	Software Developer Kit
SGBM	Semi-Global Block Matching
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SPFH	Simplified Point Feature Histogram
SURF	Speeded Up Robust Features
SWIR	Short Wave Infrared light
UGV	Unmanned Ground Vehicle
UKF	Unscented Kalman Filter
VIS	Visible light

CHAPTER 1

1 BACKGROUND

Unmanned ground vehicles (UGVs), a subset of autonomous mobile robotics, have been a source of interest and an area of research since the 1940s [1]. They vary widely in form and functionality, ranging from hoping robots modeled after grasshoppers [2] to autonomous cars that can navigate an urban environment [3]. UGVs have the ability to impact a vast portion of human society in the coming decades. The vision for these vehicles is extremely broad. Many of the environments typically traversed by UGVs have been structured. This describes numerous settings in which humans operate. Structured environments do not encompass all of the terrain in which UGVs are being called upon to operate. Settings such as a disaster areas or countryside are not ordered. Obstacles and objects in these situations are highly variable and do not follow strict rules. Thus, these environments were not researched initially due to their increased difficulty. However as robotic autonomy and perception have increased, the branch of UGVs for use in unstructured environments has continued to grow.

1.1 UGVS IN UNSTRUCTURED ENVIRONMENTS

The conditions that UGVs encounter vary significantly. They can include any terrain that exists on earth and even include extraterrestrial environments. As human civilization has progressed, we have modified our environments to fit our needs. Areas such as cities are perfect examples of this trend. They are a network of roads, sidewalks, and buildings. They are designed to increase human efficiency and comfort. To aid in this goal, these areas are structured and tend to have general rules that govern the behavior of the humans existing in these surroundings. These rules and set patterns assist in defining the operating parameters of the UGVs designed to work in these areas. However, urban areas consist of less than 4% of the earth's ground surface. Overall, less than 50% of the land mass is developed for any kind of human usage [4]. This leaves a large portion of the world closed off to UGVs designed to fit into the systems and structured environments in which humans live and work. This remaining portion of the world can be classified as unstructured. It does not have an overarching set of behaviors or structures that the UGV can know beforehand. This simple distinction adds greatly to the difficulty of operating a UGV in these settings. The UGV must not only interpret its mission objective, but it must do so in an ever changing world that does not have set rules. This vast and diverse portion of the world coupled with the human race's quest to explore and understand the whole of the earth's surface provides an equally vast and diverse realm in which UGVs can be utilized. From the exploration of active volcano craters [5] to robotic military pack mules [6], there are countless possible unstructured environments and missions for which UGVs can be designed.

1.2 UGV RESEARCH IN THE MECHATRONICS LAB

The Mechatronics Lab at Virginia Tech (MLVT) has been involved in the research and development of applied autonomous vehicles. Many of these vehicles fall into the UGV category. Beginning in 2004, Virginia Tech entered several vehicles into the Defense Advanced Research Projects Agency (DARPA) challenges [7]. These vehicles enjoyed great success in both the DARPA Grand Challenge (2004, 2006) and the DARPA Urban Challenge (2007) [3]. The competitions challenged the vehicles to navigate both unstructured and unstructured environments using onboard sensors and autonomy algorithms. A great

advance of the current state of technology was the result of these competitions. The knowledge gained by the MLVT was applied to the founding of TORC Robotics as well as a series of UGV defense and security related applications. These advances were also utilized to produce current military experimental vehicles, such as the Ground Unmanned Support Surrogate (GUSS).

1.2.1 DARPA CHALLENGES

The vehicles developed by the MLVT, along with the others in the DARPA challenges, accelerated the state of UGV technology forward at a very high rate. The UGVs from the DARPA Grand Challenge had to deal with both structured and unstructured environments. The ability of the Mechatronics Lab vehicles to interact with these situations was greatly improved over the four years of DARPA competitions. Many perception, drive-by-wire, and sensor improvements were made possible by the DARPA challenges.

Cliff was the first MLVT DARPA Grand Challenge vehicle [8]. It set the stage for the next several years of applied autonomy by the MLVT. The base vehicle for Cliff was a prototype vehicle made by Ingersoll-Rand. The computational power behind the vehicle's autonomy is from three National Instruments controllers programmed using LabVIEW. Cliff utilized an array of sensors to achieve its mission. These sensors consisted of a Light Detection and Ranging (LIDAR), stereo cameras, and an Inertial Navigation System (INS). The LIDAR and INS were the primary sensors, with the stereo vision providing support functionality. Cliff was one of fifteen vehicles selected to compete in 2004. Rocky was the second MLVT DARPA Grand Challenge vehicle. Rocky was a small four wheel drive off-road vehicle with the slightly upgraded base chassis from the one used for Cliff [9]. The autonomy software was similar in structure to Cliff, running on four National Instruments controllers running LabVIEW. The sensor configuration was also largely the same. The LIDAR was used for obstacle detection and map building. The INS was used to better measure the vehicle's motion. Both Cliff and Rocky, seen in Figure 1-1, were selected in the top twenty three spots to compete in the final stage of the DARPA Grand Challenge in 2005, Rocky was selected in the top ten and Cliff made an impressive show in a later trial to earn one of the coveted positions [10]. The MLVT learned many lessons by competing in the DARPA Grand Challenge. It enabled this team of researchers to develop control and perception algorithms that were practically applied to UGVs used in unstructured environments.

Figure 1-1. DARPA Vehicles, Rocky and Cliff. The DARPA grand challenge vehicles developed by the MLVT. (left) Rocky is the second generation vehicle that finished in the top ten of the 2006 grand challenge. (right) Cliff is the original MLVT grand challenge entry [11].

The next stage in the DARPA challenges was the Urban Challenge in 2007. This featured a more structured environment than the previous competitions, but was significantly more demanding. This competition required the vehicles to follow California driving regulations along with additional DARPA imposed constraints. All of this was done in an environment that resembled a typical urban setting. Odin was the entry from the MLVT, into the DARPA Urban Challenge [12][13]. The base vehicle for Odin was a 2005 Hybrid Ford Escape. It can be seen in Figure 1-2. The hybrid system was crucial in simplifying the power generation and conversion for the vehicle's onboard systems. This vehicle also allowed for easy drive-by-wire integration. Other system components on Odin were more complex than in the previous competition vehicles. The computing system was composed of two HP rack-mounted servers that ran multiple virtual machines to allocate hardware usage for each software module.

Figure 1-2. Odin: DARPA Urban Challenge vehicle. The MLVT entry into the 2007 DARPA Urban Challenge, work done in conjunction with TORC Robotics.

The perception system consisted of two cameras for lane detection, two forward facing Ibeo multi-plane LIDAR units, one rear facing IBEO LIDAR, and four single plane LIDAR units to eliminate the remaining blind spots around the vehicle. The perception system for Odin proved to be a very competitive solution. It used a LIDAR-based system to successfully navigate the urban environment assisted by a couple cameras. The system architecture for Odin increased in complexity to meet the trials set forward in this competition. The DARPA Urban Challenge proved to be very beneficial to the MLVT. At the conclusion of the competition, Odin finished third, behind only Stanford and Carnegie-Mellon [3]. The knowledge gained and technology developed in this event was the catalyst for the establishment of TORC Robotics, a robotics company based in Blacksburg, VA.

1.2.2 GUSS

The next UGV to be developed by the MLVT was the Ground Unmanned Support Surrogate (GUSS) [14] [15]. GUSS is a support vehicle for the United States Marine Corps (USMC) that can be seen in Figure 1-3. It was an experimental vehicle that was commissioned by the Marine Corps Warfighting Lab. It was designed to support a squad of marines by carrying their gear and water in a follow-me mode. It also could be teleoperated to provide forward reconnoitering for the squad.

Figure 1-3. Ground unmanned support surrogate developed by MLVT and TORC Robotics for the USMC

1.3 PERCEPTION

The perception subsystem was a key component of each of the MLVT's vehicles described above. The role that perception plays in the autonomy of UGVs can be seen in the following sections. Perception is defined as 'to become aware of something through senses' [16]. Perception is a key part of any living being's interaction with the environment in which it lives. There are many examples of perceptions systems that are used by animals and humans. These range from the sound-based methods used by dolphins, to the electric-field detection used by fish, and the magnetic-field detection used by birds. Each of these systems use a very unique set of sensors by which it perceives a variety of different types of information. The natural world is full of this sensory and perceptive diversity. As humans, we do not have many of these types of sensors. Instead, we rely on five main sensory types to perceive our environment: hearing, touching, tasting, smelling, and seeing. Each of these has a different functionality and is used to perceive a different part of our environment. This combination of sensors gives humans a range of data by which to interpret the world we live in. As humans strive to create robots that operate in the same world as we do, it is necessary to use sensors to give the robot an understanding of its world. Most of the sensors that are used on robots find their inspiration in the natural world. To better understand how to equip robots with sensors, we must be able to quantify what perception is and to categorize the types of sensors.

There are several levels that this awareness or perception fall into. Detection, recognition, and identification were defined by Johnson as the levels of the human perception and observation system [17]. These criteria can be used in robotics to describe the minimal amount of information needed to reach each level. The information is collected using an array of sensors that inform the robot of its environment. This information comes in forms such as range, reflectivity, texture, and emitted energy. The data that is collected by the sensors is processed using various navigation and classification techniques. The final result of robotic perception is that the robot has an understanding of the composition of the world that it observes and knows the location of obstacles and how to traverse its environment. As mentioned in 1.2.1, robotic perception was one of the key areas of advancement in the DARPA challenges. The progression of both sensors and algorithms helped the MLVT to create practical UGVs. One example of this phenomenon is in the short time from the start of the DARPA challenges, the MLVT perception sensors have moved from an array of single plane Light Detection and Ranging (LIDAR) units to a single thirty-two plane LIDAR unit [18]. As with any market, the creation of these new sensors was directly driven by

the explosion of UGV applications. The tireless efforts of many excellent researchers have also led to an ever growing number of algorithms that process and make sense of the data [19]. Other factors in the rapid expansion in the field of perception are the increase of computing power and the development of cheaper and better sensors. Since the field of perception is so broad, this work is focused on the application of visual based perception sensors.

1.3.1 ACTIVE PERCEPTION

All of the MLVT vehicles sensed their environments using a primarily active perception system. Active perception is founded in the concept of emitting energy into the UGV's environment and sensing the response to gather data on the status of the sensed area. These perception systems are typically founded on sensors such as LIDAR, Sound Navigation and Ranging (SONAR), and Radio Detection and Ranging (RADAR), which are derived from nature-based systems. The concept of all of these active sensors is the same. Each one emits a type of waveform or electromagnetic radiation and measures the intensity and time delay of the return signal. This provides the sensor with information on the distance and a general identity of the object. The practical applications of these sensors vary in sophistication and accuracy. These systems have been historically used to detect and identify objects, obstacles, and build maps upon which the UGV must base its navigation decisions[9], [13], [18], [20], [21]. These sensors have proven to be reliable. They provide accurate ranging data that informs the UGV of the location of objects in its environments. Another key benefit of these types of active ranging systems is that they provide a dense data without overwhelming current processing systems. This was an important feature in many previous robotics systems. Many of the perception algorithms that have been developed in the history of robotics have relied active ranging sensors[9], [13], [20], [21]. The MLVT UGVs have relied almost exclusively LIDAR for perception.

Several former graduate students in the MLVT have explored active perception. As mentioned above, LIDAR was the primary sensor for the previous UGVs developed by the MLVT. Stephen Cacciola's thesis covered the detection of specific obstacles using LIDAR data supplemented with visual cameras [21]. Peter King's thesis focused on sensor fusion of LIDAR, Global Positioning System (GPS), odometry, and compass data for more accurate vehicle localization. These sensor suites, in addition to others, heavily utilized LIDAR based perception [22]. The results of these approaches were quite successful and proved to be instrumental in the success garnered by the MLVT's UGVs in both the DARPA challenges and the following vehicles.

1.3.2 PASSIVE PERCEPTION

In contrast to active perception, most creatures use passive perception. As most of the robots today have garnered inspiration from the natural world, it is not surprising that passive sensing has been offered as a solution to robotic perception. Passive perception relies on detecting emitted and reflected energy from a general uncontrolled source instead of from a controlled source. The sensors required to achieve passive perception primarily are imagers, which are electro-optical (EO) sensors. This type of sensor was developed to mimic the eye. It captures the reflected and emitted light over a particular range of energies by focusing it on a sensory array. The incident light is rendered into an image. With images, humans can interpret and perceive the contents of this data. However as with active perception, the interpretation of this data is a key factor.

Passive perception has many benefits. It adds many new forms of data. This data comes in the form of dense images. Images depict the returned intensities of a specified bandwidth of energy that is captured by the camera over a discrete period of time. The intensities are stored in bins termed pixels. Visible cameras are the most common type of camera; they have been tuned for human consumers. It captures light ranging from 400 nm to 700nm. This range is split into three smaller bands by filters on the sensor. Finally the data is interpreted by the camera to produce an image with red, green, and blue color planes that are combined to produce what we know as a color image. With other types of cameras, many more color bands can be captured [23], providing information about different portions of the electromagnetic spectrum, which can be seen in Figure 1-4. All of this added data can be used to improve the ability of the robot to detect, recognize, and identify its environment.

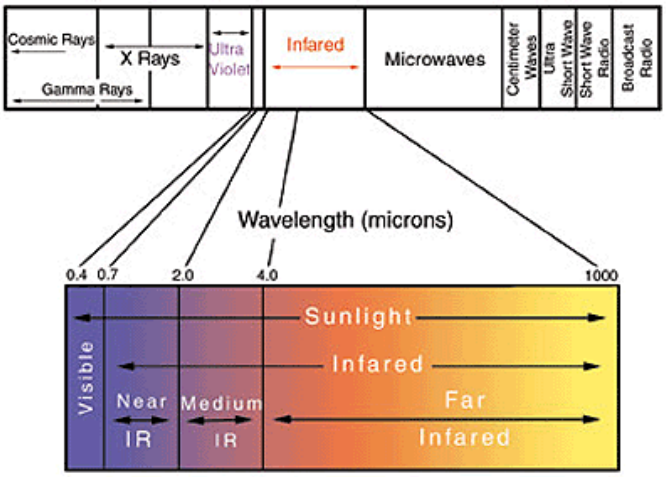

Figure 1-4. The electromagnetic spectrum showing the visible and near infrared spectrums in magnification [23].

One of the major issues with robotic vehicles is their exorbitant cost. For example, to create a UGV from a standard car, the cost would be two to three times the cost of the base vehicle, because of the extra systems that must be added. Much of this additional cost is driven by the sensory package. Current active sensors, such as the 32-plane Velodyne LIDAR [18], are $30,000 per unit. Many of the active perception systems cost as much or more than the vehicle on which they are mounted. Passive perception provides a unique solution to this problem. Digital cameras have seen a very large increase in demand since their invention, which has driven the cost downward [24]. By structuring a passive perception system in an adequate manner, passive perception systems that use off-the-shelf cameras can provide more information for less cost than current active perception systems. Another key factor driving passive perception in UGV applications is the reduction of the UGV's energy signature. With active perception systems, energy is purposefully emitted from the robot's sensors. However, the emitted energy is drastically reduced with passive systems and tends to be thermal energy, not light energy that can be easily detected by commonly available sensors.

Although passive perception has significant and valid arguments for its preeminence over active perception systems, there are several factors that discourage roboticists from incorporating such systems into UGVs. Passive perception systems can collect a wider range of data through the use of cameras and other passive sensors than their counterparts in active perception. Although this additional data can be incredibly useful, it must be interpreted by the robot's computing system. In addition to the shear amount

of data, passive perception systems do not directly return distance measurements. This must be gleaned from the images and requires a significant amount of processing power. Also, feature and obstacle detection can be more processing intensive when using passive sensor data because of the pre-processing that must be done before applying the techniques used for the active systems. To some degree, these issues have been alleviated over the past decade as the advances made in active perception have been applied to passive perception [25], [26], [27].

The promise of passive perception is alluring, yet there are many practical considerations before a wholesale migration occurs for robotic systems. The transition has already begun. Many current systems incorporate both passive and active perception systems. This solution provides a compromise between these two options. Since active perception systems are more mature, passive systems contribute a supporting role. As the technology advances, passive perception will take a more prominent role, eventually surpassing and even replacing active perception systems.

1.4 PROBLEM STATEMENT

The challenges and opportunities facing UGVs are enormous. The plentiful practical applications for UGVs are exciting. The progress made in the last decade only adds to this feeling. This is especially true for the MLVT. As UGVs in general begin to shift from active perception systems, the MLVT is seeking to help advance this area of robotics.

1.4.1 THESIS

The focus of this Masters of Science thesis is to implement a three dimensional (3D) scene re-creation for applications on UGVs using a passive perception system. This thesis will implement a variety of component algorithms to provide the finished product. It will not focus on new algorithm development, but instead merge successful techniques to allow for practical UGV 3D scene reconstruction and sensor fusion. The process will begin with data collection from the broadband cameras. Stereo matching will be applied to this data. The resulting depth information will transformed to a 3D point cloud. This information will be fed to a six degree-of-freedom simultaneous localization and mapping (SLAM) function that can handle 3D translation and rotation about each of the three axes. The result of the SLAM processing is a construction of the world model. From this point, the original images will be overlaid onto the rendered 3D environment that models the natural world. The final result will be a 3D world map with broadband data contained in point clouds. It will also be expandable for new sources of information and classification data. The source data will be the images from perception rig that has two visible-wavelength (VIS) and near-infrared (NIR) cameras.

1.4.2 JUSTIFICATION

The Advanced Perception System, Figure 1-5, is comprised of four cameras that cover four bands of the electromagnetic spectrum. These bands range from 400 nm to 14.0 μm. The JAI AD 080-GE is a two CCD camera that uses a prism to separate the visual band (400-700 nm) and the NIR band (700-1000 nm). The Goodrich SU320HX 1.7RT is Short Wave Infrared (SWIR) camera; its spectral band is 0.9-1.7 μm. The final camera is a Xenix Gobi-384. This camera senses Long Wave Infrared (LWIR) in the 8.0-14.0 μm range. These cameras provide a broad coverage of the selected band. Each camera was chosen to address a specific region of interest.

Figure 1-5. Perception camera rig. The passive perception rig with 4 cameras with 6 sensors and all of the networking and power requirements included below.

The images from this system are used for the classification of objects of interest for a mobile UGV. These objects of interest are vegetation using the VIS and NIR images, water using the SWIR images, and humans using the LWIR images. The goal of previous research in the MLVT was to augment a LIDAR based ranging system by adding classification data. The next step in this research is to incorporate the ranging information to replace the LIDAR system. This can be accomplished in many ways. However, a useful step in the direction is placing the data collected into a world model. This can be accomplished by using triangulated meshes with overlaid source image sets to create a 3D world model. Classification information can also be placed in the world model. The resulting model will be useful for path planning and further classification.

1.5 THESIS OVERVIEW

Chapter 1 of the thesis covered the background of UGV research and, specifically, on passive perception. It set the foundation for this research project by looking at the history of UGVs in the MLVT and how perception systems were used in those applications. The concept of perception, both active and passive was introduced. This chapter also expounded on the precise problem statement for this thesis and the justification of the research.

The following chapter explores the current literature on the content areas needed for this work. It begins with the topic of perception and continues to review papers and books that deal with the types of research was needed for this thesis. The second section explores several types of stereo cameras and the methods used for achieving stereo matching. The next portion looks at the research that many authors have done on the process of registering sequential point clouds using simultaneous localization and mapping methods. It concludes with a review of common techniques for rendering the final 3D data.

Chapter 3 contains the methodology and theory behind the approach taken. It expounds on the mathematical and physical models needed to achieve the 3D scene reconstruction exemplified in this research. The methods explored here begin with stereo vision and stereo correspondence. It then details the math behind the SLAM algorithm that is used in this thesis. The final section describes the data and how it is rendered.

Chapter 4 translates the methodology in Chapter 3 to the experimental procedure in Chapter 5. Chapter 4 starts with the stereo equations and progresses to the SLAM registration. It specifies how each step applies to the overall research and sets forth the data flow for the thesis. The experimental objectives, setup and results are covered in

Chapter 5 takes the processes from Chapter 4 and describes the physical implementation as well as the experimental procedure. This chapter also shows how the data was collected as well as how the methods were implemented in software. The experimental results at each stage and for the several types of experiment can be clearly seen.

Finally, in Chapter 6, the conclusions drawn from the work are laid out. The areas identified for future work and improvement are also covered in this chapter.

CHAPTER 2

2 LITERATURE REVIEW

This chapter focuses on a review of the current literature on the topics set forward in the problem statement in Section 1.4.1. It is broken down into sections covering perception, stereo vision, 3D scene reconstruction, and 3D data display techniques.

2.1 PERCEPTION

The field of robotic perception is a large one. There are many possible and varied solutions for addressing robotic perception. As set forward above, the thrust of this thesis is the examination of passive perception on UGVs. There are countless vehicles that employ passive perception systems. The following are examples of how these sensors can be used in UGVs. The Argo UGV, developed by Carnegie Mellon for the PerceptOR program, employs LIDAR, a visible spectrum camera, and a thermal IR camera [28]. The fusion of this data was used for navigation and object detection. The sensor characteristics were also tested under adverse conditions such as smoke, rain, and dust. The LWIR camera proved to provide an important source of information under these conditions. The combination of these sensors allowed the UGV to successfully operate under a wider range of conditions than it would be able to just using active perception.

Another example of a passive perception system is the work done by JPL [29]. Their system used passive perception for terrain classification and obstacle detection. They used a combination of stereo cameras in the visual, MWIR, and LWIR spectrums. The data gathered from this perception system has been very successfully employed in terrain mapping and classification. The classification results, seen in Figure 2-1, show the ability of the JPL system to classify the terrain in an effective manner with the ability to measure longer ranges over tradition LIDAR solutions.

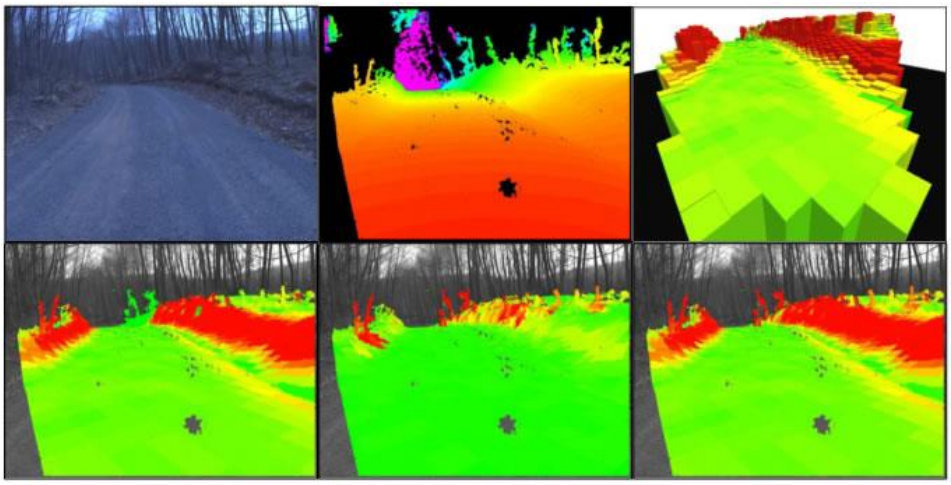

Figure 2-1. Mapping and Classification . An example of mapping and classification using a passive perception system and overlays onto the original images [29]**.**

Main features of the terrain are classified and can be used by the autonomy system. This classification data is also combined with the mapping data. The outcome is impressive. Drivable terrain, along with features of interest is clearly displayed. These examples, along with others, show the type of results that have been seen in UGVs that use passive perception.

2.2 STEREO VISION

To complete the transition from active LIDAR-based systems to passive perception systems, the ability to create a representation of the operating environment is essential. The goal of stereo vision is to provide an efficient solution for providing ranging information to the perception system. It can also be used for the detection of obstacles, both positive and negative as well as terrain classification [30]. In order to find an efficient solution for replacing the LIDAR ranging data with information gathered from EO sensors, it is necessary to examine the techniques that have been published previously. A careful evaluation of current sensors and sensor platforms will follow. From this foundational knowledge, an optimum vision ranging algorithm can be selected.

The topic of assigning a range to objects within an image is a well-developed field. There have been several different methods developed for ranging with visual sensors. Some of the most promising methods are described below. Researchers from the University of Parma have explored a multiple baseline stereo camera system [25]. The baseline for a stereo pair of cameras plays a significant role in its ability to accurately provide ranging information. The narrower baseline, the distance between the cameras used for stereo correspondence, provides better close range accuracy while the wider baseline works better at long range accuracy. This system, shown in Figure 2-2, allows the stereo ranging algorithm to use the best set of cameras for its current speed. The wider baseline pair is used at high speeds and the smaller baseline pair is used at lower vehicle speeds.

Figure 2-2. The TerraMax stereo camera setup features three cameras horizontally aligned. This provides a multiple baseline stereo camera rig [25]**.**

The goal of visual ranging systems is to provide an accurate distance measurement from the camera to the objects in the field of view. The most common way to provide this information is to calculate the disparity between every pixel in the stereo image pair. This task is very computationally and time intensive. When implementing this on a moving vehicle, the frame rate of this process must be higher than on a stationary platform. However, there is accuracy drop off if a computationally simple correlation method is used. So, there must be a compromise to find a balance between the computation time and the accuracy of the disparity values, [31].

Overlapping windows can be used to find the correlation between the stereo pair of images. Both Hirschnuller [31] and Rankin [29] used a method called SAD5. Sum of Absolute Differences (SAD) is a sliding window based cost function. SAD5 is constructed using five adjacent SAD scores to average the final resulting measurement. This is an example of a technique that provides a compromise between

computational time and accurate disparity matching. This method increases the accuracy of the measurement by averaging five surrounding measurements, but the computation time is kept reasonable by using mathematically simple methods for the construction of each measurement. In Figure 2-3, the comparison between a basic correlation method, seen in Figure 2-3 (b) and the SAD5 method can be seen Figure 2-3 (c). The SAD5 method provides a much smoother and more accurate representation of the scene that is pictured in the first frame (a). These window-based methods can be classified as block matching (BM) techniques. The techniques for finding a better disparity match shown below represent a cross section of the many varied approaches for finding the disparity between two images, which all have unique pros and cons.

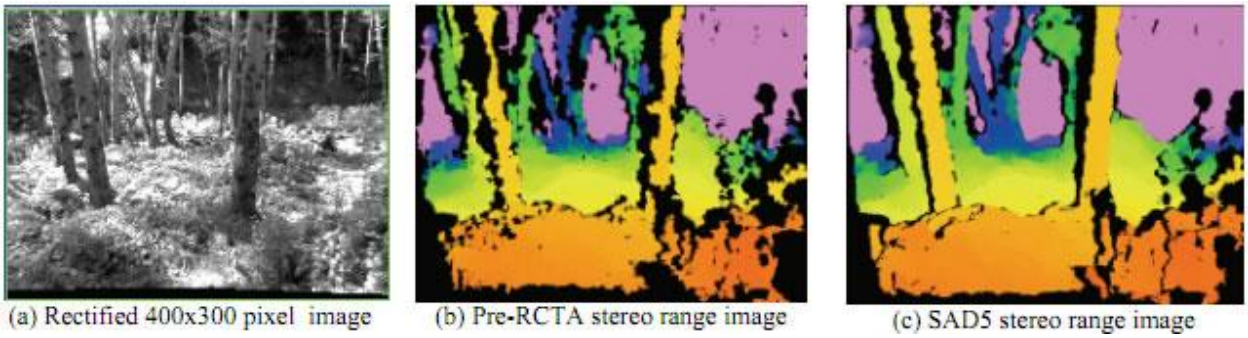

Figure 2-3. Stereo correlation techniques. (a) One of the stereo images that has been rectified. (b) a single pass SAD STEREO CORRELATION map. (C) the SAD5 stereo correlation provides a much smoother ranging map [29]**.**

OpenCV uses two types of types of stereo correspondence functions. The first is a graph cut method. It utilizes graph theory and an adjacency constraint to ensure more accurate matches between pixels on the left and right images captured from a stereo camera system in a time synchronous manner. An energy function is also applied to the matches to smooth out noise. The second method is a semi-global block matching algorithm [27], [32]. This method compares blocks of pixels using a cost function to find the minimum cost. The results of this algorithm can be seen in Figure 2-4. An energy function is then applied to the cost map to adjust the matches. The optimal disparity value is then assigned by a winner take all computation based on the result of the energy function. Post filtering can also be done to filter peaks and interpolate gaps in the results [27], [33].

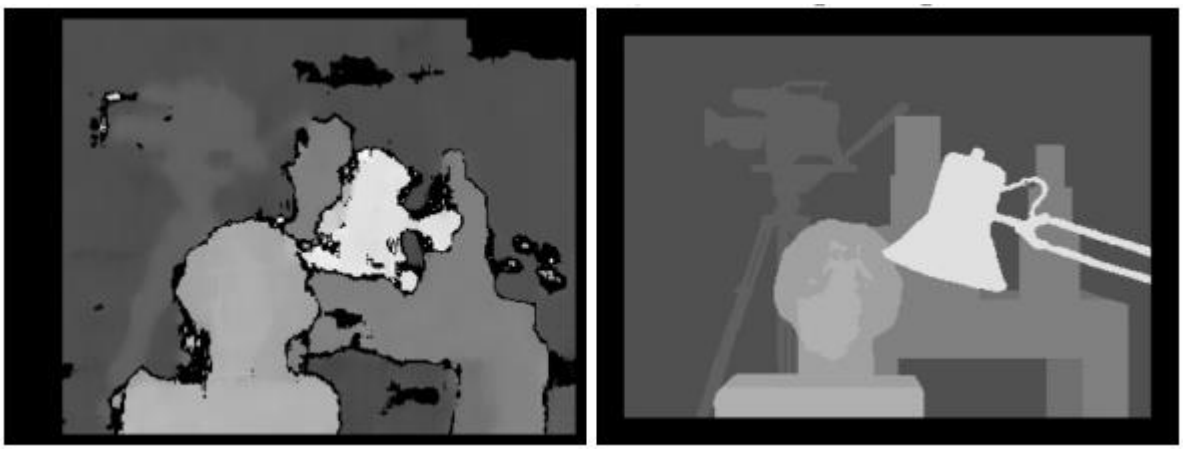

Figure 2-4. Blocking matching stereo algorithm. (a) Block matching stereo using the Tsukuba data set. (b) Ground Truth map [27]**.**

Another addition to stereo correspondence is addressed in van der Mark's paper [34]. The correlation values are found using a standard cost based correlation method, sum of squared differences. To further improve upon this technique, the authors used a parabolic approximation to estimate an equation for the resulting disparity function, shown in Figure 2-5. Since the goal of the correlation is the find the minimum cost, they were able to find this minimum to sub pixel accuracy. To further assist in the computation of the stereo correspondences, a circular buffer of the cost function was used to reduce the number of calculations required to find the complete disparity map.

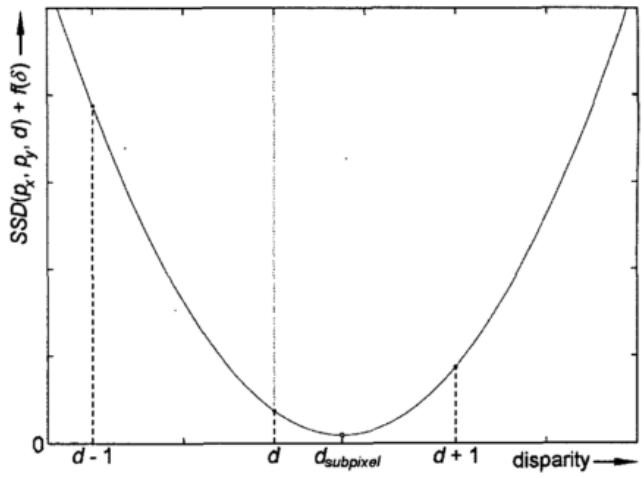

Figure 2-5. Subpixel accuracy of stereo disparity calculations. The pixels are aligned at d*k, but the minimization function can find the point in between the disparities where the ideal solution lies [34]**.**

Once the disparity values for a stereo image pair have been found accurately, the disparity values can be mapped to a physical distance using camera geometry. In addition to distance calculations, the disparity values can be used for several other purposes. Figure 2-6 introduces the concept of a v-disparity map [35]. The v-disparity map takes the disparity values in each row and places them on their own axis. The result is a graph that highlights obstacles as vertical lines and the ground plane as a diagonal line.

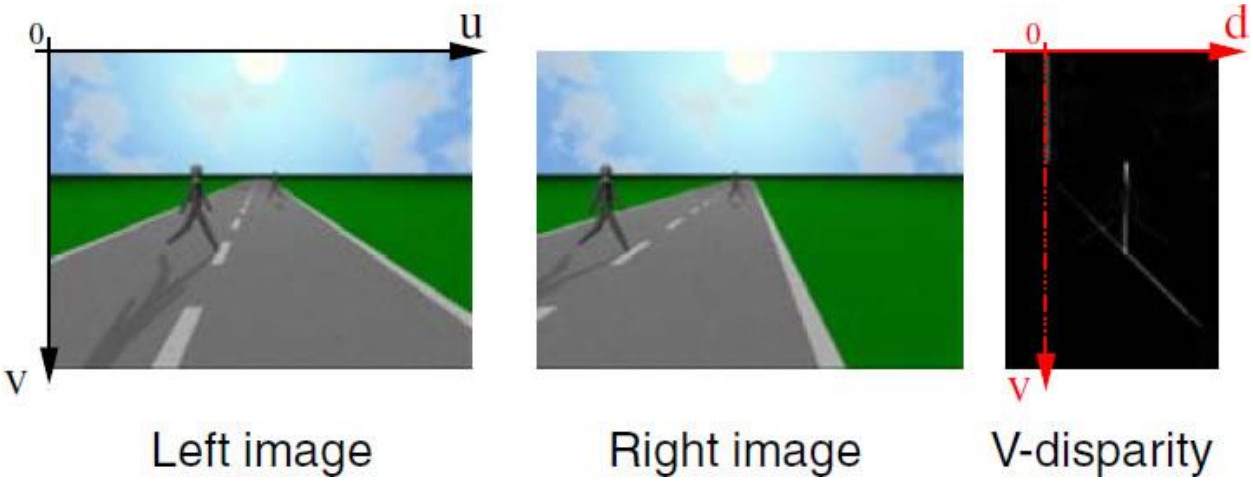

Figure 2-6. V-Disparity Map. A stereo image pair with the corresponding v-disparity map. this shows the ground plane and obstacles [35]**.**

The benefits of the v-disparity map have been further exemplified in Figure 2-7. The ground plane can be identified by the diagonal line in the v-disparity map. This is shown by the yellow line and bounded by the green lines. Another feature that comes out of this plot is the horizon line. This information can streamline the processes that follow after ranging, such as road and obstacle detection, as seen in Figure 2-7.

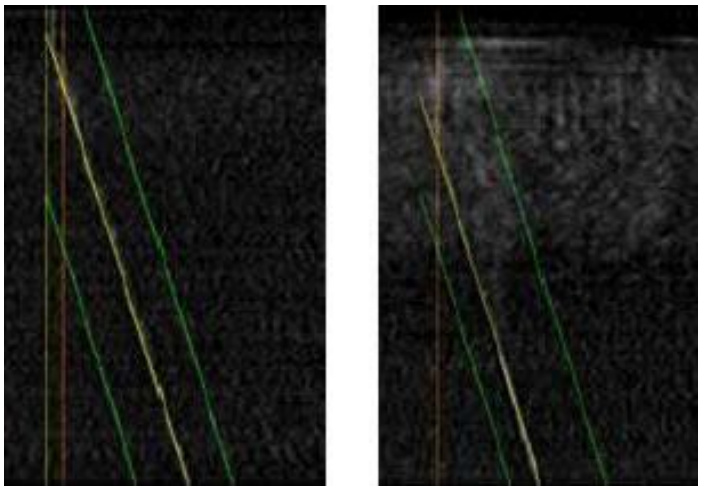

Figure 2-7. Ground Plane Identification. Examples of ground correlation lines and the bounds (yellow line and green lines respectively). The horizon is also identified by the red line [35]**.**

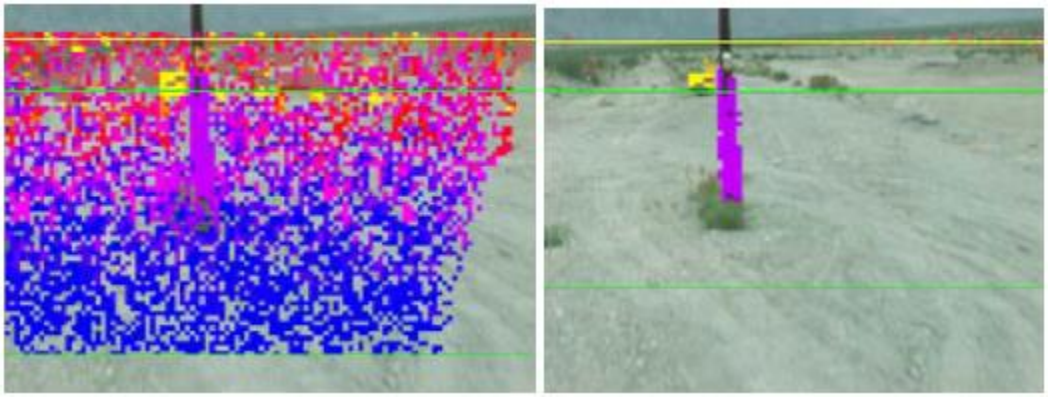

Figure 2-8. Disparity Space Image (DSI). The disparity values are assigned to each region of the image depending on the most similar region that is found on the same row of the other image. The obstacle map removes point from the ground plane and isolates potential obstacles [35]**.**

The ultimate goal of stereo ranging is to provide useful distance information. This information can be used to inform obstacle detection and identification as well as path planning. As described above, there are many different techniques for finding this distance information as well as many methods for extracting even more usable data from stereo ranging. The stereo data can then be used for registration and scene reconstruction.

Stereo vision sensors provide a comparable alternative to active perception systems. In his paper comparing active and passive ranging solutions from 2000, Hebert states that stereo systems, although lagging behind the active systems, show great promise [36]. Since that paper was written, the gap has been closed. Strecha et al. tested several different types of stereo vision systems as well as several

techniques for generating the 3D measurements and compared them to a LIDAR system. They found that the resulting stereo measurements had a mean relative error between 2.04cm and 5.03cm. With relative completeness percentages between 89% and 70% [37]. In satellite applications stereo vision systems have been shown to have much higher resolution than traditional active systems with 5cm resolution in the z-axis as compared to 30cm [38].

2.3 Reconstruction and Modeling Techniques

The next major portion of the thesis is to use the EO stereo data to reconstruct the 3D environment. This is done using SLAM to take successive point clouds and register them to each other. This creates a semi-global map of the robot's past and present path as well the portions of the world that have been observed by the sensors.

2.3.1 Simultaneous Localization and Mapping

SLAM is a process that, as its name suggests, creates a map of the sensed environment along with localizing the robot with in this environment. A great deal of research has been conducted on this topic [26], [39–43]. SLAM can be used in a wide variety of situations with an array of inputs. The general pattern of the algorithm uses range data over time to calculate the motion of the sensors and to stitch the sequential data sets into a larger model of the environment through which the robot has moved. This concept is further explored in Section 3.2. SLAM is a powerful tool that continues to be a point of research within the robotics community [41], [43].

2.3.1.1 Visual SLAM

Although many different types of sensors can be used as inputs for a SLAM algorithm, this thesis will focus on EO sensors as the sole source of data. Using the stereo vision approaches described above, these EO sensors can provide the depth measurements needed for SLAM. This approach is not a novel one, as it has been explored by several authors.

The use of stereo vision and optical flow to ensure more accurate SLAM results is an approach examined by Franke [44]. He and his colleagues used these two tools to provide a robust 6 DOF SLAM algorithm. This system was especially useful when dealing with dynamic objects. Dynamic objects are a common issue for most SLAM algorithms. By segmenting moving objects from stationary ones, this system was able to create a more accurate model and to reduce the possibility of directing a vehicle into a moving object. Several filters were also applied to the Kalman filter parameters in order to increase the speed of convergence. By using the optical flow to segment moving objects, the authors were able to improve the performance of their system. The maps that they created can be seen in Figure 2-9.

Figure 2-9. Rendered Visual SLAM. This shows the texturized point cloud that results from one implementation of vision based SLAM [45].

Another approach uses feature extraction on the images individually and then conducts stereo correlation on the feature level. This step reduces the amount of data and ensures that the points used in the map are significant. These values are input into the Sim et. al SLAM algorithm [46]. To further assist the SLAM algorithm in motion estimation this group used optical flow. The combination of these methods was proved to enhance the vehicle's environmental model and the estimate of its motion [46].

An issue with the two previous methods is their inability to handle large data sets. These two research groups addressed this issue by severe filtering of the data. This can be usable, but eliminates large portions of usable data. These methods were successful, but there are still many challenges to be mastered when using stereo vision for SLAM.

2.3.1.2 Feature Extraction

A key part of most SLAM algorithms is the ability of the robot to find, measure, and track features in its environment. Stereo vision returns a large amount of data points from each measurement since the pixel count in many of the cameras used is more than a million and the data from each pixel may have 8 bits of resolution or more. The data collected can quickly expand past the ability of the computer to process the data is a timeframe required for autonomous operation of UGVs. These measurements can also have significant amounts of noise and uncertainty. To ensure that the best features are used when executing the SLAM algorithm, it is common to utilize feature extraction algorithms.

Some the feature extraction methods that are used are Scale Invariant Feature Transformation (SIFT) [47] and Speeded Up Robust Features (SURF) [48]. By applying these algorithms to the stereo depth maps or the original images, a set of distinct features are extracted. This subset of data points is used for the SLAM process [45], [49]. Other methods for extracting more refined information from the data sets consist of segmentation, line extraction, and plane extraction, seen in Figure 2-10 [26]. All of these methods can be used to assist to filter the raw data before it is used in SLAM [50].

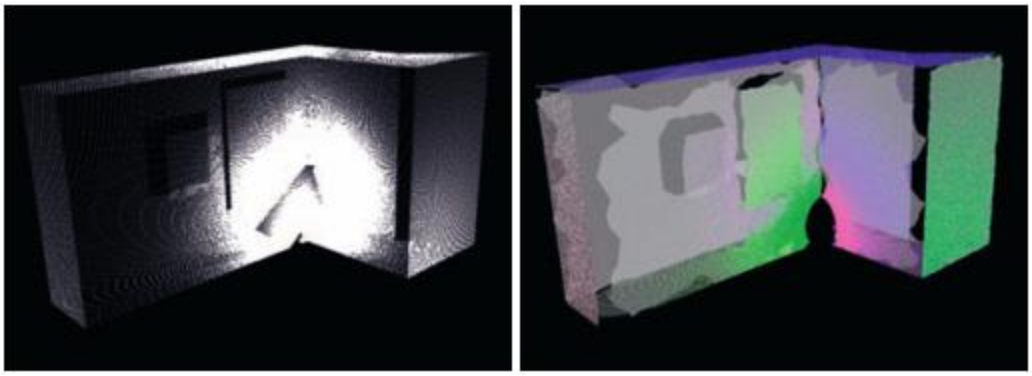

Figure 2-10. The results of SLAM with segmentation to identify planes. (A) The original point cloud scan. (b) the segmented map [26]**.**

2.3.1.3 Probabilistic Scan Matching

The scan matching technique is another one of the key features that is most commonly adjusted. There are a wide range of possible techniques. All of them are based on a recursive estimation framework that serves to ensure that the solution converges to the correct values.

Weingarten breaks the possibilities into four distinct categories [50]. The first is Kalman filters, both Extended Kalman Filter (EKF) [51] and Unscented Kalman Filter (UKF) [52]. They are one of the most popular methods and are used by many authors [49], [50]. Kalman filters use recursive estimation to converge to an estimate of the state vector, which contains the robot's pose and the observed features. It is a very efficient method and usually is able to achieve excellent results. However, its computation complexity increases drastically as the number of features increase. This can inhibit its viability when the scene is a large or complex environment.

The second method is Markov models [50]. This method represents the data as the probability distribution function (pdf) over all possible robot poses. This technique is fairly flexible, but it can only provide localization and the precision is limited to the cell size. Particle filters comprise the third variety of scan matching [50]. This is another recursive estimation technique and is a very efficient method that can handle larger data sets than the Kalman filter, however it is less precise. A type of particle filter in conjunction with visual odometry was used by Sim [46] with promising results can be seen in Figure 2-11.

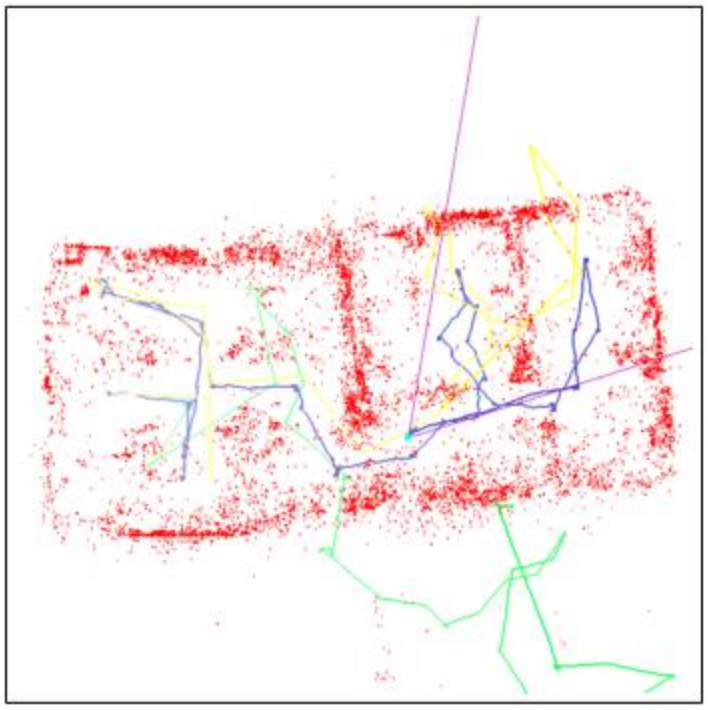

Figure 2-11. Particle Filter Slam: This shows the results of a SLAM technique that can correct position estimates from the odometry [46]**.**

The last process that Weingarten defines is direct scan matching. This is conducted using the raw 3D data from the sensors as the input to the SLAM function. This is typically performed using an Iterative Closest Point (ICP) [53] or Random Sample Consensus (RANSAC) [54] technique [50]. Scan matching removes the feature extraction steps, which can reduce the number of calculations, needed to preform SLAM. The down side to this is that the number of data points can be very large. Another side effect is that the map is not directly globally consistent.

There are other methods that Weingarten does not address. Sáez suggests an entropy minimization method for a Six Degree of Freedom (6DOF) SLAM algorithm [42]. This method uses visual odometry and dense feature extraction to feed into an entropy minimization function. The authors also use trajectory compression and local maps to eliminate data overload. This is offset by using entropy minimization to best align the local maps with the global coordinates. The impressive results can be seen in Figure 2-12.

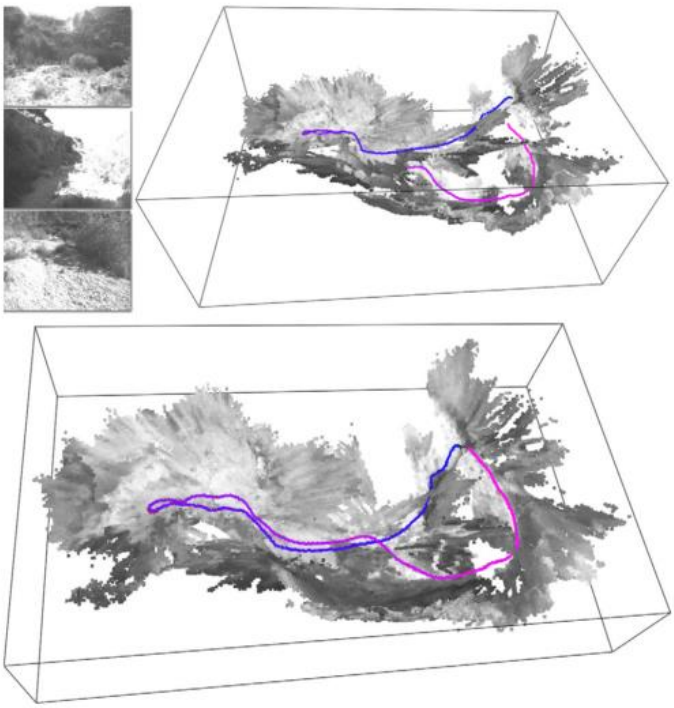

Figure 2-12. 6DOF Unstructured SLAM. (A) Source images. (B) Initial scan matching. (C) Final scan matching with no egomotion assumptions [42]**.**

2.4 DISPLAY TECHNIQUES

The ability to display data in an easily consumable format has been topic of research for many years [29], [55–58]. The data is returned from the SLAM process is semi-global point clouds constructed from the original filtered data. There are several considerations when displaying this format of data. All of the data is not needed to create an adequate representation of the environment. It can be compressed, filtered, or consolidated. Since the data is intended for machine consumption, the visualization of the data is not paramount, but it can be useful. The final goal of this process is the represent the observed data in an accurate and efficient manner.

One of the most powerful methods for visualizing a 3D world model, especially for consumption by UGV autonomy algorithms is to overlay the source images onto the point clouds. The method is called texture projection and can be an effective way of conveying a more realistic scene to the user, while maintaining source data.

2.4.1 VOXEL GRID, OCTREES, KD-TREES, AND TRIANGULATED MESHES

Four of the common block based methods for 3D display are voxels, octrees, k-d trees, and triangulated meshes. Voxels, volumetric picture elements, are 3D pixels. Voxels can be a powerful rendering technique since they extrapolate on a universal standard of 2D imaging. Each voxel is assigned a value and a size to populate the map based on the raw data points that fill into its volume. The continuous 3D world is discretized into cubes and rendered into a simple block. This method can greatly reduce the complexity and serve to compress the maps generated by the SLAM techniques. Rankin uses a voxel based map to render the results from the data that their robot generates [29]. The results of this method are shown in Figure 2-13. They have also been used in many computer games [59].

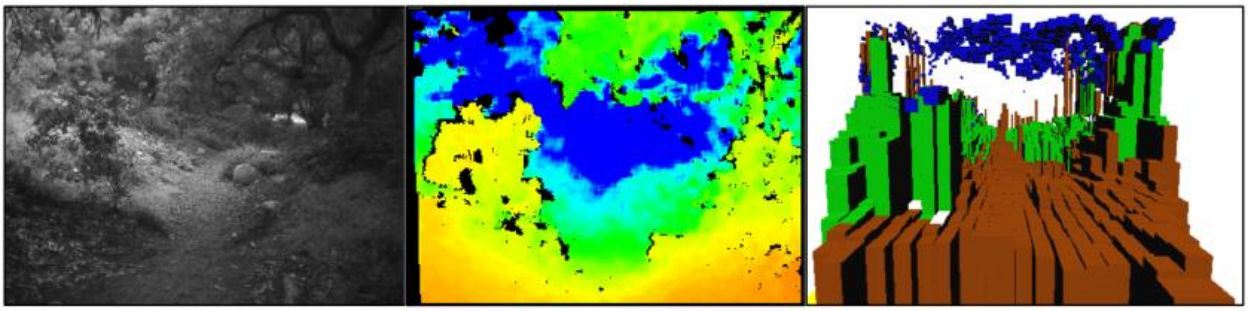

Figure 2-13. Block based rendering. (A) Source Image. (B) Stereo Depth Map. (C) Classification and voxel, 3D block, based rendering [29]**.**

Another method for 3D block based rendering is the octrees [58]. This is a 3D expansion on a quad tree that falls into a tree based structure; it can be seen in Figure 2-14. It is a much more efficient in its compression of the raw data than the voxel method. It breaks up the 3D space into eight cubes and continues with this process until a minimum threshold of data points lies in each subdivided cube.

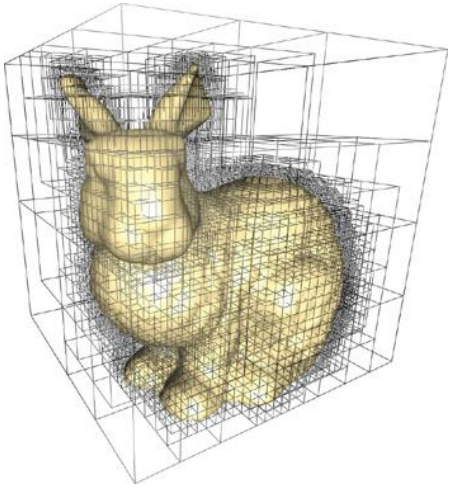

Figure 2-14. Octree rendering. Solid subdivision using the octree method to reduce the number of cells while maintaining high accuracy [58]**.**

As Figure 2-14 clearly shows the constant spaces in the image are contained in large cubes and the points that lie around the edges of the object are in small cubes. This is an effective way to render 3D objects in a block based form as it is limitable and allows for easy memory allocation. [58].The final method commonly used for spatial representation is the kd-tree method [60]. It is very similar to the octrees method, but is more flexible. It can be split along as many lines as necessary, rather than just eight. It also splits to keep points along edges rather than inside cubes. Figure 2-15 shows a 2D data set that is partially rendered with the kd-tree method. The compression and simplification of the data set can be clearly seen. Both the octrees and kd-tree methods are implemented in the Point Cloud Library (PCL) [61] and are the more commonly implemented methods because of their added compression and flexibility.

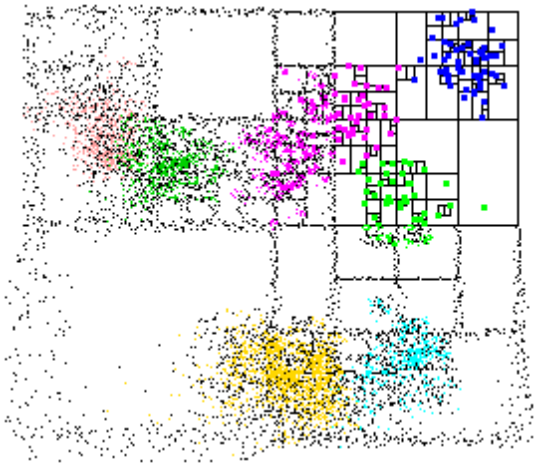

Figure 2-15. Approximate Nearest Neighbor Search. This figure shows the ability of this approach to find the key points while reducing the number of cells [62]**.**

The display of point clouds can be solved with the previous methods or by using triangulated meshes. This is most typically accomplished through the connection of the points to form triangles or other polygons. Examples of this method can been seen in Figure 2-16 [55]. The ability of this method to facilitate the easy projection of the source images into the mesh. This feature is a powerful tool for the rending of a realistic 3D environment. There are many different methods by which triangular meshes can be generated; however, several methods stand out. The one of these is the fast triangulation of unordered points and the second is the greedy projection triangulation [56], [63–65]. The greedy method is used by PCL.

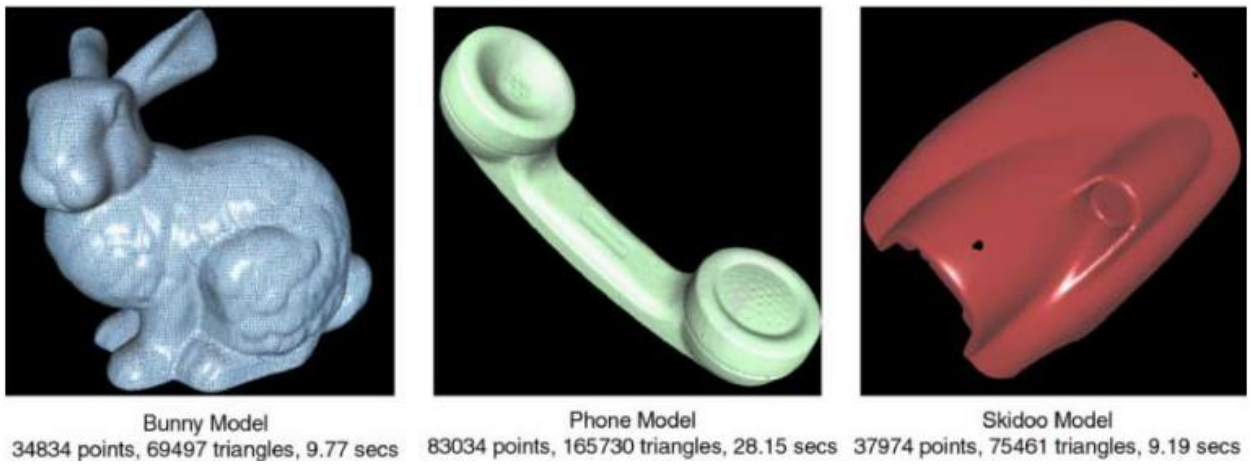

Figure 2-16. Triangle Mesh Rendering. (A) Bunny surface model. (B) Phone triangulated mesh. (C) Skidoo model with 75k triangles [63]**.**

2.4.2 Colorized Point Clouds

The display of point clouds can be solved with the previous methods or the point clouds themselves can be made viewable. This method allows the user to see what the robot sees. This process can also be described as data fusion. It begins by fusing the image color data to the point clouds data [66]. This results in a data set that contains several sources and types of information. This data can be viewed as a

simple point cloud or with the color information as well, similar to Figure 2-17. This type of data can be very useful to classification and autonomy algorithms for the application on UGVs.

Figure 2-17. Point Cloud Colorization. This dense point cloud has been fused with color images to allow for the visualization of the 3D data concurrently with the color data as well [66]**.**

The work shown in the previous sections encompasses the content matter set forward in Section 1.4.1. It begins with the topic of passive perception and progresses to cover the topics that are key to the thesis statement. The direction and topics of current research for each section was explored. This literature review served to inform and direct the research conducted in this thesis.

CHAPTER 3

3 METHODS AND FUNDAMENTAL SCIENCE

After having explored the work of many excellent researchers in the area of passive perception and world modeling, this chapter breaks down the thesis statement (1.4.1). It dissects each step need to accomplish this task and explains the fundamental principles and mathematics behind each. The methodology used in this thesis can be broken down into three sections. The first section is stereo vision. It begins with the model of a camera and follows each step need to achieve successful point cloud generation. The second section covers the task of creating the three dimensional world model using simultaneous localization and mapping. Finally, the model and original data are recombined using rendering and reconstruction methods which are explained below.

3.1 STEREO VISION

The first step in building a passive perception, the EO-based 3D scene reconstruction system is the stereo vision system. Stereo vision is the key to creating a passive 3D scene re-creation system. It serves to replace the LIDAR in providing dense depth measurements. The 3D environments are created using pairs of time correspondent images obtained from the stereo vision systems. The source images must first be rectified to account for lens distortions and other camera parameters using the calculated calibration values. The rectified images are then correlated on a pixel level to find the coincident regions of the images. From this process, depth can be calculated. This results in a 3D point cloud for each pair of images taken with the cameras.

3.1.1 CAMERA GEOMETRY

To start the process of gathering stereo image data, the sensors must be modeled. This model includes the basic functionality of the camera and contains adjustments for both intrinsic and extrinsic camera properties. Stereo vision systems rely on at least two images captured from different perspectives. This is typically done using two cameras taking pictures at the same time.

3.1.1.1 Projective Geometry

Euclidean geometry is the standard form of mathematics used in most 3D applications, such as modeling lines, points, and planes. However, there are several drawbacks with Euclidean geometry that cause it to be discarded. The first issue is that points at infinity are treated as a special case and are not easily modeled. The second reason is that when projecting a 3D point onto the image plane of a camera a perspective scaling is required. Thus, a projective geometry is typically used to model the image creation process and to reconstruct 3D space from multiple images [67].

The benefits of projective geometry are that it removes the need to do perspective scaling for each point by including a free scaling parameter and that it also resolves the infinity point issue. Points in Euclidean space are referenced as a vector, $(X, Y, Z)^T$, while projective geometry uses a 4 element vector, $(X_1, X_2, X_3, X_4)^T$. These are known as inhomogeneous coordinates and homogeneous coordinates.

The transformation between these two coordinate systems can be seen in Equations (3.1.) and (3.2.)

$$X = \frac{X_1}{X_4}, \qquad Y = \frac{X_2}{X_4}, \qquad Z = \frac{X_3}{X_4}, \qquad where\ X_4 \neq 0 \tag{3.1.}$$

More generally,

$$(X_1, X_2, \dots X_n)^T = \lambda(X_1, X_2, \dots X_n)^T, \qquad where\ \lambda \neq 0 \tag{3.2.}$$

λ is commonly called the free or homogeneous scaling factor. Using perspective geometry makes the most sense when dealing with images. Cameras capture light rays, which are most simply described by perspective geometry. The projective camera model also allows the most flexibility in transforming the images into 3D space [68].

3.1.1.2 Pinhole Camera Model

Once a coordinate geometry has been established, the next step in defining the system model is to pick a model for the sensors themselves. The goal of this model is to define the relationship between objects in the 3D environment to the 2D image coordinate frame. The most common and most logical model of a camera for this application is the pinhole camera model. This model is a simplification of most physical cameras; however, it is robust and contains parameters to adequately describe their functionality. As shown in Figure 3-1, a pinhole camera is a box with a hole in it. The reflected light from the object travels through the small hole, which is called the optical center. The image is inverted on to the sensor or image plane, shown as the back of the box.

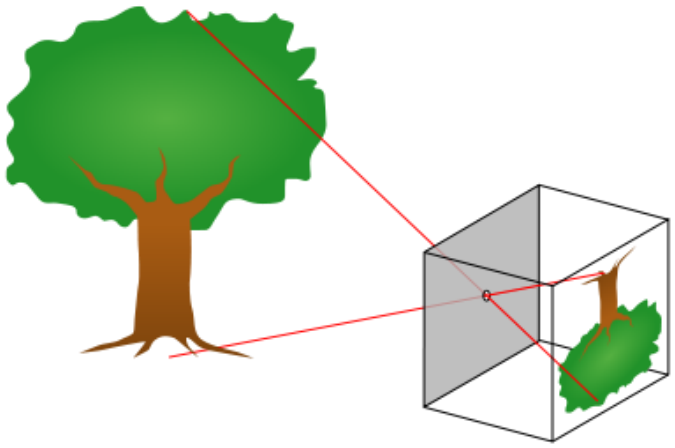

Figure 3-1. Pinhole Camera Model. The figure shows the simple pinhole camera model that can be used to define most cameras [69].

The distance from the pinhole to the image plane is the focal length. For ease of use, the pinhole camera is more commonly modeled with a virtual image plane in front of the optical center. The image plane is same distance in front of the optical center as the image plane was behind it. This preserves the same focal length and removes the coordinate inversion necessary in the previous model.

The camera model shown thus far is shown in Equation (3.3.); it is expressed in matrix notation in Equation (3.4.)

$$u = \frac{Xf}{Z}, \qquad v = \frac{Yf}{Z}, \qquad where\ f = focal\ length \tag{3.3.}$$

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \qquad where\ Z = \lambda \tag{3.4.}$$

All cameras define origin of the image as the top left pixel in the image instead of the principal point, the intersection of the Z axis with the image plane, as shown in Figure 3-2.

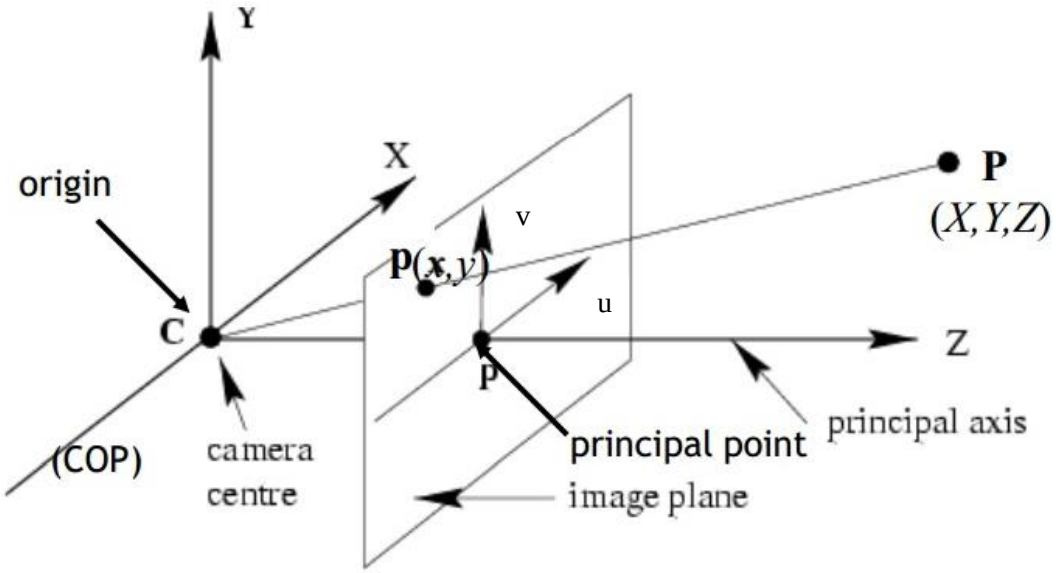

Figure 3-2. Detailed pinhole camera model. This model provides more detail on the geometry and key features of a pinhole camera model; based on the image from [70].

However, as the model shown in Figure 3-2 notes, the zero point should be the principal point. To account for this, the model can be easily adjusted defining the principal point as $(o_X, o_Y)^T$.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & o_X & 0 \\ 0 & f & o_Y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \qquad where\ Z = \lambda \tag{3.5.}$$

Furthermore, this model assumes that the physical pixels in the camera are square, with a 1:1 aspect ratio. Many cameras utilize non-square pixels, which has the effect of creating different focal lengths on both the x and y axis. Another physical parameter that must be taken into accounted is pixel skew (τ), which is when the pixels are in the shape of parallelograms instead of being rectangles. Both of these effects can be visualized in Figure 3-3.

Figure 3-3. Illustration of Skew and Aspect Ratio. (A) Shows a pixel that has no skew and has a 1:1 aspect ratio. (B) Shows a pixel that has a large amount of skew and does not have a 1:1 aspect ratio. Meaning the interior angles are not 90° and the sides are of unequal length.

$$\lambda\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}f_x & \tau & o_X & 0\\0 & f_y & o_Y & 0\\0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}=[\boldsymbol{A}\quad \mathbf{0_3}]\boldsymbol{P},\qquad where\ P=\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}^T \tag{3.6.}$$

Equation (3.6.) makes the final simplification in the basic pinhole camera model. The 3x4 matrix K is known as the intrinsic parameters matrix. It contains all of the model information to describe the functionality of the camera in the projective geometry.

The final aspect of defining the camera model is the radial and tangential distortions. This distortion is a 5-vector, **κ** (3.7.). Radial distortion is a product of the camera lens and is labeled thus because it causes straight lines to become curved. It is typically more noticeable at the edges and corners of the image. The method for modeling radial distortion is known as the inverse radial distortion model because it takes a distorted image and removes the distortion.

$$\boldsymbol{\kappa}=[\kappa_1\quad \kappa_2\quad p_1\quad p_2\quad \kappa_5] \tag{3.7.}$$

Where κ_1 and κ_2 are the radial distortion coefficients and p_1 and p_2 are the tangential coefficients. κ_5 is always assumed to be equal to zero. The radial distortion is generally derived from the camera lens and can be described the ellipse shown in the next equation. The tangential distortion is derived from the imperfections in the mounting of the sensor plane. The tangential distortion parameters will be higher if the sensor is further out of a perpendicular alignment to the center line of the camera.

To model the radial distortion, we begin with the following equation that describes the radial distortion.

$$\begin{aligned}x_{corrected}&=x(1+\kappa_1r^2+\kappa_2r^4+\kappa_5r^6)\\y_{corrected}&=y(1+\kappa_1r^2+\kappa_2r^4+\kappa_5r^6)\end{aligned} \tag{3.8.}$$

The tangential distortion is modeled at shown in Equation (3.9.). These two sets of equations are applied to the images to correct for both the radial and tangential distortion.

$$\begin{aligned}x_{corrected}&=x+(2p_1xy+p_2(r^2+2x^2))\\y_{corrected}&=y+(p_1(r^2+2y^2)+2p_2xy)\end{aligned} \tag{3.9.}$$

The derivation for the radius used in both sets of distortion equations can be seen in Equation (3.10.).

$$r_{d1} = \sqrt[3]{\frac{-q+\sqrt{\Delta}}{2}} + \sqrt[3]{\frac{-q-\sqrt{\Delta}}{2}}, \qquad where\ \Delta > 0$$
$$p = \frac{1}{k_1}, \qquad q = -\frac{r_u}{k_1}, \qquad \Delta = q^2 + \frac{4}{27}p^3 \tag{3.10.}$$

While radial distortion is an intrinsic camera property, it is not included in the intrinsic properties matrix, A. It is dealt with during the rectification process on an image level.

The practical definition of the principal point, skew, aspect ratio, and radial distortion will be covered in Section 3.1.2 since each of these values are unique to each camera and must be found empirically.

The other aspect of defining a stereo camera model is the extrinsic camera parameters. As the intrinsic parameters define each specific camera, the extrinsic parameters define their relationship to each other and to other reference frames of interest. This is a topic of interest because in order to localize any element seen by the sensors to the camera, robot, or world reference frame the pose of the cameras must be known. Pose is described as the position and orientation of the camera with respect to a specified coordinate system.

The extrinsic parameters can be defined as the sum of the rotation matrix and the translation vector. The final rotation matrix is the product of rotation matrices about each axis (Equation (3.11.))

$$R = \begin{bmatrix} c\theta & s\theta & 0 \\ -s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} c\phi & 0 & s\phi \\ 0 & 1 & 0 \\ -s\phi & 0 & c\phi \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & c\psi & s\psi \\ 0 & -s\psi & c\psi \end{bmatrix}$$
$$T = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.11.}$$

where θ, φ, and ψ are rotation angles about the X, Y, and Z axis, respectively. To completely describe the pose of the camera in space, the translation of the camera is needed. It can be simply expressed as vector from the world origin to the camera's optical center. Thus, the extrinsic parameters or essential matrix can be defined as follows.

$$E = [R \quad T] \tag{3.12.}$$

The result of this model is the ability to reference any point in 3D space to a camera pixel. The baseline the term that refers to the distance between the optical centers of each camera. It is required for inferring depth from the stereo correspondence [67] [71]. For a stereo camera system with the cameras horizontal to each other, this distance is the X value in the translation vector above.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = [A][E]\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.13.}$$

Equation (3.13.) is the final single camera model. It incorporates both the camera's intrinsic and extrinsic properties.

3.1.1.3 Stereo Camera Geometry

The next step in defining the stereo camera model is to relate the two cameras used for stereo vision. Each camera must necessarily view the environment from a different perspective, since they will not share the same optics. Using a common point space seen by the two images, the corresponding pixels are found on each image plane using the camera model derived in Section 3.1.1.2. The 3D position of the point P is space is calculated using triangulation (3.1.4) and the camera geometry. Stereo camera geometry is referred to as epipolar geometry. As Figure 3-4 shows, a 3D point (P) is connected to each camera's optical center. The pixel points for this object are the points at which each ray intersects its camera's image plane. This is calculated using the camera model. All of these points lie on a common plane. The epipolar plane, π, is the plane that is defined by the two camera centers and the point of interest in space. This plane defines an important feature of stereo vision. The epipolar plane specifies the constraint when searching the images for corresponding points.

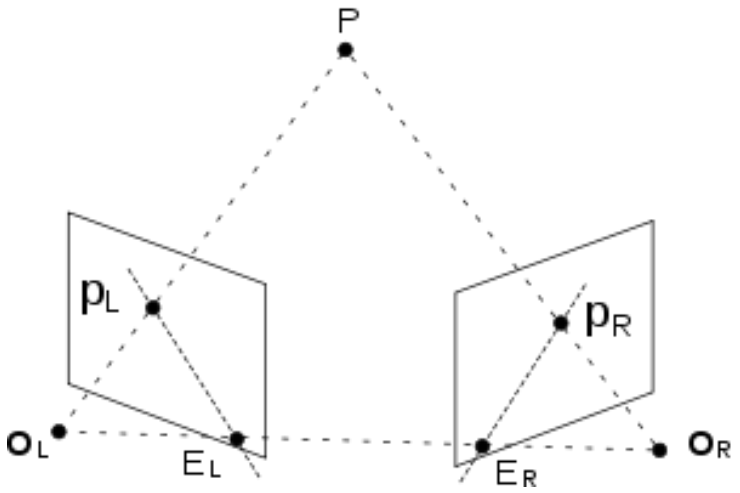

Figure 3-4. Epipolar Geometry. This figure shows the epipolar plane created by the camera centers and the point P in space. It also shows the camera planes and the epipolar lines and the baseline [72]**.**

Another feature defined by the epipolar plane is the epipolar line. This line is created by the intersection of the epipolar plane and the image plane. Due to the physical properties of cameras, it is very beneficial to ensure that the epipolar lines are parallel to the camera's pixel rows or columns. By guaranteeing that this is true, the computational complexity of finding the pixel correspondences is greatly reduced, since the correspondence must only be done along the epipolar lines instead of in two dimensions. This constraint is defined as the epipolar constraint.

3.1.2 CALIBRATION AND RECTIFICATION

The process of gathering the intrinsic and extrinsic parameters is called camera calibration. Once the cameras have been quantitatively measured, the images that are collected from that point on can be rectified. The rectification process removes sensor and lens distortions in the image and ensures that the epipolar constraint is met.

3.1.2.1 Calibration

Camera calibration only needs to be conducted once in the camera's lifetime as long as the lens and other camera peripherals remain the same. The return of the stereo camera calibration process is the intrinsic parameters for each camera, the distortion coefficients for each camera, and the extrinsic parameters that

relate the stereo pair of camera to each other. These matrices are directly used to provide image rectification.

The methods cataloged and set forward by Bouguet have become common practice for camera calibration [71]. It employs a calibration checkerboard. The checkerboard is used because it can be easily measured, made, and provides distinguishable features. The process involves taking a series of pictures in which the calibration board is fully visible in both the left and right images. In order to collect an adequate sample for calibration, more than fifteen image pairs must be taken. By moving the checkerboard around the whole space of the image and by varying its orientation, a more accurate model of the camera's intrinsic, extrinsic, and distortion parameters can be calculated. Some examples of calibration images can be seen in Figure 3-5.

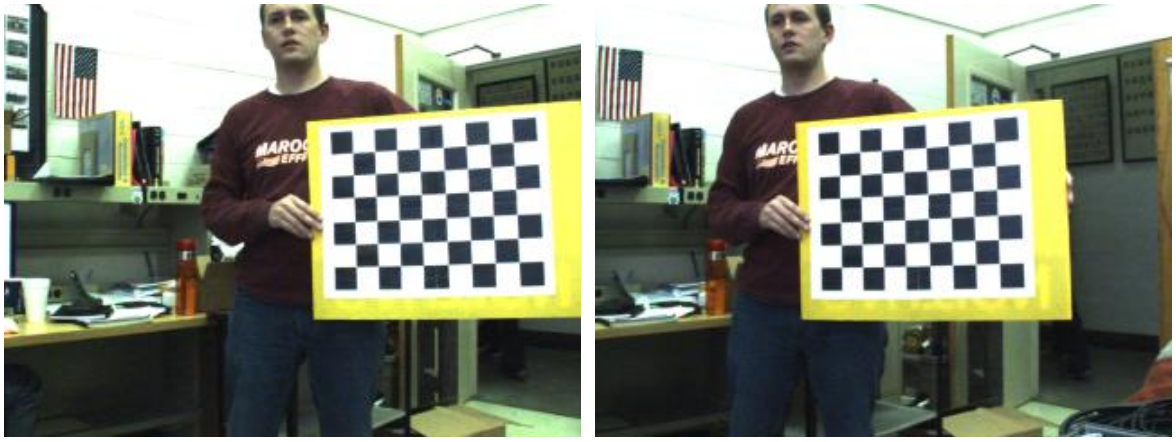

Figure 3-5. Sample Calibration Images. (Left) This is the image taken from the left camera of a calibration grid. (Right) This is the image taken from the right camera of the same calibration grid. The difference in perspective is clearly visible.

Once the calibration images have been captured, the calibration function must identify all of the checkerboard corners. The points generated by identifying each of the internal corners of the checkerboard, provide a data set by which the calibration can characterize the distortion and other parameters. The points are found to a subpixel accuracy of about 0.1 pixels. This process iteratively adjusts its camera parameter estimations. This algorithm uses the projective pinhole camera model with the position of the corner points to calculate the camera parameters. All of the information needed to model the cameras is from this process.

3.1.2.2 Rectification

Image rectification can be implemented once the calibration process has been completed. The images are rectified by using the projective pinhole camera model described in Section 3.1.1. The parameters found in the calibration process are feed into the model along with any new image collected by the cameras. The primary purpose of the rectification process is to remove the lens distortion. The resulting images have been rectified and now adhere to the epipolar constraint. All of the parameters and data required for inferring depth from stereo have been compiled.

3.1.3 STEREO CORRESPONDENCE

The concept of stereo vision is a familiar one. Humans use this technique to assess depth with our two eyes. Depth is inferred from two images of the same scene by finding the corresponding points in space

and measuring the difference in location on the images. This difference is called disparity and the process of finding the best matches by which to calculate the disparity is stereo correspondence. Due to the epipolar constraint these searches for correspondence can be simplified to one dimension.

3.1.3.1 Prefiltering

As with the other steps in this methodology, some prefiltering is helpful to ensure the best results from the primary stage of the process [32]. The main approaches used in this step are to normalize the image brightness and to enhance the image texture. The image normalization helps to reduce the effects of differences in lighting and exposure between the two cameras. Normalization spreads out the image intensities to use the full range of possible values.

$$I_N = \frac{I_U \cdot I_{high}}{I_{max}} \tag{3.14.}$$

where I_N is the normalized image, I_{high} is the maximum possible intensity value, and I_{max} is the current maximum value in the image. The next step in prefiltering is to enhance the image texture. This can be done by using a Laplacian of Gaussian filter or a normalized cap filter [32]. Both of the filters use convolving windows to apply the filter across the image. Equation (3.15.) is an example of the normalized cap filter.

$$I_c = \min\left[\max\left[I_c - \bar{I}, -I_{cap}\right], I_{cap}\right] \tag{3.15.}$$

where I_c is the center pixel in the window, $\bar{I}$ is the average intensity value in the window, and I_{cap} is a predefined positive numeric limit. Once the images have been prefiltered, they are ready to undergo stereo correspondence.

3.1.3.2 Correspondence

The correspondence problem is the key function in using stereo for depth measurement. There are many possible methods that have been developed, some of which were discussed in Section 2.2. However, the method used in this thesis falls into the block matching category. With block matching, correspondence is found using a sliding window. One image is chosen as a reference and the other is labeled as the target image. A window centered around a pixel on the reference image is selected. This window is compared to a series of windows along the epipolar lines in the target image using a cost function. The window and, thus, the central pixel on the target image that corresponds to the lowest cost is selected as the best match. The difference in pixel location along the epipolar line is defined as the disparity, d. Figure 3-6 shows an example of the stereo windowing function and how that results in the disparity calculation.

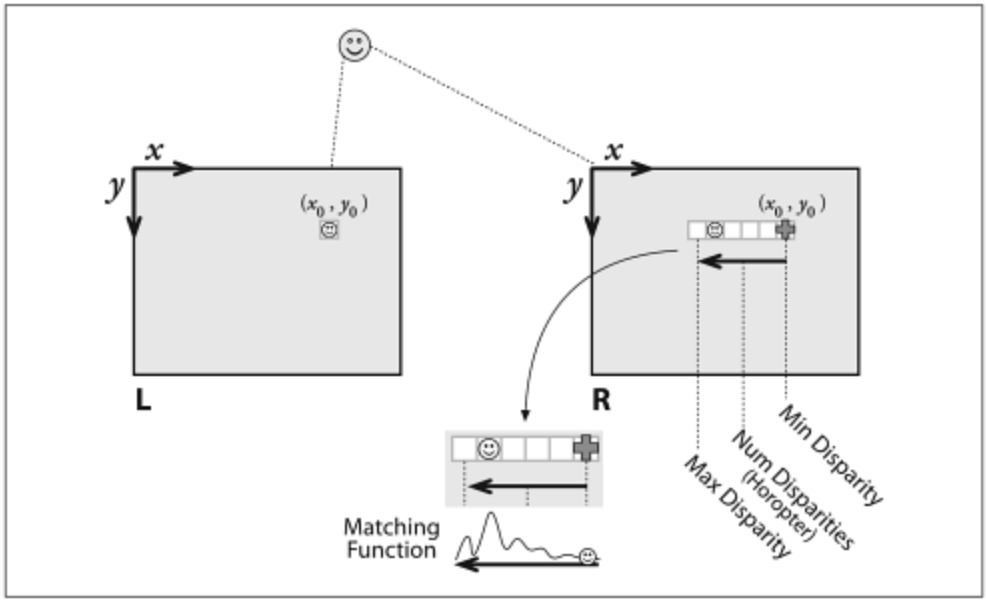

Figure 3-6. Stereo Windowing Function. This figure shows how the windowing process is conducted. A target pixel on the left image is taken as the starting pixel on the right which is scanned along the epipolar line to minimize the cost or matching function [32]**.**

$$d = x_R - x_T \tag{3.16.}$$

where d is the disparity, x_R is the reference pixel location, and x_T is the target pixel location. Equation (3.16.) is used in combination with the camera model from Section 3.1.1. The geometry behind the inference of depth can be seen in Figure 3-7; the green bar remains the same length and the difference in perspective is shown by the red bar. The relationship between the size of the disparity, difference between the two pictures, and depth can be seen in Figure 3-8.

Figure 3-7. Disparity Illustration. The concept of disparity can be seen in the set of images. The green bar is the same length in both images. The disparity in perspective is shown by the red bar in the right image. The width of the disparity is inversely proportional to the distance from the cameras.

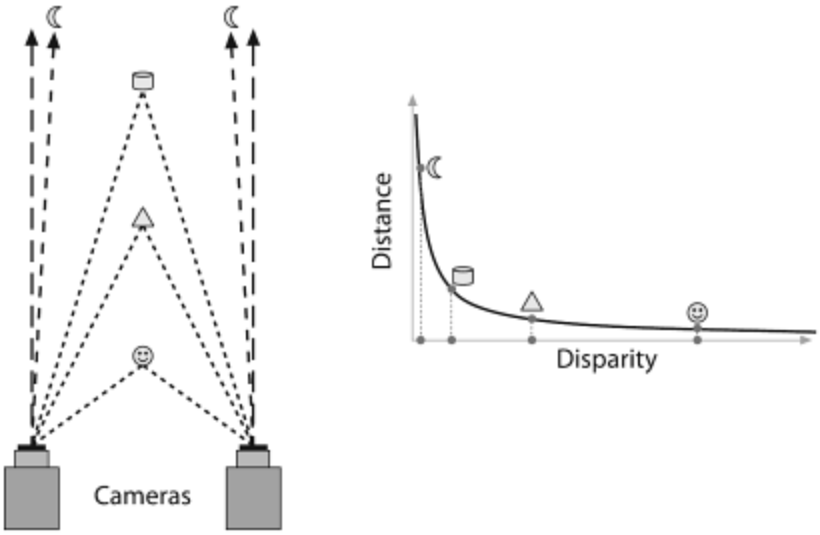

Figure 3-8. Disparity to Depth Correlation. The relationship of depth and disparity, seen in the previous figure, are further illustrated in these two illustrations [32]**.**

The cost function that is used is a SAD window [32]. It takes windows of defined size centered a specified pixel and takes the absolute difference of the sum of the pixels in the window. As the windows in the reference and target images are more similar, the cost of choosing that value decreases. The objective of the cost function is to sort through the possible solutions and find the optimal one, see Figure 3-9.

$$SAD = \sum_{(u,v)\epsilon W} |I_R(u,v) - I_T(u,v+i)| \tag{3.17.}$$

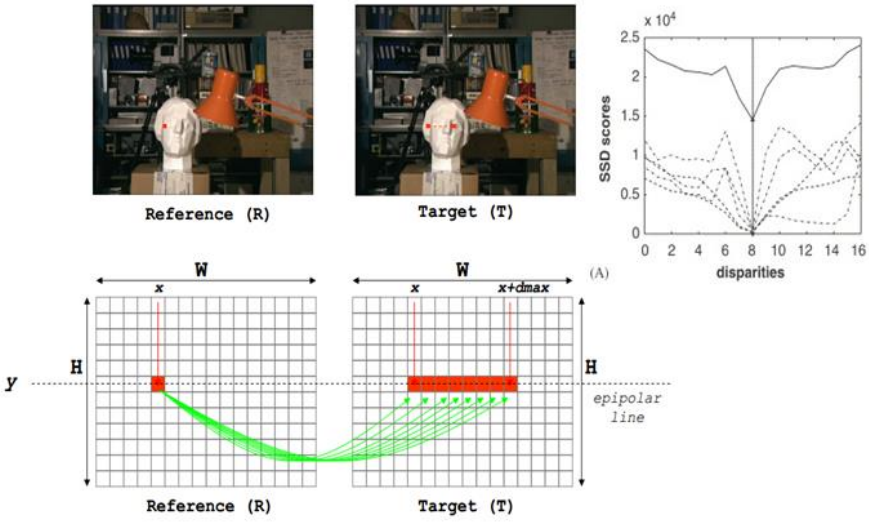

Figure 3-9. Stereo Windowing. This set of images shows the process of stereo windowing and the minimization of the cost function to find the ideal match. (A,B) Source stereo image pair. (C) The graph of a cost function over the scan width. (D,E) The reference pixel and the scanning of the target image along the epipolar line to find the ideal match [73]**.**

There are several parameters by which the stereo correspondence function can be tuned. The first are the minimum and number of disparity values. This allows the search area in the target image to be minimized and to best fit the cameras and scenes. The number of disparities defines the horopter, which is the 3D volume that the stereo vision system is able to measure. Another parameter is the window size. Larger windows are less susceptive to noise, but can suppress smaller features. The window size must be optimized to provide the most reliable matches.

There are other issues that stereo correspondence must overcome. There is the possibility of multiple matches, uniform regions of the image and repetitive patterns in the image. All of the issues are dealt with by the cost function in the post filtering process. However, the topic of occlusion is one that must be addressed. Occlusion occurs when the ordering constraint is violated. This happens predominantly at the edges of objects where the difference in perspective between the two cameras is more noticeable, such as seen in Figure 3-10. This effect is addressed by the postfiltering process.

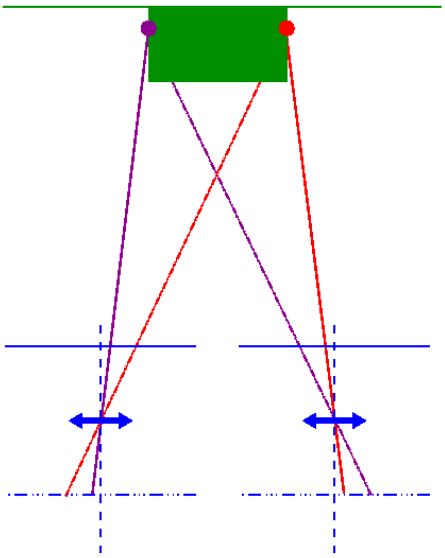

Figure 3-10. Stereo Occlusion. This shows the potential situation where occlusion would occur. This happens when one camera can see things in the scene that the other cannot. It tends to happen at depth discontinuities [74].

3.1.3.3 Postfiltering

Once the stereo disparity map has been calculated, the results are postfiltered. Postfiltering, as with prefiltering, ensures a more accurate result and helps to eliminate noise and bad matches in the correspondence process. There are two forms of postfiltering that are applied and are based on the work by Konolige [75]. The first is a texture filter. Areas of low texture in the source images are excluded from the disparity map. A uniqueness ratio is the other method that is used for post filtering. Equation (3.18.) shows how the ratio is applied. Any value that is not consistent within a certain range of the minimum match for that disparity.

$$uniquness\ ratio > \frac{(match\ value - min\ match)}{min\ match} \tag{3.18.}$$

The process of stereo correlation is one of the most important portions of the algorithm for this thesis. It combines the physical model with the raw data to produce useful information. The final product of stereo correspondence is a disparity image.

3.1.4 POINT CLOUD GENERATION AND FILTERING

The step that follows stereo vision and disparity map generation is converting this information from a 2D disparity image to a 3D point cloud. To translate the disparity values to a 3D point, the camera parameters must be used. The result is Equation (3.19.). It shows that depth (Z) is inversely related to the disparity and is dependent on intrinsic (f) and extrinsic camera properties (B).

$$Z = \frac{Bf}{d} \tag{3.19.}$$

where B is the camera baseline, f is the camera's focal length, and Z is the depth from the camera origin. This, however, is an over simplification of the physical model. To fit the inhomogeneous transformation equation, some adjustments must be made.

$$Z = \frac{f}{\left(\frac{d}{B} + b\right)} \tag{3.20.}$$

where b is the non-homogeneity adjustment constant.

At this point, all of these variables needed to do this transformation exist. Equation (3.21.) is the model of the transformation from [u,v,d,1] to [X,Y,Z,W].

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -o_x \\ 0 & 1 & 0 & -o_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{1}{B} & b \end{bmatrix} \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} \tag{3.21.}$$

The effect of this transformation is the culmination of the camera model and the stereo correlation process. It takes raw sensor readings from a pair of cameras and translates them into a 3D point cloud containing the measurements in space.

3.1.4.1 Image Overlay

To allow for the inclusion of all of the source data into the final product, the color and NIR data points are overlaid onto the point clouds generated from the disparities. Since the visible and NIR images are optically aligned, the images can be added to the point cloud in the same manner. The manner by which the stereo depth maps are generated assumes that one of the cameras is the reference camera. This means that the coordinate system for the stereo camera system is located at this reference camera. This feature allows the images taken with the reference camera to be related to the 3D points from the disparity image on a pixel level. During the transformation of the disparity map to a point cloud, the color information is tied to each point in the cloud from the source images.

3.1.4.2 Stereo Resolution

A primary factor in the transformation described above is the resolution of the spatial transformation. This is an issue in all three dimensions. There are several published methods for addressing stereo resolution or

quantization error, each of which makes different assumptions [76], [77]. Due to the camera model assumptions, each pixel records data from a solid angle. The dimension of this solid angle is defined by the pixel size and lens geometry. Thus the primary factor in addressing this issue is the pixel size. The error can be as much as d. Let z_{min} and z_{max} be the minimum and maximum ranges in the FOV. Also assuming that $0 < z_{min} \leq z \leq z_{max} < Bf/d$. The effect of the error of the depth resolution is shown in Equation (3.22.) [77],

$$\Delta z = \left(\frac{z^2}{Bf}\right) x \Delta N \tag{3.22.}$$

where z is the depth and x the pixel size, ΔN is the step change in the disparity. This function shows that the depth error is a squared relation to measured depth. An assumption of this method is that the error in the x and y directions is presumed to be zero as they are very small compared to the error in the z direction. In order to provide a data collection system that met the criteria for this thesis, the baseline and focal length were chosen to provide an acceptable range error over the desired distance.

3.1.4.3 Point Cloud Filtering

Point cloud filtering prefiltering is a very important aspect in the data refinement process. It reduces the errors in the calculation of surface normals and ensures more accurate scan matches. Even after all of the stereo filtering, outliers remain in the data sets. A statistical outlier filter was used to filter the point clouds that were generated by the stereo visions system. This filter operates on basic statistical principles. The input to the filter is the raw data set, a mean value, and a standard deviation [61].

$$Filter \geq k \pm \alpha\sigma \tag{3.23.}$$

where k is the defined mean threshold and α is the number of standard deviations that the filter will allow. The statistical outlier filter removes a point if its mean distance from the surrounding points is above the defined mean and within the range defined by ασ deviation set value. The assumption made with this filter is that the distribution of neighboring points fit a Gaussian distribution. By this assumption, the outlying data points can be effectively trimmed. This process can be seen in Figure 3-11. After the data has been filtered, it is ready to undergo the next step in the process.

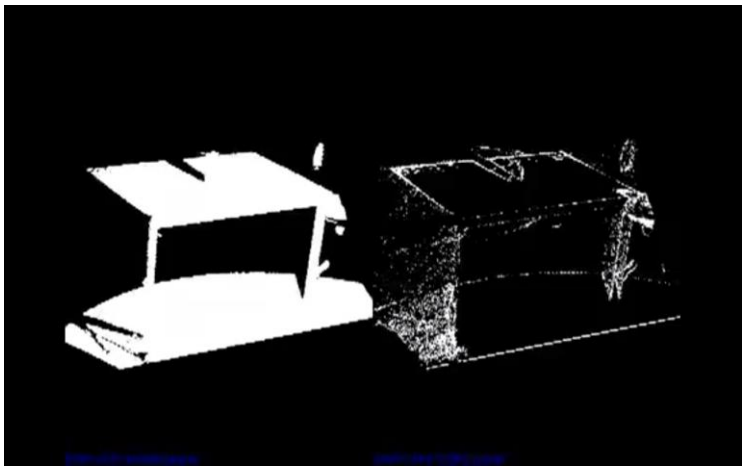

Figure 3-11. Filtered Point Cloud Data. The data on the left is the filtered point cloud of a table. The data on the right is the removed portion of the original data. The filtering was done with the statistical outlier removal filter [61]**.**

The next stage of prefiltering is a down-sampling technique. This method is based on a Voxel Grid filter [61]. This process divides the 3D space that original data set occupies and segments it into equal 3D volumetric pixels, or voxels. Each of these voxels contains any number of data points. For each voxel the centroid of its resident points is found and a single point is placed there to represent the original data. The formula for this process is shown in Equation (3.24.). This method preserves the original shape while reducing the amount of data that must be processed in the following steps [61].

$$\boldsymbol{P_c}\begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}x_1 + x_2 + \cdots + x_n\\ y_1 + y_2 + \cdots + y_n\\ z_1 + z_2 + \cdots + z_n\end{bmatrix}\begin{bmatrix}\frac{1}{n}\end{bmatrix} \tag{3.24.}$$

where Pc is the new centroid point. It is equal to the average position of the points in each dimension.

3.2 Simultaneous Localization and Mapping

The concept of SLAM is not an old one. It has developed in recent years to support the advances in robotic technology. SLAM is a very powerful category of algorithms. There are a wide variety of methods that have been used to solve the problem of concurrently sensing the robot's motion and building a map of the environment that the robot's sensors see. The following sections break down the mechanics of this process, beginning with initial scan registration, following with the recursive estimation needed to reduce error and drive the solution to the ideal and accurate value, and localization and pose estimation from the previous steps. Both pre- and post- filters play an important role in handling the complexity and inconsistency of real world data.

The premise of SLAM is the ability of an algorithm to build a map of the environment observed by a moving set of sensors and to also use the same data to deduce its position with in this world. Another key feature is that the location of all of the features and the trajectory of the system can be estimated without a priori knowledge. A two- dimensional simplification of SLAM can be seen in Figure 3-12.

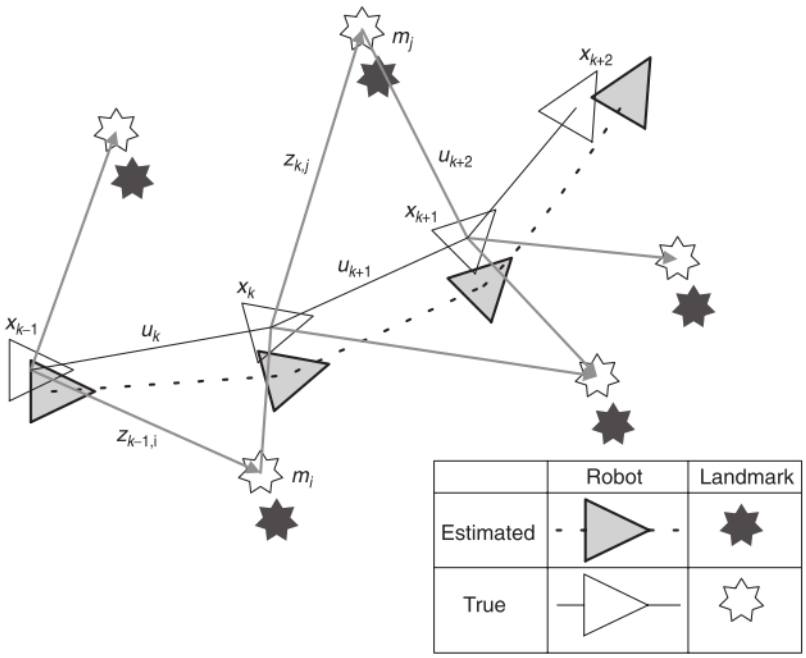

Figure 3-12. SLAM estimation. The process of estimating position, pose, and map building can be seen above. The position is estimated from the measurements of the landmarks [43]**.**

This premise is expanded into the 3D, 6DOF world. In many algorithms, the only change is the complexity of the calculations required to find the needed values. The method used in this application is not a commonly used method for SLAM specific algorithms, yet is more often applied to other forms of data registration. However, it is used in several papers for registration of point clouds [78], [79].

3.2.1 POINT CLOUD REGISTRATION

Point cloud registration, the process of aligning two point cloud data sets of the same environment from two different times and perspectives, can be a complex problem. If it is framed as an optimization problem, many algorithms can be applied toward its solution. The solution optimizes the 6DOF system by finding the ideal rotation and translation values to relate the two point clouds. The most commonly used algorithm of this type is ICP [53]. However, the algorithm used for point cloud registration in this thesis is known as Sample Consensus – Initial Alignment or SAC-IA. This algorithm sample consensus method was designed by Rusu et al. and is described as "a new sample consensus based method for bringing two datasets into the convergence basin of a local non-linear optimizer." [79] It iteratively finds an ideal solution without any information on the initial pose of the data. The two main steps of this procedure are rooted in the Fast Point Feature Historgram (FPFH) and the SAC-IA. The result of this method is a pose and position estimation between the two input data sets.

Several of the keys to this method are the speedy finding of feature points and the accurate alignment of the two 3D data sets. The feature point histograms assist the algorithm in finding a solution faster. There are also benefits to having ordered data points. Since the stereo disparity maps are turned into point clouds sequentially, the data is already ordered. This eliminates one of the steps needed to set up the registration. The characterization of FPFH is needed in order to initiate the registration process.

3.2.1.2 Fast Point Feature Histograms

Fast point feature histograms are an upgrade to Point Feature Histograms (PFH). They reduce the complexity of calculation from $O(k^2)$ calculations to O(k) calculations. Another one of the key optimizations in the FPFH is the caching of feature values, which is made possible by the ordered data. To understand the FPFHs, it is best to grasp PFHs first.

PFH relies on two pieces of data on each point in the cloud: the 3D location p(x, y, z) of the point and the estimated surface normal vector of that point *n*(nx, ny, nz). The PFH is computed via the following steps. First, all of the points, k, that lie within a sphere of radius, r, from the initial point, p, are selected. These points are selected using a k-d search tree. This sorts and orders the input data into a tree format so that the searches are more efficient. The k-d tree is a binary tree in which each point is k-dimensional. The algorithm in 2D can be described as follows in Figure 3-13:

Algorithm BUILDKDTREE(P, *depth*)
1. **if** P contains only one point
2. **then return** a leaf storing this point
3. **else if** *depth* is even
4. **then** Split P with a vertical line ℓ through the median x-coordinate into P_1 (left of or on ℓ) and P_2 (right of ℓ)
5. **else** Split P with a horizontal line ℓ through the median y-coordinate into P_1 (below or on ℓ) and P_2 (above ℓ)
6. $v_{\text{left}} \leftarrow$ BUILDKDTREE(P_1, *depth* + 1)
7. $v_{\text{right}} \leftarrow$ BUILDKDTREE(P_2, *depth* + 1)
8. Create a node v storing ℓ, make v_{left} the left child of v, and make v_{right} the right child of v.
9. **return** v

Figure 3-13. k-d tree search algorithm. This is the pseudo code to describe the 2D k-d search tree. It is expanded to 3D for this application [61].

From the resulting search tree, neighboring points can be easily identified and used for the calculation of the FPHs. The second step defines a Darboux *uvn* frame [79], Equation (3.25.), for each point pair in the defined set of points. A point pair is defined as p_i and p_j that are points in the set where $i \neq j$ and n_i and n_j are the respective estimated normal vectors of the points.

$$u = n_i \qquad (3.25.)$$
$$v = (p_j - p_i) \times u$$
$$w = u \times v$$

The next step in the PFH calculation is finding the angular variation of n_i and n_j as shown in Equation (3.26.)

$$\alpha = v \cdot n_j \qquad (3.26.)$$
$$\phi = \frac{\left(u \cdot (p_j - p_i)\right)}{\|p_j - p_i\|}$$

$$\theta = \arctan(w \cdot n_j, u \cdot n_j)$$

The resulting mesh network from the calculation of the PFH can be more clearly understood in Figure 3-14. It is a 2D abstraction of a 3D reality, but the connection and region of influence is defined by the radius, *r* around *p*, the point of interest.

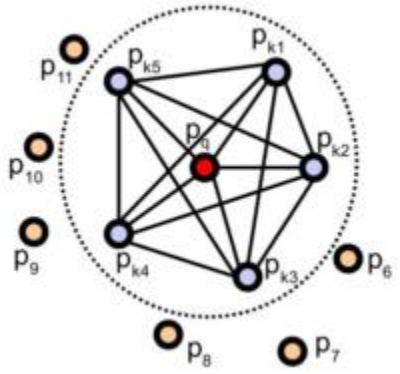

Figure 3-14. PFH Mesh Visualization. This figure shows the mesh created by the PFH calculation within the sphere that is defined by the radius, r [79]**.**

Especially in larger data sets, there can be large numbers of PFHs that are similar or even the same. This can cause a significant problem when trying to match point clouds based on these FPHs, causing a sliding problem where points do not aid in finding the correct match. To address this issue, a persistence method can be used to find the points that are more prominent. This is done by looking at what histograms are significant at different scales, by changing the *r* value. This can be simplified by approximating the PFH for a data set as a Gaussian distribution defined by Equation (3.27.)

$$\mu \pm \beta \cdot \sigma \tag{3.27.}$$

where μ is the mean PFH distance value, β is the controls the width and acts as a band stop filter for the desired interval, and σ is the standard deviation of the distance distribution. Less common points fall outside of Equation (3.27.). These points are the ones that should be selected for further analysis. To ensure that these points are actually distinctive, this technique is applied over several scales. A point is persistently unique if it is defined as follows in Equation (3.28)

$$P_f = \bigcup_{i=1}^{n-1} [P_{f_i} \cap P_{f_{i+1}}] \tag{3.28.}$$

where P_f is the final set of unique points from the various scales and each P_{fi} is the points selected at each scale.

The changes made from the PFH to the FPFH are small, yet reduce the number of operations needed for calculation from a factor of k^2 to k. This is accomplished using the SPFH, which only calculates the distances from the point of interest to the neighboring points, not the entire network, using Equation (3.26.). The second step in this process is re-determining each point's k neighbors and using the SPFH values from those to weight the final histogram. This value is called the FPFH and is detailed in Equation

$$FPFH(p) = SPFH(p) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_k}SPFH(p_k) \tag{3.29.}$$

where ω_k is the distance between p and p_k. This is best visualized in Figure 3-15. The main differences in FPFH and PFH are three-fold. First, FPFH does not fully connect all of the neighboring points; secondly, FPFH includes points up to $2r$ away from the central point; thirdly, FPFH combines SPFH values and recaptures some of the neighboring value pairs.

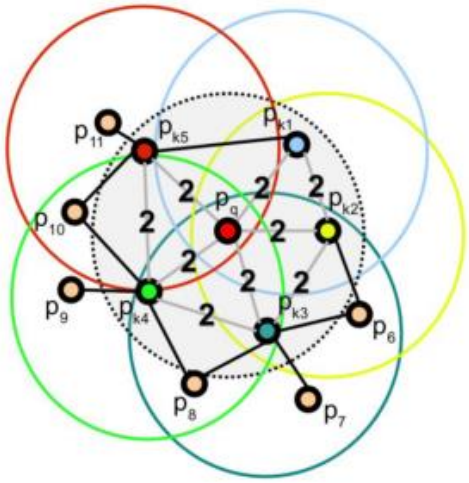

Figure 3-15. FPFH Mesh Visualization. This figure shows the mesh created by the FPFH calculation within the sphere that is defined by the radius, r. It contains points outside of the sphere and less connecting distances, however is much faster to compute [79]**.**

The next step in point cloud registration is the alignment of the two data sets, using the FPFHs calculated as described above.

3.2.1.3 Sample Consensus – Initial Alignment

The SAC-IA algorithm is an adaptation of a Greedy Initial Alignment algorithm [79]. The original algorithm is robust; but since it is a greedy algorithm, it can get stuck at local minimums instead of finding the global minimum. The goal of the SAC-IA is to allow for the sampling of a large number of correspondence candidates to make sure that the ideal match in found.

These correspondence candidates are then ranked based on a three step process. The first step is to select sample points, *s*, from the set of persistent points, P, found above. These points must also have pairwise distances above a threshold d_{min}. The second step is to find all the points whose FPFH is similar to the points in the first step. The last step is to compute the ridged transformation between the points in step one and two. A Huber penalty metric, Equation (3.30.), is also calculated to rate the quality of the transformation [79].

$$L_h = \begin{cases} \frac{1}{2}e_i^2 & \|e_i\| \le t_e \\ \frac{1}{2}t_e(2\|e_i\| - t_e) & \|e_i\| > t_e \end{cases} \tag{3.30.}$$

These three steps are repeated and the alignment that produced the lowest error is used as the initial, rough alignment.

The final step in completing the alignment is to apply a non-linear Levenberg-Marquardt algorithm to optimize the registration [80]. This technique is a damped least squares method for minimizing the error between the 3D point clouds.

$$S(\beta) = \sum_{i-1}^{m} [y_i - f(x_i, \beta)]^2 \tag{3.31.}$$

where the datum pairs are (x_i, y_i) and β is the optimization parameter. The curve f(x, β) is optimized so that the sum of the squared deviations becomes minimal. Each iteration β is replaced with a new estimate that is determined by the derivative and a damping factor that is adjusted for every iteration as well. This process is repeated until the values fall within a predefined range.

This registration process is repeated between each pair of point clouds that is collected in the data set. These sequential registrations are compiled to create a global frame point cloud. This global point cloud is the ultimate goal of the registration process and concludes the SLAM process.

3.2.2 POST FILTERING

The final stage of the SLAM process is to apply a post-filter. This filter uses a Moving Least Squares (MLS) surface reconstruction technique [64]. The result is a smoothing process that seeks to align the surface normals of the point cloud. This helps to eliminate the small data inconsistencies caused by noise in the measurements that are not easily removed by the statistical pre-filters. This technique takes advantage of an assumption from 3D modeling that there are no jump discontinuities over a small area. By resampling and aligning the normal vectors of the 3D space, the moving least squares methods ensures that this is locally true for the data set.

The MLS method returns a smoothed and continuous function from a set of unordered point samples. The function used to produce this result is show below

$$\sum_{i \in I} (p(x) - f_i)^2 \theta(\|x - x_i\|) \tag{3.32.}$$

where p minimizes the weighted least square error and is of a given polynomial power and θ is an error function. The overall effect of the post filtering of the point cloud data is the reduction in small matching errors and bad measurements from the original data through the use of a smoothing filter. This prepares the data for the next phase of operations.

3.3 DATA REPRESENTATION

One of the most important aspects of any project is to accurately and adequately represent the data that has been collected and processed. This project is no different. The culmination of the SLAM process results is a point cloud that contains all of the points collected in the previous steps. These maps contain the information that was collected by the sensors assigned to a point 3D space. Although these clouds are not ideal for human viewing, they are good sources of input data for path planning and autonomy algorithms. In order to provide easily decipherable data, all of the broadband data must be assigned to the point cloud. There are many ways to consolidate, compress, and more usably display these maps.

At this point, all of the constitute pieces needed for the application of this have been explored. This chapter provides a thorough investigation of the math, methods, and fundamental science needed to

accomplish the objective set forwards in this thesis. It began with the definition of a physical model for the cameras used to capture the data. The next step investigated one type of stereo correspondence and the associated filters to improve the collected information. It then covered the methods for translating the stereo data into 3D point clouds using the camera model derived previously. The process of registering and filtering the point clouds was then expounded upon. These sections included all of the math and methods that need to employed to achieve this process. Finally, Chapter 3 covered the display and rendering of the point clouds in a manner that conveys the data that they contain in a format that is useful to both humans and robots alike.

CHAPTER 4

4 APPLICATION OF THE FUNDAMENTAL SCIENCE

The theory and methodology set forwards in Chapter 3 must be applied and systematized to allow for an easy understanding of the experimental process. This chapter describes how the fundamental science from Chapter 3 is incorporated for the fulfillment of the thesis statement. It shows the process used to implement a 3D scene recreation and mapping algorithm using a variety of component algorithms that can be used in UGV applications and can enhance the information delivered to current autonomy and classification systems. This chapter will show how each equation is used in the process and how the data flows from one end of the system to the other. It begins with an overview of the complete algorithm then continues to show each step with its associated input data, assumptions, equations, and output data.

4.1 APPLICATION PROCESS

To achieve the implementation of a 3D scene reconstruction passive perception system, the methods from Chapter 3 were organized into a process that must be executed. The process of applying the methods detailed in Chapter 3 can be visualized in the flow chart from Appendix B. The process begins with image acquisition. These images can come from any camera, but the cameras used for this thesis were single, beam split, two CCD cameras that capture both visible and NIR images. With two of these cameras, four time-synched images are returned every time the trigger is activated. These source images can be saved to preserve the original data. The next step in the process is to apply the stereo correspondence equations. This procedure begins with the pre-filtering of the color images that have been converted to grayscale. The images are pre-filtered to enhance the stereo matching process. The stereo block matching equations from Chapter 3 are then applied to each sequential image pair.

There are several parameters for the stereo matching that can be tuned to achieve the best correspondence. These parameters will be further examined in the next section. After stereo matching is done on the source images, the resulting disparity map must be post-filtered. This process reduces the number of incorrect matches and removes most of the noise in the disparity images. Both the stereo matching and the conversion of the disparity maps into 3D point clouds require information about the camera's physical parameters. This includes intrinsic parameters which define each camera's physical model constants and the extrinsic parameters which relate the cameras in the stereo pairs in 3D space.

The next phase of the procedure is to convert the disparity maps and the source images into 3D point clouds. This is done using the stereo camera parameters from the camera calibration. Each disparity map and corresponding visual and NIR image are loaded and each pixel is translated into a 3D point using a projective transformation. Each frame is stored in its own point cloud. All of the point clouds in the data set are then pre-filtered using a statistical outlier removal filter and then a voxel grid down sampling. These two steps remove more of the noise from the stereo matching and reduce the number of points to increase the computational efficiency.

After the point clouds have been filtered, the registration process can be started. This commences by finding the FPFHs. These define important points in the point cloud that are used to register the two clouds to each other. These sets of points from each source point cloud are input into the SAC-IA

algorithm. This algorithm iteratively solves for an ideal solution to register the point clouds to each other. This is repeated until all of the clouds are registered to each other. From this point, the data that was collected is ready to be displayed in a 3D environment with either a visible or NIR image overlay. The following sections go more into depth on the steps and functionality of each phase of the process.

4.2 STEREO VISION

The development of a stereo vision system was the first of the two main steps in creating a passive perception 3D scene reconstruction system. The math for inferring depth from stereo cameras has been detailed in Chapter 3; however, this section will focus on the data flow, assumptions, and application of the mathematical algorithms to the problem of applying this to a system that can be used in UGVs.

4.2.1 STEREO ASSUMPTIONS

The key assumptions of this process are three-fold. First, the cameras are fixed in relation to each other. This is essential because the calibration that was conducted is no longer valid if the cameras shift and the disparity maps cannot accurately be transformed into 3D models. Secondly, it is assumed that the cameras view largely the same scene. If information in the image pairs does not overlap, then no matches can be found between the two. Lastly, the stereo matching algorithms assume that the local depth in an image is continuous. This assumption allows for the windowing technique to be applied to find the best match between features in the image pairs.

4.2.2 STEREO PROCESS

As stated above, the source images for the stereo matching algorithm are a pair of temporally registered color images that have been converted to grayscale. The NIR images are also passed along for future use, but do not play a part in the stereo portion of the algorithm. The grayscale image data is fed into the stereo processing functions. The images are pre-filtered using a normalized cap filter. This filter adjusts the data for lighting differences and reduces noise in the images by truncating the amount of change allowed between adjacent pixels. Figure 4-1 shows how this filter is applied to a step change.

Figure 4-1. Cap Filter Example. (A) This image shows a large difference between adjacent pixels. (B) This shows how the cap filter smooths this jump by only allowing the maximum difference to be the cap filter value. This example is over continuous space for visualization sake. In the images this would be a discrete operation, where the slope is stepped.

These filtered images are fed to the SAD stereo block matching algorithm. The key parameters that control the results of this function are the number of disparities and the SAD window size. These two variables control the closest depth that can be found by the stereo function and the amount of detail that can be matched. By having a small window size, the small features can be found, but more bad matches occur.

$$SAD = \sum_{(u,v)\epsilon W} |I_R(u,v) - I_T(u,v+i)| \tag{4.1}$$

The SAD window size is defined in the equation above as W and controls the number of pixels that are compared between the two images to find the matching cost. This results in a disparity map, seen in Figure 4-2

Figure 4-2. Stereo Vision Example. (A-B) These are the stereo pair of images used for stereo correspondence. (C) This image is the disparity map where red is close and blue is far away or no match. The shape of and relative positions of the primary objects can be clearly seen in this result.

After the initial matches have been found, the results must be post filtered. This is done using the uniqueness ratio filter [32]. This filter removes points that do not match the points in the surrounding regions. This enforces the smoothness constraint. The last filter that was applied is the texture filter [32]. This filter removes matches that come from areas of low texture. Between these two filters, many of the errant matches can be removed from the stereo disparity map, exemplified in Figure 4-3.

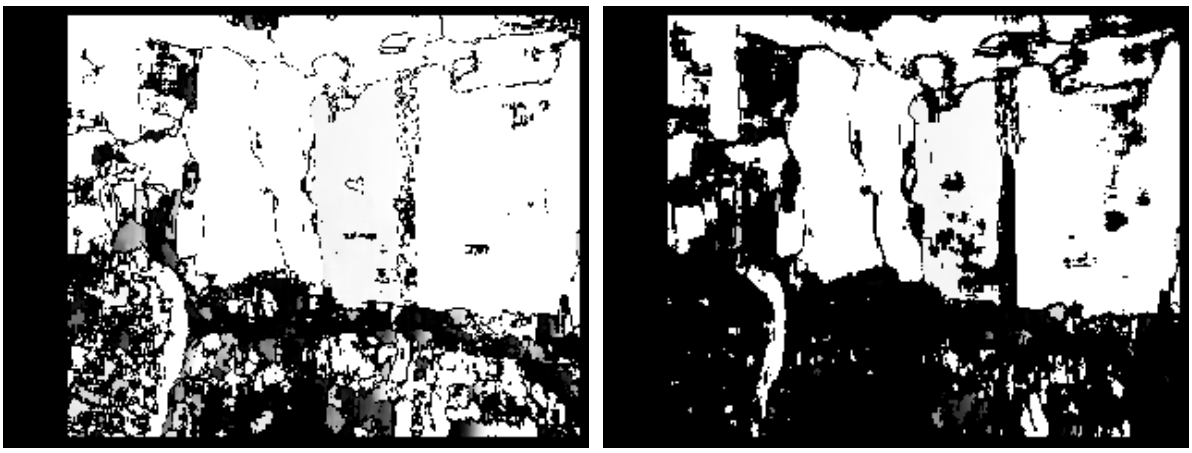

Figure 4-3. Stereo Filtering Example. (A) This image shows an indoor scene with little depth variation and no post filtering applied. (B) This is the same scene with the post filtering applied. It successfully removed a very large amount of noise.

The resulting data is a disparity map that corresponds to the original visual and NIR images and is the same size. This disparity map can be translated from a 2D image into a 3D point cloud using the camera calibration results and the projective transformation. During this process each pixel is converted from the [u, v, 1] frame to the [x, y, z] frame. The color and NIR intensity value are also stored with their corresponding points. Once all of the points in the disparity image have been translated into 3D space,

this phase of the process is complete and can be seen in Figure 4-4. The point clouds are now prepared for the next portion of the process.

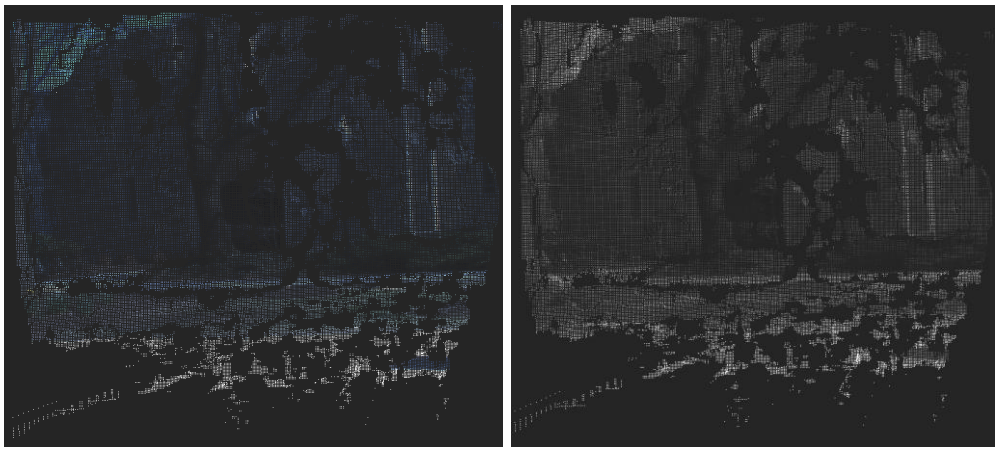

Figure 4-4. Colored Raw Point Clouds. (A) This figure shows the color image overlaid onto the point cloud created from the disparity map. (B) This is the same point cloud with the NIR information overlaid.

4.3 Simultaneous Localization and Mapping

The second main step in creating a passive perception system for 3D scene reconstruction is the SLAM segment of the algorithm. The SLAM process takes the colored point clouds from the stereo step and registers them to each other, thus creating a complete map of the environment as well as a log of the position of the sensors as they took the data. The final result of this step is the 3D world model that can be used for UGV applications.

4.3.1 SLAM Assumptions

As with the stereo process, there are several key assumptions that direct the application of the SLAM process. The first is that sequential point clouds must have some spatial overlap. This ensures that there are corresponding features that can be used to match between the two data sets. Within this assumption, there lies another, each data set must contain unique features. Without distinguishing features, it is very hard to find the correspondence between the two pieces of data. It is also assumed that both the state of the sensors and its environment can be measured and estimated concurrently. These assumptions shape the application of the theory detailed in the previous chapter.

4.3.2 SLAM Process

The process of executing SLAM upon the data received from the stereo process can be broken down into four steps. The first step is the pre-filtering of the point clouds. Each input cloud is filtered to enhance the quality before passing the data off to the following steps. The second step is the identification of the feature points in set a point clouds. These features are used by the third stage. Registration takes the features two point clouds and preforms the SAC-IA algorithm. The result of this action is a registered point cloud, a transformation matrix, and the error in the registration process which is in the form of the Euclidean distance between the point clouds. The final step is to filter the registered point clouds. This step smooths the registered output and then down samples the final point cloud. Each step is conducted sequentially, with the second and third steps being repeated until all of the data has been processed.

The first stage in the SLAM process is to pre-filter the point clouds. This procedure is directly obtained from the theory set forward in Chapter 3. Each point cloud in the data set is filtered in a two stage process. The first filter is the voxel grid down sampling. This filter is defined by one variable called the leaf size. The leaf size defines the size of the voxel that is used to average the point which lies inside that cube. The points are replaced by a new point that is located at the centroid of this sub set of points. The second phase of the pre-filtering is to apply a statistical outlier removal filter. This filter is defined by two parameters, the standard deviation and the number of points by which the statistics are calculated. The number of points helps to define the mean value of the distances between the points. The standard deviation acts as the threshold value. Any point, in the points surrounding the point of interest, that is more than the defined standard deviation away is removed from the data set. The primary goal of this set of filters is to make the input data more manageable and to eliminate outliers and bad matches in the data.

The filtered point clouds are passed to the second phase of the process, which is the feature point extraction. The features that are extracted are the FPFHs. These are points that are unique within the point cloud. They are found by creating a histogram of the distances between points within a specified range. Points that have statistically unique histograms are selected to represent the data set. Several examples of this process can be seen in Figure 4-5.

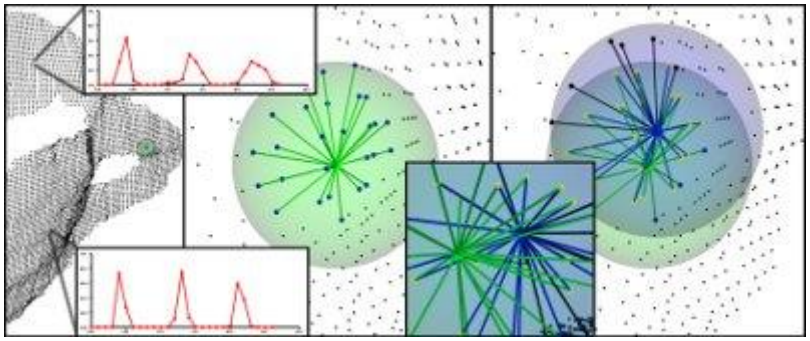

Figure 4-5. FPFH Visualization. This figure shows how the FPFHs are calculated in a 3D space as well as some of the resulting histograms [61]**.**

This array of feature points is then fed into the third step which conducts the registration. The registration process uses the SAC-IA algorithm described in Chapter 3. It conducts many trial alignments between the two input sets of feature points and assigns each with a cost value. It iterates through this process until the convergence parameters have been met. The variables that control this process are the minimum and maximum distances that the clouds can be apart. These set the bounds on the possible matches. Other parameters that define the convergence are the Euclidean fitness epsilon and the transformation epsilon. These define the distance between the two clouds before they are considered to have converged. This process yields both a registered point cloud, consisting of the two input point clouds transformed to the same coordinate system, and the transformation that was applied to create this alignment. The transformed point cloud is feed back into the process of step two and three with the next point cloud until all of the point clouds have been registered to the first coordinate system. The final step is the post-filtering. This, like the pre-filtering, is a two-step process. The first stage is to smooth the surface created by the registered point cloud data. This is done using a moving least squares filter that adjusts the points to better align the surface normals, shown in Figure 4-6. It helps to eliminate inconsistencies from the registration process.

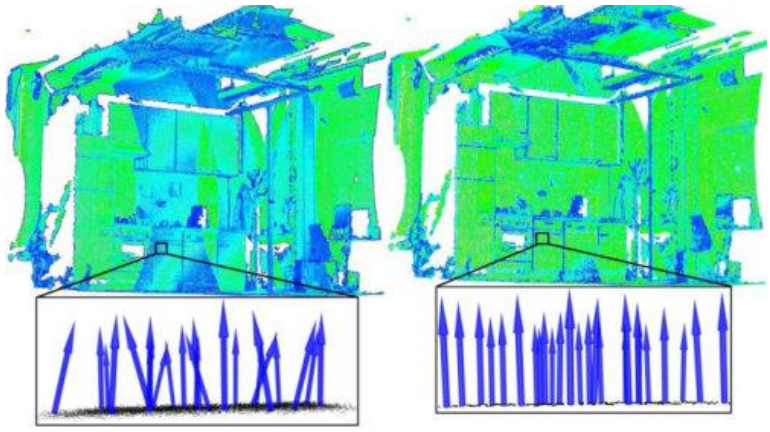

Figure 4-6. Post-filtering Example. This figure shows a before and after of two registered point clouds with MLS being applied to smooth the surface by aligning the surface normals [61]**.**

The second stage is the voxel down sample once again. The purpose of this filter is to reduce redundant points induced by overlaps in the data sets after registration. The output from this post-filter is the finished product. It is ready for display or to send to an autonomy or classification algorithm. It contains the spatially registered point clouds from a series of samples along with the original color and NIR values for each point in the cloud. These colored point clouds are the broadband 3D world models that were created using a passive perception system designed for use on UGVs.

CHAPTER 5

5 EXPERIMENT AND EXPERIMENTAL RESULTS

Chapter 5 is the application of the theory and methods detailed in Chapter 3 and the process from Chapter 4. It describes the experiment and experimental procedure used to test and validate the passive perception system that was designed to create 3D world models to augment UGV perception, classification, and autonomy systems. To begin, the test objectives are defined. The next section investigates the data collection system, its construction, its purpose, and its output. The following sections show the step- by-step process from Chapter 4 applied to the collected data and the experimental results.

5.1.1 OBJECTIVES

The objectives of these experiments are to collect data from two broadband stereo cameras, compute the scene depth with stereo correspondence, fuse the data into a world map with SLAM techniques and render the data into a solid 3D environment with the source and calculated data. The final product is a passive perception based 3D scene reconstruction system that combines existing technologies to allow for practical use on UGVs. The flow chart of the system can be seen in Appendix B.

5.1.2 DATA CAPTURE SYSTEM

The data capture system used for this experiment contains two broadband cameras. They are the JAI AD-080GE. Each camera is contained in one body with a single lens. There is a beam splitter inside the camera body that splits the light onto two separate CCDs, shown in Figure 5-1. One sensor records the visible spectrum on a Bayer filtered CCD and the other records the NIR light on a monochrome CCD. These images are output via two Ethernet ports and connected via a switch to the processing and data collection computer.

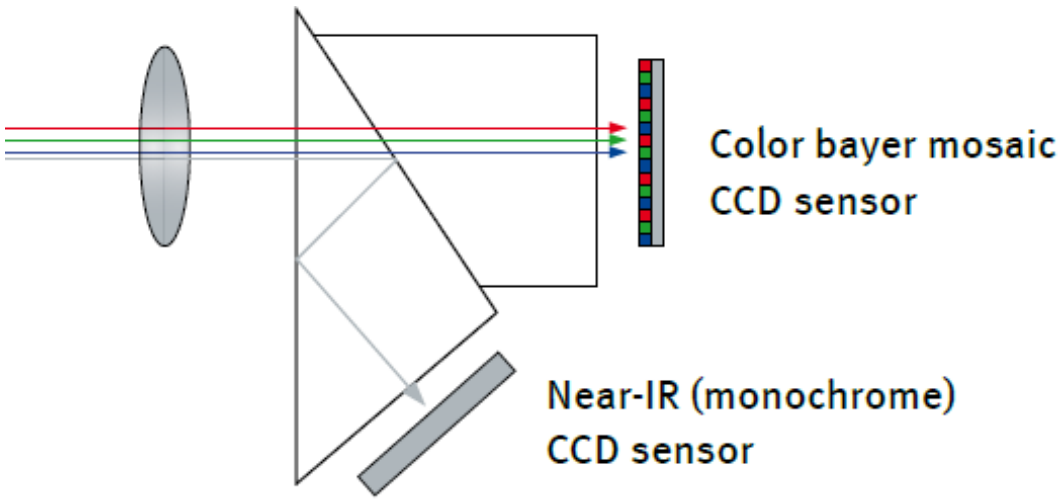

Figure 5-1. JAI Beam Splitter. The JAI camera setup can be seen in this figure. The incident light is split with a prism onto two CCD sensors. One of the sensors is for the visible spectrum and the other is for the near infrared [81]**.**

The cameras are powered by an embedded power distribution board that has features for timing and a GPS interface. The timing system is powered by a microcontroller and internal clock. This system emits a

pulse train at a set frequency. The frequency can be set programmatically. The pulse train is fed into the digital I/O pins of each camera. This timing signal ensures that the pictures from each camera are taken at the same time. Since temporal registration is a key assumption of the core algorithms, this is an essential feature of the camera system. The data capture frequency was set at 10 Hz. Each image is captured at a resolution of 1024x768 pixels. Each sample contains a combined total of eight color planes. This data rate falls below the limit of the GigE Ethernet connection. However since all of the data is captured at once, there is an instantaneous overload of the network switch's buffer memory. In order to accommodate this requirement and resulting challenge, each camera's packet size and inter-packet delay settings were changed from the default to 5796 bits/packet and 9000 clock ticks between packets. This resulted in clean images with no packet loss or delay. The other features of the passive perception rig, Figure 5-2, can be utilized in future work and to assist in integration with other systems to ensure accurate absolute timing.

Figure 5-2. Perception camera rig. The passive perception rig with 4 cameras with 6 sensors and all of the networking and power requirements included below.

The complete passive perception system is mounted on a stable metal base that can be attached to various vehicles as a perception package. The layout of the physical implementation can be seen in Figure 5-2. It contains all of the necessary power and communications interfaces needed to run the cameras. Figure 5-3 shows the data flow and power flow for the data collection system. This system was designed to provide stability and flexibility to enlarge the possible applications when testing perception algorithms.

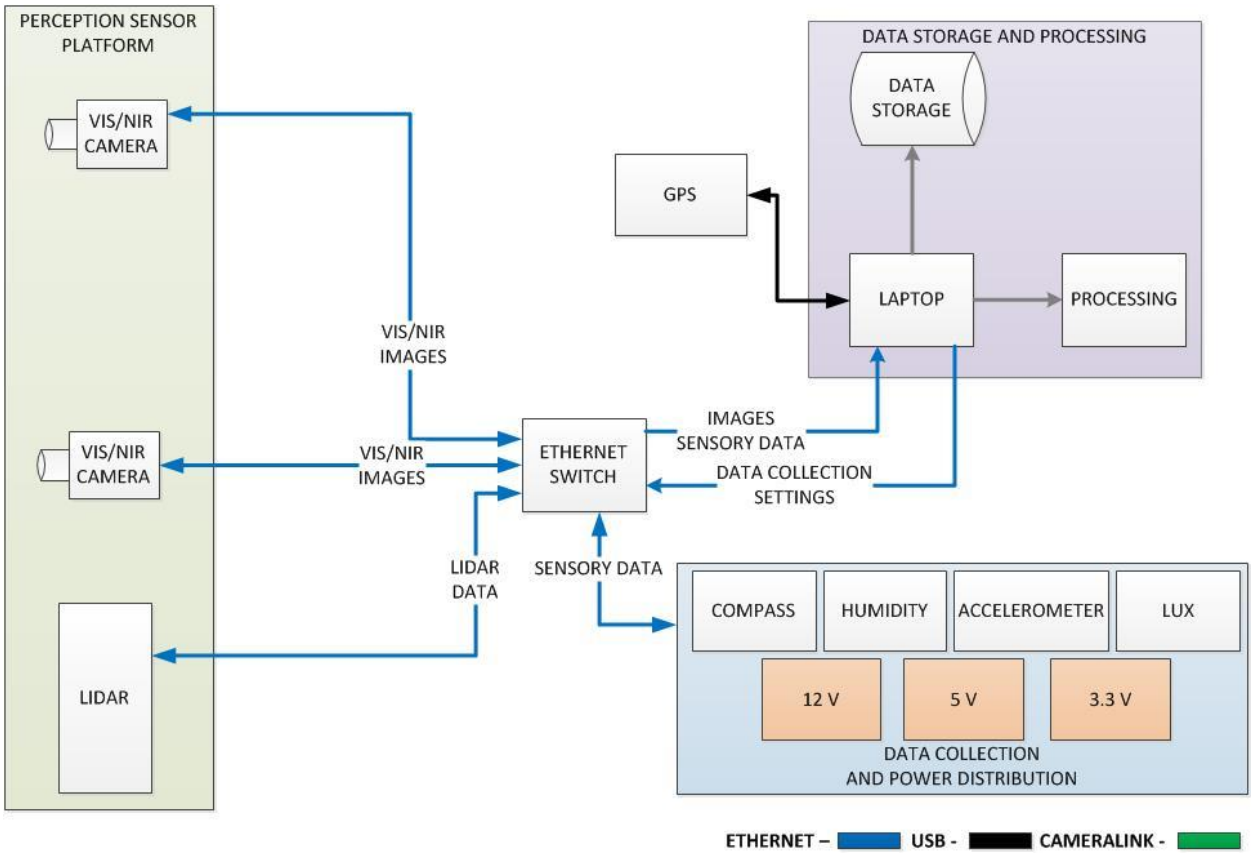

Figure 5-3. System Info and Power Diagram. This shows all of the components in the passive perception system including power and communication flows.

The full camera specifications can be seen in Table 5-1. The camera properties, camera settings, and the estimated intrinsic and extrinsic parameters are included below.

Table 5-1. Camera Specifications. These specs include all of the important pieces of information for the JAI AD-080GE cameras [81]**.**

	JAI AD-080GE VIS	JAI AD-080GE NIR
Modality	Visual	Near Infrared
Range	0.4-0.7 um	0.7-1.0 um
Resolution	1024x768	1024x768
Pixel Size	4.65 um	4.65 um
Data Link	GigE Ethernet	GigE Ethernet
Frames per second	30	30
Sensor Size	1/3"	1/3"
Lens Focal Length	6mm	6mm
FOV (°)	42	42
Voltage (V)	12	12
Current (A)	0.5	0.5
Power (W)	6	6

The complete passive perception system contains several features that fall outside of the scope of this project; however, LWIR and SWIR cameras could be integrated in the future. For a more detailed description, see Section 0 on future work.

5.1.3 TEST DATA

There were two types of data collected by the passive perception system, the NIR and VIS images. These raw images were recorded and stored to the hard drive. At each step along the process, the calculated data is stored with the source images. Figure 5-4, below, shows the raw data that was collected by the passive perception system. The processed data will be presented in the following sections.

Figure 5-4. Sample Source Images. (Top Left) Image from the left visible camera. (Top Right) Image from the right visible camera. (Bottom Left) Image from the left NIR camera. (Bottom Right) Image from the right NIR camera. It is easy to see that the objects are aligned in each column because of the optics in the JAI cameras.

The data used in this experiment was collected in several different environments to test the scalability and robustness of the system. The tests started with simple single frame data sets. These were used to test the individual functions of the program, such as stereo correlation and feature detection. The data collection was then scaled up to include indoor tests. These addressed more feature detection testing and initial loop closure. Since the focus of the research is passive world modeling for UGVs, the main portion of data collection was done in unstructured outdoor environments. These outdoor tests began in a small scale to tune the program for outdoor environments and include the broadband data in the texturization feature.

5.1.4 SOFTWARE ARCHITECTURE

The main focus of this thesis is to combine passive perception algorithms to enable their application on UGVs in unstructured environments. To facilitate this goal, the software interface was a key component. It provides the data collection, data analysis, and user interface. The data collection and analysis is the core of the work done for this thesis. The user interface, although not required, delivers key functionality by allowing the user to understand, manipulate, and view the results.

The software was written in C++. It utilized several open source or free C++ libraries, such as Qt 4.8.4, OpenCV 2.4.3 [82], and Point Cloud Library 1.6 [61]. Qt was used to construct the GUI, OpenCV was used for the image processing, and PCL was used for point cloud manipulation and registration. The usability of the software was a prime factor in its construction, both from the programmer's view and

from the user's view. The programing technique used to structure this software was the Model View Controller. This style is highly effective when using Qt's built- in functions for the sharing of data between the different parts of the program and the user. The Model View Controller is a software structure that separates the user interface from the data and manipulation of that data. It accomplishes this process by utilizing three separate fixtures: the controller, the model, and the view. The UI is comprised of the controller and the view and the model contains the core functions of the program. The controller sends commands from the user to the model in order to affect the desired change. The View is what is typically known as the GUI. It displays the results of the model for the user. Figure 5-5 clearly shows how this interaction is achieved.

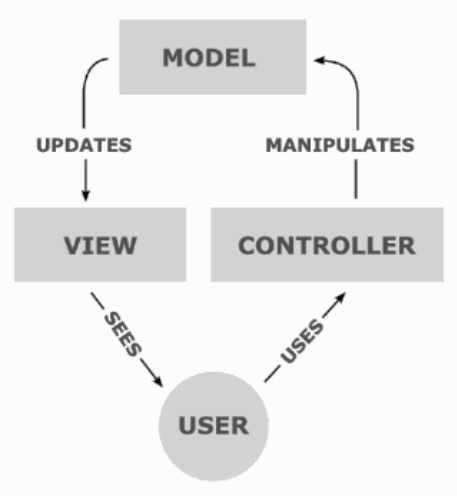

Figure 5-5. Model View Controller. The user inputs commands to the controller; the controller feeds data to the model which updates the view. This view is seen by the user [83].

The slots and signals methodology that is a key feature of Qt was used to practically realize the software architecture detailed above. Qt's slots and signals were utilized to enable the flow of data from one class to another and from the controller to the model and to the view. This methodology is quite simple. Each time one class needs to send data to another, it emits a signal, which is constructed like an empty function. The slot is constructed like a normal function and processes the data received. In most cases, these slots are in the controller/view class. These slots and signals are connected in the controller/view class called MainWindow. The program is split into three features, see Figure 5-6. Each feature contains function and variables for that specific feature. Each feature is realized in the software as a class. In the user interface allows for the selection of the level of depth that the program will run. This ranges from data acquisition, to stereo and data logging, to SLAM and world modeling. Each of these features is interfaced with through the controller, MainWindow, class.

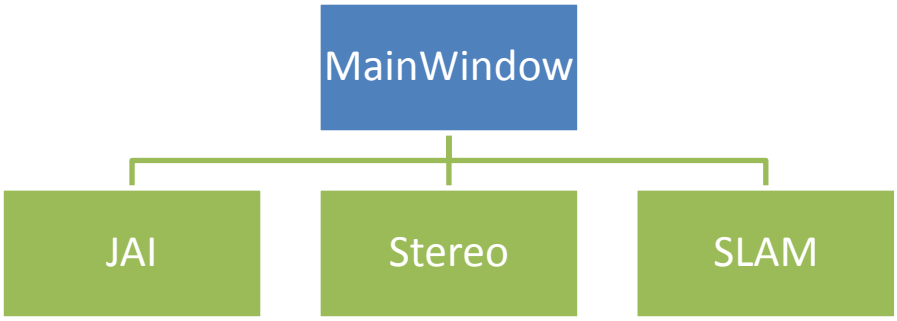

Figure 5-6. Software Architecture. The MainWindow class serves as the controller and viewer class. The other three classes serve as the model. All communication is enacted with the slot and signal architecture.

The first feature of the software is the acquisition class. It is used to interact with the cameras and to process and store the images collected. The cameras included a C++ Software Development Kit (SDK) by which the interfacing occurs. The JAI SDK is comprised of several structures and modules. The first is the factory. The factory contains cameras and acquisition functions. Each camera is discovered and then the parameters of that camera can be manipulated. Once a connection to each camera has been established, an acquisition stream is opened. From this stream, images are pulled into a class variable from a buffer. At this point the images are sent to the controller for display in the UI.

The next feature is stereo. This feature is included in the MainWindow class. The stereo operations are conducted using the methods described in 3.1.3. Several of the parameters can be tuned from the UI. This allows the user to adjust the stereo in real-time to best account for the environment. The third feature is SLAM. Since the SLAM portion of this thesis is not conducted in real time, this class reads images that were saved by the stereo class. This class also utilizes the PCL to filter, down-sample, and register the distance data collected in the stereo feature. The SLAM class used the procedure laid-out in Section 3.2. It starts with the conversion of the disparity maps into point clouds. All pixels that were filtered in the post processing of the stereo process are not counted as valid points in the conversion. The point clouds generated are stored in a vector. All of the point clouds are also filtered using the statistical outlier filter and then down sampled with the voxel grid approach. These sequential data sets are then registered using the SAC-IA method. The result is a complete point cloud representation of the recorded scenes. The final step in this process is the post-filtering. A smoothing LMS filter reduces noise and cleans up the surfaces in the data.

The next step of the process is the rending of the data. This was done in several stages. Each step of the SLAM class has an associated visualization function. The raw, filtered, and registered point clouds can be visualized. Each of these clouds has the source image data associated with each point. The software architecture in this class is such that this can be expanded to include other source images and classification results.

The main issue that was encountered when creating this software package was the requirements of the open source libraries. Once all of the correct versions libraries had been installed correctly and linked in to the C++ project, the libraries were easily accessed. Overall, the software system was constructed to meet the requirements of the project through a GUI application of several open source libraries with the functionality for the processing portion of the project.

5.2 Experimental Results

The background set forth in Chapter 1 shaped the flow and thought process used to design the experiments for this thesis. As the application is UGVs in unstructured environment, the ultimate goal of these experiments is to test the system in environments indicative of what the UGV might encounter. Using the hardware and software systems described above, a range of data was collected to adequately test the functionality of both portions of this project. The following sections recreate the methods detailed in Chapter 3 and Chapter 4 to show the steps taken and the results of the experiments. It begins with camera calibration and continues with stereo matching, SLAM, and finally rendering.

5.2.1 Camera Calibration

The calibration of the physical hardware is key to the remaining processes. It ensures that the position in 3D space of each camera is known as well as their relative position to each other, the extrinsic parameters. This calibration also encompasses finding the intrinsic parameters of the cameras. This data is essential for stereo and for the transformation of the disparity maps into 3D point clouds. Figure 5-7 shows some of the images used to calibrate the stereo camera hardware. These image pairs, taken from each of the two cameras at the same time of a common and known object, allow the calibration tools to calculate the camera parameters.

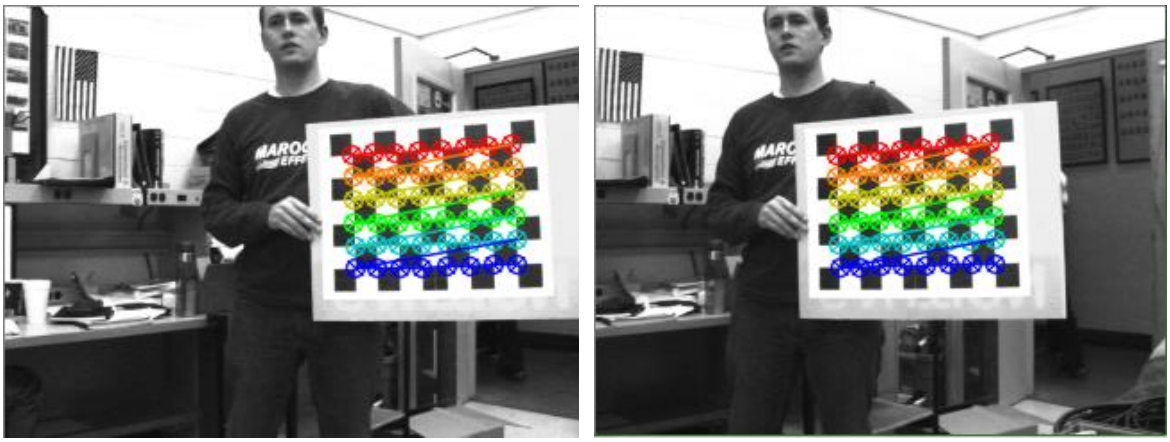

Figure 5-7. Calibration Corner Detection. The corners of the checker boards shown in Figure 3-5 are found and labeled with colored circles. This is done for each camera to find the intrinsic and extrinsic parameters.

The checkerboard grid is used because its distinct corners and high contrast regions provide for a reliable detection. The corners of the checkerboard are detected using the OpenCV functions. The returns are fed to a stereo calibration function that outputs all of the data needed for stereo camera calibration.

Table 5-2, Table 5-3, and Table 5-4 show the results of this calibration process for the intrinsic and extrinsic parameters for the cameras individually and for the stereo system.

Table 5-2. Left Camera Calibration. The data contained in this table is the intrensic parameters of the left camera.

Calibration Results – Left Camera	
Focal Length (f) (px)	[402.28267 406.08120] ± [15.29893 15.57063]
Principal Point (P)	[163.24408 61.97768] ± [15.39138 16.03065]

Skew (τ)	[0.00000] ± [0.00000] → 90.00000 ± 0.00000 degrees
Distortion (κ)	[-0.18502 -0.01664 -0.00823 -0.00111 0.00000] ± [0.06745 0.28411 0.00658 0.00366 0.00000]
Pixel error	[0.20156 0.22546]

Table 5-3. Right Camera Calibration. The data contained in this table is the intrensic parameters of the right camera.

Calibration Results – Right Camera	
Focal Length (f) (px)	[385.08865 387.90122] ± [24.14876 24.10460]
Principal Point (P)	[172.53400 66.16932] ± [27.13033 23.83721]
Skew (τ)	[0.00000] ± [0.00000] → 90.00000 ± 0.00000 degrees
Distortion (κ)	[-0.21678 0.14059 -0.00465 -0.00237 0.00000] ± [0.09099 0.18762 0.00894 0.01076 0.00000]
Pixel error	[0.19733 0.22429]

Table 5-4. Stereo Calibration. The results of the stereo calibration which combines the individual calibrations and determines the extrinsic parameters. The Rotation matrix is the Rodrigues of the rotation vector. These are computed for the right camera with respect to the left camera.

Calibration Results – Stereo Pair	
Rotation vector	[0.01243 0.00477 -0.00358] ± [0.01170 0.01586 0.00140]
Translation vector (mm)	[-268.93925 0.51587 -16.57751] ± [5.05666 3.08070 18.60634]

The stereo camera calibration can be seen in Figure 5-8. It shows the planes created by the checkerboard and the origins of the camera as well as their relationship to each other.

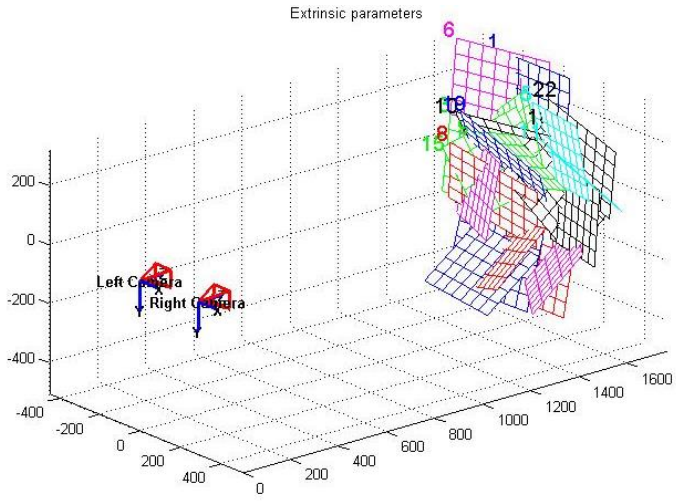

Figure 5-8. Stereo Camera Extrinsic View. The calibration planes and the cameras can be visualized. These are results of the complete stereo camera calibration.

Some more of the results of the camera calibration can be visualized in Figure 5-9, which shows the complete distortion model for each camera. This includes the effect of the **κ** vector, which contains both radial and tangential distortion.

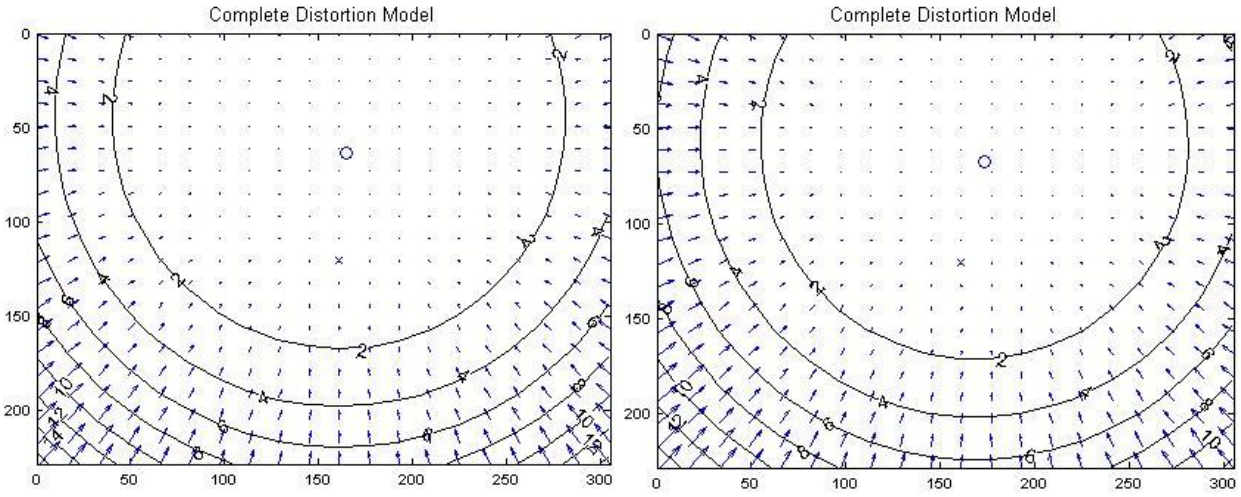

Figure 5-9. Distortion Models. (Left) The left camera distortion model. This contains both the radial and tangential distortion. (Right) The distortion model for the right camera. These figures are generated by the MATLAB Calibration Toolbox [71].

The pixel error from the calibration process can be seen in Figure 5-10. Each color represents the corner point from each checkerboard in each calibration image. It is clearly visible that one of two of the images were challenging for the calibration function to identify. This resulted in a higher error for those images. However, the average error was between 0.19 pixels and 0.22 pixels for both cameras.

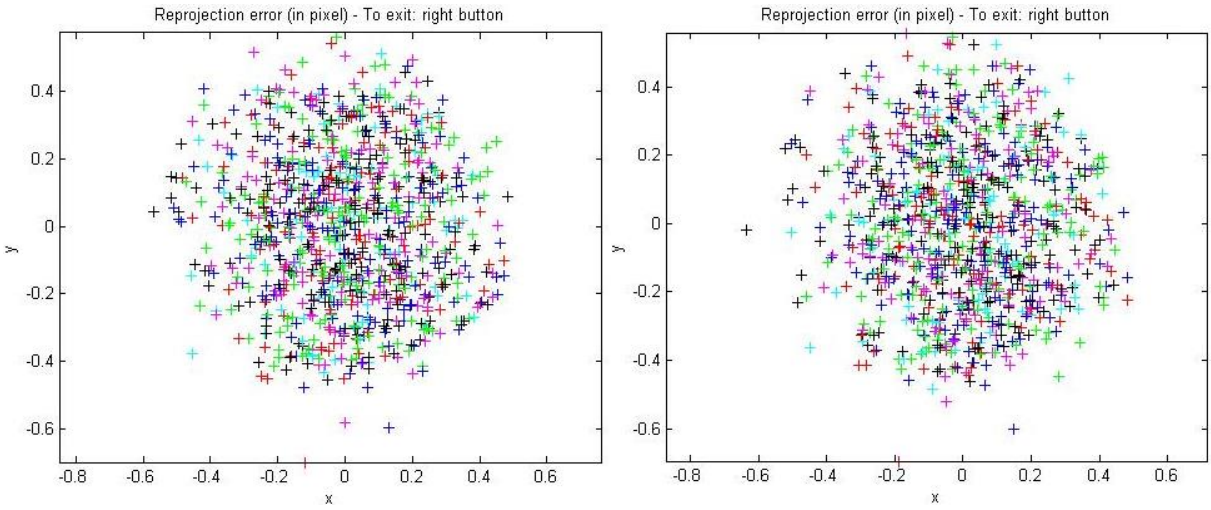

Figure 5-10. Calibration Pixel Error. (Left) This image shows the distribution of pixel error on the left camera calibration. (Right) The distribution of pixel error on the left camera calibration. The pixel errors were consistent between the cameras and across the calibration images.

The application of these parameters to the source images of the cameras takes the form of rectification. Practically, rectification removes distortions in the picture caused by the lenses and corrects any alignment issues with the two cameras. Since the cameras have been effectively modeled, both intrinsically and extrinsically, the stage is set for the next phase of the process.

5.2.2 STEREO MATCHING

The application of the camera parameters found in the previous section takes the form of stereo correlation. As detailed in Chapter 3, stereo vision uses a matching algorithm to relate points in 3D space that are viewed by each camera. The difference in the pixel location of these points translates into a disparity measurement. The disparity is turned into a depth measurement using the extrinsic camera parameters.

In order to tune the stereo matching for each situation, the software GUI allows the user to change several of the key parameters mentioned in Section 3.1.3 while several others remain fixed at the ideal values. The pre-filter has a cap of 31 and an adjustable window size. The cap is the recommended value of the OpenCV function. Once the pre-filter values have been set, the core stereo parameters can be established. The first of these is the maximum number of disparities. This value sets the search range of the stereo correlation search. It ranges from 16 to 96 in increments of 16. The step size is defined by the OpenCV function. The minimum search range is set to 0 since the relation of the cameras to each other is known. The next parameter is the correlation window size, or SAD window size. This value can be odd numbers from 3 to 31. This scale provides a large range of effects due to the window size while ensuring that the most prominent data is not lost at the larger sizes. The next set of variables focus on the stereo post-filtering. These values consist of a texture threshold, uniqueness ratio, and the speckle filter. The texture threshold values range from 3 to 21. The parameter removes areas of low texture from the stereo matches. This is essential because SAD correspondence accuracy is less than ideal in low texture areas. The next portion of the post-filter is the uniqueness ratio. This input can be adjusted from 5 to 15. This filter is a key factor in removing bad matches from the stereo calculations. It removes pixels that have largely different disparities to adjacent pixels. The final filter is the speckle filter. This filter was integrated into

the GUI, but was not utilized because the effect was not beneficial to the process conducted in these experiments. It, also, is largely similar in outcome to the uniqueness ratio. For descriptive images of the realization of these parameters with in the GUI, see Appendix A.1.

The results of the stereo matching can be adjusted using the parameters described in the previous section. These factors can be changed depending on the environments, lighting, and objects in the data collected. Some examples of the stereo matching can be seen in Figure 5-11.

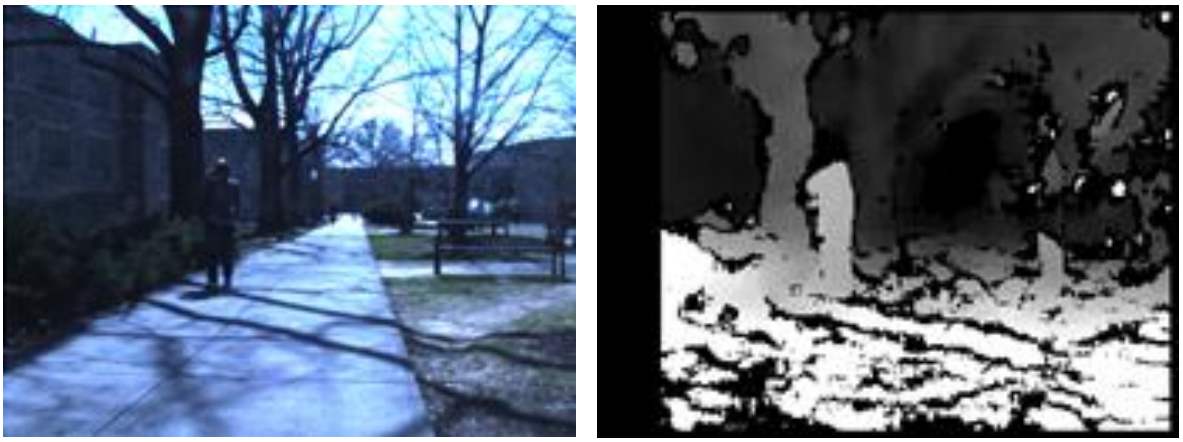

Figure 5-11. Stereo Matching. This figure shows the source visual image and the normalized stereo map. In the disparity map, the lighter colors indicate closer points and the black points are non-matches.

The ideal combination of the stereo parameters is fluid; however, in practice, an optimum configuration can be found. Although an adaptive control for the stereo parameters falls outside this thesis, it is discussed in Section 0. In general, the best parameters were tuned as followed. The pre-filter window size was set to 31 pixels. This produced a good base for the stereo correspondence. The number of disparities was set to 32. This proved to provide an adequate range to find the matches in the scenes under which the camera is designed to operate. The following calculations show the effect of the setting described previously. Equation (5.1) shows the theoretical minimum and maximum distances that are possible with the experimental setup.

$$Z_{min} = \frac{f}{\left(\frac{d_{max}}{B} + b\right)} = 3.10m \qquad (5.1)$$

$$Z_{max} = \frac{f}{\left(\frac{d_{min}}{B} + b\right)} = 98.99m$$

The disparity error function from Equation (5.1) is plotted over the range of possible depths from the previous equations. The disparity errors range from ± 0.0124 m at the minimum range to ± 12.61 m at the maximum range. These results can be seen in Figure 5-12.

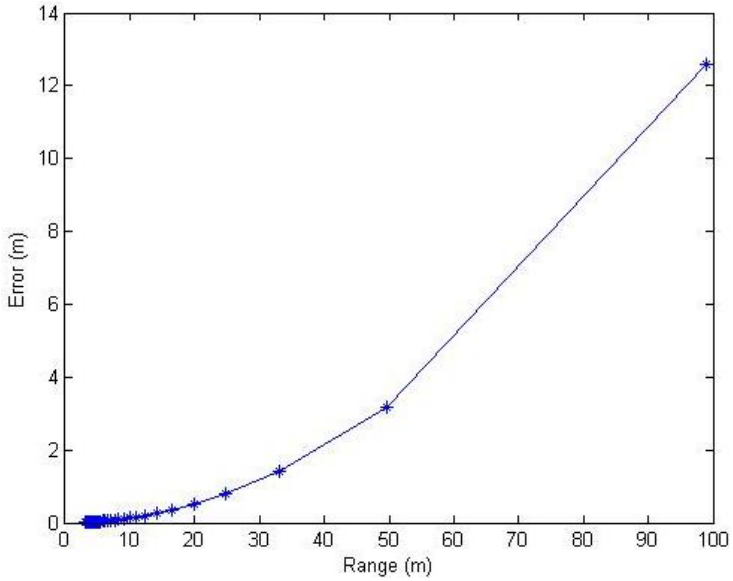

Figure 5-12. Disparity Error. The disparity error from the minimum to maximum range and the associated error.

The next stereo parameter is the SAD window size. This value has a large effect on speckle noise in the disparity map. As the size of this window is increased, it reduces this noise significantly. However, if this window is too large it can distort and obscure important features in the scene. The balance of this parameter was found to rest at a window size of 13 pixels. The final important parameter was the uniqueness threshold. This value balanced the SAD window size. It is one of the post-filtering techniques that were applied to the stereo images. By setting the uniqueness threshold value at 15, the noise left by the smaller SAD window was mostly eliminated while preserving the smaller details in the images. To test the ability of the stereo system to measure distance, a set of trials were conducted to validate the model. In order the measure the distance that the camera system had moved fixed reference point was chosen. This reference is the tree in the center of the frame shown in Figure 5-13. The disparity measured for the tree was 12. The calculated distance was 8.28 $\pm$0.088 m. The measured distance was 8.5 m. This shows a percent error of 2.59 % at this range. Much of this error is due to the quantization of the stereo disparity due to a limited bit depth of the images.

Figure 5-13. Stereo Depth Quantification. (A) The source image. (B) The resulting disparity map. The tree in the center of the frame was measured to have a disparity of 12 and a distance from the cameras of 8.5 m.

5.2.3 SLAM

As detailed in Section 2.3.1, the process of SLAM can take many forms; however, the process that was used for these experiments was detailed in Section 3.2. The process of converting the images to 3D point clouds started with the disparity maps. These disparity maps are loaded from memory and converted to point clouds using the math described in Chapter 3. Once the disparity maps are converted into point clouds, these clouds are filtered to make them ready for the registration process of SLAM. The prefiltering process consists of two main steps. The first is the statistical outlier removal filter. This filter has two input parameters: the mean K value and the standard deviation. The mean K value ranges from 1 to 100. This range covers all of the needed values for this parameter. The ideal value was found to be to 20. The standard deviation has a default value of 1.40 and a range of 0.00 to 3.99. These values can be adjusted to better account for noisy data, but the values described above proved to be quite satisfactory. The next step in the pre-filtering process is the voxel grid filter. This process divides the entire point cloud space into cubes. It, then, finds the centroid of all the points that are within this cube and reduces the data to just that one point. This effectively down samples the data while preserving the accuracy of the representation. The parameter for this portion of the pre-filter is called the leaf size. It defines the size of the cube that is used for the voxel grid. The range for this variable is 0.1 to 10.0. The ideal value, depends on the setting in which the data is taken; however, a good value was found to be 1.5 cm. The results of the pre-filtering can be seen in Figure 5-14. The statistical outlier removal filter is very effective in removing the random noise that remained in the stereo images and the voxel grid down sampling was very useful in preserving the content of the data while reducing the number of points to increase the speed in the following steps. This process assists in the SLAM registration in the next step.

Figure 5-14. Point Cloud Filtering. The first image is the source color image. This is projected onto the left point cloud is the raw unfiltered point cloud. The one on the right is the pre-filtered point cloud.

The next stage in the SLAM process is the point cloud registration. This is the primary function of SLAM. As such, it is the most computationally intensive. As with the previous steps, the registration is implemented using the algorithms described in Section 3.2.1. The point cloud registration takes two sequential filtered point clouds and compares them. It uses the SAC-IA algorithm to find the optimal transformation between the two data sets. To initialize this process, the FPFHs must be calculated. This begins by using a k-d tree search algorithm to find each point's nearest neighbors. This search is conducted on the 10 nearest tree branches. Next, the point normal vectors must be calculated. This is done using native PCL functions based on the results of the k-d tree search results. Once these steps have been accomplished, the FPFHs can be computed. The only parameter in this function is the search radius for the FPFH algorithm. The range for this factor is 1 to 15. The best value for the search radius was found to be 5. This balanced the computation time and the accuracy of the result. The search radius value could be increased in future applications by using a GPU to calculate the registration. All of the distances in the point cloud manipulation are in centimeters. After setting the search radius, there are some final parameters that need to be set before the registration algorithm can be run. The main parameters that need set are the minimum and maximum search distances for the SAC-IA function. These variables set the bounds on the search for the best match of point clouds. The minimum is set to 0 cm and the maximum is set to 45 cm, which is the distance traversed by a vehicle traveling 10 mph in 0.1 s. This is the estimated bound for the application of this system. For examples of this process, see Section 5.2.4.

Once the FPFHs have been found for each source data set, the clouds can be registered to each other. This step follows the SAC-IA procedure laid forth in section 3.2.1. Each point cloud is registered to the previously registered data, until all of them have been registered. The result of each registration process is a transformation matrix that contains the information to rotate and translate the second data set to align with the first data set. For the implementation of the SAC-IA function, 200 iterations were used for to converge to the solution. The other parameters were the minimum and maximum distances. These values put bounds on the search region for the registration algorithm. These values can range from 1 to 50 cm. Each time the registration is found the second, registered, point cloud is added to the first. This process is repeated until all of the point clouds in the data set are registered to the original data set.

The final phase in the registration process is to filter the final results. This is done by applying a MLS filter that aligns the surface normals to even out inconsistencies in the results of the registration process. The second phase of the post filtering is to conduct another voxel grid down-sampling. This condenses portions of the data that overlap, so that they have a consistent density.

5.2.4 SCALED EXPERIMENTS

In order to validate and test the procedure from Chapter 4, a set of experiments was designed to explore the features of this method. These experiments can be broken down into two main features. Originally, indoor testing was planned, but the range of the stereo system made indoor testing impractical. Thus, the first experiment is small- scale outdoor testing. The small- scale tests were designed to test the individual steps of the algorithm one at a time, to best gauge the individual performance of these steps. The next phase is the full- scale out door testing. This experiment was designed to examine the performance of the complete system.

5.2.4.1 Limited Outdoor Setting

The small- scale outdoor testing was conducted to examine, primarily, the ability of the SLAM portion of the algorithm. The portions of this experiment are a test of the registration process for rotation and a test of the registration for translation. These trials were conducted in a generic outdoor setting that contained common features that are expected for this unstructured UGV application.

The first test was a test of the ability of the SLAM algorithm to handle translation. The cameras were positioned and set of data was captured. The cameras were then moved a known distance. These two pieces of data were run through the SLAM portion of the algorithm. The cameras were moved a known distance with a simple scene being viewed. The SLAM algorithm matched the before and after measurements to achieve an estimate of the distance traveled by the cameras. The averaged results of this process were an estimate of 158.5 cm along the Z-axis. The actual motion was 66.0 cm. This results in a percent error of 40%. This error is high; however, it can be attributed to the stereo quantization error and effects that this has on the matching algorithm. The results of this experiment can be seen in Figure 5-15.

Figure 5-15. Translation Experiment. (A) This frame shows the initial point cloud. (B) This figure shows the second point cloud. (C) This is the registered output of the two clouds.

The second small- scale test was the rotation test, shown in Figure 5-16. This was designed to investigate the effectiveness of the SLAM registration to calculate the correct angle of rotation. The procedure for this test was similar to the translation test. A set of data was captured from a known orientation and then the cameras were rotated to a known angle. The cameras were positioned and set of data was captured. The cameras were then rotated a known angle. These two pieces of data were run through the SLAM portion of the algorithm. The SLAM algorithm matched the before and after measurements to achieve an estimate of the angle rotated by the cameras. The averaged results of this process were an estimate of 40.14 degrees about the Y-axis. The actual motion was approximately 45 degrees. This results in a percent error of 0.12 %.

Figure 5-16. Rotation Experiment. (A-B) These are the source images from the two different perspectives. (C) This is the point cloud from the first view point (D) and this is the combined point clouds. Some of the depth resolution is lost due to the incomplete overlap of the source point clouds.

5.2.4.2 Full-Scale Outdoor Setting

The second phase of testing involved more complex motions of the sensors to test the complete functionality of the system. This portion of the experiment was also comprised of two sections. The first is a test of the SLAM system moving along a linear path. This tested the software's ability to match multiple frames of data that contain mainly the same information. The results of this process were encouraging. The average fitness score or squared Euclidean distances for these data sets were around 0.087. The visualization of the data also showed that the sequential images were aligned well, Figure 5-17.

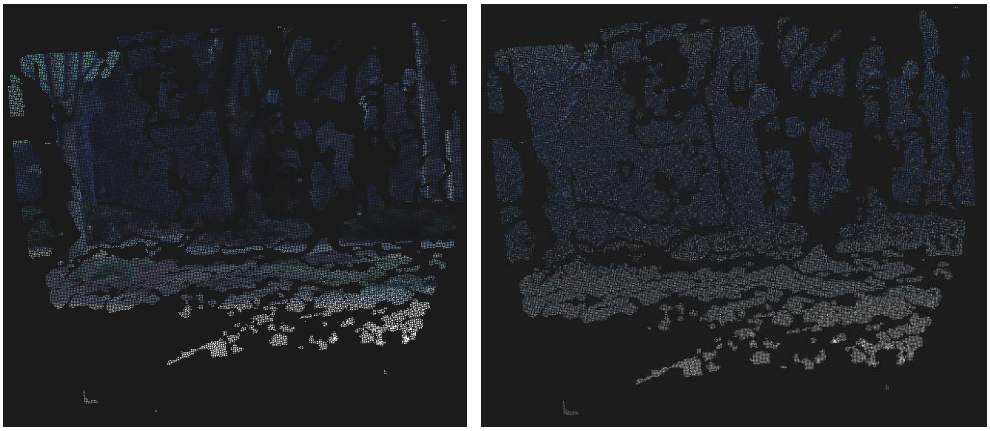

Figure 5-17. Linear SLAM Test. (A) Registered series of 10 point clouds along a roughly linear path of motion. (B) The post-filtered point cloud of the same source data.

The second phase of the full- scale outdoor testing involved a taking the sensors in a loop. This path combined both translation and rotation into a more realistic scenario. These tests were conducted in the same environment as the small scale tests, but with more motion to view more of the scene. The average fitness score was 0.524. Some of the visualized results can be seen in Figure 5-18.

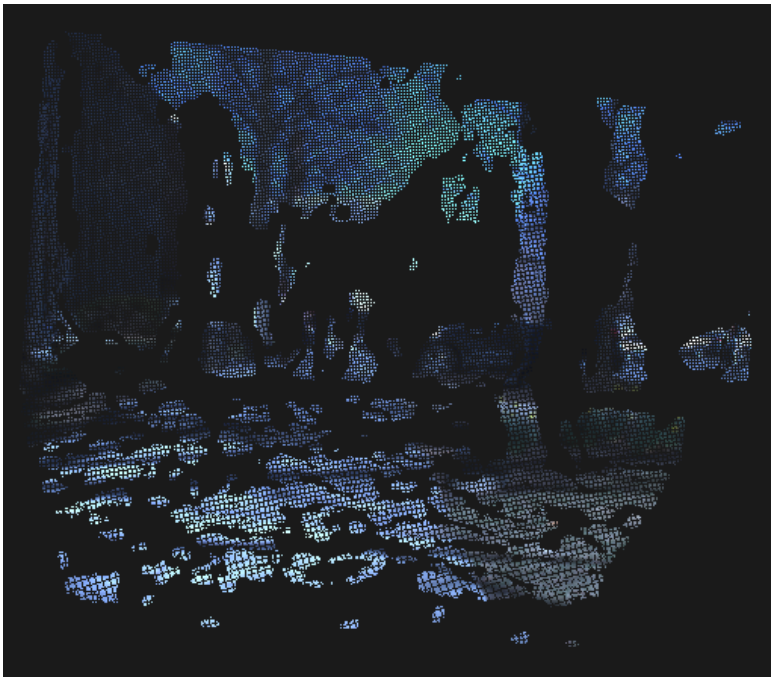

Figure 5-18. Full Scale Experiment. This figure shows the registration of 100 frames of data covering a portion of the loop taken in the Randolph quad. The side of the building as well as the sidewalk and the trees in the quad can be easily distinguished.

Both the small- scale and full- scale tests produced effective results. The small- scale tests showed that the system is fairly accurate, especially the rotation portion of the registration process. The full- scale tests, also, showed promise. The tests can be expanded in further unstructured environments and the perception system should be tested on a UGV platform. These and other ideas are discussed further in Section 0. Overall, the scaled experiments proved to be an effective test of the performance of the system's ability to create 3D world models by combining series of algorithms to process the data from a passive perception system.

CHAPTER 6

6 CONCLUSIONS AND FUTURE WORK

This is the final chapter of this thesis. It covers the conclusions derived from the previous chapter's results and discusses the possibilities for the continuation of this research as it applies to the application of this passive perception system to UGV related 3D scene reconstruction.

6.1 CONCLUSIONS

The entirety of the work encompassed in this thesis was compiled as a method by which broadband passive perception can be applied for the purposes of world modeling and scene reconstruction. This process was designed for applications related to UGVs and to replace or augment their current perception systems. The scope of this research was to apply current technologies in the areas of stereo correspondence and SLAM registration to build a model of the environment through which the UGVs travel. The first phase of the work was the implementation of a stereo vision system that could capture broadband information and preform the stereo correspondence problem. The resulting data from this process was directed into the SLAM portion of the algorithm, where it was rendered into a world model. Each of the steps in this process were implemented and tested. There were many lessons learned through this process, which are detailed in the following sections.

6.1.1 STEREO

The process of creating a stereo vision system involved defining individual camera models and relating them to each other to define the stereo camera system. This model was derived from a calibrations process that used data captured from the system of a known target to calculate the parameters for that system. Using the software system, a software interface link was created to record the visible and NIR images from the cameras. This software architecture was also used to compute the stereo correspondence using a block matching method. The disparity maps were then translated into 3D point clouds for the next phase of the process.

The results from this portion of the thesis were impressive. The stereo correspondence algorithm was able to process the incoming images in real time at 10 Hz. This rate allowed for a 45 cm maximum range of motion from the expected UGV rate of travel at 10 mph. In the primary sensing distances of the stereo system, it showed reasonable accuracy. At a mid-range distance of 8.5 m, the system exhibited an error of 2.9%. The translation of the depth information from a disparity map to a point cloud produced impressive results. The broadband information was also reintegrated in this step to allow for creation of broadband world models. The filtering techniques applied to the data proved to be very effective in removing almost all of the noise from the stereo correspondence method. The post- filtering processes were particularly helpful.

During this process, several issues had to be overcome. In the process of data collection, the camera's SDK proved to be poorly documented and did not work well with several of the other libraries needed to conduct the research. This issue was overcome by splitting the software into two programs to minimize the interaction between these sections of code. The second issue was related the error in the stereo matching. This was somewhat quantified in Section 5.2.2; however, this issue did not produce the desired

accuracy, especially at longer ranges. This also constrained the possible disparities into 32 planes. This caused more issues when trying to register the data in the SLAM procedure. In further work, this can be addressed by utilizing the sub pixel matching to provide a smoother transition between objects of different depths and to reduce the significant quantization error.

Overall, the process of matching features in the images from two stereo cameras was effective. It proved to be fast enough for the desired application and showed adequate accuracy. The disparity maps were effectively transformed into 3D point clouds and colored using the source data. The finished product from this phase of this thesis was then relayed to the second main phase.

6.1.2 SLAM

This section of the thesis, like the previous section, contains several steps that were investigated individually and corporately. It began with the pre- filtering of the individual point clouds from the stereo step. These filtered point clouds were then delivered to the feature extraction portion of the SLAM procedure. This function identified unique features using the FPFH model. These distinctive features were used to register the sequential point clouds to the global map. The results were post- filtered to smooth the resulting global map.

The results of this portion of the process were tested in several steps. Two scaled tests were conducted to test the rotation and translation portions of the SLAM registration algorithm. The rotation experiment showed very good results with a percent error of 0.12%. The translation experiment showed less promising results with a percent error of 40%. As mentioned in the previous section, much of this error can be attributed to the quantization error of the stereo and the issues that these defined planes cause the registration process. The creation of global maps was also a success. In larger data sets, the computation time was a hindrance; however, the models that were created represented the original world in a format that a UVG could interpret.

In the implementation of these methods, several issues were also encountered. The first of the difficulties was the intense computational nature of this process. Each image, taken 10 times a second, contains 76,800 pixels that are translated into 3D point clouds. Even though these clouds are down sampled, there are still an enormous number of data points, especially in a data set captured over a longer period of time. Because of the size of this data, the processing requirements are very large. As the data size increases, the processing time increases exponentially. To address this, the software was run on an AMD quad core desktop with 8 GB of RAM. In the future, this process can be shifted to a GPU to increase the speed to real time levels. For further exposition on this idea, see Section 6.2.2. As mentioned above, the size of the global maps proved to be inhibitive. This issue can be addressed by more critically selecting the feature points and by creating semi-global maps that are registered in a separate process. ADD

In summary, this portion of the algorithm performed satisfactorily with the purpose of implementing current component algorithms for 3D scene reconstruction with a broadband passive perception system. The results were analyzed and discussed. Issues that were identified in the process of implementing the theory set forward in chapter 3 were addressed. Future work that could continue and expand the current progress is discussed in the following sections.

6.2 FUTURE WORK

The work covered in this thesis is only a small portion of this field. There are many remaining opportunities to continue and extend this research. These range from applications of the passive perception broadband scene reconstruction methodologies on current UGV platforms to continued research and refinement of the current algorithms.

6.2.1 ADDITIONAL BROADBAND DATA

The images collected in the work described in this thesis was limited to the visible and near infrared spectrums. Although this information can be very useful for both navigation and classification, there are many other sources of possible information such as the data obtained from LWIR and SWIR images. This additional information can be used to allow for vehicle navigation in low light situations. It also provides more information that can be used to improve the classification algorithm's certainty. The classification data can also be displayed along with the source data and used to inform the autonomy algorithms. Other sources of information could include INS data to increase the accuracy of the registration process. Overall, this additional data could be used to expand the capability of the system and increase its accuracy.

6.2.2 ALGORITHM IMPROVEMENT

The next area that can be improved is the registration process. There are many methods that were discussed in Chapter 2. There are portions of these algorithms that could be implemented in the registration process to improve both speed and accuracy. The concept of local and global mapping will be vital for the creation of large maps. By splitting the map into smaller portions, the computation power and memory taken by these maps can be contained. This process also allows to the reduction of global error as it is reset at the start of each new map. Also, as the libraries that were used to calculate both stereo correspondence and map registration improve, the functions can be implemented on a GPU. This processor is optimized for these types of parallel operations. By running these computations on a GPU, the speed of processing can be greatly increased. This speed increase will make these algorithms applicable for UGV integration. Another possible route for expansion of this project, is to automate the control and adjustment of the camera and stereo parameters. This could be realized by creating a mapping function to adjust these parameters based on the exposure levels in the images and environment that is being viewed at the current time. This process would improve the accuracy and reliability of those portions of the algorithm.

6.2.3 UGV INTEGRATION

One of the main areas in which this thesis can be continued is in the application onto UGVs. The scope of the work described above is focused on algorithm merging. However, these results can be directly integrated onto current UGVs. By having the 3D scene information, a robot can plan a path and navigate through its environment. In addition to 3D map of the environment, the broadband data is used to classify the objects observed by the perception system. This added classification information will improve the ability of the autonomy to guide the vehicle.

Overall, a significant amount of work remains to be done in this field. By implementing the changes and additions listed above, this system can be greatly improved, making it an ideal system for applications on UGVs in unstructured environments.

REFERENCES

[1] "Update: UGV history," *Defence and Freedom*, 2009. [Online]. Available: http://defense-and-freedom.blogspot.com/2009/02/update-ugv-history.html.

[2] M. Kovac, "Self Deploying Microglider," 2009. [Online]. Available: http://lis2.epfl.ch/CompletedResearchProjects/BiomimeticJumpingMicroglider/.

[3] "DARPA Urban Challenge," 2007. [Online]. Available: http://archive.darpa.mil/grandchallenge/.

[4] "The Growing Urbanization of the World," *Columbia University*, 2005.

[5] G. Muscato, "UAV and UGV for Measurement and Exploration in Volcanic Environmnets," *DIESS University*, 2010. [Online]. Available: http://www.drexel.edu/~/media/Images/coe/News/Events/NATO Presentations/muscatoNato2010s.ashx.

[6] "BigDog," *Boston Dynamics*, 2013. [Online]. Available: http://www.bostondynamics.com/robot_bigdog.html.

[7] "DARPA Grand Challenge," *Wikipedia*. .

[8] C. Reinholtz, B. Leedy, J. Putney, J. M. Webster, and C. Bauman, "Virginia Tech Grand Challenge Team DARPA Grand Challenge 2005," 2005.

[9] A. Wicks, B. Leedy, A. Blumer, R. Wilhelm, and C. Bauman, "Virginia Tech Team Rocky DARPA Grand Challenge 2005," 2005.

[10] L. Crumbley, "Virginia Tech's Cliff and Rocky selected for Grand Challenge starting line," 2005.

[11] E. Svoboda, "PopSci's Darpa Grand Challenge Preview: Update #3," *Popular Science*, 2005. [Online]. Available: http://www.popsci.com/scitech/article/2005-10/popscis-darpa-grand-challenge-preview-update-3.

[12] "VictorTango wins $500,000 with an SUV that thinks like a driver," *Virginia Tech*, 2007. [Online]. Available: http://www.vt.edu/spotlight/achievement/2007-10-29_victortango/2007-10-29-victortango.html.

[13] C. Reinholtz, T. Alberi, J. Farmer, S. Frash, C. Terwelp, and A. Wicks, "DARPA Urban Challenge Technical Paper," 2007.

[14] "US Navy Marines Test Autonomous Unmanned Ground Vehicles," *US Navy*, 2010. [Online]. Available: http://www.defencetalk.com/us-navy-marines-test-autonomous-unmanned-ground-vehicles-27996/.

[15] "Ground Unmanned Support Surrogate, Optionally Unmanned & Autonomous Vehicles, Lighten the Load for Dismounted Warfighters," *TORC Robotics*, 2010. [Online]. Available: http://www.torcrobotics.com/case-studies/ground-unmanned-support-surrogate.

[16] "Perceive," *The Free Dictionary*, 2013. [Online]. Available: http://www.thefreedictionary.com/perceive. [Accessed: 05-Mar-2013].

[17] J. Johnson, "Analysis of Image Forming Systems," *SPIE-The International Society for Optical Engineering*, vol. 513, 1958.

[18] "Velodyne LIDAR," *Velodyne*, 2012. [Online]. Available: http://velodynelidar.com/lidar/lidar.aspx.

[19] M. W. Spong and M. Fujita, "Control in Robotics," 2011.

[20] A. J. Dalton, "Autonomous Vehicle Path Planning with Remote Sensing Data Sensing Data," 2008.

[21] S. J. Cacciola, "Fusion of Laser Range-Finding and Computer Vision Data for Traffic Detection by Autonomous Vehicles," no. December, 2007.

[22] P. H. King and C. F. Reinholtz, "A Low Cost Localization Solution Using a Kalman Filter for Data Fusion A Low Cost Localization Solution Using a Kalman Filter for Data Fusion," 2008.

[23] "WHAT IS FAR INFRARED (FIR) ?," *SmartyHealth*, 2002. [Online]. Available: http://www.smartyhealth.com/fir.html.

[24] Y. Zhao, "Why are Prices Falling Fast ? An Empirical Study of the US Digital Camera Market," pp. 1–32, 2006.

[25] C. Caraffi, S. Cattani, and P. Grisleri, "Off-Road Path and Obstacle Detection Using Decision Networks and Stereo Vision," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 4, pp. 607–618, Dec. 2007.

[26] J. Weingarten, "feature-based 3d slam," vol. 3601, 2006.

[27] S. Latour, S. Droppelmann, M. Heuting, and M. van der Veen, "Stereo Vision using the OpenCV library," no. June, 2010.

[28] T. Peynot, J. Underwood, and S. Scheding, "Towards reliable perception for Unmanned Ground Vehicles in challenging conditions," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1170–1176, Oct. 2009.

[29] A. L. Rankin, A. Huertas, and L. H. Matthies, "Stereo-vision-based terrain mapping for off-road autonomous navigation," *Proceedings of SPIE*, vol. 7332, pp. 733210–733210–17, 2009.

[30] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin, "Terrain Perception for DEMO I11," no. Mi, 2000.

[31] H. Hirschmuller, "Improvements in Real-Time Correlation-Based Stereo Vision," 2001.

[32] A. Bradski, Gary; Kaehler, *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, Inc., 2013, p. 441.

[33] K. Kornyakov and V. Eruhimov, “Realtime Computer Vision with OpenCV,” pp. 1–17.

[34] W. van der Mark, F. C. a. Groen, and J. C. van den Heuvel, “Stereo based navigation in unstructured environments,” *IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No.01CH 37188)*, vol. 3, pp. 2038–2043, 2001.

[35] A. Broggi, C. Caraffi, P. P. Porta, P. Zani, and V. Ingegneria, “The Single Frame Stereo Vision System for Reliable Obstacle Detection used during the 2005 DARPA Grand Challenge on TerraMax TM,” pp. 745–752, 2006.

[36] M. Hebert, “Active and Passive Range Sensing for Robotics,” no. April, 2000.

[37] C. Strecha, D. Ettlingen, L. Van Gool, and C. Ethz, “On Benchmarking Camera Calibration and Multi-View Stereo for High Resolution Imagery,” 2008.

[38] S. Gehrke, K. Morin, M. Downey, N. Boehrer, T. Fuchs, N. W. Geomatics, and D. E. Models, “Semi-global matching: an alternative to lidar for dsm generation?,” 2008.

[39] L. M. Paz, P. Pini, and J. D. Tard, “Large-Scale 6-DOF SLAM With Stereo-in-Hand,” vol. 24, no. 5, pp. 946–957, 2008.

[40] V. Group, D. De Ciencia, D. Computaci, and J. M. S, “Underwater 3D SLAM through Entropy Minimization,” no. May, pp. 3562–3567, 2006.

[41] T. Bailey and H. Durrant-Whyte, “Simultaneous Localization and Mapping (SLAM):,” no. September, pp. 108–117, 2006.

[42] J. M. Sáez and F. Escolano, “6DOF entropy minimization SLAM for stereo-based wearable devices,” *Computer Vision and Image Understanding*, vol. 115, no. 2, pp. 270–285, Feb. 2011.

[43] H. Durrant-Whyte and T. Bailey, “Simultaneous Localization and Mapping : Part I,” 2006.

[44] U. Franke, C. Rabe, and S. Gehrig, “6D-Vision : Fusion of Stereo and Motion for Robust Environment Perception,” pp. 216–223, 2005.

[45] J. Mcdonald, C. Cadena, and J. J. Leonard, “6-DOF Multi-session Visual SLAM using Anchor Nodes.”

[46] R. Sim, P. Elinas, M. Griffin, J. J. Little, and C. Intelligence, “Vision-based SLAM using the Rao-Blackwellised Particle Filter.”

[47] D. G. Lowe, “Object recognition from local scale-invariant features,” *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pp. 1150–1157 vol.2, 1999.

[48] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF : Speeded Up Robust Features,” pp. 404–417, 2006.

[49] W. Brink and W. Brink, “Stereo Vision as a Sensor for EKF SLAM,” pp. 19–24.

[50] J. Weingarten and R. Siegwart, “EKF-based 3D SLAM for structured environment reconstruction,” *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3834–3839, 2005.

[51] G. A. Einicke and L. B. White, “Robust Extended Kalman Filtering,” *IEEE Transactions on Signal Processing*, vol. 47, no. 9, pp. 2596–2599, 1999.

[52] S. J. Julier and J. K. Uhlmann, “A New Extension of the Kalman Filter to Nonlinear Systems,” 1997.

[53] P. Besl and N. D. McKay, “A Method for Registration of 3-D Shapes,” *IEEE Transactions on Pattern and Machine Learning*, vol. 14, no. 2, pp. 239–256, 1992.

[54] M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Apphcatlons to Image Analysis and Automated Cartography,” *Communications of the ACM*, vol. 24, no. 6, 1981.

[55] T. Tajbakhsh, “3D Polygonal Representation of Dense Point Clouds by Triangulation, Segmentation, and Texture Projection,” *SPIE-IS&T Electronic Imaging*, vol. 7526, p. 75260C–75260C–8, Feb. 2010.

[56] B. A. Cohen, “Approximation by Greedy Algorithms,” vol. 40, no. 8, pp. 2–3, 2007.

[57] A. Üngör, “Computational Geometry: Kd-Trees and Range Trees,” 2009. [Online]. Available: http://www.cise.ufl.edu/class/cot5520fa09/CG_RangeKDtrees.pdf. [Accessed: 04-Jan-2013].

[58] S. Lefebvre, S. ; Hornus, and F. Neyret, “Chapter 37. Octree Textures on the GPU,” 2005. [Online]. Available: http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter37.html.

[59] “Voxel,” *Wikipedia*. .

[60] I. Wald and V. Havran, “On building fast kd-Trees for Ray Tracing, and on doing that in O(N log N),” *2006 IEEE Symposium on Interactive Ray Tracing*, pp. 61–69, Sep. 2006.

[61] “Point Cloud Library,” 2013. [Online]. Available: http://pointclouds.org/. [Accessed: 04-Feb-2013].

[62] D. M. . Mount and S. Arya, “ANN: A Library for Approximate Nearest Neighbor Searching,” 2010. [Online]. Available: http://www.cs.umd.edu/~mount/ANN/.

[63] M. Gopi and S. H. K. Rishnan, “A Fast and Efficient Projection-Based Approach for Surface Reconstruction,” pp. 0–7, 2002.

[64] B. Mederos, L. Velho, and L. H. De Figueiredo, “Moving least squares multiresolution surface approximation,” *16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003)*, pp. 19–26, 2003.

[65] N. J. Mitra and A. Nguyen, “Estimating surface normals in noisy point cloud data,” *Proceedings of the nineteenth conference on Computational geometry - SCG ’03*, p. 322, 2003.

[66] A. Abdelhafiz, B. Riedel, W. Niemeier, T. U. Braunschweig, and G. Str, “TOWARDS A 3D TRUE COLORED SPACE BY THE FUSION OF LASER SCANNER POINT CLOUD AND DIGITAL PHOTOS.”

[67] Y. Morvan, “Acquisition, Compression and Rendering of Depth and Texture for Multi-View Video,” 2009.

[68] S. Birchfield, “An Introduction to Projective Geometry (for computer vision),” 1998. [Online]. Available: http://vision.stanford.edu/~birch/projective/node1.html. [Accessed: 04-Jan-2013].

[69] “Pinhole Camera,” *Wikipedia*. 2013.

[70] S. Banerjee, “Pin-Hile Camera Model,” 2010. [Online]. Available: http://www.cse.iitd.ernet.in/~suban/vision/geometry/node3.html.

[71] J.-Y. ; Bouguet, “Camera Calibration Toolbox for Matlab,” 2010. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/.

[72] “Epipolar Geometry,” *Wikipedia*. 2010.

[73] S. Mattoccia, “Stereo Vision: Algorithms and Applications,” *University of Bologna*, 2012. [Online]. Available: http://www.vision.deis.unibo.it/smatt/Seminars/StereoVision.pdf.

[74] N. Kiryati, “Stereo - Computer Vision,” 2013. [Online]. Available: http://www.eng.tau.ac.il/~nk/.

[75] K. Konolige, R. Avenue, and M. Park, “Small Vision Systems : Hardware and Implementation.”

[76] J. J. Rodriguez and J. K. Aggarwal, “Quantization error in stereo imaging,” *Proceedings CVPR ’88: The Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 153–158, 1988.

[77] S. Ahuja, “How to select a proper Disparity Value for Stereo-Vision,” 2009. [Online]. Available: http://siddhantahuja.wordpress.com/2009/07/24/how-to-select-the-proper-disparity-value-for-stereo-vision/. [Accessed: 04-Feb-2013].

[78] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, “Towards 3D Point cloud based object maps for household environments,” *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, Nov. 2008.

[79] R. B. Rusu, N. Blodow, and M. Beetz, “Fast Point Feature Histograms (FPFH) for 3D registration,” *2009 IEEE International Conference on Robotics and Automation*, pp. 3212–3217, May 2009.

[80] A. W. Fitzgibbon, “Robust registration of 2D and 3D point sets,” *Image and Vision Computing*, vol. 21, no. 13–14, pp. 1145–1153, Dec. 2003.

[81] “JAI AD-080 GE Datasheet.” JAI, pp. 1–2.

[82] Itseez, “OpenCV,” 2013. [Online]. Available: http://opencv.org/.

[83] “Model View Controller,” *Wikipedia*. .

A.Additional Images

A.1 GUI Images

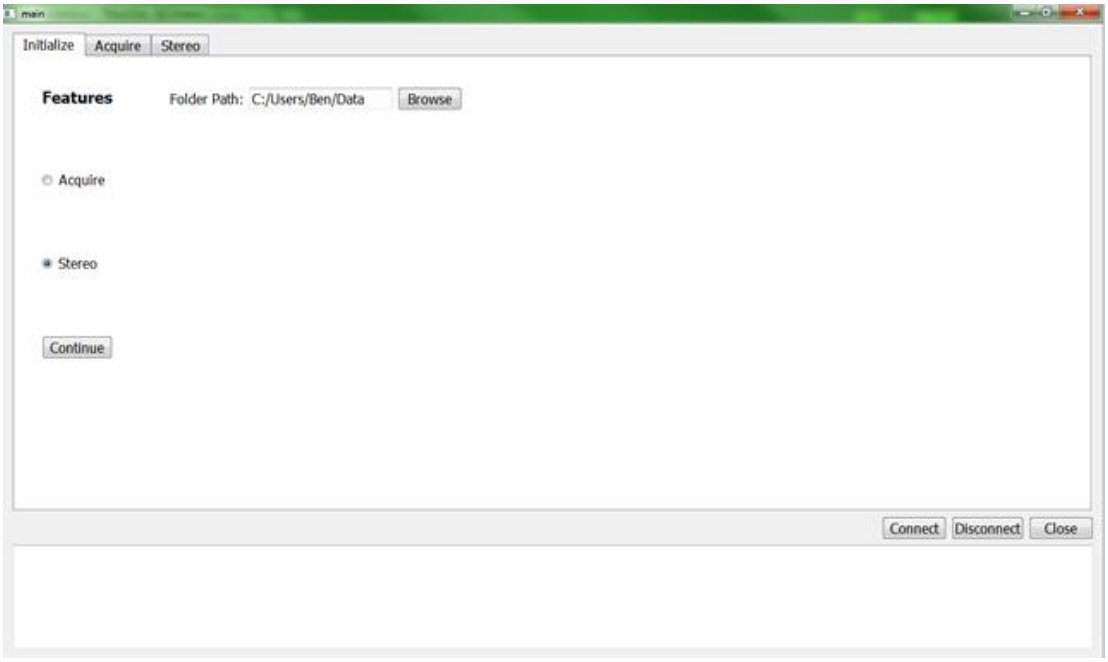

Figure A-1. Initialization GUI. This is a screenshot of the startup for the data capture software. There are tabs for each process in the program. This program can view the cameras and collect raw and stereo data. The user also specifies the path to which the files are saved.

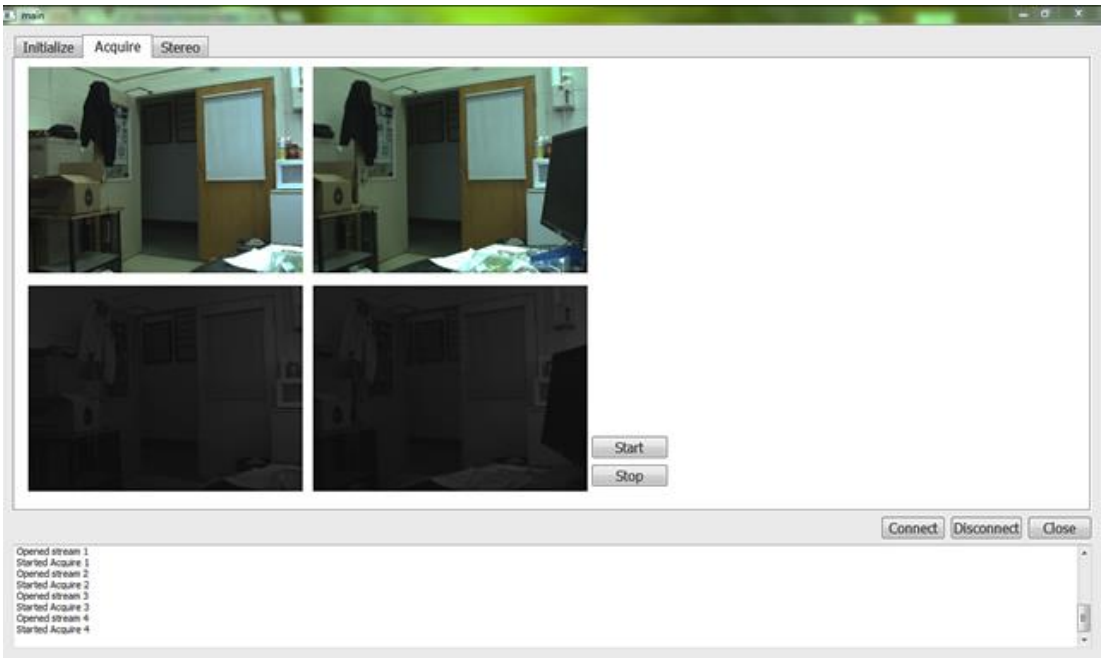

Figure A-2. Acquire Frame of the GUI. This is a screenshot of the acquire frame for the data capture software. Each of the four raw images can be seen.

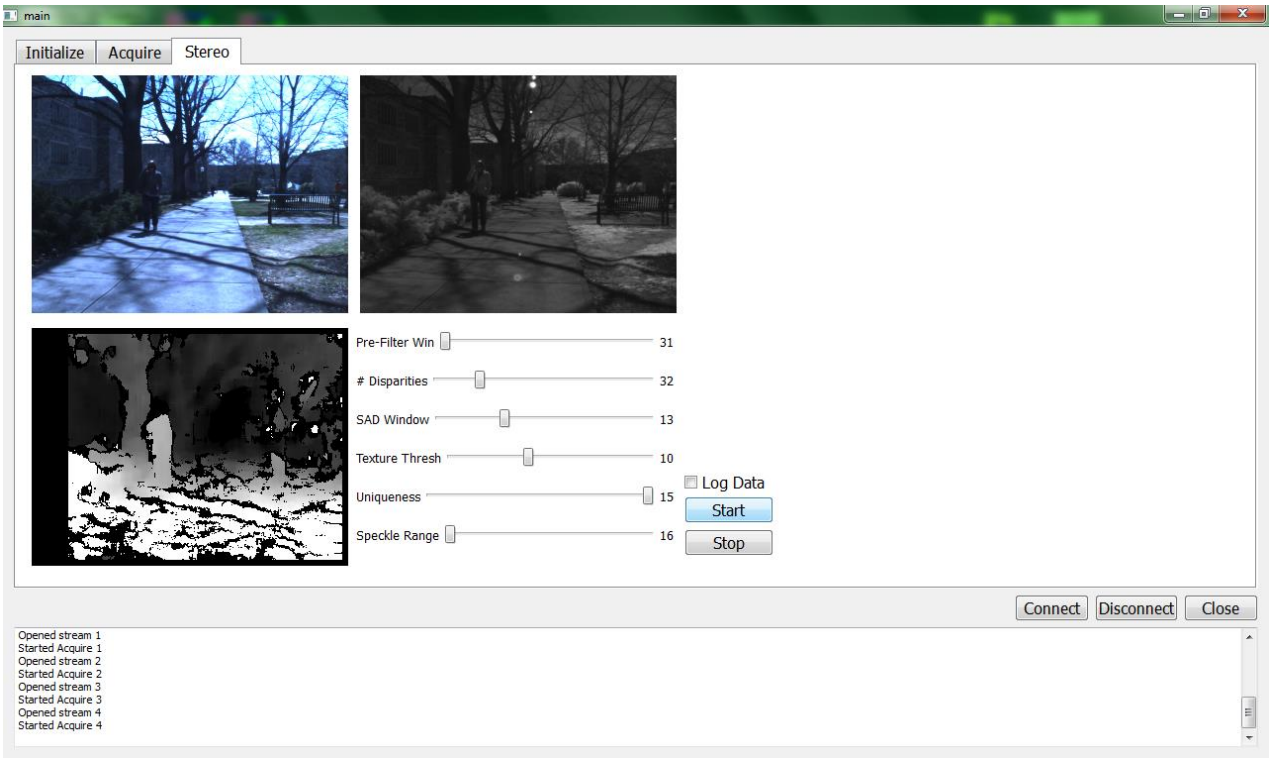

Figure A-3. Stereo Frame of the GUI. This is a screenshot of the acquire frame for the data capture software. Each of the four raw images can be seen.

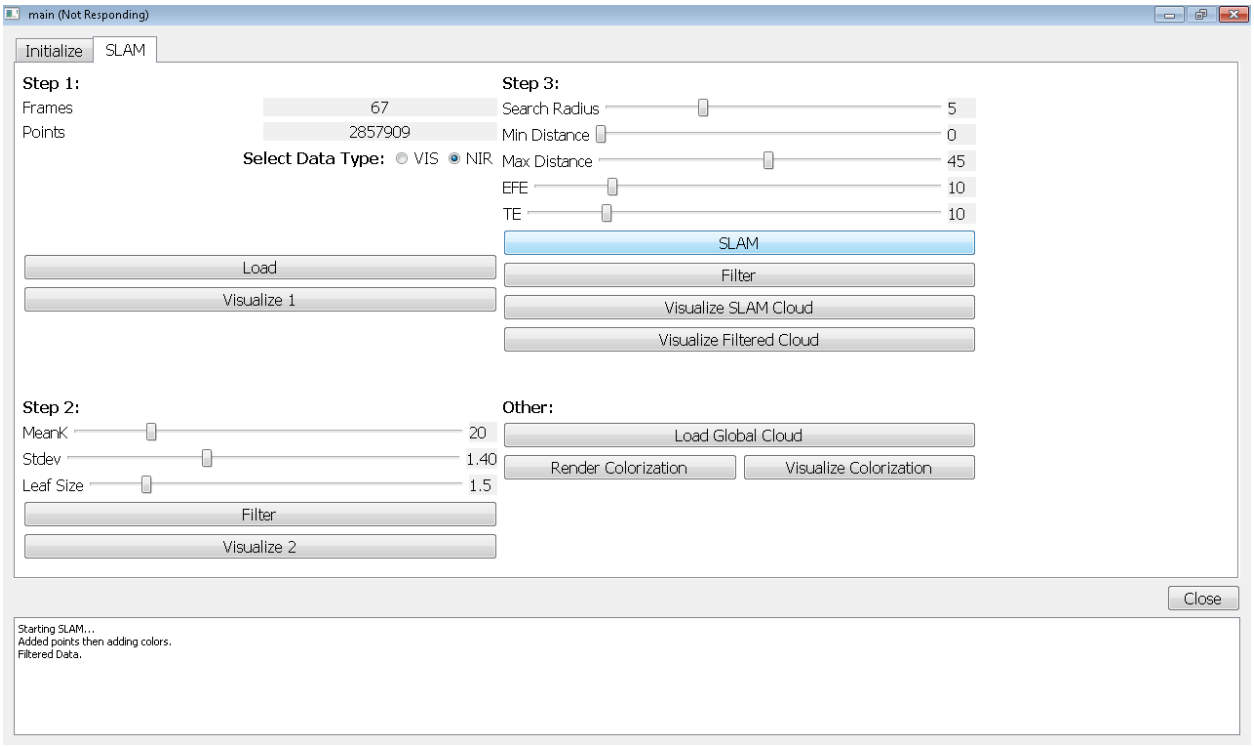

Figure A-4. SLAM and Visualization Frame of the GUI. This is a screenshot of the SLAM and vizualization frame for the data rendering software. There are four main steps that involve loading the data, prefiltering, SLAM, and meshing/texturizing.

B. Other Attachments

B.2 Flow Chart

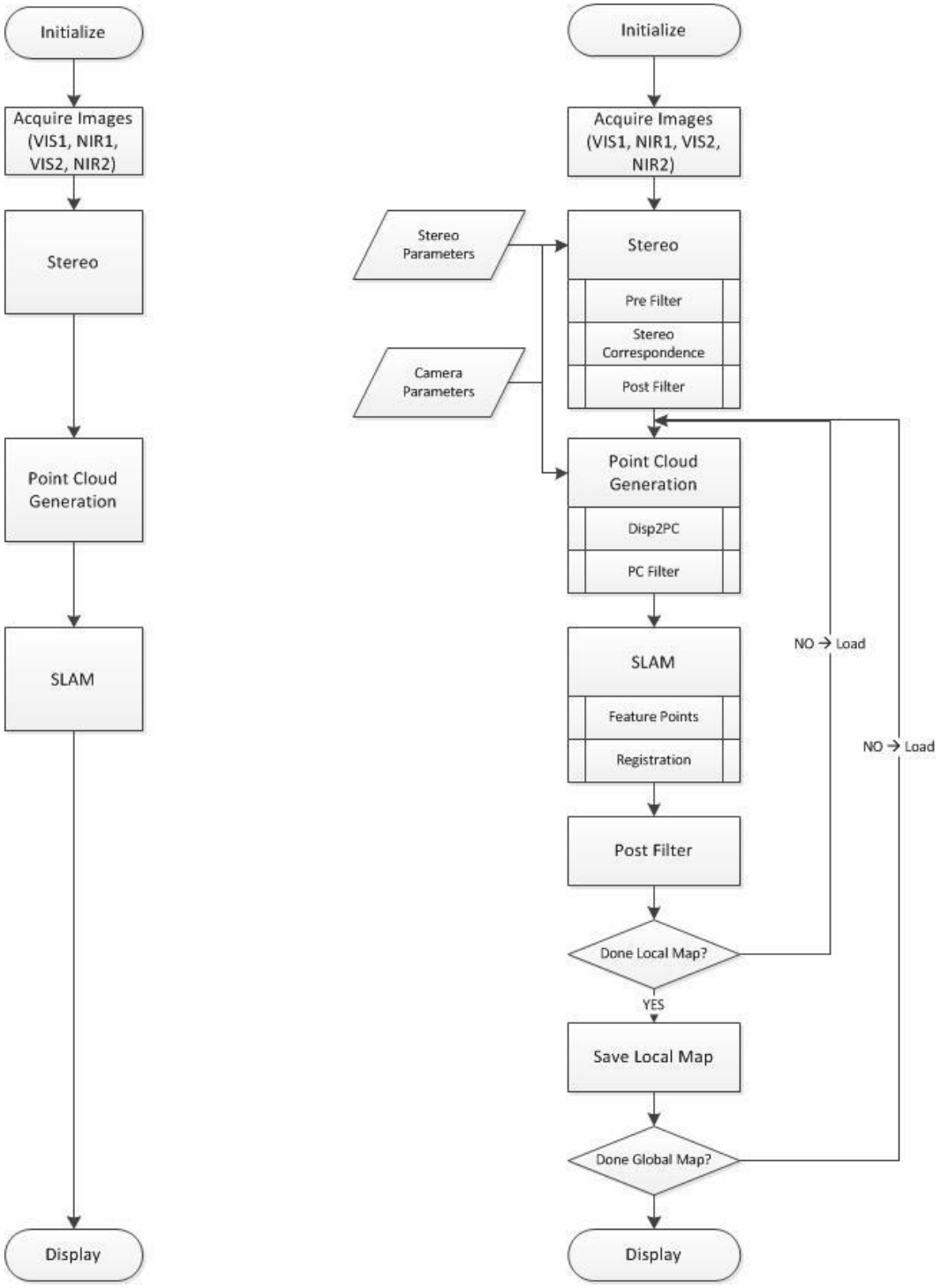

Figure B-1. Program Flow Chart (a) This is the simplified version of the algorithm. (b) The more complete diagram of the data flow.