Quantifying Aerial LiDAR Accuracy of LOAM for
Civil Engineering Applications

Derek Anthony Wolfe

A project submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of
Master of Science

Kevin W. Franke, Chair
John D. Hedengren
Daniel P. Ames

Department of Civil Engineering
Brigham Young University
April 2016

Copyright © 2016 Derek Anthony Wolfe
All Rights Reserved
ABSTRACT

Quantifying Aerial LiDAR Accuracy of LOAM for Civil Engineering Applications

Derek Anthony Wolfe
Department of Civil and Environmental Engineering, BYU
Master of Science

There is growing demand for remotely sensed data obtained through small Unmanned Aerial Vehicles (sUAVs). The use of aerial LiDAR has been incorporated as an alternative to current Structure from Motion (SfM) 3-D modeling. Using Linux to integrate ROS to connect the VLP-16 LiDAR with LOAM, near real time 3-D point cloud models are developed. This paper addresses the challenges of integrating different technologies and explores one current complete solution to using LiDAR with a UAV. The custom platform was tested in both indoor and outdoor environments. Control datasets were developed to help determine if mounting the sensor on a UAV causes too much distortion of the raw data. In general, the 3-D models attain accuracies within the 3 cm accuracy limitation of the sensor. Models not attaining 3cm accuracy were generated from outdoor environments where structure is not available for the LOAM to use as constraining features for point cloud registration. This paper serves as an instruction document outlining the operation and use of this tool. Future advances for this tool include creating a more user friendly application, as the current tool requires many small operations that have the potential to be automated.

Keywords: Unmanned Aerial Vehicle, UAV, Unmanned Aerial System, UAS, LiDAR, Laser Odometry and Mapping, Velodyne VLP-16, Robot Operating System, ROS, 3-D Modeling, Point cloud registration, SfM, DJI S1000
ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Dr. Franke as my committee chair, for his constant leadership, loyalty, schedule management, and confidence in me and this research effort. Dr. Hedengren and Dr. Ames, as my committee members, have provided encouragement, patient guidance, and useful critiques of my research. I would also like to thank the C-UAS industrial members for their interest in my research and for providing the necessary funding to complete this work. I appreciate Dave Duggins of Real Earth and Ji Zhang of CMU for their expert advice with regards to LiDAR and LOAM, Abe Martin of BYU for his constant assistance and profound problem solving capabilities, and Brandon Reimschissel of BYU for his resourcefulness in helping create a reliable platform, his aeronautics experience, and for the various flight tests that would not be possible without him.

My grateful thanks are also extended to Samantha Ruggles, Chris Bender, and Byron Yates as my fellow student researchers who assisted me with various miscellaneous tasks throughout the course of the project. I would also like to thank my parents and in-laws for their support and encouragement throughout my study. Finally and most importantly, I would like to thank my wife for her positive reassurances, enthusiastic support, patience, sacrifice, and constant love throughout the duration of this project.
# TABLE OF CONTENTS

List of Tables ......................................................................................................................... vii

List of Figures ......................................................................................................................... viii

1. Introduction ......................................................................................................................... 1
   1.1. Problem Statement ......................................................................................................... 1
   1.2. Objectives .................................................................................................................... 2

2. Background ......................................................................................................................... 3
   2.1. Remote Sensing ............................................................................................................. 3
   2.2. Unmanned Aerial Vehicles (UAVs) ............................................................................. 5
   2.3. 3-D Modeling ................................................................................................................ 7
   2.4. LiDAR ........................................................................................................................... 12
   2.5. LOAM .......................................................................................................................... 16
   2.6. ROS .............................................................................................................................. 17

3. Platform Selection and Specs ............................................................................................. 19

4. Accuracy Testing ................................................................................................................ 23
   4.1. Indoor Accuracy Testing ............................................................................................. 23
   4.2. Outdoor Accuracy Testing .......................................................................................... 30
      4.2.1. Fletcher Building Courtyard ................................................................................. 30
      4.2.2. Rock Canyon Park ............................................................................................... 34
4.2.3. BYU Football Stadium ................................................................. 36

5. Instructional ......................................................................................... 43

5.1. How to Configure Linux for ROS ...................................................... 43

5.1.1. Downloading ROS ........................................................................... 43

5.1.2. Creating a Working ROS Directory ................................................. 45

5.1.3. Obtaining and Installing the ROS Packages .................................... 47

5.2. Using the Terminal ............................................................................. 48

5.3. ROSbag ............................................................................................. 51

5.4. RVIZ and the Points Viewer Tool ..................................................... 53

5.4.1. Using the Points Viewer Tool ......................................................... 53

5.5. LOAM driver ....................................................................................... 57

5.5.1. Method 1: Raw Sensor Data and Real Time Point Clouds ............ 57

5.5.2. Method 2: Pre-Recorded ROSbag Files or PCAP Files ................ 59

5.6. ROSbag to PCD .................................................................................. 62

5.7. CloudCompare ................................................................................... 65

5.7.1. Opening and Colorizing the Point Cloud ...................................... 66

5.7.2. Navigating the Point Cloud ............................................................ 70

5.7.3. Manipulating the Cloud ................................................................. 71

5.7.4. Measurements with the Cloud ....................................................... 72

6. Future Work and Other Applications .................................................. 75
7. Conclusion .......................................................................................................................... 77

References .................................................................................................................................. 79

Appendix A: Ubiquiti WiFi Radio Configuration and Setup .................................................. 81

Appendix B: Digital Appendix .................................................................................................. 83
LIST OF TABLES

Table 4-1: Box Dimensions and Measurements .................................................................27
Table 4-2: UAS Model Dimension Comparison .................................................................27
Table 4-3: Control Model Dimension Comparison .............................................................28
Table 4-4: Real Dimensions vs. Digital Dimensions ............................................................33
Table 4-5: Dimensions of BYU Football Field .....................................................................40
Table 5-1: Common Commands Used in the Terminal .........................................................50
Table 5-2: Typical Point Cloud Formats Used in CloudCompare ........................................66
Table 5-3: Typical Tools Used in CloudCompare .................................................................72
Table 5-4: Subset Commands of the Point Picking Tool in CloudCompare .........................73
LIST OF FIGURES

Figure 2-1: Common remote sensing technologies. (a) Nikon D700 digital camera, (b) FLIR TG 165 thermal camera, (c) radar sensor used to measure speed of vehicles in transit, and (d) Velodyne HDL-32 LiDAR scanner. ................................................................. 5

Figure 2-2: Common sUAV platforms: (a) DJI Phantom 2, smaller quadrotor and camera, (b) DJI S1000 heavy lift platform, octocopter, (c) Ritewing 81” Zephyr XL, fixed wing platform for longer flight times. ............................................................................................. 7

Figure 2-3: Unique features in each photo are selected and matched with similar features throughout the entire photoset to create a 3-D model of the Sydney Opera House. .................. 9

Figure 2-4: 3-D mesh (left) and 3-D point cloud (right) of the Book Cliffs. .................................................................................. 9

Figure 2-5: Distance measurement in CloudCompare. ............................................................................................................. 11

Figure 2-6: Change detection performed in CloudCompare from a before and after model. The scale is in meters and the values on the scaler field legend associated with different colors are in meters. The color shows the different degrees of change between before and after models. ......................................................................................... 11

Figure 2-7: Faro X330 TLS. ................................................................................................................................. 13

Figure 2-8: A point cloud generated from a TLS scan. ................................................................................................. 14

Figure 3-1: DJI Spreading Wings S1000. .................................................................................................................. 20

Figure 3-2: Velodyne VLP-16 LiDAR. .................................................................................................................. 21

Figure 3-3: Deans to barrel plug connector to power VLP-16. .................................................................................. 21

Figure 3-4: Ubiquiti PicoStation (left) and NanoStation (right). .................................................................................. 22
Figure 3-5: Modified PoE connector to give power to Ubiquiti PicoStation and to retrieve data from VLP-16 interface box. .................................................................22

Figure 4-1: Wooden boxes of known dimensions. .................................................................24

Figure 4-2: (Top) FB290F looking South-East. (Bottom) FB290F looking North-West. .................................................................24

Figure 4-3: (Top) Partial elevation view of UAS model. (Bottom) Plan view of UAS model. .............................................................................................................................25

Figure 4-4: (Top) Partial elevation view of control model. (Bottom) Plan view of control model. .............................................................................................................................26

Figure 4-5: UAS model showing the change in vertical distance or height as 0.302 m. ............29

Figure 4-6: Control model showing the change in vertical distance or height as 0.317 m. ..........29

Figure 4-7: System used to manually gather data in Fletcher courtyard. .................................30

Figure 4-8: Fletcher courtyard. ..................................................................................................31

Figure 4-9: Point cloud of Fletcher courtyard with EDL shader enabled. .................................31

Figure 4-10: Plan view of Fletcher courtyard with EDL shader enabled. Trailer indicated by red outline ..................................................................................................................32

Figure 4-11: Digital measurement of trailer in CloudCompare. ..................................................33

Figure 4-12: Picture of Rock Canyon Park, Provo, UT with a red square outlining the pavilion .............................................................................................................................35

Figure 4-13: Point cloud model of Rock Canyon Park, Provo, UT. ...........................................35

Figure 4-14: BYU football stadium ............................................................................................37

Figure 4-15: Best UAS model of BYU football stadium............................................................38

Figure 4-16: All misaligned frames of BYU football stadium ....................................................38
Figure 4-17: (Top) BYU stadium plan view processed by Real Earth. (Bottom) BYU stadium side perspective view processed by Real Earth.

Figure 4-18: Screenshot of field width dimensioning (change in X only).

Figure 4-19: Screenshot of field length dimensioning (change in Y only).

Figure 4-20: Screenshot of railing height dimensioning (change in Z only).

Figure 5-1: Screenshot of terminal after catkin_make command has been executed.

Figure 5-2: Download zip package for Velodyne-master.

Figure 5-3: Screenshot of the Linux terminal application.

Figure 5-4: Text file showing popular commands used in the terminal window.

Figure 5-5: Using the Tab key in a Linux terminal.

Figure 5-6: Determining the referenced directory within the terminal window.

Figure 5-7: RVIZ being activated in Linux.

Figure 5-8: Adding the PointCloud2 topic to the RVIZ display.

Figure 5-9: Selecting the /velodyne_points topic.

Figure 5-10: Visualizing raw sensor data after the correct topic has been selected.

Figure 5-11: Screenshot of VeloView sensor configuration.

Figure 5-12: Recording data on VeloView.

Figure 5-13: Generated PCD files after running the bag_to_pcd command in ROS.

Figure 5-14: Drag and drop all PCD files from Windows Explorer into CloudCompare’s workspace.

Figure 5-15: Merging multiple clouds in CloudCompare.

Figure 5-16: Using the PCV tool in CloudCompare.

Figure 5-17: Using the scalar field options in CloudCompare after PCV.
Figure 5-18: (Top) before PCV colorization. (Bottom) after PCV colorization. ........................................69

Figure 5-19: Activating the trackball in CloudCompare. .................................................................70

Figure 5-20: (Left) complete stadium model before segmentation. (Right) north end zone left after segmentation. ..................................................................................................................71
1. INTRODUCTION

There is growing demand for remote sensing data obtained through small Unmanned Aerial Vehicles (sUAVs). Not only are UAV platforms becoming more affordable, but sensor counterparts are also becoming cheaper, smaller, and lighter. At BYU, research is being done in computer vision techniques that converts aerial imagery into 3-D models through Structure from Motion (SfM) (Martin et al. 2016a; Martin et al. 2016b; Palmer et al. 2015). While being successful in generated highly detailed and accurate 3-D models, industry has a modeling standard of using light detection and ranging (LiDAR) to create 3-D maps. In order to augment the current SfM research, a LiDAR system has been purchased and incorporated in the 3-D model making research. LiDAR model generation can serve as a land surveying alternative to obtain relatively accurate (3 cm) 3-D models that can be visualized in real time or near real time.

1.1. PROBLEM STATEMENT

There is a need to estimate distances, areas, or volumes at construction sites or in an existing building without the lengthy processing time to generate 3-D geometry. If preliminary calculations are desired for an old building renovation, but the construction documents cannot be procured, it can take large amounts of resources and time before engineers can get started with introductory analysis/design. However, the use of a sUAV to quickly and methodically collect data via special sensors can allow for rapid point cloud generation. This study takes a look into
real-time point cloud generation through Laser Odometry and Mapping (LOAM). Real-time point generation and cloud registration can lead to faster engineering analysis and easy geometry verification. LOAM research is attributed to Ji Zhang of Carnegie Mellon University (Zhang 2014). Registering point clouds with the Velodyne VLP-16 is also a challenge. A significant portion of this research is focused on creating a suitable unmanned aerial system (UAS) that supports the Velodyne LiDAR. The data is then processed, analyzed, and accurately exported into usable LiDAR point cloud formats. Typical commercial UAS of this type can range from $60,000 to $100,000. This study shows the feasibility of a platform that costs less than $20,000.

1.2. OBJECTIVES

Following a semi-comprehensive background dialogue the author introduces the development concepts and implementation that has occurred in this study. The research and presentation of the following results are proposed to meet the following objectives:

1. Identify a suitable UAS platform for sustainable LiDAR data collection
2. Compare the 3-D models to ground truth to verify the accuracy and usefulness of both the LiDAR sensor and data gathering UAS with both indoor and outdoor case studies
3. Instruct the user on how to use the selected UAS
2. BACKGROUND

The integration of LiDAR, unmanned aerial vehicles (UAVs), point cloud generation, and computer processing techniques allows for innovative solutions to civil engineering challenges. However, it must be noted that each of these technologies and their joint integration are heavily researched areas, and this report alone should not be viewed as a comprehensive volume to understand these technologies. Rather, this report should serve as a guide to understand the basics of these technologies to give the reader elementary insight into how these tools can be applied on real-world civil engineering problems.

Several new terms and technologies are introduced in a methodical order to serve as a foundation for this study. The fields of study related to this LiDAR-based point cloud generation technology are: remote sensing, UAVs, 3-D modeling, LiDAR, laser odometry and mapping (LOAM), and robot operating system (ROS). A brief synopsis of each topic is covered in the proceeding report.

2.1. REMOTE SENSING

Remote sensing is a way to acquire data from a specific site or object from a distance away from the object, most commonly taking the form of aerial or satellite imagery (Bolstad 2012). Satellite imagery is easy accessible and is almost commonplace for anyone living today. For instance, Google Earth® allows the user to virtually navigate the entire world without
physically having to be there. For engineers, this is a tremendous help as site visits can be first made virtually and areas of significance can be identified and later visited in person for further inspection. Bolstad points out that “The most common forms of remote sensing are based on reflected electromagnetic energy. When energy from the sun...strikes an object, a portion of the energy is reflected. Different materials reflect different amounts of incoming energy, and this differential reflectance gives objects a distinct appearance.” Essentially, the visible light spectrum in the form of photographs produces data that can be analyzed (It should be noted that although digital cameras are most common, other remote sensing technology exists like thermal/infrared cameras, radar, and LiDAR, which can also be used to gather data that regular cameras cannot obtain, see Figure 2-1).

Photographs can yield information like area coverage, geometric accuracy, and create a permanent record from a specific point in time. With the addition of Global Positioning System (GPS) technology, location can be associated with each image and the image can be assigned to a specific area on the earth. With a limited knowledge (like an object with known dimensions), one could capture many photos of an area, create a mosaic by overlapping the same areas of different images, and measure approximate distances all with a set of 2-D imagery. The science behind using photographs in surveying and mapping to measure distances is called photogrammetry. Photogrammetry is defined by American Society of Photogrammetry and Remote Sensing (ASPRS 2012) as “the art, science, and technology of obtaining reliable information about physical objects and the environment, through processes of recording, measuring, and interpreting images and patterns of electromagnetic radiant energy and other phenomena” (ASPRS 2012). Images are a rich source of spatial information and have been used as a basis for mapping for more than seven decades (Bolstad 2012).
If satellite imagery does not achieve the detail that is required for object recognition or investigation, aerial imagery is used to obtain close up data. In the past airplanes have been mounted with remote sensing technology. However, for more intimate detail, UAVs have grown in popularity as a new vehicle for obtaining aerial imagery.

2.2. UNMANNED AERIAL VEHICLES (UAVS)

UAVs are a technology that have been rapidly developing and have crossed interdisciplinary lines throughout engineering. An early definition of what a UAV (commonly referred to as a “drone”) was brought to the public from Peter van Blyenburgh of the European Unmanned Vehicle Systems Association (ERO UVS) who defined them as “uninhabited and reusable motorized aerial vehicles, which are remotely controlled, semi-autonomous, autonomous, or have a combination of these capabilities, and that can carry various types of payloads” (Blyenburgh 1999). A payload is defined as the capacity of the UAV to carry objects that are not associated with the UAV itself. A typical payload consists of a camera mounted on a gimbal (a device used to both stabilize the camera and to turn the camera about certain axes).
UAVs come in various shapes and sizes that are governed by the function of the aircraft. For instance, if a UAV is needed to travel over long distances, a fixed wing platform that can conserve energy by gliding and travel at high speeds may be preferable. In contrast, if a UAV is desired to take close-up imagery of a cliff face, a quad rotor platform may be more desirable as it can fly at much slower speeds within close proximity of a target allowing the user to take stable imagery. Figure 2-2 shows three different types of platforms that are commonly used for remote sensing. A more complete and detailed report on how platform selection vs. camera type and the interaction between the two can be found in the work of Dr. Kevin Franke at BYU. UAVs are a shining light in the terms of data collection, giving the user many different ways to photograph sites or features that would be too difficult to obtain by car or even by hand. The International Society for Photogrammetry and Remote Sensing (ISPRS) has said that UAV systems used for remote sensing have soared in the past four years and have deemed UAVs as “a new appeal for scientists, who will now be able to conduct research in a much more flexible way” (Everaerts, J. 2008).

Having a skilled pilot can make all the difference when trying to acquire usable datasets. When one is in the field and adjustments need to be made it can be beneficial to use the skills of an experienced pilot to get closer/further, fly faster/slower, or perform a specific flight path that may be hard to reprogram in the field. However, full autonomy has its advantages when modeling long linear features because remote control transmitters can fall out of range easily.
One limitation of UAVs are that the majority of small UAVs require battery power. For platforms that are required to lift heavy DSLR payloads becomes a challenge as more battery power is used to support such loads and flight times are reduced.

2.3. 3-D MODELING

For the scope of this report, 3-D modeling is broken into three categories: data collection, processing techniques and modeling software, and the different types of model formats. Each one of these subcategories have been researched extensively by other scholars and stand on their own, but only their relevance toward this report’s application was noted. However, before going further the question must be asked, “Why do we model anything?” If there are 2D images available, what is the point of making a full-fledged model? The answer is simple and relates to the amount of information needed to solve a problem. It can be really difficult to measure distance in three dimensions, not to mention volume from simply looking at a photograph. 3-D models allow the user to investigate features or specific sites to perform analysis such as change detection. Imagine trying to estimate how much material in a landslide has moved from just viewing before and after pictures. It may be possible, but still very hard to be done accurately.
D modeling gives the user the ability accurately estimate quantities such as volume while also preserving high resolution detail to be used for future examination.

Data collection can come in many forms, but most commonly and as discussed previously through the process of remote sensing. Once a feature or an area is desired to be investigated more closely data can be acquired by taking digital photographs, thermal imagining, or LiDAR scanning (see LiDAR section of background). Using UAVs can reduce the time it takes to acquire raw data from a site and allow multi viewpoints of specific features. Many different viewpoints may not be attainable by hand as traversing the terrain may not be possible.

Once the data has been obtained, modeling moves into the processing phase. For 3-D models generated from photographs a common processing technique is SfM, which is an area of computer vision. Frank Dellaert of Carnegie Mellon University sheds light on the process of SfM and computer vision, explaining “A primary objective of computer vision is to enable reconstructing 3-D scene geometry and camera motion from a set of images of a static scene…Given a set of image feature trajectories over time, solve for their 3-D positions and camera motion. This classical formulation has been studied in the context of structure from motion” (Dellaert et al. 2000). Principally, SfM uses an algorithm to detect similar features in each image from a photoset comprising many different viewpoints and deem them the same feature. A simplified visual aid is shown in Figure 2-3, showing three vantage points of the Sydney Opera House in Australia and how SfM creates 3-D geometry. The 3-D model is first generate as a lot of points that are collectively referred to as a point cloud or a 3-D point cloud model. Each point are assigned an x, y, z position and can be so closely spaced together that they give the appearance of one solid object. From this point cloud, a mesh (or a surface) can be generated which interpolates a face between adjacent points to create an actual solid surface.
For example, a 3-D model was created of the Book Cliffs in Southern Utah. A point cloud model and a mesh model were both created for different purposes. The 3-D models can be seen in Figure 2-4.

Figure 2-3: Unique features in each photo are selected and matched with similar features throughout the entire photoset to create a 3-D model of the Sydney Opera House.

Figure 2-4: 3-D mesh (left) and 3-D point cloud (right) of the Book Cliffs.
A reliable software for generating SfM point clouds is Agisoft™ PhotoScan Professional, which is the point cloud generator of choice at the BYU Process Research and Intelligent Systems Modeling (PRISM) lab. PhotoScan uses an SfM algorithm to recreate 3-D geometry. It has the ability to include reference points where manual GPS points can be referenced in each image to ensure an accurate model for measuring distance. Once the model is generated it can be exported to a cloud analysis software. An open source cloud computing software that is widely used is CloudCompare. Within CloudCompare, distances can be measured, areas and volumes can be estimated, and change detection can take place. Figure 2-5 shows a simple distance calculation being performed in CloudCompare. Boxes 8 and 9 were spaced over a meter apart and the distance tool calculated there horizontal distance between them was 1.14 meters. Figure 2-6 shows the results of a change detection that happened near Lake Havasu City, AZ. The site was imaged twice by a UAV, once before any soil compaction [deep dynamic compaction (DDC)], and a second time after DDC had occurred causing a uniform drop in elevation of about 1.25 feet. The scale on the right of the figure is in meters.

The one drawback of using SfM techniques and PhotoScan is the time it takes to develop a model start to finish. With image collection taking close to 4 hours, manual pre-processing the photoset 5 hours, computer processing on average for high detail 65 hours, all together totaling 74 hours. A complete model may not be fully attainable until 3 days after gathering the data. For some applications, that is too long or costly for manual effort.
Figure 2-5: Distance measurement in CloudCompare.

Figure 2-6: Change detection performed in CloudCompare from a before and after model. The scale is in meters and the values on the scaler field legend associated with different colors are in meters. The color shows the different degrees of change between before and after models.
2.4. LIDAR

The point cloud models showcased in Figure 2-6 were generated from the SfM program *PhotoScan*, however, the “industry standard” for 3-D modeling is based on using LiDAR technology (Autodesk 2014). The National Oceanic and Atmospheric Administration (NOAA) gives a comprehensive definition of what LiDAR is, “LiDAR, which stands for Light Detection and Ranging, is a remote sensing method that uses light in the form of a pulsed laser to measure ranges...These light pulses—combined with other data—generate precise, three-dimensional information about the shape of the Earth and its surface characteristics” (NOAA 2015). In short, a laser pulse is emitted from the sensor at a specific inclination. By knowing the speed of light the sensor can measure the time it takes for the pulse to reflect off a surface and come back to the sensor and by doing some simply math, the sensor converts time to distance. With the addition of a GPS, the LiDAR sensor can not only measure the distance between itself and an object, but tag that object with specific position data (either local or global coordinates). Once all the data is recorded it can be processed and a complete georeferenced map can be generated and analyzed. The process of combining multiple scans is known as registering the point cloud. Generally, the closer the scanner is to an object, the amount of lasers that sample points from the object increase, which increases the point density. Objects that are far away tend to have lower densities because the lasers are emitted at different orientations and spread out over large distances.

Traditionally, a terrestrial LiDAR device was set up over a fixed, known point and starts scanning. Once scanning was complete, the device was relocated to a different viewpoint that was also GPS referenced and scanning was performed again. The process was repeated until enough data was recorded to create an accurate 3-D model. However, this style of data gathering is not the only way to use the device today. Autodesk, a computer aided design (CAD) software
company, has posted literature breaking LiDAR into four distinct categories: Stationary terrestrial LiDAR scanning (STLS), mobile terrestrial LiDAR Scanning (MTLS), high-altitude airborne LiDAR, and low-altitude airborne LiDAR (Autodesk 2014). The method described previously is known as STLS or just TLS. An example of a terrestrial laser scanning device is a Faro® TLS scanner, which is capable of sampling up to 976,000 points per second with a sampling accuracy of ±2mm (Faro 2013). A Faro TLS scanner can be seen in Figure 2-7 and a TLS point cloud model of a building can be seen in Figure 2-8.

![Image of Faro X330 TLS](image)

Figure 2-7: Faro X330 TLS.
Figure 2-8: A point cloud generated from a TLS scan.

For the scope of this report only the TLS and a subset of low-altitude airborne LiDAR is discussed. Research involving LiDAR and small UAVs is becoming increasingly rapid (Watts et al. 2012). As advancements in sensor technology and UAV flexibility have made progress, the incorporation of both technologies is a new field of research. Independently LiDAR and UAVs have taken great strides in development over the years and much is written about both. However, using the technology in tandem leaves much to be desired.

In 2011, Yi Lin presented one of the original studies about using LiDAR with mini-UAVs for fine-scale mapping. The study explains how traditional aerial LiDAR (high altitude) is used to monitor large areas but with limited density. It was economical unfeasible to try and scan areas with more than one laser pulse per square meter. Local data at the fine scale level was lost due to low density and demonstrates the need for a fine scaled, local solution. The experiment
focus primarily on fine scale mapping which deals with height estimation, small object detection, and even intensity based-road extraction. Yi describes his study as “a pioneered mini-UAV-borne LiDAR system” (Lin 2011).

In Australia, research was conducted at the School of Geography and Environmental Studies at the University of Tasmania in Hobart with respect to UAV-LiDAR systems. Luke Wallace discusses the application of using LiDAR to map and inventory Forests. The paper aims to, “present the development of a UAV-borne LiDAR system using lightweight and low-cost sensors, and demonstrate its capability of collecting spatially dense, accurate, and repeatable measurements for forestry inventory applications” (Wallace 2012). The study shows the increasing rate at which applications of UAVs and LiDAR are being developed and tested as worthwhile options to gather accurate data.

Registering aerial LiDAR into usable point cloud data can be very difficult. It consists of a dynamically moving sensor that is requiring both GPS position and orientation data from an inertial measurement unit (IMU) to ensure accuracy. Such additions to the LiDAR sensor can also be very costly. Global Navigation Satellite Systems (GNSS) that combine both GPS and IMU data can cost tens of thousands of dollars. As stated previously, the sUAV can only carry a specific payload. Adding sensors to the UAV (each requiring its own battery) can quickly accumulate payload, and the subsequent flight times can be drastically reduced. Ji Zhang of Carnegie Mellon University has developed a way to accurately register point clouds in real time by using software to simulate position data and orient each scan of laser data so that it matches with previous scans. This real time scanning and registering is referred to as LOAM (Zhang 2014).
2.5. LOAM

Many applications of LiDAR mapping require the LiDAR to move. As discussed previously this can create a problem as precise location of the sensor is required if proper registration of the points is to be accomplished. If the money is available, one could use a fixed coordinate system approach using very expensive, highly acute sensors associated with the LiDAR. However, a different approach is taken from Ji Zhang as he developed the LOAM technology. He explains, “The method achieves both low-drift and low-computational complexity without the need for high accuracy ranging or inertial measurements. The key idea in obtaining this level of performance is the division of the typically complex problem of simultaneous localization and mapping (SLAM), which seeks to optimize a large number of variables simultaneously, by two algorithms. One algorithm performs odometry at a high frequency but low fidelity to estimate velocity of the LiDAR. Another algorithm runs at a frequency of an order of magnitude lower for fine matching and registration of the point cloud…. Specifically, both algorithms extract feature points located on sharp edges and planar surfaces, and match the feature points to edge line segments and planar surface patches, respectively” (Zhang 2014). The methodology involves first determining the sensors location by odometry (the use of data to estimate change in position over time) relative to its starting location (Zhang 2015). Once odometry is estimated, the second algorithm is finding similar features in each successive frame (one full laser scan) and lining them up based on structure that is apparent in each frame (sharp edges, corners, and the intersection of planes).

LOAM can be supplemented with IMU data if a sensor is available, which can help reduce error by relying both on simulated orientation data and IMU orientation data and disposing of frames that are not necessary. One of the original iterations of LOAM was posted
with its source code onto the popular open source robotics website: https://github.com, which is the version LOAM that is used in this report and study. Additional advances have been made with the LOAM code by implementing more sophisticated SLAM algorithms. The algorithms allow point clouds to be generated as data is gathered. These advances are not available to the public as the inventor of LOAM has started a business, Real Earth, where the algorithm is commercialized for scanning registration. To use this web based service the user would go to the webpage www.realearth.us where there is a 3-D model generator and other products that are available to supplement the Velodyne LiDAR sensor. In order to apply the capabilities of LOAM the reader must first become familiar with Robot Operating System (ROS) which is only used on Linux.

2.6. ROS

To associate the LOAM processes with the LiDAR sensor, there needs to be an interface platform that facilitates communication between the two (or more) devices. Robot operating system, or ROS, was developed by a group of researchers primarily comprising students from Stanford University and the University of Southern California. The collaboration published a journal article titled, “ROS: an open-source Robot Operating System” which summarizes ROS’ original purpose as, “not an operating system in the traditional sense of process management and scheduling; rather, it provides a structured communications layer above the host operating systems of a heterogeneous compute cluster” (Quigley et al. 2009). ROS allows for on-board machines (or sensors) to have a bridge to off-board machines (like the LOAM software).

ROS.org is a website that explains the new developments in robotics and applications of ROS and is often associated with github.com. GitHub is a website where source code is published and often times is freely accessible for download. According to ROS.org, “ROS is a
flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. Why? Because creating truly robust, general-purpose robot software is hard. From the robot's perspective, problems that seem trivial to humans often vary wildly between instances of tasks and environments. Dealing with these variations is so hard that no single individual, laboratory, or institution can hope to do it on their own. As a result, ROS was built from the ground up to encourage collaborative robotics software development” (ros.org/about-ros).

The simple knowledge that ROS is used to help two entities communicate is the fundamental principal behind this research effort and the complexities of writing code is not discussed in this report. However, knowing how to call out ROS on a Linux platform is discussed in the instructional part of this report.
3. PLATFORM SELECTION AND SPECS

The UAS consists of three components: the platform, the sensor and the Wi-Fi radio network. With all the components connected correctly the total weight of the platform is 16 lbs. and the maximum safe flight time is 17 minutes.

The DJI “Spreading Wings” S1000 drone (see Figure 3-1) was selected as the base platform for the UAS. The S1000 can carry heavier payloads compared with other sUAVs and can harness great stability at low flight speeds. The current setup provides the S1000 with 16 lbs. of payload. Using two parallel Glacier 6-cell 22.2 volt 8,000mAh batteries, the S1000 can theoretically achieve 17 minutes of flight time; only 12 minutes of flight time are permitted in each flight as a precautionary measure to prevent the batteries from failing the S1000 mid-flight. A modified Pixhawk 3DR autopilot is used to help stabilize the platform in flight. The pilot uses a Spektrum DX8i controller to communicate with a Spektrum AR8000 with a PWM Encoder that connects to the Pixhawk aboard the S1000.
A Velodyne VLP-16 “Puck” 3D LiDAR (Figure 3-2) sensor is mounted on the S1000. It provides a $360^\circ$ horizontal field of vision (FOV) and $\pm 15^\circ$ vertical FOV. The Puck supports 16 channels of spinning lasers to provide approximately 300,000 points per second within a measurement range up to 100 m and $\pm 3$ cm of typical accuracy. The Puck’s low power consumption (approximately 8 watts) and low weight (830 grams) allows it to take measurements from aboard the S1000 while keeping the payload light to allow for extended flight time. Currently, the Puck is powered by a Zippy Compact 3-cell 11.1 volt 2200mAh battery. To connect the battery to the VLP-16 interface box a customized power plug was used, as seen in Figure 3-3. On one side is a deans plug and the other side is a 2.1mm barrel plug.
The Wi-Fi radio network consists of two parts: a PicoStation on the drone in the air and a NanoStation attached to a computer on the ground (Figure 3-4). The Ubiquiti PicoStation M2 HP unit mounted on the S1000 has an outdoor range of up to 500 m and is powered by a third Glacier 6-cell 22.2 volt 8,000mAh battery by a means of a modified power cord (Passive PoE Injector Cable made by Adafruit, modified with deans plug shown in Figure 3-5), but that battery could be replaced by a smaller battery to decrease the payload on the drone. The Ubiquiti NanoStation M2 unit is attached to a computer and is powered by another battery. The PicoStation transmits the data gathered by the Velodyne Puck to the NanoStation which then
sends the data to the computer on the ground. This allows the computer to store the data and build a point cloud in real time while the drone is still in flight. When conditions are optimal the Wi-Fi pair can send and receive up to 150 Mbps, which is well within the demand of the Puck which sends out information at around 8 Mbps. To ensure good throughput AirMAX needs to be enabled when configuring both Wi-Fi units (the PicoStation must be locked onto the NanoStation). For proper configuration see Appendix A.

Figure 3-4: Ubiquiti PicoStation (left) and NanoStation (right).

Figure 3-5: Modified PoE connector to give power to Ubiquiti PicoStation and to retrieve data from VLP-16 interface box.
4. ACCURACY TESTING

Two main types of accuracy testing were performed to validate the use of this LiDAR mapping platform, indoor and outdoor. Within the indoor and outdoor tests, two types of subtests were performed, gathering datasets both with and without the sUAV. The reason for the subtests was to determine if physically mounting the LiDAR sensor on a UAV would limit or restrict the sensor. The main concern is posed as follows, “Will the high frequency vibrations induce too much disturbance to produce reliable raw sensor data for model processing?”.

4.1. INDOOR ACCURACY TESTING

The primary indoor testing was conducted inside of the BYU Fletcher building in room 290F. Figure 4-2 shows two different views of the room. A set of wooden boxes (shown in Figure 4-1) with known dimensions and distances from each other were lined up along the North wall. The purpose of the boxes were to determine if the resulting 3-D model would preserve the dimensions within allowable error (3cm due to limitations of the sensor). A brief flight was conducted moving the UAS approximately 8 meters between take off, panning the room West to East, panning back East to West, and then landing the UAS. A secondary test was performed to collect control data. The LiDAR was mounted on a movable table and was manually moved back and forth along the same path as the UAS.
Figure 4-1: Wooden boxes of known dimensions.

Figure 4-2: (Top) FB290F looking South-East. (Bottom) FB290F looking North-West.
After running the raw data through the LOAM processing and cloud registration, the resulting point clouds were colorized in grayscale. The point cloud generated from the UAS is shown in Figure 4-3, while the point cloud generated by manually maneuvering the sensor (control data) is shown in Figure 4-4.

Figure 4-3: (Top) Partial elevation view of UAS model. (Bottom) Plan view of UAS model.
Figure 4-4: (Top) Partial elevation view of control model. (Bottom) Plan view of control model.

Qualitatively the models look similar. The outline of the boxes show up in the plan view distinctly. The geometry of the room appeared to be consistent with reality, with no distortion or skewed surfaces showing up. To estimate accuracy, the distance between the boxes and their height and width dimensions were measured and were recorded in Table 4-1. The boxes (2-7) are lined up from left to right, numerically in ascending order in Figures 4-3 and 4-4. A comparison of the actual Box I.D. 3 dimensions versus how they were measured digitally in UAS model and
control model are shown in Table 4-2 and Table 4-3, respectively. It is important to note that on Tables 4-2 and 4-3 there is information labeled as either absolute or relative error. A negative sign (-) indicates that the model measurements are smaller than the actual dimensions, where positive data indicates that the model overestimates distance.

Table 4-1: Box Dimensions and Measurements

<table>
<thead>
<tr>
<th>Box I.D.</th>
<th>Height (in)</th>
<th>Width (in)</th>
<th>Distance to Next Box (in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>10</td>
<td>13.75</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>8</td>
<td>14.25</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>6</td>
<td>15.25</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>4</td>
<td>15.25</td>
</tr>
<tr>
<td>7</td>
<td>12</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4-2: UAS Model Dimension Comparison

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Actual (in)</th>
<th>Model (in)</th>
<th>Absolute Error (in)</th>
<th>Relative Error</th>
<th>Absolute Error (cm)</th>
<th>Percent of Allowable Error Relative to 3 cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width (in)</td>
<td>10</td>
<td>10.19</td>
<td>0.19</td>
<td>1.9%</td>
<td>0.48</td>
<td>16.1%</td>
</tr>
<tr>
<td>Height (in)</td>
<td>12</td>
<td>11.89</td>
<td>-0.11</td>
<td>-0.9%</td>
<td>-0.28</td>
<td>9.3%</td>
</tr>
<tr>
<td>Distance to Box 4 (in)</td>
<td>13.75</td>
<td>13.31</td>
<td>-0.44</td>
<td>-3.2%</td>
<td>-1.12</td>
<td>37.3%</td>
</tr>
</tbody>
</table>
Table 4-3: Control Model Dimension Comparison

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Actual (in)</th>
<th>Model (in)</th>
<th>Absolute Error (in)</th>
<th>Relative Error</th>
<th>Absolute Error (cm)</th>
<th>Percent of Allowable Error Relative to 3 cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width (in)</td>
<td>10</td>
<td>9.84</td>
<td>0.16</td>
<td>1.6%</td>
<td>0.41</td>
<td>13.5%</td>
</tr>
<tr>
<td>Height (in)</td>
<td>12</td>
<td>12.48</td>
<td>-0.48</td>
<td>-4.0%</td>
<td>-1.22</td>
<td>40.6%</td>
</tr>
<tr>
<td>Distance to Box 4 (in)</td>
<td>13.75</td>
<td>12.56</td>
<td>1.19</td>
<td>8.7%</td>
<td>3.02</td>
<td>100.8%</td>
</tr>
</tbody>
</table>

Both Table 4-2 and Table 4-3 have a column titled “Percent of Allowable Error Relative to 3 cm” which shows a calculation that takes the absolute error in centimeters and dividing it by 3 cm as that is given accuracy of the sensor. In both the UAS and the control datasets, it appears that both are within the 3 cm accuracy of the sensor, with the exception of the control data dimension “Distance to Box 4”. However, it is only slightly over 100%. An example of the UAS model measurements for Box 3 are shown in Figure 4-5, and the example of the control model measurements are shown in Figure 4-6.

It is interesting to note that in almost every category the UAS model is superior to the Control model. The UAS model seems to preserve distance better than the control model because it has the advantage of having slight fluctuation in elevation, whereas the control data had the sensor mounted and recording at a fixed elevation throughout the test. This suggests that the LOAM does better if it has multiple elevations or viewpoints, which is reassuring as terrestrial LiDAR is typically more reliable when multiple perspectives are scanned.
To determine if there is an actual trend occurring due to change in elevation, a statistical analysis needs to be performed on several measurement datasets to conclude of any statistical significance. A thorough statistical analysis was not performed in this study.
4.2. OUTDOOR ACCURACY TESTING

Outdoor testing was performed at two different locations. One set of data was gathered manually in the courtyard of the Fletcher building at BYU. Unfortunately, an outdoor test in the courtyard with the UAS is not permitted by BYU as there is a “no fly on campus” policy. A second set of UAS data was gathered at Rock Canyon Park in Provo. A third test using a more sophisticated LOAM algorithm was performed at the BYU Football Stadium.

4.2.1. FLETCHER BUILDING COURTYARD

The sensor was hard-wired into the processing computer, so no Wi-Fi connection was necessary. The sensor was wheeled around the Fletcher courtyard on a moving cart, as seen in Figure 4-7. A picture of the courtyard can be seen in Figure 4-8 and the resulting point cloud model can be seen in Figure 4-9.

Figure 4-7: System used to manually gather data in Fletcher courtyard.
Figure 4-8: Fletcher courtyard.

Figure 4-9: Point cloud of Fletcher courtyard with EDL shader enabled.
To check the accuracy of this model the length of the doors on the white trailer located in the North-East corner (see Figure 4-10) of the courtyard was measured. The image of the digital measurement is shown in Figure 4-11 and the measurements from real life and in the model were recorded in Table 4-4.

Figure 4-10: Plan view of Fletcher courtyard with EDL shader enabled. Trailer indicated by red outline.
Figure 4-11: Digital measurement of trailer in CloudCompare.

Table 4-4: Real Dimensions vs. Digital Dimensions

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Actual (cm)</th>
<th>Model (cm)</th>
<th>Absolute Error (in)</th>
<th>Relative Error</th>
<th>Percent of Allowable Error Relative to 3 cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>183</td>
<td>185</td>
<td>2</td>
<td>1.1%</td>
<td>67%</td>
</tr>
</tbody>
</table>
The "Percent of Allowable Error Relative to 3 cm" from Table 4-4 was 67% meaning that the 3 cm limitation of the sensor was still not limiting the model after it underwent LOAM processing. As was the case with the indoor test, the outdoor test proves both qualitatively and quantitatively accurate as it preserves the geometry of the courtyard. In the previous figures the colorization is based on intensity values. Everything that is not very reflective appears in blue, increasing to green, highly reflective in yellow, and most reflective in red. Typically the pavement returns blue intensity and the metallic license plate on the trailer returns red.

As encouraging as these results are, the parameters of this experiment were ideal for LOAM processing. LOAM does very well in highly structured environments (the sensor is almost completely enclosed by walls and it allows the LOAM processing very easy computations as it matches up similar features between frames) and almost resembles an indoor environment. The next phase of testing at Rock Canyon Park shows how the sensor performed when there was not a lot of structure between frames or even within the 100 m range of the sensor.

4.2.2. ROCK CANYON PARK

Along the far East side of Rock Canyon park there is a pavilion that is located where the test was conducted. There is a large open field that lay before the pavilion as outlined in Figure 4-12 (image not taken on same day as flight). During the test there was snow on the ground. Snow may or may not affect the processing as it reflects sunlight causing a washout affect where the sensor may be susceptible to glare. The pavilion sits on a bank that also has public restrooms to the north and a section of foliage to the south. This amount of structure was believed at first to be sufficient, however after processing the data it was observed that the model was not of usable quality. The pilot flew the UAS in a simple flight path no more than 100 feet from the pavilion. The resulting model is shown in Figure 4-13.
Figure 4-12: Picture of Rock Canyon Park, Provo, UT with a red square outlining the pavilion.

Figure 4-13: Point cloud model of Rock Canyon Park, Provo, UT.
The model shown in Figure 4-13 yields very discouraging results. It is almost impossible to discern where the pavilion (most likely located in the middle of the image). After further examination and from collaboration with Dave Duggins from Real Earth (Formerly of CMU in the Robotics Institute) he suggested, “It appears that this was an open field with very little 3-D structure, so there is no point cloud that we could register. The sensor was in the correct orientation and the flight wasn’t too bad, I would guess, but without structure, there is nothing to match between successive frames.” He further explains, “Ji (the inventor of LOAM) suggested pitching the sensor up a little (15 to 20 degrees) facing the hill, so the back of the lidar can still see the ground behind the platform (also means that best results are obtained with the minimum amount of occlusion due to mounting)”.

The problem with the open field behind the structure, even with pitching the sensor as Ji suggested, is that all of the points hitting the ground (remember it was snow covered) look so similar to the algorithm that it cannot deceive unique features enough to accurately align the frames and register the point cloud. For any future exploration where a similar site is required to be modeled, it would be recommended that the site not only have more structure, but that the UAS fly no more than 20 feet from the structure to ensure enough detail is captured in the raw data that the LOAM could use for frame registration. However, it may be more beneficial to use SfM techniques if these criteria cannot be met.

4.2.3. BYU FOOTBALL STADIUM

After testing both the highly structured outdoor environment and the low-to-none environment, it was practical to test at a medium structured site. The BYU Lavell Edwards football stadium proved to be a great option as it has a wide open athletic field with 128 meters by 68 meters dimensions surrounded by stadium seating on all four sides. It provides an
environment where certain constraining features were only in the FOV of the sensor in as low as two sides and as much as three. There was always one side of the stadium that was outside of the 100 m. range of the sensor. It forced the sensor to rely on what was usually directly in front of it for LOAM processing. The UAS was flown at an altitude of about 20-30 feet above the outer perimeter of the field at the face of the first row stands. There was one complete loop of the stadium completed, which took about 3.5 minutes to complete. An aerial view of the BYU stadium (image not taken on same day as flight) is represented in Figure 4-14 and the processed UAS model is shown in Figure 4-15.

Figure 4-15 shows the best model that could be made of the stadium from the current open source version of LOAM that was accessible for this project (only 273 out of 1357 frames were registered correctly). Unfortunately, the algorithm does get confused and misaligns the frames to result in a spiral warped model shown in Figure 4-16. From the efforts of Real Earth, this same raw data set was processed with their updated and improved LOAM algorithm that yielded very promising results that is shown in Figure 4-17 with the colored line indicating the flight path of the UAS.

![BYU football stadium.](image)
Figure 4-15: Best UAS model of BYU football stadium

Figure 4-16: All misaligned frames of BYU football stadium.
Figure 4-17: (Top) BYU stadium plan view processed by Real Earth. (Bottom) BYU stadium side perspective view processed by Real Earth.
In Figure 4-15 the LOAM was able to register the first 273 frames correctly and produce a usable model that fairly well represents the stadium with intact geometry. As the LOAM continued to process the model started to deform. The registration couldn’t decide which wall was the correct wall that it started to start a new datum to align the rest of the frames from. If the model is rotated, it is very apparent that two distinct models began to form and are superimposed over each other (Figure 4-16).

The updated LOAM algorithm used by Real Earth was able to generate a very crisp and clean model with very little noise (Figure 4-17). The entire stadium was able to be completed and qualitatively is very aesthetically assuring compared to the attempts made to register the point cloud by the previous version of LOAM. Dimensioning along three axes were measurement in CloudCompare and were compared to the real life dimensions. The real life dimensions were performed by both total station surveying and hand tape measuring methods. The actual vs digital measurements are compared and shown in Table 4-5. Corresponding CloudCompare screenshots of each measurement are shown in Figure 4-18, Figure 4-19, and Figure 4-20.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Actual (m)</th>
<th>Model (m)</th>
<th>Absolute Error (cm)</th>
<th>Relative Error</th>
<th>Percent of Allowable Error Relative to 3 cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of Field</td>
<td>127.800</td>
<td>127.844</td>
<td>-4.40</td>
<td>-0.03%</td>
<td>147%</td>
</tr>
<tr>
<td>Width of Field</td>
<td>68.785</td>
<td>68.748</td>
<td>3.70</td>
<td>0.05%</td>
<td>123%</td>
</tr>
<tr>
<td>Height of Railing</td>
<td>1.780</td>
<td>1.758</td>
<td>2.20</td>
<td>1.24%</td>
<td>73%</td>
</tr>
</tbody>
</table>
Figure 4-18: Screenshot of field width dimensioning (change in X only).

Figure 4-19: Screenshot of field length dimensioning (change in Y only).
Over large distances the sensor does accumulate error, however over a total of 127.8 meters the cumulative error is only 4.4 cm and the relative error is only -0.03% (meaning the model under-predicted the length of the field by 0.03%). However, because this compounding error is greater than 3cm (the accuracy of the sensor) it can be speculated that the LOAM processing is slightly effecting the accuracy of the model. After looking at the “Percent of Allowable Error Relative to 3 cm” in Table 4-5, it shows that for the length measurement the value is 147% (meaning 47% of the error is greater than 3 cm, which accounts for 1.4 cm of the total 4.4 cm absolute error). Point densities are on the order of 800 points per square meter.

As the qualitative aspects of the model is pleasing and the quantitative accuracy is also within justified means, there is a caveat. While vastly superior to the open source LOAM, the upgraded LOAM algorithm is not a free registration service. Real Earth now charges $0.15/meter of sensor movement. While this charge is not overly taxing, for large models (such as any long linear feature modeling like transmission lines or canals) the cost can add up quickly. Upload raw sensor data to [http://www.realearth.us/3d-model-generator.html](http://www.realearth.us/3d-model-generator.html) for the updated LOAM processing.
5. INSTRUCTIONAL

The Instructional section should serve as a comprehensive guide to using this LiDAR UAS. However, there are two caveats that are not be discussed in this report. First, how to actually fly the platform is not outlined in this report. To fully use this tool an experienced UAV pilot is required to operate and fly. Secondly, how to install Linux as an operating system on a laptop or desktop is not be discussed. This report assumes the reader has the availability of a Linux machine.

5.1. HOW TO CONFIGURE LINUX FOR ROS

Ubuntu version 14.04.3 was used in this study and is recommended for the Linux OS and ROS Indigo was used. Configuring Linux for ROS is broken into 3 sub categories: Downloading ROS, creating a working ROS directory, and obtaining and installing the ROS packages.

5.1.1. DOWNLOADING ROS

ROS.org outlines the steps for installing ROS very thoroughly. Go to http://wiki.ros.org/indigo/Installation/Ubuntu and follow the installation steps. Each step is associated with commentary to help guide the user (ROS.org commentary was not added to this report). Open up a terminal (see Section 5.2 Using the Terminal) and copy and paste each line of code from the instructions into the terminal (to paste into a terminal the hotkey is Ctrl + Shift +
V as opposed to just Ctrl + V) and press Enter. Below are each successive line of code, in order, required to complete installation. Depending on the capacity of the user’s computer, steps 4, 9, and 10 can take several minutes. Certain commands will also prompt the user to type in “y” meaning yes, followed by pressing Enter, to complete the download steps. Step 10 is not necessary for ROS but is necessary for using Velodyne data. It should be noted that an internet connection is required as ROS repositories and packages are downloaded from the internet. It is recommended the user go through some ROS tutorials available online at [http://wiki.ros.org/ROS/Tutorials](http://wiki.ros.org/ROS/Tutorials) to become familiar with the interface and commands.

1. sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

2. sudo apt-key adv --keyserver hkp://pool.sks-keyservers.net --recv-key 0xB01FA116

3. sudo apt-get update

4. sudo apt-get install ros-indigo-desktop-full

5. sudo rosdep init

6. rosdep update

7. echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc

8. source ~/.bashrc

9. sudo apt-get install python-rosinstall

10. sudo apt-get install libpcap-dev
5.1.2. CREATING A WORKING ROS DIRECTORY

It is important to create a folder (directory) and designate that as the primary folder that all the ROS packages are linked to. To make a directory open up a terminal and type in “mkdir /home/username/foldername” and press Enter. Let’s break down this statement. “mkdir” is a function command that will make a directory. The second part of the statement “/home/username/foldername” is telling Linux where to make the directory, it is saying, “go into the home folder then to the username folder (this is typically the account that is signed into when logging onto the Linux workstation, for example if the user’s computer name was “derek”, the following would be typed /home/derek/foldername) then make a folder named “foldername”. The last term of the file-path will be the name of the folder. In this case it was “foldername” but it could be whatever the user wants. Let’s say that the user will name the new folder “ROS” (mind the capital letters; if the user was to make reference to this folder later and typed in “ros” or “Ros” it would not work as the file-path names are case sensitive). To ensure that the user actually created a folder and it was put into the right place, access the “Files” icon which looks like a filing cabinet (it is the Linux equivalent of Windows Explorer) and search for the newly created “ROS” folder.

Once the new folder is created we need to turn this folder into a workspace, specifically a catkin workspace (most ROS packages are developed as a catkin package). A catkin workspace is a folder where the user modify, build, and install catkin packages (ROS Wiki June 17, 2014). To transform the newly created folder named “ROS” into a catkin workspace we open up the terminal and enter three commands:

1. `mkdir /home/username/ROS/src`
2. `cd /home/username/ROS`
3. `catkin_make`
The user will see the terminal run the processes showing that the catkin workspace is being created (see Figure 5-1). To ensure that the environment for the catkin workspace is configured correctly, ROS has to be sourced. Open the terminal and type “source /opt/ros/indigo/setup.bash” which allows ROS to be sourced correctly. To ensure this was done correctly, open up the “Files” application and press “Ctrl + H” to access hidden files. A file entitled “.bashrc” should appear. Open up “.bashrc” and a text editor will appear. Scroll to the bottom of the document and the line of code should appear that was just typed into the terminal “source /opt/ros/indigo/setup.bash”. If it did not work the user can type the line in manually and save the “.bashrc” file.

![Screenshot of terminal after catkin_make command has been executed.](image)

Figure 5-1: Screenshot of terminal after catkin_make command has been executed.
5.1.3. OBTAINING AND INSTALLING THE ROS PACKAGES

There are three ROS packages that need to be downloaded and installed into the “ROS” folder. Two of the three (Velodyne-master and pcl-ros) will be obtained by visiting the ROS.org website and manually downloading the packages. The third package, LOAM, can only be obtained from a flash-drive in the possession of BYU professor Kevin Franke (kfranke@et.byu.edu) or via email correspondence with the author Derek Wolfe (derekanthonywolfe@gmail.com) as the package is no longer publically available. To obtain the Velodyne-master package, go to https://github.com/ros-drivers/velodyne and select download zip on the right of the webpage as seen in Figure 5-2. Once the folder is downloaded it can be unzipped and stored in the “src” folder within the “ROS” catkin folder. This same process will be repeated for the pcl-ros package (found at https://github.com/rospersonception/perceptionpcl/tree/indigo-devel) and for the LOAM package, however, the LOAM package will be downloaded offline.

![Download zip package for Velodyne-master.](image)

Figure 5-2: Download zip package for Velodyne-master.
Once all three packages are downloaded and placed into the “src” folder, the user is ready to execute the catkin_make command again. This will be done in two steps in the terminal window:

1. cd /home/username/ROS
2. catkin_make

Once again the terminal will actively update the progress of the install and may require the user to enter “y” for yes to continue installation. Depending on the size of the packages and the processing capability of the computer, this can take several minutes to complete.

After completion, the user must source the “ROS” folder to the “ROS” environment that was sourced previously. This is done by opening up the terminal and entering:

1. cd /home/username/ROS/devel
2. source devel/setup.bash
3. echo $ROS_PACKAGE_PATH /home/youruser/catkin_ws/src:/opt/ros/indigo/share:/opt/ros/indigo/stacks

To ensure the sourcing was done correctly, the user can access the “.bashrc” file from the “Files” application and scroll to the bottom and look for “source ~/ROS/devel/setup.bash” (if the user’s folder was titled something other than “ROS” look for the correct folder name). If it does not appear the user may manually enter the text and save the “.bashrc” file. After this process is complete close all of the terminal windows.

5.2. USING THE TERMINAL

The terminal is frequently used to send commands to different functions of ROS. It is similar to the “Command Prompt” application on a Windows computer. It is important to
understand how to use the terminal and to be aware of what it yields as feedback. For someone new to Linux and the terminal it can be challenging to use. However, outlined below are a few important commands and hotkeys used while using the terminal. The terminal window can be seen in Figure 5-3, the red box indicating the desktop shortcut icon that is pressed to open the terminal window.

![Figure 5-3: Screenshot of the Linux terminal application.](image)

Sometimes it is easy to create a text file that contains the most common commands that are entered into the terminal. Instead of typing in the commands all the time, they can be copy and pasted in from a text document like the one shown in Figure 5-4. When copying from a text document, the hotkey is “Ctrl+C”, however, when copying directly from the terminal the hotkey is “Ctrl+Shift+C” pressed simultaneously. Inside the terminal “Ctrl+C” will terminate any command that has been ordered or end a process that has been executed/running in that terminal window. The same procedure is done for pasting lines of text into the terminal, the hotkey combination is “Ctrl+Shift+V” instead of “Ctrl+V”. Some other common terminal commands and their description are outlined in Table 5-1.
Figure 5-4: Text file showing popular commands used in the terminal window.

Table 5-1: Common Commands Used in the Terminal

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rosmmsg</td>
<td>Displays message data structure definitions</td>
</tr>
<tr>
<td>rostopic list</td>
<td>displays run-time information about topics</td>
</tr>
<tr>
<td>catkin_make</td>
<td>Build a catkin workspace</td>
</tr>
<tr>
<td>rqt_bag</td>
<td>graphical tool for viewing data in ROS bag files</td>
</tr>
<tr>
<td>cd</td>
<td>Change directory</td>
</tr>
<tr>
<td>clear</td>
<td>Clear a command line screen/window for a fresh start.</td>
</tr>
<tr>
<td>df</td>
<td>Display used and available disk space</td>
</tr>
<tr>
<td>mkdir</td>
<td>Create a new directory</td>
</tr>
<tr>
<td>pwd</td>
<td>Displays the pathname for the current directory.</td>
</tr>
<tr>
<td>printenv</td>
<td>grep ROS</td>
</tr>
</tbody>
</table>
For a lot of commands in the terminal if the “Tab” key is pressed the terminal will auto-complete statements. For example if the user is trying to change directories (cd) into a different directory, the command would start off, “cd /home”. If the user was to type “cd/h” and press tab directly after h, it would automatically complete “cd/home”. Tab completion is very useful and helps save the user a lot of time. If more than one option is available after pressing tab, the terminal will typically list out the potential options. Listing out options can also be accomplished by pressing the Tab key twice. The Tab list is illustrated in Figure 5-5.

![Figure 5-5: Using the Tab key in a Linux terminal.](image)

5.3. ROSBAG

Understanding the ROSbag command and using its capabilities is essential for using this technology. According to ROS.org a rosbag is “a set of tools for recording from and playing back to ROS topics. It is intended to be high performance and avoids deserialization and reserialization of the messages”. In short, it is the way that any ROS topic is recorded and stored. The user can subscribe to a published ROS topic and the rosbag will record all the information
from that topic until the user commands it to stop. ROSbag is also the way that previously recorded datasets can be “played” and simulate a robot publishing the saved topic. The two most important rosbag commands are “rosbag play” and “rosbag record”.

To play previously recorded data the user would open up the terminal enter the command:

```shell
rosbag play /home/derek/ROS/trial1.bag
```

This command is telling ROS to open up the file named “trial1.bag” stored in the “ROS” directory and to play all the topics stored on that ROSbag. It is important to note that the rosbag command is a ROS operation and the roscore command most have been previously executed before using any rosbag tools.

To record data from a robot publishing a ROS topic (sensor data) open up the terminal and enter the following command (note that roscore has to be executed before any rosbag commands can be initiated):

```shell
rosbag record -O rosbag_trial1.bag /velodyne_packets
```

This command is telling ROS to start subscribing to data from the velodyne_packets topic and recording it in a file named “rosbag_trial1.bag”. Depending on what directory the terminal is referencing (by looking at the text in the terminal that is located before the typed in command will tell the user what directory the terminal is referencing) is where the ROSbag will be saved. Figure 5-6 shows that the current directory is “Test” located within the “velodyne” directory and inside the “Test” directory is where the ROSbag file would be saved if the “rosbag record” command was executed. If no directory is listed, it means that the “home” directory is referenced.
5.4. RVIZ AND THE POINTS VIEWER TOOL

RVIZ (ROS visualization) is a 3-D visualizer for displaying sensor data and state information from ROS (sdk.rethinkrobotics.com/wiki/Rviz). An in-depth user’s manual can be found at http://wiki.ros.org/rviz/UserGuide. Both the standard point viewer for visualizing raw Velodyne laser data and the LOAM data will use RVIZ. A good way to check if the installation was performed properly is by doing a basic RVIZ test and verifying that data is being transferred via ROS interfacing.

5.4.1. USING THE POINTS VIEWER TOOL

The first step is to open a terminal window. Secondly, turn on the VLP-16 and connect the interface box to the computer via Ethernet. Some computers require that the I.P. address be configured so that the computer will recognize the sensor data. On Linux, a simple line of code in the terminal can configure the IP address. In the terminal type “sudo ifconfig eth0 192.168.100 netmask 255.255.255.0”. If the user is familiar with Linux they can manually configure the network connection by clicking on the network icon at the top right of the screen and accessing the “edit connections” option.
In the open terminal window type in the command, “roslaunch velodyne_pointcloud VLP16_points.launch” and press “Enter”. The command is telling ROS to look for Velodyne points and publish the information. Next, the user needs to have ROS subscribe to Velodyne points topic, which can be visualized in RVIZ. Open up a second terminal window and type, “roslaunch rviz rviz –f velodyne” and press “Enter”. An RVIZ application screen will launch and look like Figure 5-7. In the bottom left under the “Displays” sidebar there will be a bottom entitled “Add” (see Figure 5-8) and press it. A dropdown window will appear showing types of topics, select “PointCloud2” and press OK. The new “PointCloud2” topic will show up in the “Displays” sidebar. Expand the section and next to the “Topic” field select /velodyne_points (Figure 5-9). Once selected, RVIZ will display the points that the VLP-16 is publishing in real time, showing the current frame of sensor data, which can be exemplified in Figure 5-10. As the user desires, preferences can be changed and saved in RVIZ to meet the needs of specific applications. For example, the Display sidebar has a field titled “Style”, which can be set to Points, Squares, Flat Squares, Spheres, or Boxes.
Figure 5-7: RVIZ being activated in Linux.

Figure 5-8: Adding the PointCloud2 topic to the RVIZ display.
Figure 5-9: Selecting the /velodyne_points topic.

Figure 5-10: Visualizing raw sensor data after the correct topic has been selected.
5.5. LOAM DRIVER

The LOAM driver can be used directly by feeding the tool raw sensor data or via a pre-recorded .bag file. The advantage of having both methods available is being able to produce real-time/near real-time point clouds or to post process and create point clouds at some point after the data has been collected. Each method is processed similarly, however, the configuration at the beginning is different. Both methods will be outlined.

5.5.1. METHOD 1: RAW SENSOR DATA AND REAL TIME POINT CLOUDS

If raw data is being streamed directly to the computer from the sensor, make sure the Ubiquiti Wi-Fi radios are connected and selected as the network connection (if a non-UAV application or simple operating test is desired via a hard-wired connection from the VLP-16 interface box to the computer’s Ethernet port, then make sure network is configured correctly, as outlined above). To begin processing real-time point clouds, three steps are to be taken. Each step will be accomplished in a separate terminal window. First, a ROSbag file needs to start recording topics being published by the LOAM driver. Secondly, ROS has to be told to start looking for Velodyne points. Lastly, the ROS command has to be executed to startup the LOAM driver.

The reason a ROSbag has to be recording is because the data needs to be recorded somewhere so that it can be accessed again for analysis. The LOAM driver simply runs the points it receives through a processing stage that is visualized in an RVIZ window and will simultaneously publish a registered point cloud topic. If nothing is subscribing to the registered cloud topic it will not be saved and will not be able to be transformed into a usable point cloud. To subscribe to the registered cloud open up a terminal and enter the following command:
1. rosbag record -o trial_1 /velodyne_cloud_registered

Depending on what directory the terminal command line is referencing when executing the command, the .bag file will start recording in that location. The “-o” is telling the ROSbag to title the recorded .bag file with the prefix “trial_1” and usually will be followed with a numerical time stamp. If a “-O” (note the capital letter) was used instead of a lowercase letter the file would be named exactly “trial_1” with nothing following that designation. The second term “/velodyne_cloud_registered” is telling the .bag file what topic to subscribe too.

The second command will be entered in a different terminal window. ROS needs to be aware that velodyne sensor data is being sent to the computer. To enable ROS to look for velodyne data a simple command line is entered into the terminal:

2. roslaunch velodyne_pointcloud VLP16_points.launch

When the command is executed the user will see the processing activate within the terminal.

The third command will enable the LOAM driver. The third command will be entered into the third unused terminal by typing:

3. roslaunch loam_velodyne loam_velodyne.launch

To finish recording the .bag file in terminal one, simply press “Ctrl+C” and the command will terminate the recording operation. Check to see that the .bag file was recorded correctly by checking if the file size is bigger than 4.1kB (an empty .bag file will have a file structure that takes up 4.1kB of data). If the bag size is not bigger than 4.1kB it is likely that the user told to .bag file to subscribe to the wrong topic (usually happens if the topic is not spelled correctly; it is case sensitive.)
5.5.2. METHOD 2: PRE-RECORDED ROSBAG FILES OR PCAP FILES

It the user doesn’t have the ability to bring a workstation into the field that is configured to run LOAM, it is convenient to bring a laptop that can simply record raw data and process it at a later time. Such can be accomplished on either a Windows or Linux OS. The basic Velodyne package comes with software “VeloView” that allows the user to view raw sensor data, much like the point viewer RVIZ application in Linux discussed previously in Section 5.4-RVIZ and the Points Viewer Tool. VeloView is available for download at http://www.paraview.org/Wiki/VeloView under the “How to Obtain” section. It is likely that the computers network connection IP address needs to be configured to allow sensor data to pass through the computer.

5.5.2.1. WINDOWS

To configure the IP address on a Windows machine the user will need to open up the “Network and Sharing Center” and select “Local Area Connection”. Select “Properties” and click on “Internet Protocol Version 4 (TCP/IPv4)” and select “Properties”. Change the settings from automatically obtaining the IP address to use the following IP address:

**IP address**: 192.168.1.100 (100 as example, any number except 201 works)

**Subnet mask**: 255.255.255.0

**Default gateway**: N/A

Once the IP address is configured and VeloView is installed the user can begin the process of recording data. Inside VeloView, the user will click on the “File” tab and select “Open” then “Sensor Stream”. A configuration window will appear and the user will select VLP-16 followed by “OK”, allowing the program to view data from the correct sensor (see Figure 5-11). Raw sensor data should appear in the display window and the user can orient the view as desired. When the user is ready to collect data they will select the red record button (seen in
Figure 5-12) and an output window will appear allowing the user to name and save the recorded file. Once “Save” is pressed, VeloView will begin saving the raw sensor data in the form of a .pcap file until the red record button is pressed again to stop recording.

Figure 5-11: Screenshot of VeloView sensor configuration.

Figure 5-12: Recording data on VeloView.
Once the PCAP file is transferred to the workstation that will be processing the data, it needs to be converted to a ROSbag file. Three separate terminal windows are required to perform this conversion. Open up three terminals and in each terminal and execute the following commands (mind the order):

1. roscore
2. rosrunc record -O your_vlp16_filename.bag /velodyne_packets
3. rosrunc velodyne_driver velodyne_node _model:=VLP16 _pcap:=/your/pcap/path/data.pcap _read_once:=true

The first command should look familiar. Roscore will enable ROS. The second command is telling ROS to record a .bag file. The term that follows “-O” will be the name of the recorded .bag file (it can be whatever the user wants to name it and “your_vlp16_filename.bag” is an example). The last term of command two “/velodyne_packets” instructs the .bag file what topic to subscribe to and record. The third command will activate a specific ROS driver that was downloaded back in Section 5.1.3 (Velodyne_master). It will run the .pcap file through the ROS driver and publish the topic “/velodyne_packets” that the ROSbag in command two is subscribing too. The term that follows “_pcap:=” is the file path where the .pcap file is stored.

For example, if the .pcap file was saved in the “ROS” folder within Derek’s home directory and the .pcap was titled “Trial1.pcap” the user would type, “/home/derek/ROS/Trial1.pcap”. The last term of command three is telling ROS to only read the .pcap file once. The user can watch terminal three because once it finishes reading the .pcap file a notice will come up saying “end of file reached – done reading” at which point the user can switch to terminal 2 and stop recording the .bag file by pressing “Ctrl+C”.

61
5.5.2.2. LINUX

If a Linux computer was used in the field, the user would need to record a ROSbag file that would be subscribing to the topic “/velodyne_packets”. For more instruction with rosbag files see section 5.3-ROSbag.

5.5.2.3. BOTH OPERATING SYSTEMS

Regardless of the operating system used to obtain raw sensor data, once a ROSbag file has been created it can be run through the LOAM driver. Follow the instructions from Method 1 and execute the three lines of code discussed there. Once everything is set up one final command needs to be executed:

1. rosbag play /home/derek/your_vlp16_filename.bag

Play the ROSbag file that was either converted from the Windows .pcap or directly recorded on Linux. The last term of this command is the file path to the .bag file that will be run through the LOAM driver.

5.6. ROSBAG TO PCD

Once a ROSbag file is created from subscribing to the /velodyne_cloud_registered topic it needs to be converted into a point cloud file know as a PCD. According to pointclouds.org, PCD stands for Point Cloud Data and is a type of file format structure used in the Point Cloud Library (PCL). PCL is a large scale, open project for 2-D/3-D image and point cloud processing (PCL 2016). A PCL-ROS command is used to transform the ROSbag file into a usable point cloud format that can be opened in CloudCompare. The command will activate a specific ROS driver that was downloaded back in Section 5.1.3 (pcl_ros). An example of how to convert the
ROSbag to a PCD file is outlined below (roscore must be running before this command can be executed):

```bash
rosrun pcl_ros bag_to_pcd /home/derek/ROS/trial_1.bag /velodyne_cloud_registered /home/derek/ROS/PCD_Trial1
```

In this example, the command to turn a ROSbag file into a PCD was executed and inside the terminal the user should see processing taking place. The second term is telling ROS what file path to follow to find the ROSbag file. The third term is telling ROS what topic to turn into a PCD file. The forth term is telling ROS to store the results in a file path that leads to a folder titled “PCD_Trial1”.

Depending on the length of the recorded ROSbag file and how many frames of data are saved in that ROSbag the command will create multiple PCD files, each representing different frames of data (shown in Figure 5-13) that can all be imported simultaneously into CloudCompare and will line up correctly. Interestingly, the Linux version of CloudCompare does not yet have the PCD extension. The PCD files need to be transferred to a Windows or Mac computer running CloudCompare that has the PCD extension. An example of importing the PCD files into cloud compare can be seen in Figure 5-14 which showcases the drag and drop approach. All the PCD files can be selected in the “DB Tree” and merged into one cloud by using the “Merge multiple clouds” tool show in Figure 5-15.
Figure 5-13: Generated PCD files after running the `bag_to_pcd` command in ROS.

Figure 5-14: Drag and drop all PCD files from Windows Explorer into CloudCompare’s workspace.
5.7. CLOUDCOMPARE

*CloudCompare* offers a wide variety of functionality related to working with and analyzing point clouds. It is recommended that the user take some time and read over the user’s manual that can be found by going to the following website: [http://www.cloudcompare.org/](http://www.cloudcompare.org/). On the website the user can also use tutorials to help aid in educating themselves about the capabilities of *CloudCompare*. *CloudCompare* can work with almost any point cloud format. Common point cloud formats are outlined in Table 5-2. Some important tools in *CloudCompare* will be outlined below.
Table 5-2: Typical Point Cloud Formats Used in CloudCompare

<table>
<thead>
<tr>
<th>File Formats</th>
<th>Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLY</td>
<td>.ply</td>
<td>A polygon file format, developed at Stanford University by Turk et al</td>
</tr>
<tr>
<td>STL</td>
<td>.stl</td>
<td>A file format native to the stereolithography CAD software created by 3-D Systems</td>
</tr>
<tr>
<td>OBJ</td>
<td>.obj</td>
<td>A geometry definition file format first developed by Wavefront Technologies</td>
</tr>
<tr>
<td>ASCII</td>
<td>.asc</td>
<td>ASCII point cloud file (X,Y,Z, etc.)</td>
</tr>
<tr>
<td>LAS</td>
<td>.las</td>
<td>ASPRS LiDAR point clouds</td>
</tr>
<tr>
<td>E57</td>
<td>.e57</td>
<td>ASTM E57 file format</td>
</tr>
<tr>
<td>BIN</td>
<td>.bin</td>
<td>CloudCompare own format</td>
</tr>
</tbody>
</table>

5.7.1. OPENING AND COLORIZING THE POINT CLOUD

To open a point cloud file, simply press the open icon (file folder) and load in the point cloud, or perform the drag and drop method, which was discussed previously and as outlined in Figure 5-14. Often times with LiDAR data, especially if using the outlined approach noted throughout this Instructional Section, adding a scalar field will help to visualize the data. A recommended form of adding a scalar field is by selecting the cloud in the “DB Tree” on the left panel and then clicking on the “Plugins” tab on the menu bar and selecting “P.C.V. (Ambient Occlusion)” which will create lighting/shadow effects. A window will appear, shown in Figure 5-16, and the “Count” option should be set to 512 before pressing “OK”.

66
Once the PVC is complete the user will scroll down on the “Properties” panel to the “Scalar Field” section and select “Illuminance (PCV)” for the “Active” field (Figure 5-17). Once the scalar field is active the user can change the color scale to any of the preloaded options or create a unique colorization for the scalar field (see the CloudCompare user’s manual for more in depth explanations). A scale bar is also able to show up to correlate the colors with different metrics that are defined by the user. Occasionally, to help the viewer see depth, the EDL shader command can be used. A before and after example of what a point cloud will look like from performing the PCV command is shown in Figure 5-18.
Figure 5-17: Using the scalar field options in CloudCompare after PCV.
Figure 5-18: (Top) before PCV colorization. (Bottom) after PCV colorization.
5.7.2. NAVIGATING THE POINT CLOUD

Using the mouse is a convenient way to move the point cloud around. By default, using the mouse wheel will zoom in and out. Using the left click will activate the orbit command, which will allow for rotation about all three axes about a certain point. The right click allows the user to pan left or right and up and down. If a 3-D mouse is available it significantly reduces the difficulty of maneuvering the point cloud. Sometimes it is helpful to toggle-on the trackball by clicking on the “Set pivot visibility” and selecting “always visible” as shown in Figure 5-19. The track ball helps the user know what axis to rotate about and where the rotation center is set at.

![Figure 5-19: Activating the trackball in CloudCompare.](image)
5.7.3. MANIPULATING THE CLOUD

There are a few key tools to be familiar with when using cloud compare. Being an open source program CloudCompare has so much functionality and so many features, however it can sometimes lack in how robust it can be. Sometimes large datasets (~4GB and larger) can bog down the program and will often crash the program. In times like this, it may benefit the user to cut down the cloud and only analyze the section of model that is relevant to a specific task.

Using the clip tool can be ideal for a large data model. Use the clip tool by first clicking on the parent model inside the “DB Tree” side panel. Once the cloud is selected, activate the “Segment: tool which is located on the upper toolbar that has the scissor icon. For example if the cloud in Figure 5-19 was too large and only the north end zone was being used in analysis, the user could use the “Segment” tool to clip out the unwanted model. The segmentation process is shown in Figure 5-20. The use can opt to save or delete the clipped (segmented) out section. Saving the section of cloud that only has the north end zone as a .bin file and deleting everything else would allow the program to function faster.

![Figure 5-20](image)

*Figure 5-20: (Left) complete stadium model before segmentation. (Right) north end zone left after segmentation.*
A few other common tools are outlined in Table 5-3 with their name and respective description. For further instruction, it is recommended to see the CloudCompare user’s manual.

Table 5-3: Typical Tools Used in CloudCompare

<table>
<thead>
<tr>
<th>Tool</th>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merge</td>
<td>![Icon]</td>
<td>Combine multiple point clouds into one.</td>
</tr>
<tr>
<td>Clone</td>
<td>![Icon]</td>
<td>Create a duplicate of the selected point cloud.</td>
</tr>
<tr>
<td>Point Picking</td>
<td>![Icon]</td>
<td>Point information, distance between 2 points, angles between 3 points, etc.</td>
</tr>
<tr>
<td>Set Current View Mode</td>
<td>![Icon]</td>
<td>Change the perspective of the model: orthographic perspective, object-centered perspective, and viewer-based perspective.</td>
</tr>
<tr>
<td>Pick Rotation Center</td>
<td>![Icon]</td>
<td>Select one point in the model to center the rotation center from.</td>
</tr>
<tr>
<td>Subsample a Point Cloud</td>
<td>![Icon]</td>
<td>Creates a scaled down version of the parent point cloud for easier analysis.</td>
</tr>
<tr>
<td>Align Clouds</td>
<td>![Icon]</td>
<td>Align 2 separate clouds by selecting at least 4 equivalent points in each model.</td>
</tr>
<tr>
<td>EDL Shader</td>
<td>![Icon]</td>
<td>Real time shading filter that enhances very small features on blank clouds or meshes. Helps with depth.</td>
</tr>
</tbody>
</table>

5.7.4. MEASUREMENTS WITH THE CLOUD

By using the “Point Picking” tool showcased in Table 5-3 the user can measure distances easily within the model. Once the “Point Picking” tool is selected a subset of commands pops up in the upper right-hand corner of the screen. The subset commands are listed in Table 5-4. It may be necessary to increase or decrease point size so that selecting the correct point is possible. Hover the mouse around the top left of workspace window and “default point size - +” will appear. Increase or decrease as appropriate, or manually set the point size within the “Properties” side panel of the selected point cloud.
Table 5-4: Subset Commands of the Point Picking Tool in CloudCompare.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Information</td>
<td>![Image]</td>
<td>Individual point information: X, Y, Z, and scalar value.</td>
</tr>
<tr>
<td>2 Point Distance</td>
<td>![Image]</td>
<td>Calculate 1-D, 2-D, and 3-D distance between points.</td>
</tr>
<tr>
<td>3 Point Triangle</td>
<td>![Image]</td>
<td>Select 3 points and calculate angles.</td>
</tr>
<tr>
<td>2-D Rectangle Label</td>
<td>![Image]</td>
<td>Label and area for quick referencing.</td>
</tr>
<tr>
<td>Save</td>
<td>![Image]</td>
<td>Save measurements.</td>
</tr>
<tr>
<td>Undo</td>
<td>![Image]</td>
<td>Undo measurement actions.</td>
</tr>
<tr>
<td>Cancel</td>
<td>![Image]</td>
<td>Cancel Point Picking tool.</td>
</tr>
</tbody>
</table>

Measuring areas can be more difficult. There is no preset tool that yields the area of a given shape except for the “3 Point Triangle” tool described in Table 5-4. The typical way is to record several “2 Point Distance” measurements in the shape of the area that the user desires and recording all of the coordinates to be used later in an area by coordinate method calculation. For more insights on different cloud analysis tools, see the online CloudCompare forums or the user’s manual.
6. FUTURE WORK AND OTHER APPLICATIONS

There are many different ways to use a 3-D model. Once a 3-D model is generated it can act as a digital record of a specific place at a specific point in time. Models can be used to perform change detection between models generated of the same place but during different points in time. Distances can be measured easily. For example, if a 3-D model was generated from scanning several miles of transmission lines in a utility corridor, the user can easily measure cable deflection in the model instead of manually going out to the field and measuring each deflection by hand. The user also has the flexibility to measure areas or even volumes when considering how much fill is required or how much cut is needed. After compaction has occurred on a site the model can help estimate how much total compaction was achieved.

However, simply generating highly detailed 3-D models is not enough. What other information can be extracted from these models to aid in civil engineering practice? With the capabilities of LiDAR, more than 3-D geometry can be reconstructed. A different dataset than RGB colors (photography) is recorded known as intensity. In general, intensity data is a measure of how reflective a surface is. Different surfaces and materials will reflect differently. Proper calibration allows for surfaces to be classified and categorized with respect to intensity values. Very accurate terrestrial laser scanners (TLS) have been tested and methods are available for classifying surfaces. There is a lack of research being conducted that focuses on if less accurate aerial LiDAR can achieve similar results. There is a need for surface classification once a 3-D
model has been generated. The methods of Darrin Burton of York University, originally conducted with a TLS, discuss how surface classification based on intensity is capable (Burton 2013). Darrin Burton’s 2011 study could be compared with the results of aerial LiDAR and will determine if current aerial technology is a viable option for surface classification (Burton 2011).

Functionality of the equipment combined with a simplified user experience of using the tool would create an effective means of implementing the tool in practice. There are a lot of complexities involved with using this aerial LiDAR tool as discussed previously. Future work can address these complexities by creating a simplified user experience via a user interface that would eliminate many of the terminal-based ROS commands through a software application.

A more complete accuracy analysis of this UAS would be desirable. Conducting more indoor and outdoor tests with control data sets could also aid in determining what applications this platform would be better suited for. It would be beneficial to test how close the sensor needs to be from an object to obtain the best model quality.
7. CONCLUSION

Using sUAVs to gather data to be used for 3-D modeling is an effective tool for model development. Using this aerial LiDAR UAS system has proven to be a reliable alternative to SfM computer vision models. Using the customized platform outlined in this report, the user is able to attain 17 minute flight times with a payload of 16 pounds. Indoor applications seem to be more practical for this UAS as the current LOAM algorithms perform better in highly structured environments better. Current indoor accuracy’s range from 1-3 cm. Outdoor applications can be successful, however, testing has indicated that large open fields with little structure can confuse the LOAM algorithm.

Outdoor testing with sufficient structure to constrain the model has yielded accuracies around 2 cm. Outdoor testing without structure has shown that the model can be almost indistinguishable and accuracy could not be quantified. With the advances in LOAM algorithms from Ji Zhang and from the company Real Earth, outdoor models, such as the BYU stadium, can achieve accuracies of ±4 cm of accuracy over long distances (127.8 m) while preserving qualitative precision by generating models that look more aesthetic pleasing than the open source version of LOAM.

With further research and advances to this UAS, the current ROS workflow could be simplified to provide the user with an easier user experience.
REFERENCES


Appendix A: Ubiquiti WiFi Radio Configuration and Setup

Individual Setup

Ubiquiti NanoStation and PicoStation Set up.

See Instructional video https://www.youtube.com/watch?v=g4LNQv2mU7E

Current Configuration:

**PICOSTATION**

To access go to plug PicoStation into computer via Ethernet and open google chrome and type in the IP address as follows: https://192.168.1.2 and the Ubiquiti user-interface should open up as long as the Internet firewall will permit it.

Username: Pico1

Password: BYU_UAV

IP-192.168.1.2

Access Point SSID: Test

Correct Time Zone

**NANOSTATION**

To access go to plug NanoStation into computer via Ethernet and open google chrome and type in the IP address as follows: https://192.168.1.3 and the Ubiquiti user-interface should open up as long as the Internet firewall will permit it.

Username: Nano1

Password: BYU_UAV
IP-192.168.1.3

SSID-testAirMaxGround

Correct Time Zone

**AirMax Setup**

Setup PicoStation as an Access Point and Enable WDS and AirMax

Setup NanoStation as a Station and Enable WDS and AirMax

On the NanoStation lock onto the PicoStation’s IP address 192.168.1.2
APPENDIX B: DIGITAL APPENDIX

- Loam Source Code
- PCL_ROS Source Code
- Velodyne_Master Source Code