Multi–LiDAR placement, calibration, and co–registration for off-road autonomous vehicle operation

By

William Meadows

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical and Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

August 2019

Multi–LiDAR placement, calibration, and co–registration

for off-road autonomous

vehicle operation

By

William Meadows

Approved:

John E. Ball
(Major Professor)

Christopher Archibald
(Committee Member)

Qian Du
(Committee Member)

James E. Fowler
(Graduate Coordinator)

Jason M. Keith
Dean
Bagley College of Engineering

Name: William Meadows

Date of Degree: August 9, 2019

Institution: Mississippi State University

Major Field: Electrical and Computer Engineering

Major Professor: John E. Ball

Title of Study: Multi–LiDAR placement, calibration, and co–registration for off-road autonomous vehicle operation

Pages of Study: 47

Candidate for Degree of Master of Science

For autonomous vehicles, 3D, rotating LiDAR sensors are critically important towards the vehicle's ability to sense its environment. Generally, these sensors scan their environment, using multiple laser beams to gather information about the range and the intensity of the reflection from an object. For multi–LiDAR systems, the placement of the sensors determines the density of the combined point cloud. I perform preliminary research on the optimal LiDAR placement strategy for an off–road, autonomous vehicle known as the Halo project. I use simulation to generate large amounts of labeled LiDAR data that can be used to train and evaluate a neural network used to process LiDAR data in the vehicle. The performance metrics of the network are then used to generalize the performance of the sensor pose. I also, describe and evaluate intrinsic and extrinsic calibration methods that are applied in the multi–LiDAR system.

Key words: Autonomy, LiDAR, machine learning, neural network, calibration, vehicle, off road, LiDAR simulation, co-registration, optimization

DEDICATION

I dedicate this thesis to my parents for supporting and funding me during my college career. With their help I was able to enjoy a much higher standard of living during my time in college than would otherwise be possible on my own.

ACKNOWLEDGEMENTS

I would like to thank many of my co-researchers on the Halo project including Matthew Doude, the principal investigator on the project. I thank Chris Goodin, the primary developer of MAVS, for creating such a detail-oriented simulation software. I thank the autonomy team on the project including Dr. Lalitha Dabbiru who performed some prior research with the LiDARs and SqueezeSeg.

I would also like to thank several individuals who have worked on the vehicle electrification over the past few years: David Zhu, Chad Leachman, Brandon Powell, Tommy Sloan, Sean Sheehan, Russel Hanby, Michelle Price, Matthew DeRocher, and Joeseph Gaddis.

Additionally, my academic advisor Dr. John Ball, has been a great source of guidance throughout my graduate education. He's provided me with several fantastic research opportunities and outlets to present my research and network.

Portions of this work were performed in association with the Halo Project, a research and development project with the Center for Advanced Vehicular Systems (CAVS) at Mississippi State University. Learn more at http://www.cavs.msstate.edu.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF TABLES vi

LIST OF FIGURES vii

LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE ix

CHAPTER

1. INTRODUCTION 1

   1.1 The Halo Project 2
   1.2 Motivation 2

2. BACKGROUND 4

   2.1 Halo Project 4
   2.2 Vehicle Simulation 6
   2.3 MAVS 9
      2.3.1 MAVS Terrain Generation 9
      2.3.2 MAVS LiDAR Simulation 10
   2.4 LiDAR Calibration 12
   2.5 LiDAR Pose Analysis 14

3. METHODOLOGY 16

   3.1 LiDAR Pose Analysis 16
      3.1.1 MAVS Simulation 20
      3.1.2 Data Projection 21
      3.1.3 SqueezeSeq Qualification 23
   3.2 Intrinsic LiDAR Calibration 24
   3.3 Extrinsic LiDAR Calibration 27

4. RESULTS 28

4.1 Data Projection 28
4.2 CNN Performance 30
4.3 LiDAR Calibration 33

5. CONCLUSIONS AND FUTURE WORK 40

REFERENCES 44

LIST OF TABLES

3.1 SqueezeSeg training and evaluation configuration parameters. 24

4.1 Calibration Parameters for one LiDAR. 34

4.2 Average Variance of the LiDAR Data Before and After Calibration. 35

4.3 Maximum Distance Error of the LiDAR Data at 5m Before and After Calibration. 35

LIST OF FIGURES

2.1 LiDAR sensor placement. 6

2.2 Halo vehicle. 7

2.3 Example of MAVS terrain and lighting model. 10

2.4 A simulated meadow scene. 11

2.5 A simulated desert scene. 11

3.1 Quanergy M8 LiDAR. 17

3.2 LiDAR beams. 17

3.3 Vehicle coordinate system. 19

3.4 Rotation coordinates for the front LiDARs. 20

4.1 Simulated LiDAR data. 28

4.2 Label classification accuracy vs. the vertical resolution of the input image. . 29

4.3 Average number of data pixels per image vs. increasing vertical resolution. 29

4.4 Simulated LiDAR data with an 8px vertical resolution 31

4.5 Simulated LiDAR data with a 64px vertical resolution 31

4.6 User accuracy of CNN across all tested positions 31

4.7 The normalized number of data points contained in an image set for each LiDAR pose. 32

4.8 Wall measurement without calibration. 35

4.9 Wall measurement with calibration. 36

4.10 Front view of a flat wall. 37

4.11 Side view of a flat wall. 37

4.12 View of the co–located LiDAR point cloud data indoors. 38

4.13 Top view of the two areas of interest for the vehicle. 39

LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE

ADAS	Adaptive Driver Assistance System
CAVS	Center for Advanced Vehicular Systems
CNN	Convolutional Neural Network
FOV	Field of View
GPS	Global Positioning System
HIL	Hardware–in–the–loop
IMU	Inertial Measurement Unit
MSE	Mean Squared Error
MSU	Mississippi State University
α	Euclidian angle of rotation about the axis in the direction of motion of the vehicle.
β	Euclidian angle of rotation about the axis perpendicular to the direction motion of the vehicle.
ϕ	Azimuth angle
θ	Zenith angle
x, y, z	Three-dimensional coordinate points
ρ	Intensity of a reflection from a LiDAR point
l	Label of a LiDAR point
H, W	Height and width respectively of the input image to the neural network
C	Vector containing simulated LiDAR data.
d	Range in meters of a LiDAR point
N	Number of LiDAR measurements
n	Number of laser beams in a LiDAR sensor

$\vec{n}$	Estimated normal vector of a flat wall
e_i	Error between a calibrated point cloud and $\vec{n}$
K	Derived calibration parameters
P_x, P_y, P_z	Calibrated three-dimensional coordinate points
β_c	Calibrated rotational angle of the LiDAR in radians
D	Calibrated Euclidean distance of a LiDAR point.
D_{ret}	= Measured Euclidean distance of a LiDAR point
D_{xy}	D_{ret} projected onto the x–y plane
D_c	Distance correction calibration parameter
V_o	Vertical offset calibration parameter
H_o	Horizontal offset calibration parameter
θ_c	Vertical angle calibration parameter
ϵ	Rotational angle calibration parameter

CHAPTER 1

INTRODUCTION

Rotating multi–beam LiDARs are an integral part of the sensing capabilities in the emerging autonomous vehicle sector. Typically, they can operate with a $360°$ field of view (FOV) and perform well in both dark and illuminated environments. These are very valuable properties since those two feats are difficult to achieve with cameras. As sensing technology has progressed, LiDAR technology has done the same with the development of sensors with an increasing number of beams and the usage of multiple LiDARs in a single system. However, processing data from a multi–LiDAR system can be challenging, and it is not always clear whether the gains of multiple sensors can justify the additional economic and computational cost. When considering a multi–LiDAR system, it should be designed in a manner that the sensors are calibrated to work with one another, maximizing the contribution from all sensors. This thesis discusses my efforts using simulation to determine the best placement of three LiDAR sensors in a multi–LiDAR configuration. I also detail methods used to calibrate and co-locate multiple LiDAR sensors and analyze their effectiveness.

1.1 The Halo Project

The Halo Project is a research and development project within the Center for Advanced Vehicular Systems (CAVS) at Mississippi State University (MSU) [27, 28]. This project showcases CAVS researchers capabilities over a range of timely research areas, including advanced batteries, lightweight materials, additive manufacturing and human–computer interaction; the central technical challenge of the project, however, is autonomous navigation in unstructured, off–road environments. To further that goal, I am working with a team of CAVS researchers to develop a full–stack autonomy solution which is not reliant on prior environmental knowledge. Our chosen approach to this difficult problem is to develop that segments its environmental model into broad classes rather than attempting to identify individual objects (it sees the forest, not the trees). While the overall autonomous system architecture is like conventional robotics architectures, the specific environmental and planning models incorporate unique features and require special attention to data passing structures and more complex algorithms to deal with the highly complex environment.

1.2 Motivation

My thesis discusses the development and fine–tuning a multi–LiDAR system on the Halo project vehicle, which is a 2014 Subaru Forester developed for autonomous off–road navigation and modified to be an all–electric four–wheel–drive vehicle. Some of the techniques used are adopted from known methods of calibration of a single sensor. My goal is to use sensor simulation to design a multi–LiDAR system and scale known calibration methods to the system.

The contribution of this thesis is an expansion on prior research [14] focused on optimal positioning of LiDARs in a multi–LiDAR system. As part of this preliminary research, I take a novel approach to the analysis by using a neural network to qualify the effectiveness of different sensor poses. Additionally, I perform this analysis using a new, physics–based simulation software designed to simulate autonomous vehicles and the sensors integrated in them.

This thesis is organized as follows: Chapter 2 discusses prior research in the fields of LiDAR calibration and simulation, along with an overview of the autonomous vehicle using the multi–LiDAR system. Chapter 3 explains my methodology. Chapter 4 discusses results. Chapter 5 summarizes my conclusions and lists potential future work.

This thesis contains work from the following publications:

W. Meadows et al., Multi–LiDAR placement, calibration, coregistration and processing on a Subaru Forester for off–road autonomous vehicle operation, SPIE Defense + Commercial Sensing, 2019 [25].

CHAPTER 2

BACKGROUND

This chapter discusses background on the Halo project and the scene simulator. Additionally, it details the state–of–the–art research in LiDAR calibration, and LiDAR placement strategies.

2.1 Halo Project

Since the conclusion of the 2005 DARPA Grand Challenge [35], much of the following effort on autonomous vehicles research has gone into developing capabilities for structured environments. Waymo, Tesla, General Motors and others have demonstrated vehicles which can reliably interpret man–made infrastructure, such as lane markings and signs, and navigate accordingly. However, less than one percent of Earth is paved [33], leaving many applications (such as military, construction, or even unpaved roads) uncovered by current technology. The Halo project aims to further research for these types of applications. It will do so by using state–of–the–art autonomous vehicle technology such as machine learning and advanced sensors to develop a vehicle capable of autonomously navigating an unstructured environment.

In the Halo project, the autonomous system utilizes multiple sensors to perceive its environment: LiDARs, cameras, inertial measurement units (IMU) and global positioning system (GPS) receivers. Cameras provide color imagery and are used extensively for object detection. The LiDARs give 3D information, useful for detecting objects as well as characterizing the roughness of the terrain, which is critical in off–road applications, which generally, do not have well–defined or smooth roadways. The IMU measures the pitch, yaw, and roll of the vehicle along with forces due to acceleration such that obstacles can be localized effectively, even when the vehicle is traversing rough terrain. The GPS unit employs real–time kinematic (RTK) positioning to localize the global position of the vehicle with centimeter–level precision. The Halo system uses multiple eight–beam, mechanically rotating LiDAR units, as opposed to a single LiDAR sensor with a larger number of beams. This provides two primary benefits for this application. First, multiple, distributed sensors are more likely to see around occlusions, such as trees, and avoid sensor shadows. Second, the fields of view of multiple sensors may be strategically overlapped to provide increased resolution in key areas of interest without "oversensing" areas of less interest. While one LiDAR sensor is mounted on the roof of the vehicle, two additional LiDAR sensors are mounted at an angle such that the midplanes of their fields of view are nearly normal to the ground plane. This placement means that the beams of these two sensors sweep across the terrain directly in front of the vehicle and intersect, forming a grid pattern.

The Halo project utilizes all these sensors for path planning. However, the focus of this thesis is determining the optimal placement of the multiple LiDARs. The top–mounted LiDAR provides a $360°$ field of view, while the two side LiDARs help gather information about the ground immediately in front of the vehicle. The three LiDAR point clouds are co–registered and present a rich set of 3D data to the processing algorithms, which analyze the surface roughness and detect obstacles. This information (as well as information from the cameras and other sensors) forms inputs to the path planner. Figure 2.1 shows the Forester vehicle with the three LiDAR sensors. Each of the LiDAR sensors are highlighted in green and are shown in their general mounting locations. Figure 2.2 shows a picture of the modified Subaru Forester in the CAVS high–bay. One front–mounted LiDAR is visible on the front bumper.

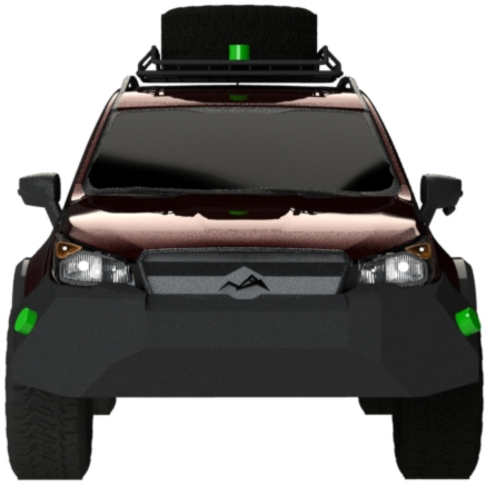

Figure 2.1

LiDAR sensor placement.

2.2 Vehicle Simulation

Some of the earliest forms of vehicle simulation evolved during the advent of adaptive driver assistance systems (ADAS) technology [10, 21]. These simulation platforms often

Figure 2.2

Halo vehicle.

tested a driver's reaction to simulated ADAS signals [10, 21]. Maag et al. used a vehicle simulator, constructed from a modified car chassis and a projected screen. The vehicle body was mounted on hydraulic actuators that provided realistic motion feedback to the driver [21]. The simulation environment provided the driver with early warnings about a braking from a leading car and showed that ADAS was effective from a human–computer interface standpoint. However, this type of simulation relied on perfect knowledge. Hanke et al. took a more realistic approach by developing a framework that allowed ADAS sensors to be modeled in simulation, allowing vehicle controllers to execute their own policies and providing a platform for hardware–in–the–loop (HIL) testing [10]. This approach more accurately resembles modern autonomous car simulation as it is based on imperfect information provided by sensors in the vehicle. However, when considering a simulation platform for an autonomous car, the human responses, typically associated with ADAS simulation, are removed and replaced with the autonomous driving policy.

A common conclusion regarding vehicle simulation is that testing algorithms such as perception and path planning algorithms in a real-world environment can be expensive and dangerous [4, 21, 34]. This motivates researchers and developers in the field of autonomous vehicles to use simulated data to evaluate their algorithms. A driving force in autonomous vehicles has been the development of commercial, on–road autonomous cars. For that reason, vehicle simulators designed to replicate urban environments are more common than off-road simulators [1, 4, 22].

Tallavajhula concludes that when simulating an outdoor environment, the ground truth of that scene is not necessarily known, nor can its features be fully recognized by sensors such as LiDARs [34]. They propose a framework for sensor simulation that involves taking in-situ measurements of an outdoor scene and fitting a simulated model to the captured data [34]. This is accomplished by generalizing “scene primitives” from the collected data. Rather than representing an object such as a tree as a high–fidelity model in simulation, their approach is to cluster non-ground points and represent them as several ellipsoids. The ground plane was determined by fitting a triangular mesh to the collected data. The usefulness of having a simulated environment of a real scene is that a different simulated agent can generate sensor data from different positions and poses than were seen in the data collection, or different sensor models may be used. The strengths of this approach are that it is based off real data, so the accuracy of the simulated scene vs. the collected data is known. This approach has been shown to be an effective method of simulating LiDAR data, however, it is not applicable to sensors such as cameras. When considering cameras, the feature resolution was increase dramatically, and lighting effects must be considered.

2.3 MAVS

The MSU Autonomous Vehicle Simulator (MAVS) is a software library for simulating autonomous vehicles in realistic, digital terrain [8]. MAVS provides a real–time, physics–based simulation of LiDAR, GPS, camera, and other sensors used in autonomous navigation, as well as the environmental factors that affect the performance of those sensors. Because MAVS readily integrates with vehicle simulation software, it can be used for evaluating the performance of autonomous perception and navigation software. Additionally, MAVS can be used to provide labeled data for training autonomy algorithms. Finally, MAVS can be used to implement simulated test cases for refining autonomous behaviors, testing performance, or simulating scenarios that are too dangerous for live tests. MAVS was used to aid in placing the front LiDAR sensors by producing labeled, simulated data that was used to visualize the beam pattern and point density of the combined pint cloud [14]. In this instance, several sensor orientations were simulated, allowing the best orientation to be chosen. Furthermore, MAVS has been used to generate datasets that can be used to train a convolutional neural network (CNN).

2.3.1 MAVS Terrain Generation

MAVS terrains are generated by first creating a surface with specified low and high frequency roughness using Perlin noise [11]. A trail is automatically created through the terrain following the natural terrain contours. Finally, plants are “grown” over a period of 20 years to create a realistic plant distribution [2]. An example output MAVS terrain is shown in Figure 2.3.

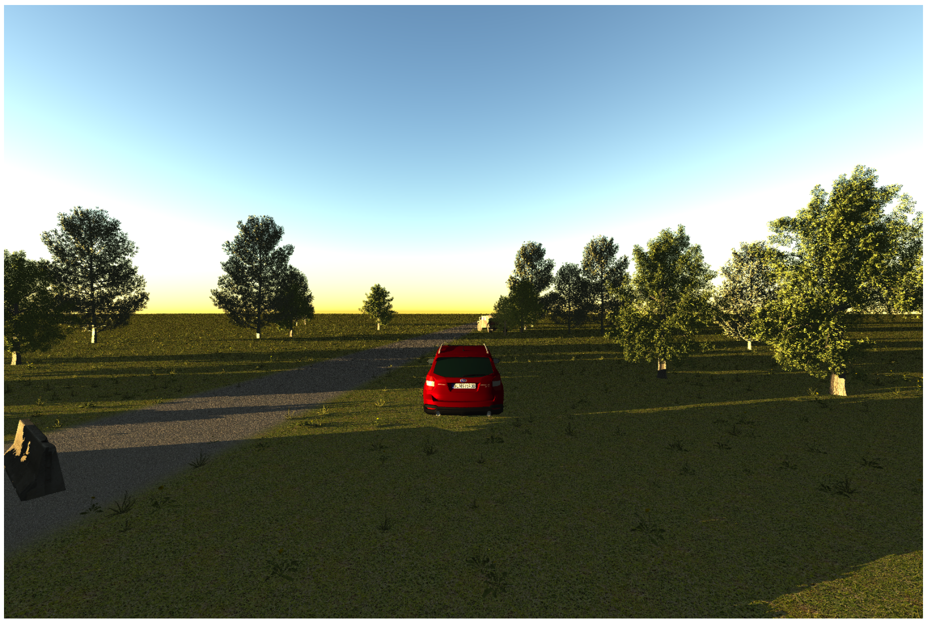

Figure 2.3

Example of MAVS terrain and lighting model.

The plant growth in the MAVS environment simulation is directed by defining "ecosystems," which contain parameters such as the typical plant type, size distribution, growth rate, and ability to compete with other vegetation. Current MAVS ecosystems include a forest in the Southeastern United States, a meadow, and a desert in the Southwestern United States. The desert and meadow ecosystems in MAVS are shown in Figure 2.4 and Figure 2.5, respectively.

2.3.2 MAVS LiDAR Simulation

The MAVS uses high–fidelity ray tracing, built on the Embree kernel [36] to simulate the physics of each LiDAR-beam pulse interacting with the environment. Each point of the LiDAR scan is simulated using nine rays, with the laser beam shape and divergence

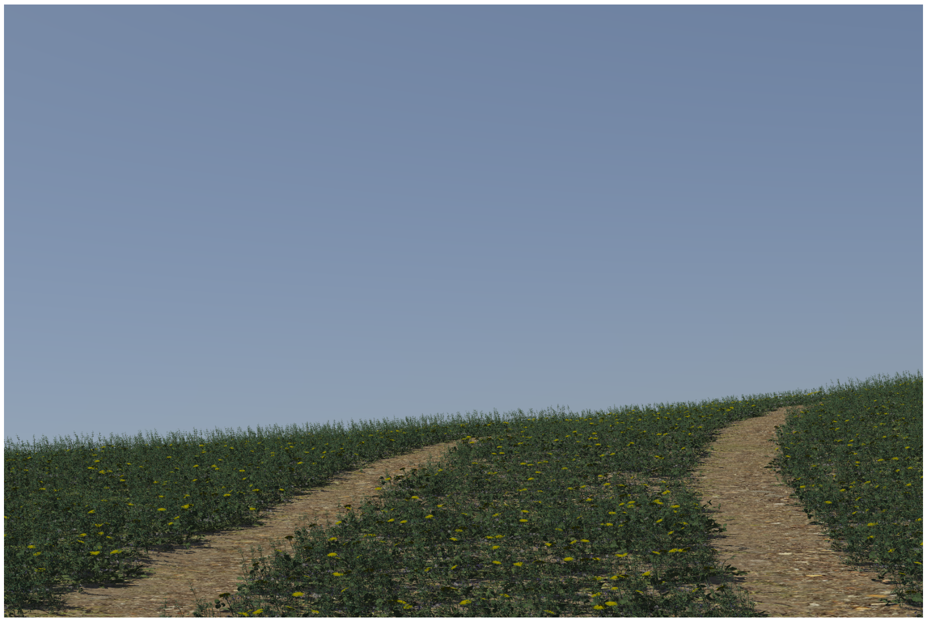

Figure 2.4

A simulated meadow scene.

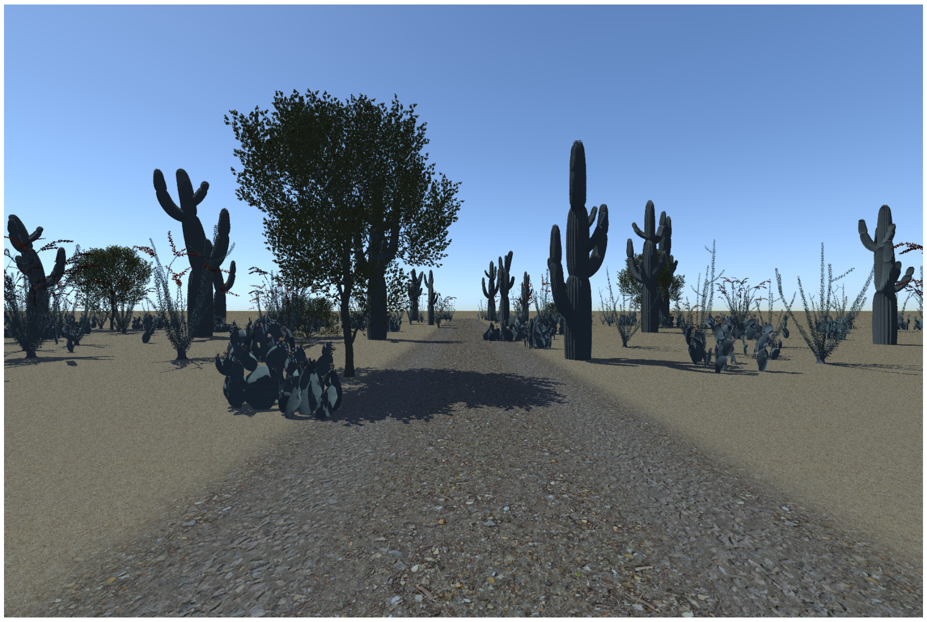

Figure 2.5

A simulated desert scene.

approximated by the orientation of the rays. For each ray, the intersections with the environment are found and the reflected intensity is calculated with a modified, physically based Phong model [20]. By oversampling the beam in this way, the effects such as multiple returns and mixed pixels can be simulated [8]. The effect of participating media like dust and rain are calculated in MAVS using a power–law scattering formula [7, 19].

2.4 LiDAR Calibration

Offline calibration methods are some of the simplest, most popular approaches to intrinsic LiDAR calibration [26] N. Muhammad and S. Lacroix present an offline, intrinsic LiDAR calibration method inspired by the calibration parameters used in a Velodyne, 64–beam LiDAR system [26]. They used data collected by the multi–beam LiDAR to scan a flat surface. A non–linear equation solver was used to tune five calibration parameters for each of the 64 sensors associated with the beams of the device. The cost function used was directly related to the related to the distance measured from the surface. Since the theoretical variance across all points across a perfectly flat plane is zero, minimizing the variance in the distance of these measurements is an effective and commonly used method to calibrate the LiDAR's intrinsic range measurements [18, 26]. Intrinsic calibration, although effective, is often not emphasized as much as extrinsic calibration because the intrinsic properties of the sensor are often not subject to change.

Levinson and Thrun show that intrinsic calibration can be accomplished in an unstructured environment [18]. Their method is accomplished using data collected while the sensor is moving through the environment, creating a more densely populated point cloud.

This is quite different from Muhammad and Lacroix's method which used data recorded in a structured, stationary environment. However, the basic goals of the cost functions are the same. Each are trying to reduce the distance between a point measurement and a plane with a calculated normal. However, in the case with Levinson and Thrun's work, the normal of this plane is defined by the nearest neighbors of the point selected for measurement. This is done under the assumption that most adjacent points lie on contiguous surfaces. Due to computational constraints, some concessions were made such as only adjusting one calibration parameter per sensor, per iteration. Because of this concession the cost function is not guaranteed to decrease after each iteration, whereas Muhammad and Lacroix's method, as a strictly offline algorithm, modifies each calibration parameter for every sensor and iteration using the appropriate non-linear solver. I chose to use Muhammad's and Lacroix's method for intrinsic calibration for its simplicity and because it does not require other forms of localization hardware to combine point cloud measurements. Additionally, modifying all calibration parameters per beam simultaneously reduces the chance that the cost function converges to a local minimum.

For extrinsic calibration between multiple LiDAR sensors, there has been more motivation to develop more automatic, online calibration methods [5, 17, 23]. He et al. [12] examined calibrating two LiDARs in a noisy industrial/city environment. They extracted features such as lines, planes and cones and optimized the feature matching by minimizing geometric distances between the two LiDARs. This method worked well in that environment but might not be suitable for a more–complex wooded environment where the geometric features used may not be readily available. [12]. Maroli et al. developed a method to estimate roll, pitch and yaw for multiple LiDARs by using ground plane alignment for roll and pitch and a genetic algorithm for yaw estimations [23]. Jaw et al. co–register LiDARs by extracting 3D lines [16]. Again, this method would not be suitable for wooded applications due to their high complexity and lack of clean lines. There are also papers that address LiDAR/Camera co–registration, such as Habib et al. [9], Ding et al. [3], Pandey et al. [29], and Levinson et al [17]. However, the focus of this paper is multi–LiDAR calibration and co–registration.

2.5 LiDAR Pose Analysis

Although much research has been devoted to calibrating multi–LiDAR systems, there is seemingly very little on the topic of how their physical positioning may affect their performance.

Some work was completed previously on the Halo project vehicle to characterize the coverage area of two front mounted LiDAR sensors [14]. Their paper analyzed the simulated ground coverage of the beams mounted at various angles. My research builds upon these results by simulating a wider variety of poses and uses a different metric to grade each pose. When simulating the pose of the sensors, Hudson, Goodin, Doude, and Carruth varied the mounting angles about one axis of rotation, leaving the other fixed. I believe that rotating a sensor about a second axis could result in a more optimum placement of the sensor. Additionally, Hudson et al. considered the point cloud density in front of the vehicle as the grading metric for each sensor placement. Alternatively, I will analyze the effectiveness of each orientation using a neural network, similarly how the data would be processed by the autonomous driving computer. I acknowledge that the point cloud density in certain areas of the point cloud generated by the sensors does not necessarily correlate to the autonomous driving computer's ability to make a correct classification.

CHAPTER 3

METHODOLOGY

The LiDARs simulated and evaluated were Quanergy M8 LiDARs [32]. This sensor was a rotating, 3D, scanning LiDAR with eight beams paced approximately three degrees apart. The Quanergy LiDARs were chosen based on an overall assessment of their capabilities and cost. Furthermore, researchers at CAVS had experience with these LiDARs on previous projects related to industrial autonomous vehicles [38, 37]. Figure 3.1 shows a picture of the Quanergy M8 LiDAR and Figure 3.2 shows the eight beam locations relative to the $z = 0$ plane of the LiDAR (this would be the horizontal plane if the LiDAR is mounted upright). The LiDAR provides $360°$ coverage with eight beams, and scans at rates from 5 to 20 Hz. The point cloud data is output through a gigabit ethernet interface.

3.1 LiDAR Pose Analysis

The multi–LiDAR system involved in this paper consisted of three Quanergy M8 LiDAR sensors.[32]. One sensor was mounted in an upright position at the top of the vehicle, giving it a relatively unobscured, $360°$ view of the environment. The other two sensors were mounted at oblique angles on either side of the front bumper as shown in figure 2.1. The vehicle, a modified Subaru Forester, was designed as an off-road-capable autonomous vehicle. The intent of placing the additional LiDAR sensors low and near the outside of the

Figure 3.1

Quanergy M8 LiDAR.

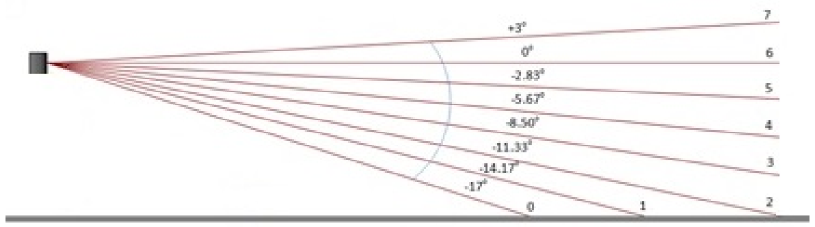

Figure 3.2

LiDAR beams.

car was to gather more information about the terrain directly in front of the vehicle and near its sides. The experiments described in subsequent sections were intended to determine the optimal positioning of the two sensors at the front of the vehicle while the top LiDAR remained fixed. The positions of the front sensors were mirrored such that rotations applied to one, were mirrored when applied the other LiDAR.

I propose that the optimum positioning of the LiDARs in this multi–LiDAR system can be determined using simulated data evaluated using a neural network. This method is focused on applications in autonomous vehicles, since most autonomous systems rely on neural networks to classify objects in their environment. This makes the neural network that processes LiDAR data in the vehicle, the most relevant qualifier of how well the LiDAR positioning may perform in the vehicle. Simulated data was generated for all LiDARs with various poses on the vehicle. Each pose was defined by two Euclidean angles, α and β, representing rotations about the x and y axis of the vehicle, as shown in Figure 3.4. There was no need to rotate the sensor about the z axis, as this is the axis of rotation of the beams and would have no effect on the data. For example, when $\alpha = 90°$, the tops of both sensors are pointed outwards from the sides of the vehicle, and when $\alpha = -90°$, both sensors are pointed inwards to the vehicle. Similarly, when $\beta = 90°$ the tops of the both sensors are pointed forward relative to the vehicle, and when $\beta = -90°$, they are oriented backwards. Throughout this analysis, I considered a limited range of viable poses: $\alpha = [45°, 90°]$ and $\beta = [31°, 68°]$. This range was chosen for practicality, reducing the search space and recognizing the physical limitations in the mounting system in the bumper.

For each simulation at a given pose, the data generated was used to train SqueezeSeg [39], the neural network used to process data on the vehicle. SqueezeSeg is a convolutional neural network–based model for LiDAR point cloud segmentation. SqueezeSeg projects a 3D LiDAR point cloud onto a spherical surface and uses a 2D CNN to predict point-wise labels. Structurally, it is based off SqueezeNet [15], and it uses the squeeze and excitation network [13] strategy. Other similar methods include PointNet by Qi et al. [30], however, PointNet is more computationally complex than SqueezeSeg.

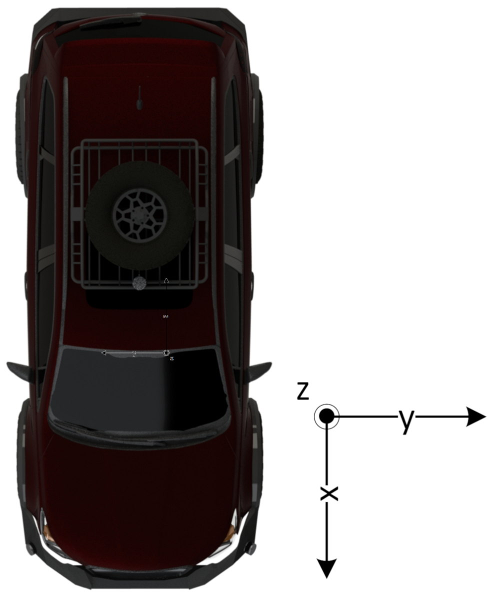

Figure 3.3

Vehicle coordinate system.

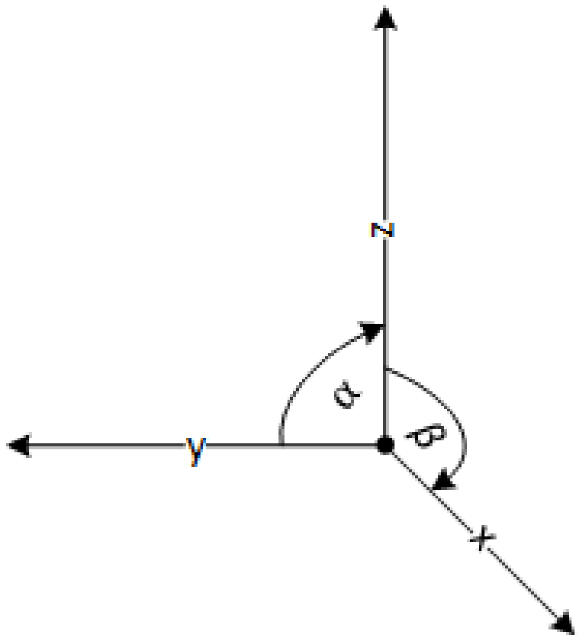

Figure 3.4

Rotation coordinates for the front LiDARs.

3.1.1 MAVS Simulation

Rather than performing tedious LiDAR data collection and labeling, MAVS was used to generate the large amounts of labeled data required to train and test SqueezeSeg. In simulation, three instances of the Quanergy M8 LiDAR were initialized at their positions on the vehicle model as discussed in section 3.1. Next, randomly generated environments based on three common types of biomes, were created. These biomes included a forest, a meadow and a desert, each with a flat surface and appropriate vegetation. Each point within the environment was labeled as either ground, vegetation, or a tree. In the simulated LiDAR data, these labels are represented at 1, 2, and 3 respectively. Simulated data from each sensor was generated as the vehicle moved through the environment. Rather than generating data from the sensors using a very large environment, it was more computationally efficient to generate data of the sensors moving through multiple, smaller instances of the randomized environment. The output data consisted of a sparse point cloud as is typically observed with LiDAR data. Each point in the point cloud contains data about its coordinates in 3D space, the intensity of the reflection as seen by the LiDAR, and the label of the object associated with that point.

3.1.2 Data Projection

Each simulated data point consisted of vector $a = \{x, y, z, \rho, l\}$, where x, y, and z represent the location of each point in 3D coordinates. ρ represents the intensity of the reflection measured by the sensor, and l is a label associated with each point. It is worth noting that MAVS generates the labels of each point automatically. If the analysis used collected LiDAR data, then point labels would have been created manually, either by some algorithm or by humans. In either case, the labeling process would have been time–consuming and would have most likely contained errors due to the large number of points involved. This provided a significant advantage to using simulated data.

Prior to training the neural network, the LiDAR data was transformed to spherical coordinates. This was done to represent the sparse, three–dimensional LiDAR point cloud as a denser two-dimensional image that can be used in a convolutional neural network such as SqueezeSeg. The process of doing so was largely adapted from Wu et al. [39]. For each pose in a simulated scene, point cloud data from all three LiDARs was combined in a unified, 3D coordinate system. Similarly to [39], only data with a positive valued

x-coordinate was considered. However, unlike Wu et al. [39], I was able to utilize data spanning the full $180°$ front field of view of the vehicle rather than being limited to a $90°$ FOV. Using equations 3.1 and 3.2, the $H \times W \times C$ tensors used as the input for the network were formed. H and W respectively describe the height and width of the tensor. Values of zenith (θ) and azimuth (ϕ) angles were evenly sampled to form the $H \times W$ image using equations 3.1 and 3.2. The vector C contains similar information to a, where $C = \{x, y, z, \rho, d, l\}$. d represents the range to the target in meters, calculated by $d = (x^2 + y^2 + z^2)^{0.5}$. In the case that there were multiple data points sampled to one index of H and W, the point with the highest intensity value was used. If there was no data associated with a pixel, C remained a zero vector as it was initialized. This introduced a fourth label, 0, indicating an unknown data point that was representative of free space.

$$\theta = arcsin\frac{z}{\sqrt{x^2 + y^2 + z^2}} \tag{3.1}$$

$$\phi = arcsin\frac{y}{\sqrt{x^2 + y^2}} \tag{3.2}$$

Previously with SqueezeSeg and other neural networks designed to process data from a single LiDAR, H is equal to the number of beams in the sensor. This is because when considering single LiDAR, the data in the H dimension is limited by the number of beams. However, this is not the case in a multi–LiDAR system where the sensors are mounted at oblique angles relative to one another. The limited vertical resolution of one LiDAR can be supplemented by the high angular resolution of another. For this reason, I was free to select H based upon the computational efficiency and performance of the neural network. Prior to performing the pose analysis, I compared the performance capabilities of SqueezeSeg trained on simulated data with various values of H. Each of these trials used a common dataset that had been resampled to varying values of H in order to determine which resolution would be ideal for the pose analysis.

3.1.3 SqueezeSeq Qualification

I used SqueezeSeg as a benchmark for several simulated poses of the LiDAR sensors. For each pose, equal sized batches of labeled data were generated to train and evaluate the network. The batch sizes and network parameters are listed in table Table 3.1. Configuration parameters not listed retained their default values. The labels used to train the network were the labels introduced in section 3.1.2. Each frame in a batch was converted to images using the process described in section 3.1.2. During this process the total mean and standard deviation of all the images in the batch were calculated. The vectors of the mean and standard deviation were used to normalize each of the batches by subtracting the mean and dividing by the standard deviation. The network was trained by starting by initializing pre-existing weights, included with SqueezeSeg, that were generated by training using the KITTI dataset [6, 39]. The network was then trained with the MAVS simulated imagery using the parameters listed in table Table 3.1.

The trained network was evaluated using a separate evaluation set of images, not included in the training set. I used the confusion matrix from the evaluation set to classify the performance of each sensor pose. Particularly, I used the user's accuracy calculated from the confusion matrix according to equation 3.3 as the accuracy metric.

Table 3.1

SqueezeSeg training and evaluation configuration parameters.

SqueezeSeg Network Parameters	
Learning rate	0.005
Learning decay rate	0.1 per 20,000 iterations
Number of iterations	25,000
Momentum	0.9
Input size	$8 \times 512 \times 5$
Batch size	12 images
Training set size	1296 images
Evaluation set size	324 images
Number of classes	4

$$UserAccuracy = \frac{TruePositives}{TotalClassifications} \tag{3.3}$$

3.2 Intrinsic LiDAR Calibration

For each of the three Quanergy M8 LiDARs, I used Muhammad's and Lacroix's method to perform the intrinsic calibration [26]. I recorded $N = 4$ measurements taken of a flat wall at one–meter increments ranging from two to five meters. A greater range of measurements would have been preferred, however, the $21.45°$ vertical FOV [32] of each LiDAR would require a larger wall than I readily had access to. I found it more beneficial to make single measurements, where all beams would simultaneously intersect the wall than attempting to limit the vertical FOV. The point cloud data was segmented by restricting the azimuth angle, ϕ, for each measurement. This was done by observation such that every point that did not lie on the reference wall was removed from the data. The same restriction of the azimuth angle was applied to data from each beam so that each sensor had a nearly equal number of data points associated with it in order to remove bias from the calibration. The point cloud data was represented in three dimensional coordinates, where each point was represented by a position vector [x, y, z]. For a single scan at a given distance, I used principal component analysis to determine the plane that best fit the data. The normal vector of this plane, $\vec{n}$ was determined by the third principal component, as the first two components define the axes along the plane. This plane is representative of the ideal, flat wall that I intended to measure. As seen in equation 3.4, the error of each scan, e_i was calculated as the perpendicular distance of each point from the plane after the current calibration parameters, K were applied. [P_x, P_y, P_z] represented the calibrated position vector, and [$\bar{x}$, $\bar{y}$, $\bar{z}$] was the mean position vector for a scan. The mean squared error (mse) over all scans was calculated according to equation 3.5.

$$e_i = ([P_x, P_y, P_z]] - [\bar{x}, \bar{y}, \bar{z}]) \cdot \vec{n} \tag{3.4}$$

$$mse = \frac{1}{N} \sum_{i=1}^{N} mean(e_i^2) \tag{3.5}$$

The calibration parameters contained in K were defined by Muhammad and Lacroix as the distance correction (D_c), vertical offset (V_o), and horizontal offset (H_o) in meters, and the vertical angle (θ_c), and rotational angle (ϵ) in radians.

Equations 3.8, 3.9, and 3.10 show how the calibrated point coordinates, P_x, P_y, and P_z, are calculated from these five calibration parameters [26]. Parameters such as D_{ret}, the Euclidean distance from the sensor to a point in meters and D_{xy}, that distance projected onto the xy–plane were calculated using $D_{ret} = (x^2 + y^2 + z^2)^{0.5}$ and $D_{xy} = (x^2 + y^2)^{0.5}$. β_c represents the rotational angle of the LiDAR in radians after the rotational angle correction has been applied ($\beta_c = tan^{-1}(\frac{y}{x}) - \epsilon$). Because manufacturing variances can occur independently between each transmitter and receiver in the LiDAR sensor, each beam had its own set of calibration parameters associated with it.

$$D = D_{ret} + D_c \tag{3.6}$$

$$D_{xy} = D * cos(\theta_c) - V_o * sin(\theta_c) \tag{3.7}$$

$$P_x = D_{xy} * sin(\beta_c) - H_o * cos(\beta_c) \tag{3.8}$$

$$P_y = D_{xy} * cos(\beta_c) + H_o * sin(\beta_c) \tag{3.9}$$

$$P_z = D * sin(\beta_c) + V_o * cos(\theta_c) \tag{3.10}$$

The calibration parameters stored in K were determined iteratively using a non–linear programming solver, $fmincon$, from MATLAB 2017b with the optimization toolbox, to minimize the mse of the point cloud data as different calibrations were applied. The function $fmincon$ was run with its default settings, aside from increasing the "Max Function Evaluation" parameter from its default value of 3000 to 40,000 to remove it as a binding constraint. By default, $fmincon$ uses the interior point algorithm to minimize a function [24]. The objective function was $\min\limits_{K} E(K, x)$ where $E(K, x)$ was a function that calculated e_i for a given LiDAR point cloud and calibration matrix, K such that mse is minimized. The output of this function, K was represented as an 8×5 numeric matrix. Each row contained the five calibration parameters for a single beam in the sensor. Prior to the minimization of mse, K was initialized to its expected values, $K_n = [0, 0, 0, v, 0]$ where v is the advertised vertical angle for each of the n beams [32].

3.3 Extrinsic LiDAR Calibration

To combine data from all three sensors to a common frame of reference, extrinsic calibration was needed. To facilitate this the LiDARs were installed on the vehicle and positioned with poses that were within our parametric search space. I used Quanergy's Q-view application to visually inspect and align common features within the common area of interest between all three sensors [31]. Furthermore, the application was used to refine the calibration after the user had sufficiently aligned the images. Extrinsic calibration was performed on two sensors at a time and Q-View produced the position and quaternion rotation angles to combine the separate point clouds. With the precise rotation and positional offsets known, point cloud data from each LiDAR in the system was transformed to the same frame of reference.

CHAPTER 4

RESULTS

Results from simulation and training SqueezeSeg and the evaluation of the calibrations are discussed below.

4.1 Data Projection

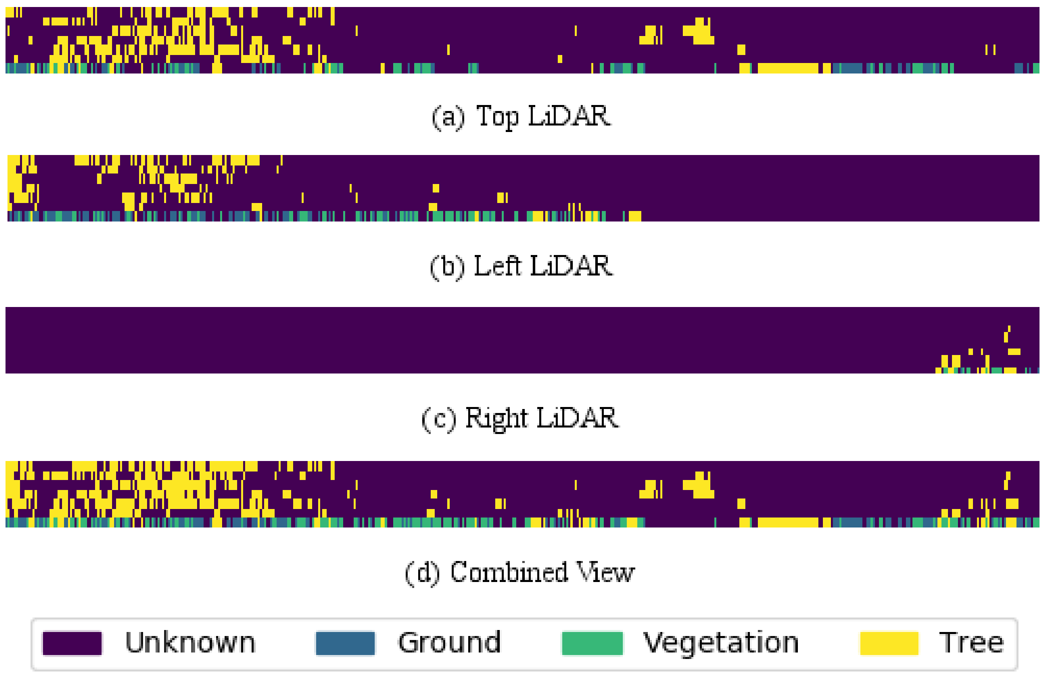

Figure 4.1

Simulated LiDAR data.

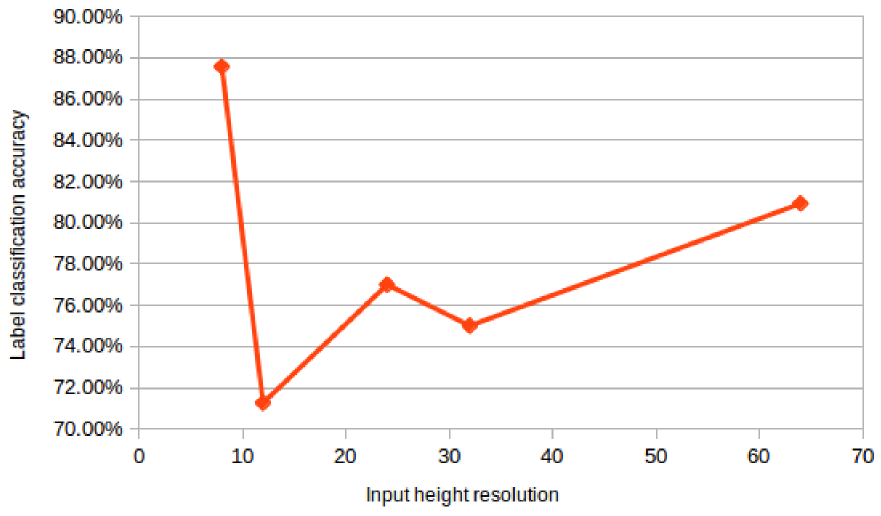

Figure 4.2

Label classification accuracy vs. the vertical resolution of the input image.

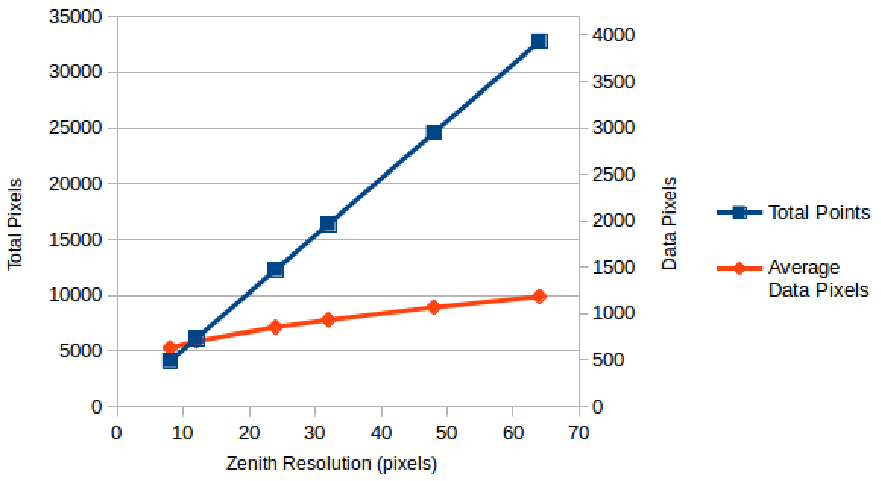

Figure 4.3

Average number of data pixels per image vs. increasing vertical resolution.

Figure 4.1 show the labels of individual LiDAR images after they are formed using the spherical projection process described in section 3.1.2. Each image presents some common and some unique data when shown in the combined view. Figure 4.2 depicts the label classification accuracy of SqueezeSeg vs. several different vertical resolutions for the input data. "Label classification accuracy" is defined at the percentage of labeled pixels classified correctly, not considering pixels with an "unknown" label. Doing this prevents the results from being skewed by high classification accuracy for pixels with the unknown label, since it is quite trivial for the neural network to learn how to classify null pixels with perfect accuracy. From Figure 4.3, increasing the vertical resolution increases the average amount of points in each image, as expected. However, doing so significantly increases the number of unknown points in the image. Figure 4.2 shows that the SqueezeSeg performed best when using the vertical resolution that is native to a single LiDAR. It is likely that the addition of more null pixels with higher vertical resolutions reduces the effectiveness of convolution operations in SqueezeSeg. It does seem that the classification accuracy does tend to increase with higher vertical resolutions, but the gains are not nearly significant enough to warrant the additional processing power required for a larger input. To compare training images with different vertical resolutions, Figure 4.4 shows the image sampled with $H = 8$ and Figure 4.5 shows the same image sampled with $H = 64$.

4.2 CNN Performance

Figure 4.6 shows the user accuracy of the trained segmentation neural network across all evaluated values of α and β. Some linear interpolation was done to produce an evenly

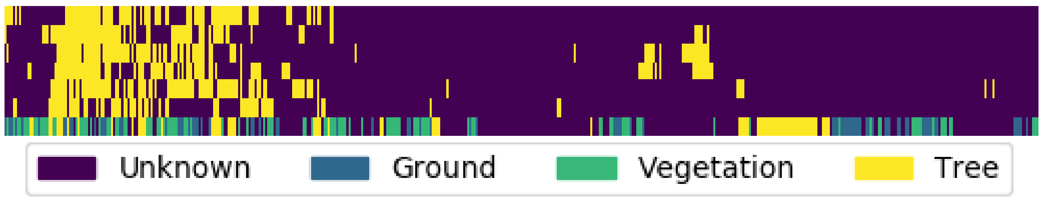

Figure 4.4

Simulated LiDAR data with an 8px vertical resolution

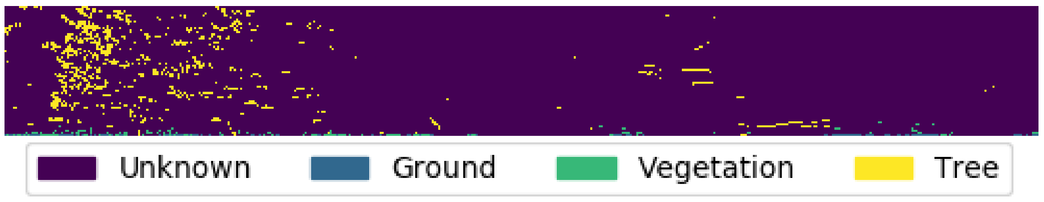

Figure 4.5

Simulated LiDAR data with a 64px vertical resolution

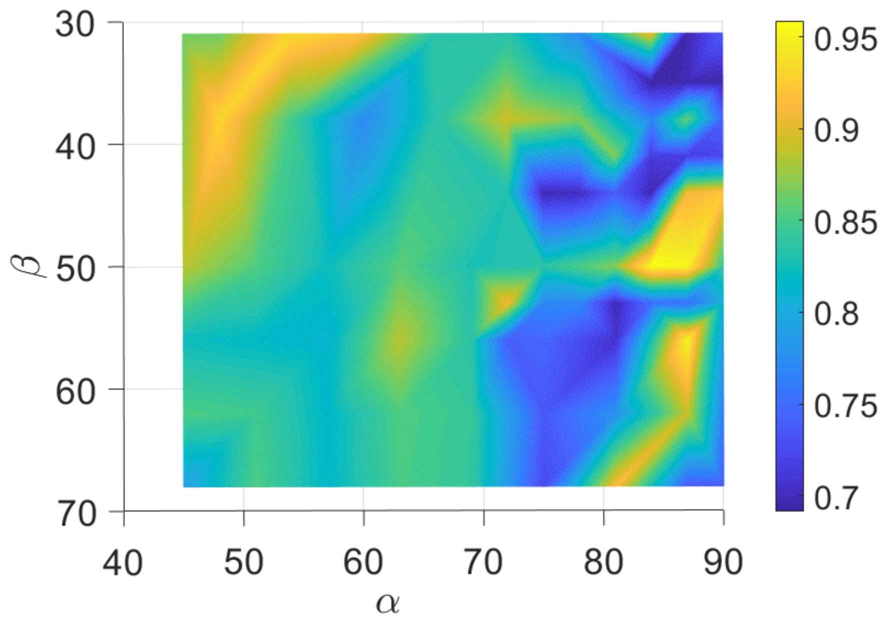

Figure 4.6

User accuracy of CNN across all tested positions

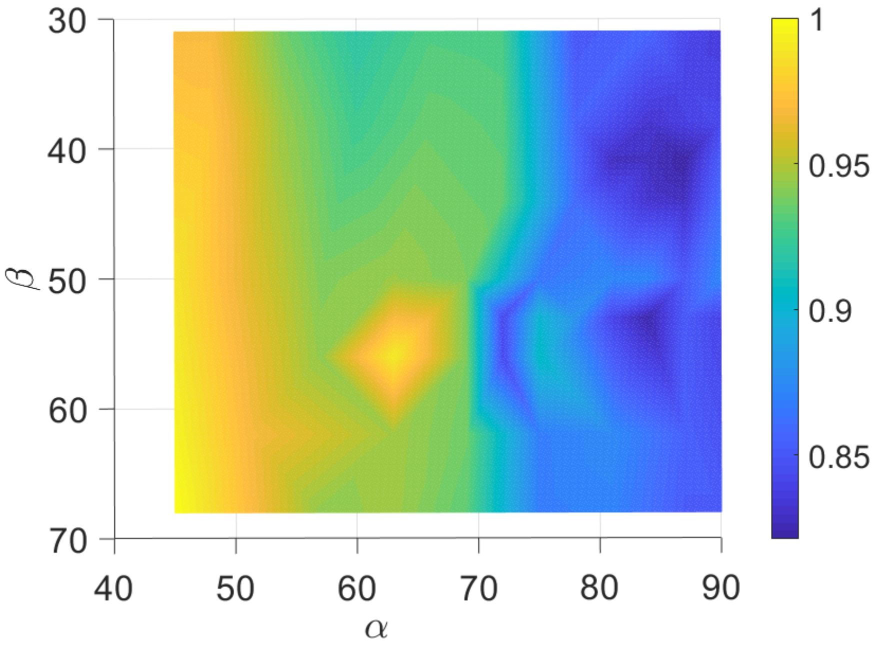

Figure 4.7

The normalized number of data points contained in an image set for each LiDAR pose.

sampled figure. Upon inspection it is difficult to identify an ideal pose for the sensors. Values of α near $90°$ produces varying results that include the highest and lowest accuracy rates observed. Lower values of α resulted in more consistent performance. Figure 4.7 depicts the total number of data points registered in the image set for each pose. When considering the amount of data in the image set, the positioning of the LiDAR can affect how much data falls within the field of view of the image. In our simulation, smaller values of α tended to produce data that fit better within the vertical field of view of the image. A clear trend across our simulations was that the amount of data seen by the neural network increased as α decreased and β increased, producing the maximum number of points in the $(45°, 68°)$ pose. The less data at higher values of α could also correlate with my observation of inconsistent user accuracy at those poses.

4.3 LiDAR Calibration

The minimizing function was able to find local minimum that satisfied its default optimality constraints within 40,000 iterations. Computationally, the program required approximately one minute to converge on a modern desktop workstation. This is significantly faster than the amount of time required suggested by Muhammad and Lacroix. Undoubtedly, the increase in speed was because I was calculating the calibration matrix for a LiDAR with several fewer beams (8 vs. 64 beams). An example of a calibration matrix for one LiDAR is listed in Table 4.1.

Unlike the 64 beam LiDAR used to develop this calibration method, the Quanergy M8 LiDAR does not seem to exhibit any characteristic patterns in its calibration parameters,

Table 4.1

Calibration Parameters for one LiDAR.

Beam	D_c(m)	V_o(m)	H_o(m)	θ_c(rad)	ϵ(rad)
7	-0.01473	-0.08415	0.0212	0.05277	-0.00551
6	-0.0107	-0.01875	0.01387	-0.01989	-0.00445
5	-0.01161	-0.01305	-0.00321	-0.06718	0.00205
4	-0.00563	0.09364	0.00566	-0.14457	-0.00078
3	-0.02251	0.00824	0.02617	-0.17176	-0.00777
2	0.00152	0.10712	0.01867	-0.2434	-0.00533
1	0.01086	0.0199	-0.00646	-0.28688	0.00155
0	-0.00565	0.08957	0.03551	-0.31713	-0.00771

such as the alternating horizontal offsets seen in the former. All parameters seem to be small, nominal values as expected, except the vertical offset for some beams, which sometimes deviated approximately 10 cm from its expected value of 0. The effect of this can be seen in beam zero and two (the first and third beams counted from the bottom) in Figure 4.8 and Figure 4.9. Despite these dramatic alterations to the image, the calibration reduced the average variance across all measurements by factors ranging from approximately 20–50% as seen in Table 4.1. Additionally, Table 4.3 shows that the maximum error of the measurements of the flat wall decreased by a significant magnitude for each LiDAR.

Figure 4.10 and Figure 4.11 show a scans of a flat surface after performing the extrinsic calibration between all three LiDARs and co–locating each of their point clouds. Upon inspection of the front view, Figure 4.10, there are several visual cues that indicate that the extrinsic calibration is not precise. The target is a rectangular board positioned so that its height was normal to the floor. Examining the top of the board the beam from the right LiDAR (in green) extends approximately 5cm above the extent of the beams from the left

Table 4.2

Average Variance of the LiDAR Data Before and After Calibration.

	Average Variance before Calibration (mm)	Average Variance after Calibration (mm)	Improvement
LiDAR A	168.0	84.64	49.62%
LiDAR B	55.17	42.83	22.36%
LiDAR C	65.44	31.92	51.22%

Table 4.3

Maximum Distance Error of the LiDAR Data at 5m Before and After Calibration.

	Maximum Difference before Calibration (cm)	Maximum Difference after Calibration (cm)	Improvement
LiDAR A	13.3	9.5	28.57%
LiDAR B	4.7	4.1	12.77%
LiDAR C	5.1	4.1	19.53%

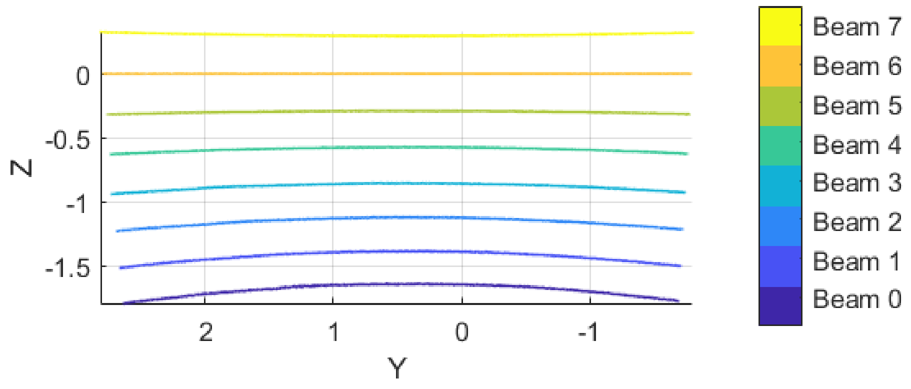

Figure 4.8

Wall measurement without calibration.

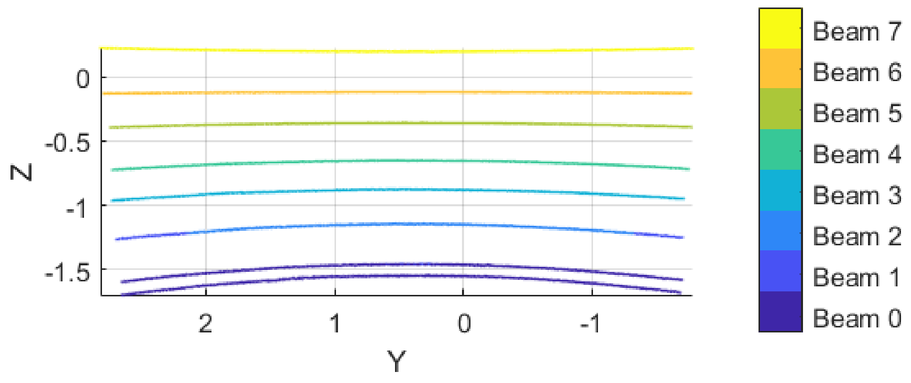

Figure 4.9

Wall measurement with calibration.

LiDAR (in blue). Additionally, I found that the right LiDAR data was rotated relative to the top LiDAR (in red) so that some beams fall within top sensor's beams of the wall while others have ends that do not. The side view, Figure 4.11 also shows that the sensors are generally co–registered but lack further refinement. The slight angle of the wall in the xz–plane is the result of a slight downward tilt in the top LiDAR. For this visualization all points were transformed to the frame of reference of the top LiDAR, giving a slightly skewed perspective.

The point cloud data shown in Figure 4.12 depicts the simultaneous beam pattern of all three sensors. The intersection of the three point clouds creates two areas of interest in front of the vehicle shown by Figure 4.13. The dense area of interest is representative of where three point clouds intersect, and the less dense area of interest is created by the intersection of at least two point clouds. The previous Figure 4.10 was measured within the dense area of interest. For the general sensor placement for the Halo vehicle, the area

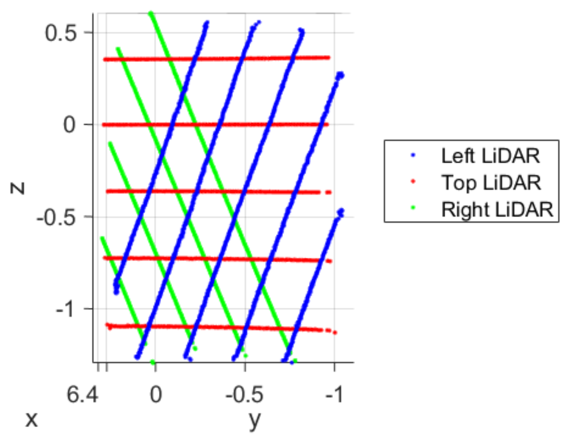

Figure 4.10

Front view of a flat wall.

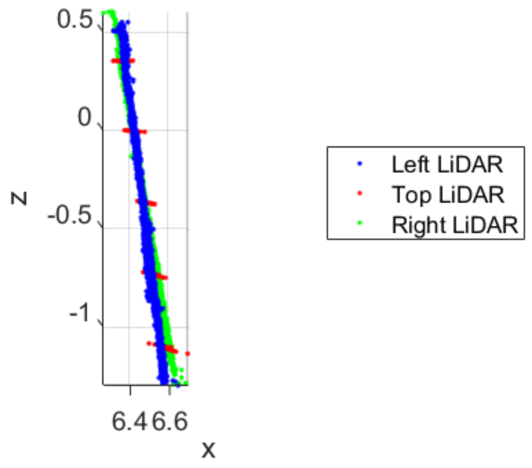

Figure 4.11

Side view of a flat wall.

of interest was located directly in front of the vehicle. The densest area is directly in front of the vehicle and four to eight meters in front of the vehicle, and the less dense area, spans a wider field of view and extends to the maximum range of the sensors.

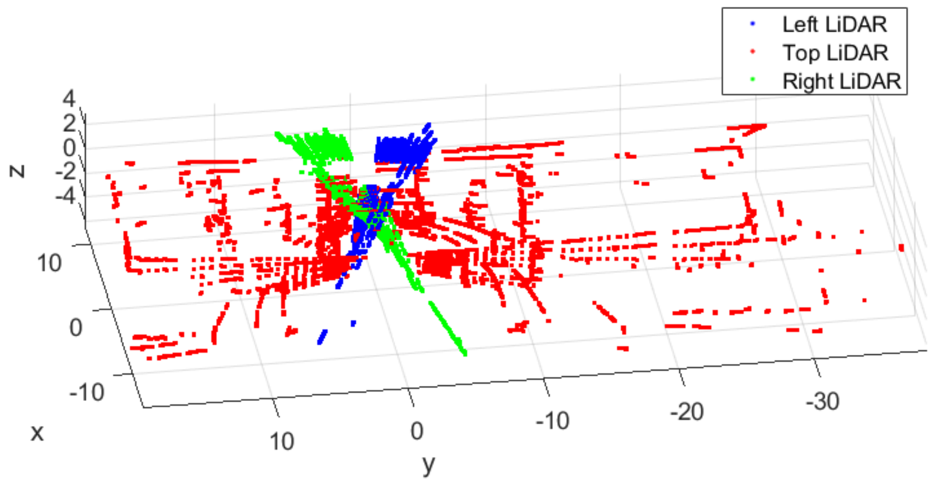

Figure 4.12

View of the co–located LiDAR point cloud data indoors.

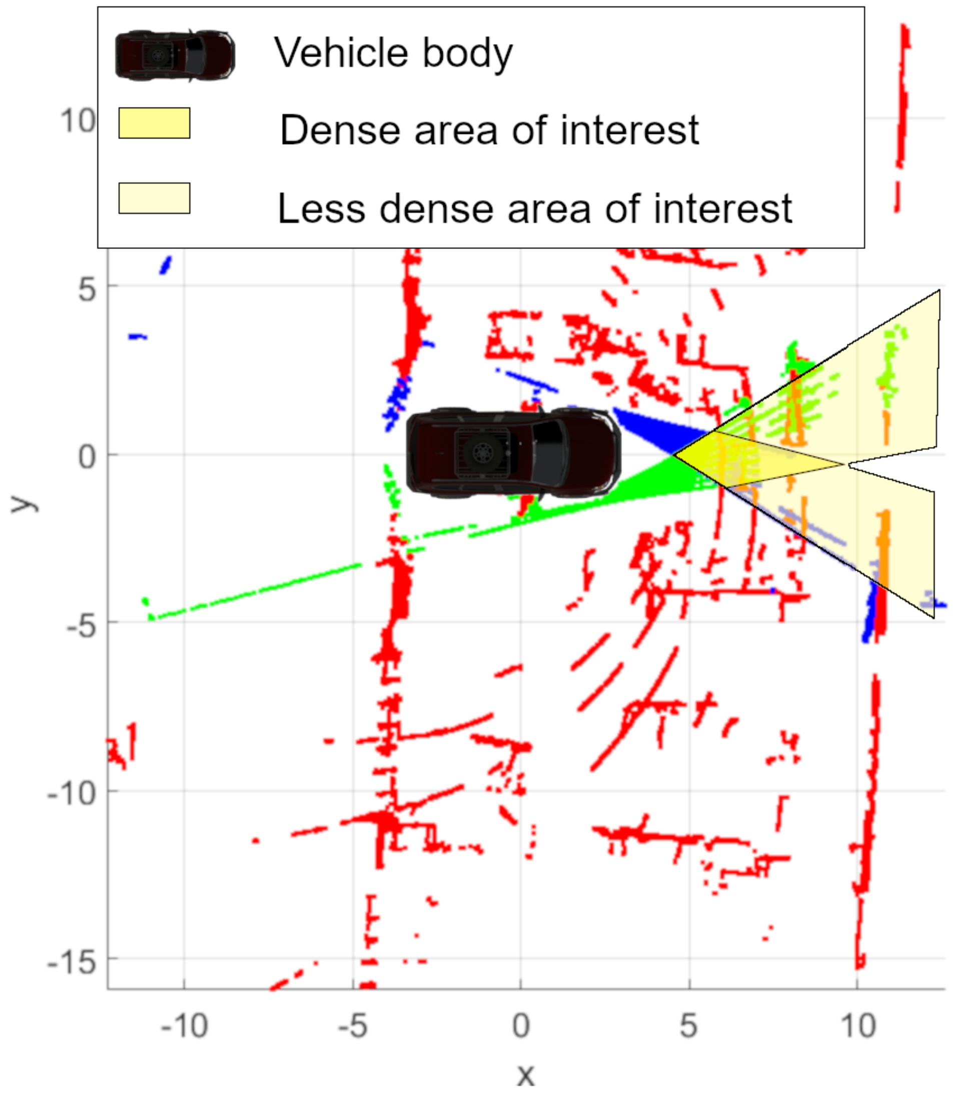

Figure 4.13

Top view of the two areas of interest for the vehicle.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

The usage of simulated data was crucial for the parametric, LiDAR pose analysis. The alternative in situ collection of such a large amount of data is logistically infeasible, especially considering the effort that would be required to label such data at single pixel resolution. Additionally, simulation facilitates a high level of precision and consistency that cannot be expected from in situ collection. I believe that the data is valid since I have observed similar performance from SqueezeSeg using either the simulated MAVS data or the KITTI data. MAVS continues to undergo further research and development. Currently, efforts are being made to evaluate is ability to generate realistic data from other sensors such as cameras.

Interpreting data from chapter 4.2, I determined that the best LiDAR positioning for the application of the Halo vehicle is approximately $\alpha, \beta = (50^{\circ}, 35^{\circ})$. Considering the user accuracy, this is not where the peak accuracy occurred. Since the peak accuracy occurred in a range that was particularly unstable, I determined that the more stable range of angles was more desirable. This is especially true considering the body of the vehicle can vary several degrees relative to the ground as the vehicle moves. Additionally, this rotation lies in a range of above–average–density data which can contribute to the neural network's accuracy. The number of points within the area of interest, and the shape of the area of interest are largely dependent on the positioning of the LiDAR sensors. Mounted at near perpendicular angles, the sensors will have a very small intersection producing the area of interest. Contrary, mounted at nearly parallel angles, the sensors will have the largest intersection of data and therefore the largest point cloud density.

When considering processing data from a multi–LiDAR system, I identified two viable methods. The first is the one presented in section 4.1 where data from all sensors is combined into an image representative of the system's area of interest. In the case of SqueezeSeg, this should be done without attempting to increase the vertical resolution of the area of interest, as I found that decreases accuracy and increases the computational complexity. An effective, alternative method of processing data from a multi–LiDAR system could be to use multiple, independent instances of SqueezeSeg for each LiDAR, where each instance would classify data from its respective sensor and consider the intersection between sensors. This would create multiple regions of interest and achieve a similar level of intersection in the images. The benefit of this approach is that each instance of squeeze set would operate using the native resolution of the eight–beam LiDAR, and more information can be processed, resulting in a wider FOV for the LiDARs if desired. In this implementation, the designer likely gains linear scalability and more efficient processing at the cost of higher processing overhead.

Considering the LiDAR rotation analysis, particularly Figure 4.6 and Figure 4.7, I am not confident that my simulation discovered an absolute optimum position for the total maximum range of rotation angles. This is because I only considered a limited range of poses that were applicable to the Halo vehicle. Further testing could discover more optimal posed. For example, as seen in Figure 4.7, the total number of points per image set seems to consistently increase as α achieves values lower than our test interval. The higher density of points could yield better classification results and may be explored in the future by considering a wider range of poses that can be deployed on a future vehicle.

The results from chapter 4.3 show that for the individual LiDARs, intrinsic calibration beyond the factory calibration can improve the accuracy. By showing that calibration using my method improves the error at every range tested, I can conclude that the intrinsic calibration improves the accuracy of the measurement at all tested ranges without introducing inaccuracies at some ranges. Error could be introduced to a point cloud if a calibration minimizes the total mean squared error but introduces less significant, localized errors. Using a flat wall as an intrinsic calibration target is beneficial because of its simplicity. However, the flat wall's simplicity may be the cause of the large vertical offsets seen in some of the beams when using this calibration method. Although, I've found no significant evidence of correlation between the vertical offset parameter and other calibration parameters, it is possible that some correlation exists. Such a correlation could explain the calculation of large vertical offsets while continuing to reduce the mean square error. Using a calibration target with more sophisticated geometry would increase the complexity of the calibration procedure but may increase independence between the calibration parameters.

Additionally, more precise methods for extrinsic calibration should be investigated. The simple Q–view tool was not adequate for the type of LiDAR system built for the Halo vehicle. Q–view's fine–tuning algorithm did not perform well when there was not significant intersection between the two point clouds under calibration. Additionally, performing the course alignment from the human perspective is quite difficult and time consuming. This task becomes more complicated and less accurate as the rotational axis of each sensor approaches orthogonal angles with other LiDAR sensors such as with the case of the Halo system.

CAVS has also recently invested in developing a dedicated, off-road autonomous mobility test track. A 50-acre site incorporates uneven terrain, mature trees, lowlands, and various natural features with which to benchmark autonomous system capability. Plans are in place to add further differentiation of soil types, including sand and rocks. The autonomous vehicle test site will provide a realistic and varied environment with which to validate the autonomous system and the effectiveness of the LiDAR system, trained using simulated data from MAVS, to generalize real world input.

REFERENCES

[1] X. Chen, "Engineering Uber's Self-Driving Car Visualization Platform for the Web,", Dec 2018.

[2] O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz, "Realistic modeling and rendering of plant ecosystems," *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 1998, pp. 275–286.

[3] M. Ding, K. Lyngbaek, and A. Zakhor, "Automatic registration of aerial imagery with untextured 3d lidar models," *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[5] C. Gao and J. R. Spletzer, "On-line calibration of multiple LIDARs on a mobile vehicle platform," *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 279–284.

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[7] C. Goodin, D. Carruth, M. Doude, and C. Hudson, "Predicting the Influence of Rain on LIDAR in ADAS," *Electronics*, vol. 8, no. 1, 2019, p. 89.

[8] C. Goodin, M. Doude, C. Hudson, and D. Carruth, "Enabling off-road autonomous navigation-simulation of LIDAR in dense vegetation," *Electronics*, vol. 7, no. 9, 2018, p. 154.

[9] A. Habib, M. Ghanma, M. Morgan, and R. Al-Ruzouq, "Photogrammetric and LiDAR data registration using linear features," *Photogrammetric Engineering & Remote Sensing*, vol. 71, no. 6, 2005, pp. 699–707.

[10] T. Hanke, N. Hirsenkorn, B. Dehlink, A. Rauch, R. Rasshofer, and E. Biebl, "Generic architecture for simulation of ADAS sensors," *2015 16th International Radar Symposium (IRS)*, June 2015, pp. 125–130.

[11] J. C. Hart, "Perlin noise pixel shaders," *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*. ACM, 2001, pp. 87–94.

[12] M. He, H. Zhao, F. Davoine, J. Cui, and H. Zha, "," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1828–1835.

[13] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," *CoRR*, vol. abs/1709.01507, 2017.

[14] C. Hudson, C. Goodin, M. Doude, and D. Carruth, "Analysis of Dual LIDAR Placement for Off-Road Autonomy Using MAVS," 08 2018, pp. 137–142.

[15] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$1MB model size," *CoRR*, vol. abs/1602.07360, 2016.

[16] J.-J. Jaw and T.-Y. Chuang, "Registration of ground-based LiDAR point clouds by means of 3D line features," *Journal of the Chinese Institute of Engineers*, vol. 31, no. 6, 2008, pp. 1031–1045.

[17] J. Levinson and S. Thrun, "Automatic Online Calibration of Cameras and Lasers," 06 2013.

[18] J. Levinson and S. Thrun, *Unsupervised Calibration for Multi-beam Lasers*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 179–193.

[19] P. A. Lewandowski, W. E. Eichinger, A. Kruger, and W. F. Krajewski, "Lidar-based estimation of small-scale rainfall: Empirical evidence," *Journal of Atmospheric and Oceanic Technology*, vol. 26, no. 3, 2009, pp. 656–664.

[20] R. R. Lewis, "Making shaders more physically plausible," *Computer Graphics Forum*. Wiley Online Library, 1994, vol. 13, pp. 109–120.

[21] C. Maag, D. Muhlbacher, C. Mark, and H. Kruger, "Studying Effects of Advanced Driver Assistance Systems (ADAS) on Individual and Group Level Using Multi-Driver Simulation," *IEEE Intelligent Transportation Systems Magazine*, vol. 4, no. 3, Fall 2012, pp. 45–54.

[22] A. C. Madrigal, "Waymo Built a Secret World for Self-Driving Cars,", Dec 2018.

[23] J. M. Maroli, Ü. Özgüner, K. Redmill, and A. Kurt, "Automated rotational calibration of multiple 3D LIDAR units for intelligent vehicles," *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.

[24] Mathworks, "Constrained Nonlinear Optimization Algorithms - MATLAB & Simulink,", https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html#brnpd5f, Accessed 3/19/19.

[25] W. Meadows, C. Hunson, C. Goodin, L. Dabbiru, B. Powell, M. Doude, D. Carruth, M. Islam, J. E. Ball, B. Tang, and et al., "Multi–LiDAR placement, calibration, co–registration and processing on a Subaru Forester for off-road autonomous vehicle operation," *Defense Commercial Sensing*. May 2019, Society of Photographic Instrumentation Engineers.

[26] N. Muhammad and S. Lacroix, "Calibration of a rotating multi-beam lidar," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 5648–5653.

[27] M. S. U. Newsroom, "'Halo Project' supercar continues automotive engineering excellence at MSU,", Jan 2018.

[28] M. S. U. Newsroom, "MSU debuts 'Halo Project' supercar in Las Vegas,", Nov 2018.

[29] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic Extrinsic Calibration of Vision and Lidar by Maximizing Mutual Information," *Journal of Field Robotics*, vol. 32, no. 5, 2015, pp. 696–722.

[30] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[31] I. Quanergy Systems, "Q-view,", http://downloads.quanergy.com/#qview, 2019, Accessed 3/19/19.

[32] *M8 Sensor User Guide*, Quanergy Systems, Inc., 482 Mercury Dr. Sunnyvale, CA 94085-4706, 2018.

[33] M. Roser and H. Ritchie, "Land Use: Our World in Data,", https://ourworkdindata.org/land-use#land-cover-of-the-world-today, 2019, Accessed 3/19/19.

[34] A. Tallavajhula, *Lidar Simulation for Robotic Application Development: Modeling and Evaluation*, doctoral dissertation, 2018.

[35] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. v. Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, 9 2006, pp. 661–692.

[36] I. Wald, S. Woop, C. Benthin, G. S. Johnson, and M. Ernst, "Embree: a kernel framework for efficient CPU ray tracing," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, 2014, p. 143.

[37] P. Wei, J. E. Ball, and D. T. Anderson, "Fusion of an Ensemble of Augmented Image Detectors for Robust Object Detection," *Sensors*, vol. 18, no. 3, 2018, p. 894.

[38] P. Wei, L. Cagle, T. Reza, J. Ball, and J. Gafford, "LiDAR and Camera Detection Fusion in a Real-Time Industrial Multi-Sensor Collision Avoidance System," *Electronics*, vol. 7, no. 6, 2018, p. 84.

[39] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud," *CoRR*, vol. abs/1710.07368, 2017.