



University of Kentucky
UKnowledge

Theses and Dissertations--Biosystems and
Agricultural Engineering

Biosystems and Agricultural Engineering


2018

ASSESSING THE SPATIAL ACCURACY AND PRECISION OF LIDAR FOR REMOTE SENSING IN AGRICULTURE

Surya Saket Dasika

University of Kentucky, surya.dasika@uky.edu

Author ORCID Identifier:

 <https://orcid.org/0000-0001-9031-887X>

Digital Object Identifier: <https://doi.org/10.13023/etd.2018.266>

[Right click to open a feedback form in a new tab to let us know how this document benefits you.](#)

Recommended Citation

Dasika, Surya Saket, "ASSESSING THE SPATIAL ACCURACY AND PRECISION OF LIDAR FOR REMOTE SENSING IN AGRICULTURE" (2018). *Theses and Dissertations--Biosystems and Agricultural Engineering*. 56.

https://uknowledge.uky.edu/bae_etds/56

This Master's Thesis is brought to you for free and open access by the Biosystems and Agricultural Engineering at UKnowledge. It has been accepted for inclusion in Theses and Dissertations--Biosystems and Agricultural Engineering by an authorized administrator of UKnowledge. For more information, please contact UKnowledge@lsv.uky.edu.

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Surya Saket Dasika, Student

Dr. Michael Sama, Major Professor

Dr. Donald Colliver, Director of Graduate Studies

ASSESSING THE SPATIAL ACCURACY AND PRECISION OF
LIDAR FOR REMOTE SENSING IN AGRICULTURE

THESIS

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science in Biosystems and Agricultural Engineering in the
College of Engineering
at the University of Kentucky

By

Surya Saket Dasika

Lexington, Kentucky

Director: Dr. Michael Sama, Professor of Biosystems and Agricultural Engineering

Lexington, Kentucky

Copyright © Surya Saket Dasika 2018

ABSTRACT OF THESIS

ASSESSING THE SPATIAL ACCURACY AND PRECISION OF LIDAR FOR REMOTE SENSING IN AGRICULTURE

The objective of this whole study was to evaluate a LiDAR sensor for high-resolution remote sensing in agriculture. A linear motion system was developed to precisely control the dynamics of LiDAR sensor in effort to remove uncertainty in the LiDAR position/velocity while under motion. A user control interface was developed to operate the system under different velocity profiles and log LiDAR data synchronous to the motion of the system. The LiDAR was then validated using multiple test targets with five different velocity profiles to determine the effect of sensor velocity and height above a target on measurement error. The results indicated that the velocity of the LiDAR was a significant factor affecting the error and standard deviation of the LiDAR measurements, although only by a small margin. Then the concept of modeling the alfalfa using the linear motion system was introduced. Two plots of alfalfa were scanned and processed to extract height and volume and was compared with photogrammetric and field measurements. Insufficient alfalfa plots were scanned which prevented any statistical analysis from being used to compare the different methods. However, the comparison between LiDAR and photogrammetric data showed some promising results which may be further replicated in the future.

KEYWORDS: LiDAR, Validation, Remote Sensing, Measurement Error, Precision Agriculture, Alfalfa

Surya Saket Dasika

Signature

July 20, 2018

Date

ASSESSING THE SPATIAL ACCURACY AND PRECISION OF
LIDAR FOR REMOTE SENSING IN AGRICULTURE

By

Surya Saket Dasika

Dr. Michael Sama

Director of Thesis

Dr. Donald Colliver

Director of Graduate Studies

July 20, 2018

Date

Dedicated to my grandmother, I could not have done it without you.

Thanks for all the love you showed and the knowledge you taught.

Thanks for all the memories...

ACKNOWLEDGMENTS

I first and foremost need to thank my parents for encouraging and supporting me throughout my life and to lots of love they showed me. I am also deeply indebted to my uncle for everything he does.

My sincere gratitude to my mentor and advisor Dr. Michael Sama for guiding me throughout these two years. Without his right guidance, the project would not have been accomplished successfully. Thanks for all the dedication and faith he showed in me. I would also like to thank my committee members Dr. Joseph Dvorak and Dr. Bruce Walcott for their valuable feedback. I would like to thank Dr. Joseph Dvorak again for provision of photogrammetric data.

Special thanks to Mr. Christopher Good and Mr. Luis Felipe Pampolini at the University of Kentucky for their contributions to the software and the hardware aspects of the project. I would also like to extend my thanks to Mr. Ali Hamidisepehr for his help with the various aspects of the project.

Special thanks also to Mr. Eric Roemmele of Applied Statistics Lab at University of Kentucky for his help with the statistical analysis. I am also grateful to Mr. Lee Rechten and Mr. Edward Hutchens at the University of Kentucky Agricultural Machinery Research Laboratory for their help with the fabrication of the linear motion system.

Thanks to all the people of BAE Department at University of Kentucky for constant support and friendship for these past two years. A special gratitude goes out to all my well-wishers and friends for constant encouragement through my career.

TABLE OF CONTENTS

Acknowledgments iii

Table of Contents iv

List of Figures x

List of Tables xv

List of Equations xvi

List of Additional Files xvii

CHAPTER 1: Introduction 1

 1.1 Low Altitude Remote Sensing in Agriculture 1

 1.2 LiDAR in Remote Sensing 3

 1.2.1 Background 3

 1.2.2 Height Measurements 5

 1.2.3 LiDAR Sensor 5

 1.3 Objectives 6

 1.4 Thesis Outline 7

CHAPTER 2: Linear Motion Test Fixture Development 8

 2.1 Summary 8

 2.2 Introduction 8

 2.2.1 Objectives 9

 2.3 Materials and Methods 10

2.3.1	Design Constraints	10
2.3.2	Test Fixture Components	10
2.3.3	Torque Requirements of the System	13
2.3.4	Motion Control System	14
2.3.5	Validation of Linear Motion System Performance	17
2.3.6	Safety Precautions	19
2.4	Results and Discussion	19
2.4.1	Frame Movement Validation	19
2.4.2	Position Control Validation.....	20
2.4.3	Steady-State Control Validation.....	20
2.5	Conclusions	21
CHAPTER 3: Software Control Development and LiDAR Validation Experiment.		22
3.1	Summary.....	22
3.2	Introduction	22
3.3	Materials and Methods	24
3.3.1	User Control Interface.....	24
3.3.2	Data Processing and Analysis	27
3.3.3	Test Target Experiment.....	30
3.3.4	Point Cloud Processing of the Test Target.....	32
3.4	Results and Discussion	34

3.4.1	LiDAR Firmware Version.....	34
3.4.2	Test Target Experiment.....	35
3.4.3	Statistical Analysis of the Test Target.....	41
3.5	Conclusions	46
CHAPTER 4: Physical Modeling of Alfalfa using LiDAR Integrated onto a Linear		
Motion Test Fixture		
4.1	Summary.....	48
4.2	Introduction	48
4.3	Materials and Methods	51
4.3.1	Test Setup.....	51
4.3.2	Point Cloud Processing of Alfalfa.....	52
4.3.3	Field Measurements	56
4.4	Results and Discussion.....	57
4.4.1	Alfalfa Physical Parameters	57
4.4.2	Comparison with Field Measurements and Photogrammetric	
	Measurements	58
4.5	Conclusions	60
CHAPTER 5: Summary and Conclusions		
5.1	Summary of Work	62
5.2	Conclusions	63

5.3 Future Work.....	63
CHAPTER 6: Appendicies	64
Appendix A. Linear Motion System CAD Models	64
A.1 Linear Frame	64
A.2 Carriage Assembly.....	73
A.2.1 Wheel Assembly	80
A.3 LiDAR Assembly.....	82
A.4 Motion Control.....	87
A.5 Linear Motion Test Fixture Full Assembly	93
Appendix B. Linear Motion Test Fixture Wiring Diagram	94
Appendix C. Linear Motion Control Software and LiDAR Data Collection Software	95
Appendix D. LiDAR Data Post Processing for Test Target.....	101
D.1 Point Cloud from Raw LiDAR Data.....	101
D.1.1 Authorship Information	101
D.1.2 Automating Files and Preallocation.....	101
D.1.3 Height of Ground	101
D.1.4 Velocity Profile Calculations	102
D.1.5 Hexadecimal to Decimal Conversion	103
D.1.6 Finding Missing Azimuth Values	106
D.1.7 Conversion of Polar Coordinates to Cartesian Coordinates	106

D.1.8 Defining the Point Cloud	110
D.2 Feature Extraction from Point Cloud	111
D.2.1 Authorship Information	111
D.2.2 Obtaining the Z Coordinate of the Whole Test target.....	111
D.2.3 Roi's for the Test Targets	112
D.2.4 Feature Extraction.....	115
D.2.5 Sort and Write to Excel File.....	118
D.3 Mean Height.....	120
D.3.1 Authorship Information	120
D.3.2 Code for the Function	121
Appendix E. LiDAR Test Target CAD Models	122
Appendix F. LiDAR Post Processing for Alfalfa	125
F.1 Feature Extraction from Point Cloud of Alfalfa.....	125
F.1.1 Contents	125
F.1.2 Authorship Information	125
F.1.3 Limits of the Quadrat and Extraction of Point Cloud of Alfalfa	125
F.1.4 Cube Method for finding the volume	126
F.1.5 alphashape Method	126
F.1.6 Block Method	127
F.1.7 Octreee Method	128

F.1.8 Finding the min and max heights in $n \times n$ square	128
F.1.9 Statistical distribution	129
F.1.10 Lodged Plants	130
F.1.11 Applying Threshold.....	131
F.1.12 Sort and Write to Excel File	131
Appendix G. LiDAR Data Statistical Analysis Software	132
References.....	137
Vita.....	142

LIST OF FIGURES

Figure 1-1: Waveform of the LiDAR mapping a tree showing its peak returns	4
Figure 1-2: (a) DTM of the tree (b) DSM of the tree	5
Figure 1-3: Velodyne VLP-16 LiDAR	6
Figure 2-1: (a) Carriage assembly with the major components identified (b) Integration of LIDAR onto the carriage assembly using the multipurpose mount	12
Figure 2-2: (a) Frame Assembly with the major components identified and (b) Sliding joint connecting the top and bottom frame assemblies.	13
Figure 2-3: Motion control system electronics enclosure box with major components identified.	15
Figure 2-4: Isometric view of the linear motion test fixture CAD model. The timing belt and cable carrier are not shown.	16
Figure 2-5: Complete linear motion system assembly designed to translate a LiDAR sensor at constant velocity over a 1x1x1 m test volume.....	17
Figure 2-6: Validation of the linear motion system performance using a robotic total station and target prism.	18
Figure 2-7: Frame movement validation showing the average magnitude of movement.	19
Figure 2-8: Position validation showing the deviation of distance of LiDAR from the theoretical distance.....	20
Figure 2-9: Steady-state velocity validation showing the deviation of steady state velocity of LiDAR from the theoretical steady state velocity.	21
Figure 3-1: User control interface to select different velocity profiles for the LiDAR	25

Figure 3-2: Data logging flow diagram showing the order of steps followed in order to log the data from LiDAR	27
Figure 3-3: Data processing flow diagram showing the order of steps performed to acquire the object height model.	29
Figure 3-4: (a) Orientation and placement of the test target underneath the LiDAR (b) Physical layout showing the arrangement of heights of the test target.	31
Figure 3-5: LiDAR point cloud of (a) Test target at a velocity of 1.0 m/s (OHM) and (b) Extracted 0.8 m targets at a velocity of 1.0 m/s.	33
Figure 3-6: Temperature response of the LiDAR sensor as exhibited by measured distance the ground surface at varying LIDAR sensor temperatures.	35
Figure 3-7: Box and whisker plots of estimated height error for (a) white targets vs. actual height (b) white targets vs. actual velocity (c) black targets vs. actual height and (d) black targets vs. actual velocity.	36
Figure 3-8: Box and whisker plots of standard deviation for (a) White targets vs. actual height (b) White targets vs. actual velocity (c) Black targets vs. actual height (d) Black targets vs. actual velocity.	38
Figure 3-9: (a) Relationship showing the correlation of point density with velocity at different heights (b) Relationship showing the correlation of point density with height at different velocities.	40
Figure 4-1: Quadrat of dimensions 1.0 m x 1.0 m x 1.0 m placed in an alfalfa field. ...	51
Figure 4-2: Modeling the alfalfa using the linear motion system over a quadrat.....	52
Figure 4-3: Points cloud of (a) Alfalfa and the quadrat (b) Quadrat (c) Alfalfa inside the quadrat.....	54

Figure 4-4: Histograms of heights of alfalfa from (a) LiDAR at a steady-state velocity of 1.0 m/s and (b) Photogrammetry	60
Figure 6-1: Cross sectional area of T-slotted aluminum (1010, 80/20 Inc.).....	64
Figure 6-2: Cross sectional area of T-slotted aluminum (1020, 80/20 Inc.).....	64
Figure 6-3: Cross sectional area of T-slotted aluminum (2020, 80/20 Inc.).....	65
Figure 6-4: Cross sectional area of T-slotted aluminum (b1010, 80/20 Inc.).....	66
Figure 6-5: Cross sectional area of T-slotted aluminum (b1020, 80/20 Inc.).....	66
Figure 6-6: A 2 x 2 bent bracket used for connecting two T-slotted aluminum joints.	67
Figure 6-7: A 1 x 1 bent bracket used for connecting two T-slotted aluminum joints.	68
Figure 6-8: A 4 x 4 bent bracket used for connecting two T-slotted aluminum joints.	69
Figure 6-9: A 2 x 2 flat bracket used for connecting two T-slotted aluminum joints. .	70
Figure 6-10: A 4 x 2 flat bracket used for connecting two T-slotted aluminum joints.	70
Figure 6-11: Brace on one end of the top assembly to join two linear rails together. .	71
Figure 6-12: Brace on one end of the top assembly to join two linear rails together. .	71
Figure 6-13: Bracket for integrating the motion control plastic box to the top assembly.	72
Figure 6-14: Sliding joint that connects the top and frame assembly.....	72
Figure 6-15: Multipurpose mount to integrate different sensors and instruments.....	73
Figure 6-16: Base mount to secure multipurpose mount to the carriage rail.....	74
Figure 6-17: Carriage rail to translate back and forth on the linear rail.	75
Figure 6-18: Multipurpose mount to secure LiDAR assembly to the multipurpose mount.	76

Figure 6-19: Upper part of the belt mount to transfer the force from timing belt to the carriage assembly.....	77
Figure 6-20: Lower part of the belt mount to transfer the force from timing belt to the carriage assembly.....	78
Figure 6-21: Cable carrier mount to secure the flexible carrier to it to translate along with the carriage assembly.....	79
Figure 6-22: Wheel designed to translate along the indentation of the linear rail.....	80
Figure 6-23: Ball bearing on either side of the wheel to ensure uniform distribution of force on the wheel.....	81
Figure 6-24: Top clamp to secure the LiDAR	82
Figure 6-25: Bottom clamp to secure the LiDAR.....	83
Figure 6-26: Top bracket to enclose the LiDAR assembly	84
Figure 6-27: Bottom part of the LiDAR assembly to enclose it.....	85
Figure 6-28: Mount for attaching all the instrumentation in LiDAR assembly.....	86
Figure 6-29: Stepper motor (STP-MTRH-34066D, AutomationDirect).....	87
Figure 6-30: Motor bracket for securing the stepper motor to the frame.	88
Figure 6-31: Pulley bracket for securing the pulley to the frame.	89
Figure 6-32: Timing belt pulley to transfer the rotary motion from the stepper motor to the linear motion of the system.....	90
Figure 6-33: Bearings on either side of the pulley to ensure uniform distribution of force on the pulley.....	91
Figure 6-34: Cross sectional view of L-series timing belt.....	91
Figure 6-35: Limit switch bracket for securing limit switches to the frame.....	92

Figure 6-36: Linear motion system with the dimensions of the top and frame assembly satisfying the design constraints. 93

Figure 6-37: Wiring schematic for the motion control hardware. 94

LIST OF TABLES

Table 3-1: Data Collection procedure of the LiDAR with the order of velocities	32
Table 3-2: Average error values of the white and black targets at different velocities and heights.....	37
Table 3-3: Average standard deviation values of the white and black targets at different heights and velocities	39
Table 3-4: Average intensity values of white and black targets at different heights and velocities	41
Table 3-5: Statistical analysis of the white targets: (a) Error model (b) Velocity tukey grouping table	42
Table 3-6: Statistical analysis of the black targets: (a) Error model (b) Velocity tukey grouping table	43
Table 3-7: Statistical analysis of the white targets: (a) Standard deviation model (b) Velocity tukey grouping table (c) Height tukey grouping table	44
Table 3-8: Statistical Analysis of the black targets: (a) Standard deviation model (b) Velocity tukey grouping table (c) Height tukey grouping table	45
Table 3-9: T-test of the errors between white and black targets.....	46
Table 4-1: Average physical properties across six replications of alfalfa for plot 1 at different velocities	57
Table 4-2: Average physical properties across six replications of alfalfa for plot 2 at different velocities	58
Table 4-3: Field based measurements for plots 1 and 2	58
Table 4-4: Alfalfa physical parameters obtained from the photogrammetry.....	59

LIST OF EQUATIONS

Equation 2-1 13

Equation 2-2 13

Equation 3-1 28

Equation 3-2 28

Equation 3-3 28

Equation 3-4 29

Equation 3-5 29

LIST OF ADDITIONAL FILES

File Name	Description	Size (Bytes)
hex2value_new.m	The MATLAB script used to obtain a 3D point cloud from raw LiDAR data by converting hexadecimal to decimal data and using velocity values to project into local 3D coordinate frame of linear motion system	21,621
meanheight.m	The function was used to compute the raw heights of all the points, mean height, standard deviation, mean intensity, missing targets for the point clouds of the individual targets	2,985
pointcloudalfa_forw.m	The MATLAB script was used to extract statistical parameters of the physical properties of the alfalfa plant	11,061
pointcloudvalforw.m	The MATLAB script was used to extract heights of each point, mean height, standard deviation, mean intensity, point density on a single test target from the point cloud of the whole test target	17,221
plate_analysis_final.sas	The SAS Script is used to find the statistical differences of errors and standard deviations of black and white targets for different velocities and heights. The script can be used to find the statistical differences of errors between black and white targets	8,773
LiDAR_Data.xlsx	Raw LiDAR Data of Test Targets	49,341,642
sas_results.xlsx	Results obtained from SAS	562,697

CHAPTER 1: INTRODUCTION

1.1 Low Altitude Remote Sensing in Agriculture

Low altitude remote sensing in agriculture (LARS) is the use of unmanned aircraft systems (UAS) or manned aircraft systems (MAS) at low altitudes to remotely estimate crop physical and biological parameters. LARS is important for small-to-medium farm holdings (C. Swain et al., 2018) and provides an alternative to relatively low resolution satellite imagery, which makes most of the fields on smaller farms look uniform and represented by too few pixels (Lamb and Brown, 2001). Site specific management or intra-field management with high resolution field data is beneficial as it monitors the local requirements in the field. Monitoring site specific data in real time will be the future of agriculture as it takes spatial and temporal variability into account instead of averaging the variability across the field and time into a single measurement. Variable rate information, although is more computationally complex, can potentially provide more accurate and precise information, which balances the cost of using it.

UAS offers high resolution remote sensing data at ultra-low altitudes and slow speeds with a high degree of flexibility and high efficiency than MAS (Guo et al., 2012; Huang et al., 2016). The use of unmanned aircraft systems (UAS) to estimate a crop's physical and biological properties has increased due to improvements in accuracy and efficiency (Malveaux et al., 2014). Especially as the cost of UAS decreases, it will become more and more viable option for remote sensing in the future. The problem with the small UAS are due to its current limitations in terms of payload capacity, range and stability (Guo et al., 2012; Zhang and Kovacs, 2012). But a recent study by Huang et al., 2016 suggested that UAS can alone provide enough accurate data or can complement MAS for crop

management. The results from different experiments over the past few years also showed the capability of using UAS for remote sensing in agriculture (Guo et al., 2012; Zhang and Kovacs, 2012; Huang et al., 2016; C. Swain et al., 2018).

Crop physical properties, such as canopy height, canopy volume, and leaf area index (LAI) are useful for estimating nutritional and fertilizer requirements (Stamatiadis et al., 2010), providing indication of health and potential yield (Cui et al., 2010; Colaço et al., 2017), modeling evapotranspiration, photosynthesis, and crop yield (Bonan, 1993), and understanding interactions between plants and solar radiation, water, and nutrients (Nie et al., 2016). UAS-based remote sensing typically have higher spatial and temporal resolutions than satellite and conventional aerial imagery (Zhang and Kovacs, 2012), which makes it particularly suitable remote sensing at the individual plant level.

There are multiple techniques used in capturing the 3D information of the environment. Most commonly used techniques are laser, radar and ultrasonic ranging sensors (Dworak et al., 2011). Laser, radar, and ultrasonic sensors are all active remote sensing techniques, which use time-of-flight (TOF), interferometry, or triangulation techniques to map the environment (Dworak et al., 2011; Vázquez-Arellano et al., 2016). The active remote sensing techniques provided their own energy source to be reflected/transmitted back and measured by the sensor whereas the passive remote sensing methods uses the energy available naturally like sunlight to be reflected to be measured by the sensor. The passive remote sensing techniques typically use image processing methods and are less accurate than active remote sensing techniques due to sunlight and environment issues (Kelly and Di Tommaso, 2015; Pittman et al., 2015; Sun et al., 2018).

In the future, LARS is likely to overtake traditional remote sensing in terms of use as it provides the opportunity of real-time monitoring and phenotyping of crops and plants with higher spatiotemporal resolutions.

1.2 LiDAR in Remote Sensing

1.2.1 Background

The active remote sensing techniques can be used any time of the day and are typically more accurate for 3D modeling, but there are some limitations among commonly used active sensors. Ultrasonic sensors are not particularly suitable for crop remote sensing due to high attenuation, divergence, sensitivity to wind, and limited range. High-resolution radar sensors are expensive and typically exhibit a time delay for data acquisition, while low-cost radar sensors do not have the resolution required for remote sensing of crop physical structure (Dworak et al., 2011). High-resolution LiDAR, as compared to other ranging sensors, has higher spatial and temporal precision, and can map the environment more accurately (Lefsky et al., 2002). LiDAR performance is robust in a wide range of environmental conditions making it suitable for agriculture use (Lin, 2015). Multiple studies have shown the utility of LiDAR for mapping crop or tree physical properties (Estornell et al., 2011; Zhang and Grift, 2012; Arnó et al., 2013; Kelly and Di Tommaso, 2015).

LiDAR is a type of ranging sensor and an active remote sensing technique that uses ultraviolet, infrared or visible light to map the environment. Originated in the 1960's, it uses time of flight technique to measure the distance between the object and the LiDAR. There are primarily two types of LiDAR based on the way the return signals are recorded. They are full waveform and the discrete return LiDAR as shown in the Figure 1-1. When

a single laser pulses emitted from a LiDAR hits an object, there may be multiple returns depending upon the geometric characteristics of the object. Full waveform records the whole pulse received whereas the discrete return only records the peaks of the signal received. As the full waveform LiDAR records the whole pulse received it is potentially provides a more accurate and precise response, but it has more data, requires more processing time and has less commercial software packages designed to process the data over large areas (Kelly and Di Tommaso, 2015).

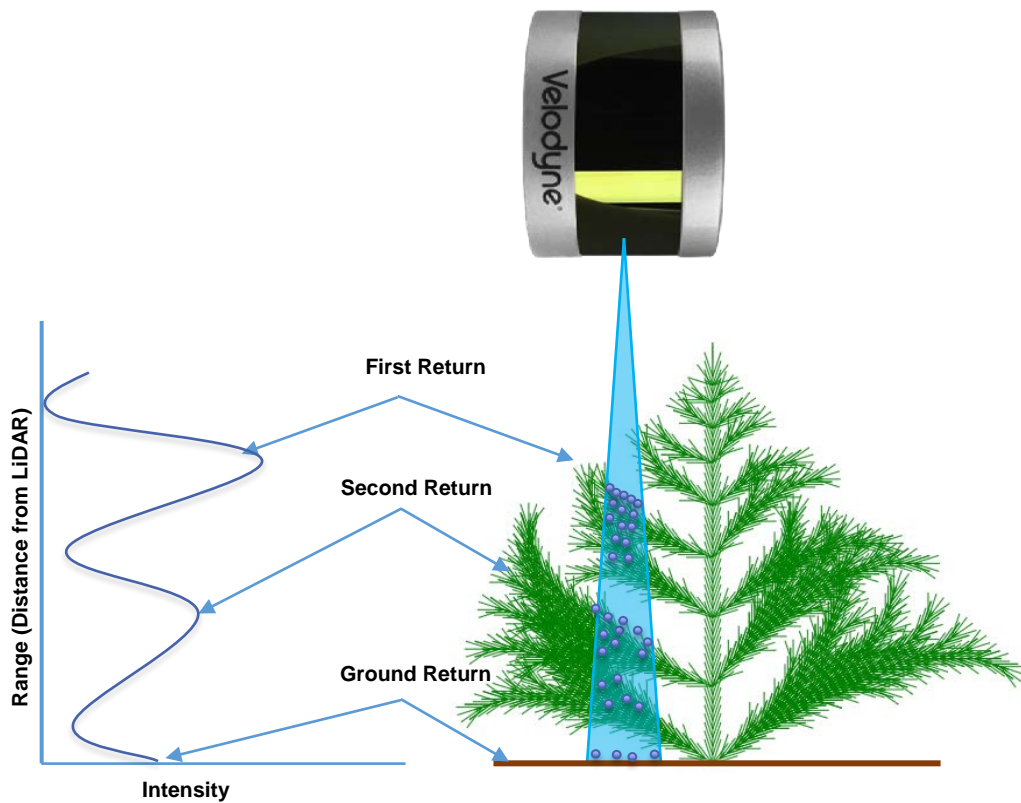


Figure 1-1: Waveform of the LiDAR mapping a tree showing its peak returns

LiDAR is an excellent at capturing the 3D information of the environment. Mapping applications range from measuring canopy physical parameters in agriculture to obstacle avoidance for autonomous vehicles to land and construction surveying to pollutant

modeling. Modern LiDAR systems are easier to deploy and can be integrated into a wide range of platforms such as UAS, etc.

1.2.2 Height Measurements

A common technique to quantify the physical height of an object relative to the ground plane was to obtain object height model (OHM) or canopy height model (CHM). OHM can be computed from the digital terrain model (DTM) and digital surface model (DSM) of an object which can be obtained through LiDAR. The DTM represented ground measurements whereas the DSM consists of objects detected above ground level. The DTM was subtracted from the DSM to obtain the OHM, and the OHM was used to extract height measurements in a given region of interest. Figure 1-2 shows an example of DSM and DTM of a tree mapped from above it.

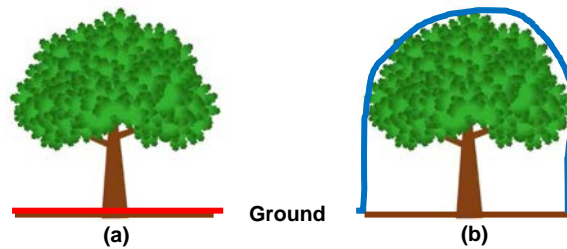


Figure 1-2: (a) DTM of the tree (b) DSM of the tree

1.2.3 LiDAR Sensor

A 16 channel LiDAR (VLP-16, Velodyne LiDAR, San Jose, CA) (Figure 1-3) was used for the study. The LiDAR communicates with a webserver GUI with the default IP of the server: 192.168.1.201 once turned on. The webserver can be used to customize the settings of the LiDAR. From the webserver interface, the LiDAR can be configured to report three different types of returns last, strongest and dual returns, which reports both the last and strongest returns. The LiDAR can also be configured to rotate 300 - 1200 rpm. The LiDAR

sends 754 UDP (User Datagram Protocol) data packets per second in the strongest and last returns configuration and 1508 data packets per second in the dual returns configuration and maps the environment in spherical coordinates. Each UDP data packet consists of 12 data blocks consisting of rotation angles (azimuth and elevation), time of flight distances, calibrated reflectivity measurements and a timestamp associated with it. The LiDAR operates within a range of 1 m - 100 m, has 360° horizontal (azimuth) field of view with resolution of 0.1° - 0.4° and 30° vertical (elevation) field of view with a resolution of 2°. LiDAR has a typical accuracy of ± 3 cm. In this whole study, the LiDAR was configured to only report the strongest returns (generally the closer object) and to rotate at 600 rpm.

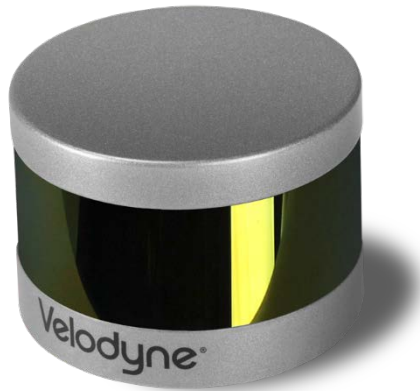


Figure 1-3: Velodyne VLP-16 LiDAR

1.3 Objectives

The overall objective of this study was to evaluate a LiDAR sensor for remote sensing in agriculture. The specific objectives were as follows:

- 1) Develop a linear motion test fixture for evaluating a LiDAR system used for remote sensing in agriculture.
- 2) Validate the system using a test target and determine the effect of target height and

LiDAR velocity on measurement error.

- 3) Model the physical structure of alfalfa using the test fixture and correlate with field measurements and 3D measurements made using photogrammetry.

1.4 Thesis Outline

Chapter 1 introduces the thesis topic, provides background information, and outlines specific objectives for this thesis. Chapter 2 introduces the development of the linear motion system and the rationale behind it. This chapter also validates the performance of linear motion system. Chapter 3 details the development of software to operate the linear motion system, log the data from LiDAR and post process the raw LiDAR data to extract the useful information. This chapter also validates the system and determines the effect of target height and LiDAR velocity on measurement error. Chapter 4 details the process of modeling physical parameters of alfalfa using LiDAR on a linear motion system. This chapter also compares the LiDAR data with the field and photogrammetry measurements. Chapter 5 discusses the future applications of this system and the additional work this system aims to produce. Chapter 6, the appendix, presents pertinent information not presented in the body of the text as well as extended figures which provides more thorough details of the all the drawings, software used in this study.

CHAPTER 2: LINEAR MOTION TEST FIXTURE DEVELOPMENT

2.1 Summary

To summarize this chapter, a linear motion test fixture was developed based on design constraints of usability and versatility to control the dynamics of the LiDAR. The linear motion system consisted of top assembly which included a carriage assembly that translated back and forth via force supplied by the timing belt to which the LiDAR was integrated. It also consisted of a frame assembly to raise the LiDAR sufficiently high above the ground and a motion control system to run the stepper motor and power the sensors. The linear motion test fixture was validated for position control, steady state velocity, and frame movement using a robotic total station. The results demonstrated the effectiveness of the test fixture for eliminating most of the uncertainty presented in traditional LiDAR deployment platforms used for remote sensing.

2.2 Introduction

For remote sensing, LiDAR can be integrated onto fixed-wing or multi-rotor UAS to achieve high-resolution real-time monitoring of crop physical properties at field scale. UAS typically consists of an aerial platform and an autopilot or flight control systems that includes a global positioning system (GPS) and an inertial measurement unit (IMU). Position and velocity are usually tracked by the GPS and IMU, and/or by additional on-board sensors (e.g., motion flow cameras). US government official website for GPS (Department of Defense, 2008) states that the global user average for position accuracy was ≤ 0.715 m and velocity accuracy was ≤ 0.006 m/s with 95% probability. On top of that the turbulence in the environment, the skill of the operator piloting the UAS also adds to the uncertainty to the position, velocity and orientation measurements. A common

assumption is that the UAS is moving at a constant velocity (Sun et al., 2018) or GPS/IMU are accurate enough to predict position at any point in time. Sugiura et al., 2003 also stated that UAS do not have precise control over velocity, which changes due to wind speed and will add inaccuracies in LiDAR measurements. In all these cases, there is some uncertainty in the velocity and position of UAS at each time interval that contributes to LiDAR measurement error. The position and velocity of an UAS are susceptible to uncertainty due to environmental conditions and will be hard to accurately track them at the spatiotemporal resolutions that LiDAR data can be collected. The uncertainties in the position and velocity at each instant will induce inaccuracies in the 3D projection of the LiDAR measurements.

One of the factors that affects the UAS is the overall stability and other factor that arise from the center of gravity (CG) of a UAS. The lower the center of gravity is in comparison to UAS control axis, the less stable it is. Integrating too many sensors at the same time and having too much weight on the lower part of an UAV lowers the CG making it less stable. A portable high-resolution LiDAR usually weighs anywhere between 0.3 kg to 2 kg. Integrating LiDAR onto a UAS will have a large impact on the COG and most likely reduce stability.

Prior to implementing these techniques on large fields, the methods must be evaluated in a more controlled manner to reduce external effects on the LiDAR measurement error and to determine the upper-limit on accuracy and precision of the LiDAR measurements.

2.2.1 Objectives

The objective of this chapter is to develop a linear motion test fixture to control the dynamics of the LiDAR. The specific objectives were as follows:

- 1) Design and fabricate the linear motion system test fixture according to the design constraints according to the design constraints of the system.
- 2) Develop a motion control system to drive the stepper motor and supply power to the whole system.
- 3) Validate the performance of the linear motion system.

2.3 Materials and Methods

2.3.1 Design Constraints

In this study a linear motion test fixture was built to control the dynamics of a multi-channel 3D LiDAR sensor. The linear motion test fixture needed to be built in such a way that it can be deployed in the field by transferring it inside a trailer. Therefore, constraints on maximum dimensions were determined from the inner dimensions of the trailer. The test fixture was also required to be robust to external disturbances – otherwise, it would induce uncertainty in the LiDAR measurements. Finally, the linear motion test fixture should also be corrosion resistant for deploying it in the moisture saturated field conditions. In summary, three design constraints of the system were defined based on its usability and versatility:

- 1) Maximum dimensions of the whole system should be less than 1.8 m x 1.8 m x 3.6 m to fit inside an existing cargo trailer.
- 2) The system should be easy to carry, light weight, and robust to external disturbances.
- 3) Components encountering moisture should be corrosion resistant.

2.3.2 Test Fixture Components

The linear motion test fixture consisted of a linear rail which was built from anodized aluminum extrusion ((1020, and 2020), 80/20 Inc., Columbia City, IN) due to a hard

surface finish, ability to assemble into a rigid structure without permanent bonds, and corrosion resistance against moisture saturated field conditions. A carriage assembly (Figure 2-1a) was designed to translate back and forth along the linear rail. The carriage assembly consisted of different parts designed and fabricated on a 3D printer. It consists of a carriage rail with wheels that slides on the t-slotted linear rail extrusion. The wheels were designed in such a way that the offset between the wheel width and the linear rail extrusion was loose enough to move smoothly without allowing for unwanted lateral movements. Each wheel assembly consisted of the plastic wheel and two stainless steel flanged ball bearings on each side (57155K305, McMaster Carr, Elmhurst, IL). The carriage assembly also consisted of a multipurpose mounting plate containing different sized holes organized in a regular pattern to accommodate different sensors. The LiDAR was integrated onto the multipurpose mounting plate using a 3D printed bracket (Figure 2-1b), and translated back and forth with the carriage assembly via force supplied by a timing belt (7959K26, McMaster Carr) (Figure 2-1a). The timing belt and pulley (1304N11, McMaster Carr) sub-assembly are connected to the stepper motor and were attached to the carriage assembly using a clamping mechanism. The clamping mechanism allowed tension in the timing belt to be set so that the belt cannot slip around the pulley. A flexible cable carrier (55835K93, McMaster Carr) (Figure 2-1a) was also used for interfacing the LiDAR to a PC for data acquisition without entanglement in the linear motion system. The cable carrier was secured to the carriage assembly using a custom 3D-printed mount.

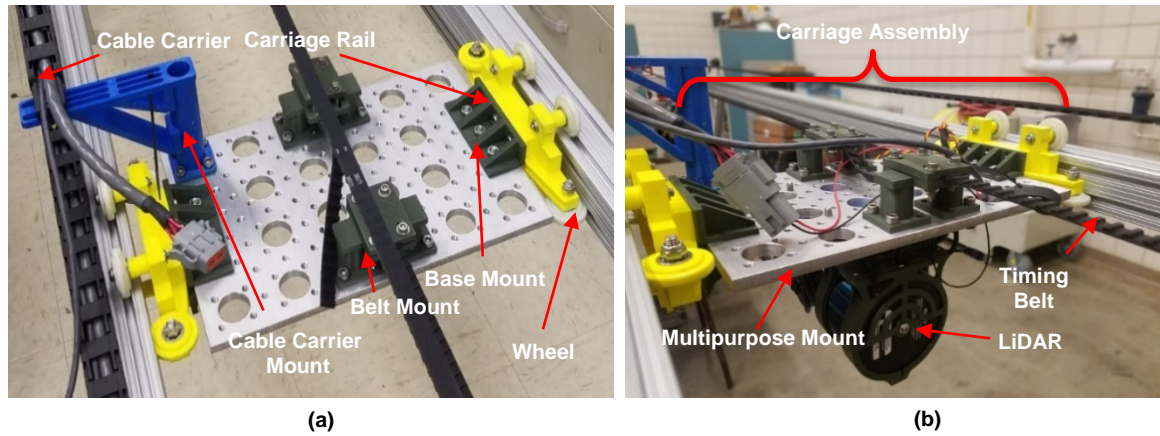


Figure 2-1: (a) Carriage assembly with the major components identified (b)

Integration of LIDAR onto the carriage assembly using the multipurpose mount

The linear motion system was mounted onto a frame assembly to set the height of the LiDAR at approximately 2 m above the ground, allowing targets up to 1 m tall to be scanned. The top assembly and frame assembly were connected by means of a sliding joint Figure 2-2b for easy dismount. The frame assembly had wheels mounted under each leg to improve indoor mobility. Both the top and frame assemblies had diagonal braces on the corners to improve rigidity. The dimensions of the top and frame assembly are less than the maximum constrained dimensions, thus satisfying all initial dimensional constraints. Refer to Appendix A for the CAD drawing of all the parts in the Figure 2-1 and Figure 2-2.

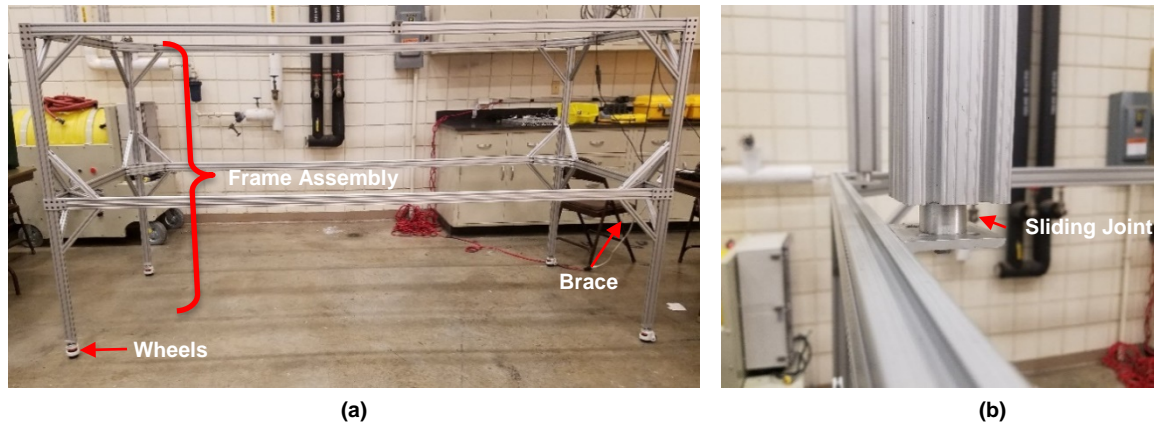


Figure 2-2: (a) Frame Assembly with the major components identified and (b) Sliding joint connecting the top and bottom frame assemblies.

2.3.3 Torque Requirements of the System

The torque requirements of the system were calculated as follows:

$$T = r \times F \quad \text{Equation 2-1}$$

$$F = Ma \quad \text{Equation 2-2}$$

Where

T = torque required to move the system (N m)

F = force needed to move the system (N)

r = pitch radius of timing belt pulley (m)

M = mass of carriage and LiDAR assembly (kg)

a = desired acceleration (m/s^2)

The maximum desired acceleration of the system was initially set at 5 m/s^2 to assist with motion component specification. A stepper motor (3.06 N m, STP-MTRH-34066D, AutomationDirect, Cumming, GA) was coupled with a 2.73 cm pitch radius timing pulley to provide torque for translating the timing belt. The required torque based upon the carriage mass and pitch radius was 0.41 N m, resulting in factor of safety of approximately

7.5. This allowed for higher accelerations to be achieved, but more importantly, sufficient torque was available to ensure that the stepper motor would not miss a step under normal operation conditions – a necessary criteria to ensure accurate position/velocity control.

2.3.4 Motion Control System

All motion control electronics were housed in a plastic enclosure box (NEMA Series, NBA 10176, Bud Industries, Willoughby, OH) (24) and mounted to the side of the top assembly for protecting against environmental and moisture saturated field conditions. Electronic components consisted of a 70 VDC power supply (STP-PWR-7005, AutomationDirect), which supplied power to a regeneration clamp (STP-DVRA-RC-050, AutomationDirect). The regeneration clamp is used to protect the system from back-EMF produced by the motor during high decelerations. The regeneration clamp then powered a bi-polar stepper motor controller (STP-DVR-80100, AutomationDirect), which emitted pulses to drive the stepper motor at a desired velocity profile. Additional power supplies (5/12/24 V) were integrated into the motion control system to supply power to LiDAR and for future instrumentation. The motion control system also consists of terminal strips and a plastic housing to have additional protection to the wires from the environment. Wiring was passed through the plastic enclosure box using cable glands. Refer to Appendix B for the wiring schematic of the motion control system.

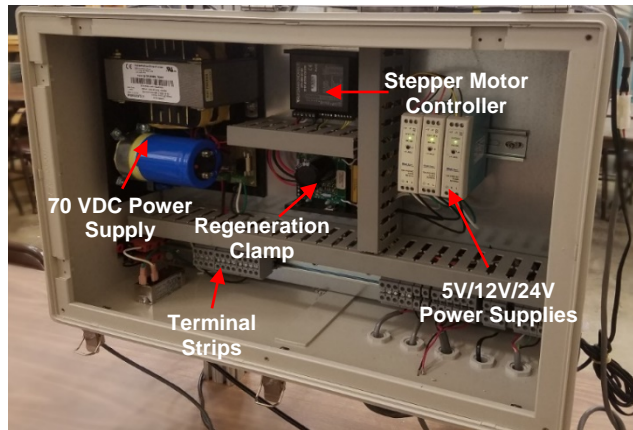


Figure 2-3: Motion control system electronics enclosure box with major components identified.

The stepper motor controller came with its own customized language called serial command language (SCL) for controlling the stepper motor. It was developed to make it simple to control the stepper motor through the serial port. It consists of several predefined commands to run the stepper motor in a desired velocity profile. Example commands used included:

- 1) “DI200/DI-200”: sets the displacement of the stepper motor in clockwise (CW)/ counter clockwise (CCW) to 200 rev.
- 2) “VE20”: sets the steady state velocity of stepper motor to 20 rev/s
- 3) “AE2/DE2”: sets the acceleration/deceleration to 2 rev/s²
- 4) “FL”: executes the above lines of code

By executing the four example commands shown, the stepper motor accelerated at 2 rev/s² to a steady-state velocity of 20 rev/s and decelerated at 2 rev/s² to rest with a total rotational displacement of 200 rev. Each revolution of the stepper motor was equivalent to the linear distance along the pitch circumference of the pulley attached to stepper motor. Therefore, desired linear dynamics had to be converted into motor velocity commands.

Carriage displacement was given the highest priority. That meant if the desired acceleration was such that it never reached the desired steady-state velocity in a given distance, then the desired steady-state velocity would not be reached.

Figure 2-5 refers to the rendered image of orthogonal view of CAD model of the whole linear motion system. Figure 2-6 refers to picture taken of the assembled linear motion system.

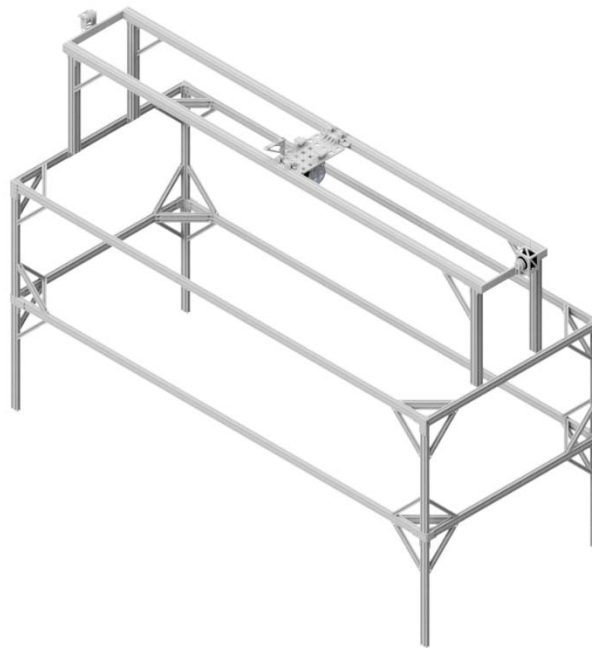


Figure 2-4: Isometric view of the linear motion test fixture CAD model. The timing belt and cable carrier are not shown.

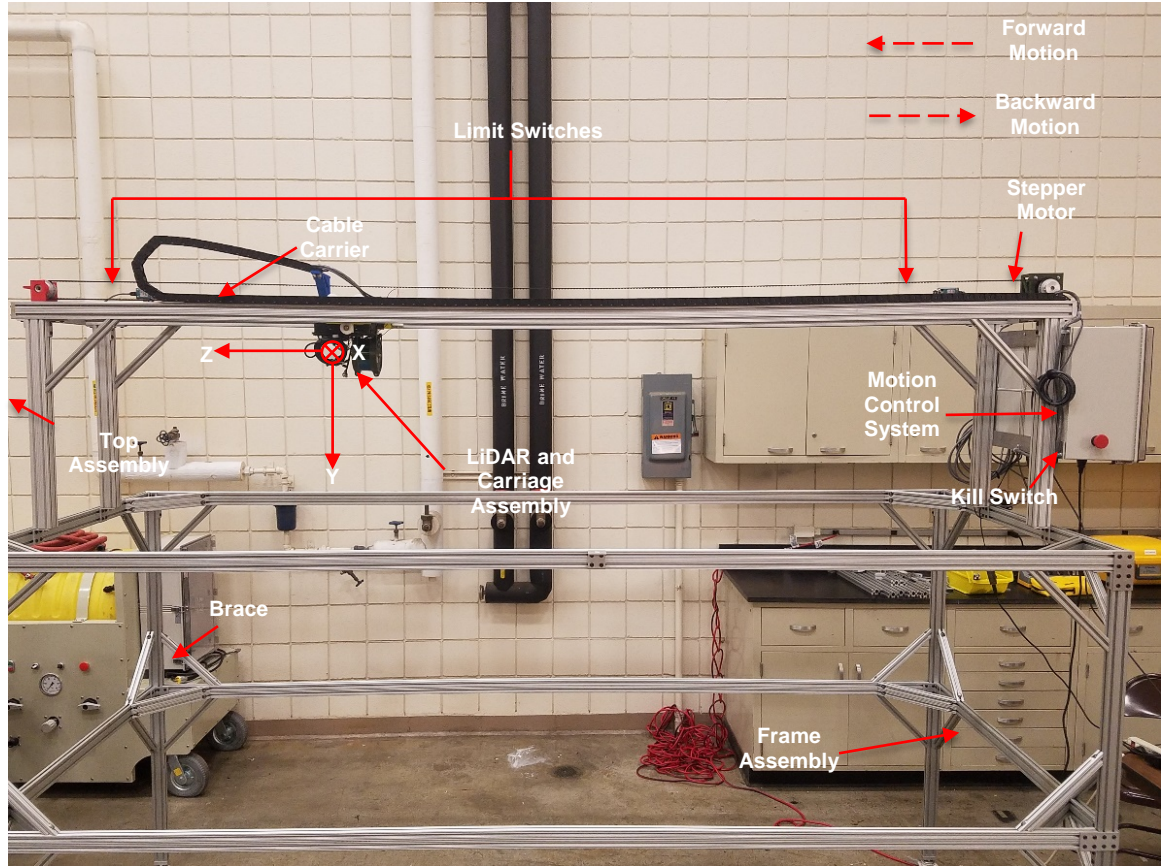


Figure 2-5: Complete linear motion system assembly designed to translate a LiDAR sensor at constant velocity over a 1x1x1 m test volume.

2.3.5 Validation of Linear Motion System Performance

The linear motion system performance was validated using a total station (S5, Trimble Inc., Sunnyvale, CA) (Figure 2-7). The total station automatically tracked a prism (MT1000, Trimble Inc.) and was used for three different validation experiments:

1. Frame movement validation – the prism was fixed to a horizontal rail on the frame assembly to detect motion in the linear test fixture frame due to acceleration/deceleration of the carriage.
2. Position validation – the prism was fixed to the carriage assembly for determining the displacement accuracy of the carriage assembly when moving

between locations.

3. Steady-state velocity validation – the prism was fixed to the carriage assembly for determining the velocity accuracy of the carriage assembly under steady-state movement.

Frame movement validation measurements were taken at 7 different velocity profiles (distance = 2.0m, acceleration/ deceleration = 1 to 5 m/s², velocity = 1 to 2 m/s) with 2 replications. Position control validation measurements were taken at 4 different distances (0.5, 1.0, 1.5, and 2.0 m) with 3 replications. Steady-state velocity validation measurements were taken at 3 different velocities (0.1, 0.5, and 1.0 m/s) with 3 replications for each velocity. The measurements were logged at 2.5 Hz through a serial port and saved in a text file.



Figure 2-6: Validation of the linear motion system performance using a robotic total station and target prism.

2.3.6 Safety Precautions

The linear motion system (Figure 2-6) was equipped with a kill switch to disengage power in case of emergency. Limit switches (SZL-VL-A, Honeywell International Inc., Morris Plains, NJ) were also placed on either end of the linear rail to disengage the stepper motor power whenever the carriage assembly tripped the limit switch – thereby protecting the linear motion system from potential mechanical failure due to improper operation or software malfunction. The limit switch closest to the plastic enclosure box was also used to index the start position of the linear motion system.

2.4 Results and Discussion

2.4.1 Frame Movement Validation

The frame movement is expressed as the average magnitude of movement across the three axes of the linear motion system when the LiDAR was in motion. Data from the robotic total station showed little movement in the frame (< 1 mm) when accelerating/decelerating the carriage up to 5 m/s^2 . The average magnitude of movement was between 0.0001 m and 0.0002 m (Figure 2-8).

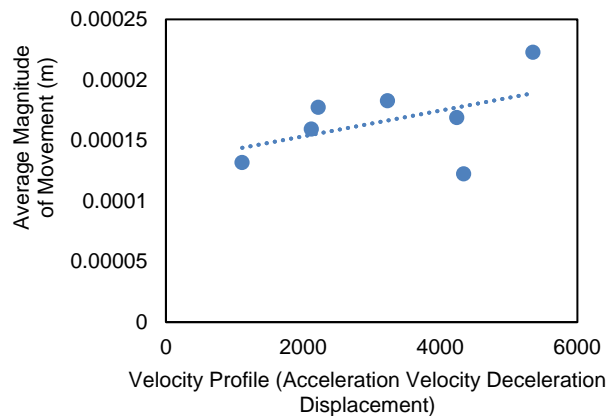


Figure 2-7: Frame movement validation showing the average magnitude of movement.

2.4.2 Position Control Validation

Position error was expressed as the difference between the distance measured by the total station and the theoretical displacement of the linear motion test fixture. The absolute position error of the linear motion test fixture was a function of distance travelled, resulting in 2 millimeters of error per meter travelled. The average absolute position ranged between 0.001 m and 0.005 m (Figure 2-9).

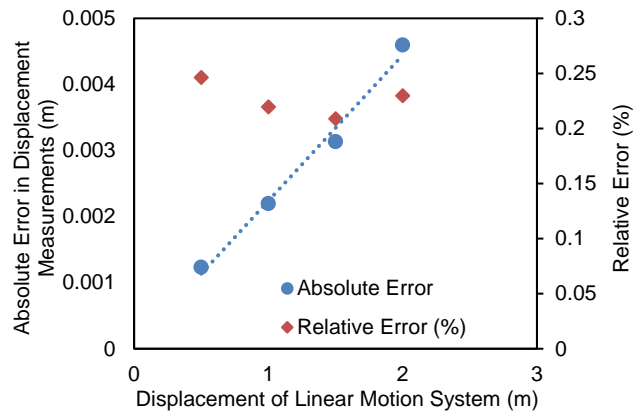


Figure 2-8: Position validation showing the deviation of distance of LiDAR from the theoretical distance.

2.4.3 Steady-State Control Validation

The error in the steady-state velocity was defined as the difference between the actual velocity measured by the total station and theoretical velocity of the linear motion test fixture. The absolute steady-state velocity error of the linear motion test fixture was 2 mm/s of error when travelling at 1m/s. The average error in steady-state velocity ranged from 0.0004 m/s to 0.002 m/s (Figure 2-10).

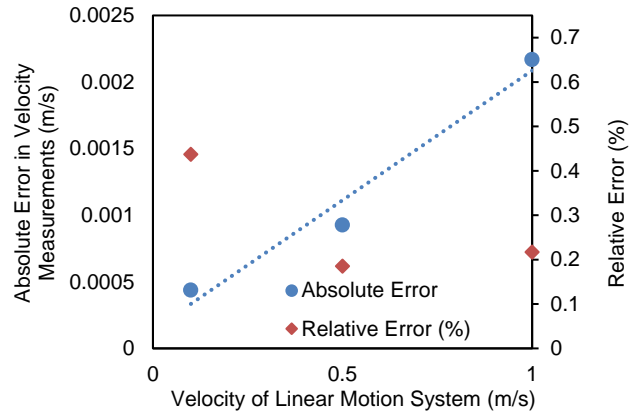


Figure 2-9: Steady-state velocity validation showing the deviation of steady state velocity of LiDAR from the theoretical steady state velocity.

2.5 Conclusions

Results showed that the linear motion test fixture constrained the LiDAR to a known path at a known velocity to a high level of accuracy and precision. Both the error in position and velocity were an order of magnitude smaller than the specified measurement accuracy of the LiDAR sensor (± 3 cm) and were considered negligible when projecting LiDAR data from the sensor spherical coordinate system to a local Cartesian coordinate system. Position and velocity error were likely due to error when calibrating the robotic total station into the linear motion test fixture's local coordinate system. No backlash was detected when moving the carriage back and forth to the same position.

CHAPTER 3: SOFTWARE CONTROL DEVELOPMENT AND LIDAR VALIDATION EXPERIMENT

3.1 Summary

To summarize this chapter, a user control interface was developed to operate the linear motion test fixture from Chapter 2. The user control interface recorded LiDAR data in the background using a second processing thread to parse and store data into a Comma-Separated Values (CSV) file. The LiDAR was then validated using multiple test targets at five different velocity profiles and six replications to determine the effect of sensor velocity and height above a target on measurement error. Generalized linear mixed models were fitted with the error and standard deviation as the response and velocity, actual height, and their interaction as the fixed effects to determine if there were significant differences in error and standard deviation for different velocities and heights. The results indicated that the velocity of the LiDAR was a significant factor affecting the error and standard deviation of the LiDAR measurements, although only by a small margin.

3.2 Introduction

The data from the LiDAR should be synchronized to the motion of the LiDAR to remove any positional error in the 3D projection of the LiDAR data. The point cloud obtained from the LiDAR data should be processed to acquire the information needed. The processing used in this study was based on Sun et al., 2018. The raw LiDAR data obtained in spherical coordinates was converted to Cartesian coordinates. OHM was obtained from subtracting the DTM from DSM to extract the height information of the object scanned and then fused with velocity profile data to project the height data in 3D point cloud.

The main advantage of using the linear motion test fixture was the ability to repeatedly move the LiDAR in different velocity profiles. The point density of the LiDAR on a target is a function of velocity. If controlled for external factors, point density follows a power series with a coefficient of -1 (Point density \propto 1/Velocity). As the velocity of the UAS increases the point density decreases for a constant LiDAR sampling rate. Lower point density reduces spatial resolution, which may obscure the true physical structure or provide insufficient data for accurate target classification. LiDAR data reduction was discussed by Liu and Zhang, 2008 and they concluded that the reduction in data doesn't significantly affect the accuracy of DEM, however most DEMs are at a much larger scale than what is being investigated in this study. They also concluded that the LiDAR data can be reduced to an extent to remove all the trivial elements or outliers and keep the remaining ones. Decrease in LiDAR data when deploying on a UAS will be useful as the UAS can travel faster and cover more area without significantly affecting the accuracy of DEM. Decrease in LiDAR data also meant that less computational complexity, fast processing times and file handling which are essential in real time monitoring.

One of the research questions this study seeks to understand is to how fast a LiDAR sensor can travel without having any significant deviation from its measurement due to decrease in point density at very high spatial resolutions (> 1 point per cm). This study also investigates how the relative height of targets within a given field-of-view (FOV) affect point density for a given velocity as a result of the geometry between the LiDAR and the targets. A predefined test target can be used to quantify LiDAR performance and perform statistical tests on the LiDAR data to show any significance of difference in height measurements with different sensor velocities and target heights.

Zhang and Grift, 2012 did similar research on estimating the crop height where they found an increase in average error of the stem height with an increase in the velocity of LiDAR due to reduction in point density. Sanz-Cortiella et al., 2011 did a 3D LiDAR experiment on multi-purpose test rail where the LiDAR was driven at 3 speeds and 2 angular resolutions and found a linear relationship between number of impacts and leaf area. Selbeck et al., 2010 also did a research project on a LiDAR scanner where he observed less to no deviation for the height measurements of a maize crop between different velocities.

This study expands on previous work by performing tests on a well-defined target to improve the ability to detect significant differences in measurement error based upon LiDAR velocity and target height.

The overall objective of this chapter is to validate the LiDAR sensor using defined input geometries. The specific objectives were as follows:

- 1) Develop a user control interface to control the linear motion test fixture and record the LiDAR data
- 2) Develop software to process the raw LiDAR data collected with the linear motion system
- 3) Validate the system using a test target and determine the effect of target height and LiDAR velocity on measurement error

3.3 Materials and Methods

3.3.1 User Control Interface

A PC-based user control interface was developed to control the velocity profile of the linear motion system and to record LiDAR data. The user control interface shown in Figure

3-1 was developed in Python (2.7, Python Software Foundation, DE) using the Tkinter graphic user interface (GUI) toolkit. Tkinter functions were imported in Python and a class was defined for software configuration. Combo boxes were used to provide preset values controlling acceleration, velocity, deceleration, and total displacement. Preset values could be overwritten by entering in custom values into any combo box, but were software limited in terms of their maximum values. Four buttons were implemented to provide the following capability:

1. Connect – opens the serial port and send the necessary commands through the serial port to initialize the stepper motor controller
2. Disconnect – closes the serial port, releasing the stepper motor controller
3. Run – simultaneously sends motion commands to the stepper motor controller to run the carriage assembly in the desired velocity profile and connects to the UDP port to log the data from LiDAR
4. Start Position – sends a series of commands to the stepper motor controller to ensure the system starts at the same place every time by slowly moving the carriage assembly in the forward direction until it strikes the limit switch.

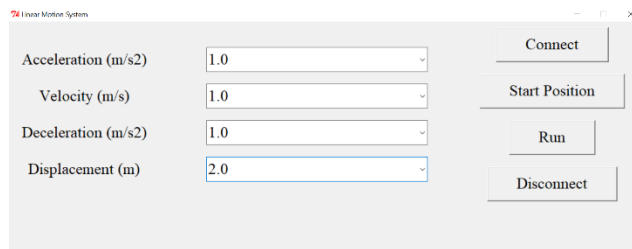


Figure 3-1: User control interface to select different velocity profiles for the LiDAR

After connecting to the serial port and clicking ‘Start Position’, the carriage assembly translated towards the start position limit switch until limit switch was pressed, at which point the carriage immediately stopped and the linear motion system location was set to

machine zero. The carriage assembly was then moved 0.1 m in the opposite direction and a local zero was set. The offset between the machine and local zero locations provided a “soft limit” to ensure that the mechanical limit switches would not be reached under normal operating conditions.

The data logging flow diagram shown in the Figure 3-2 illustrates the process of simultaneously sending commands to the stepper motor and started a data acquisition background thread after clicking the ‘Run’ button. The background thread consisted of a connection to a UDP port with an IP address and port number specified configured to match the LiDAR data stream. A comma-separated values (CSV) file was created each time the ‘Run’ button was pressed with the date and time as the file name. The LiDAR data blocks were encoded into hexadecimal format and parsed into the CSV file. A carriage position background thread continuously sampled the position of the carriage as it translated back and forth by polling the stepper motor controller. When the carriage assembly returned to the original start position, an event was raised to break the data acquisition background thread, thereby stopping the data logging and closing the CSV file. Refer to Appendix C for the motion control and data collection software developed in python.

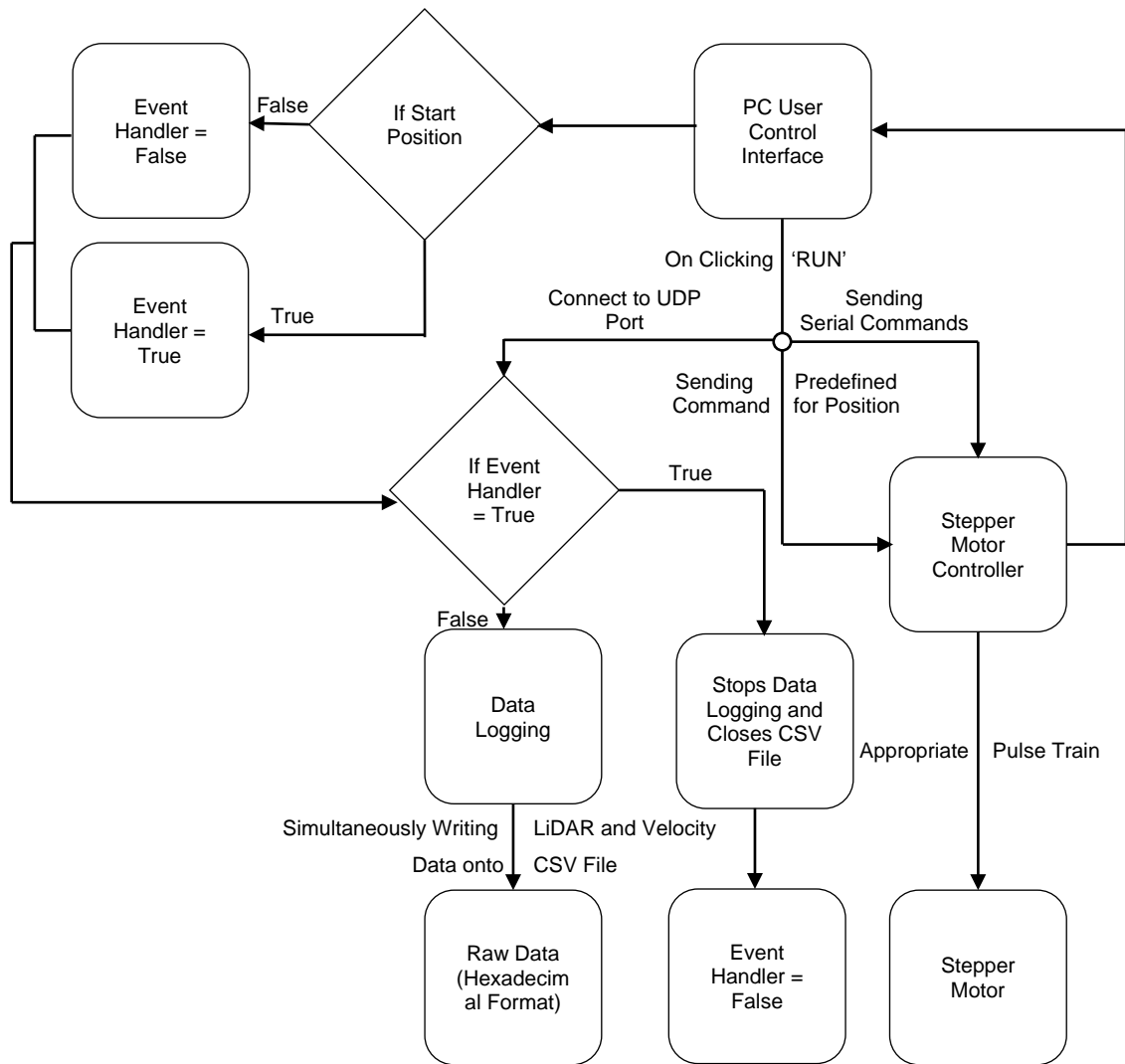


Figure 3-2: Data logging flow diagram showing the order of steps followed in order to log the data from LiDAR

3.3.2 Data Processing and Analysis

The work flow of the data processing is shown in the Figure 3-3. Initially, the raw LiDAR data was fed into a MATLAB script (R2016a, MathWorks, Natick, MA). A 1x16 cell structure array titled *distance* with each cell corresponding to a particular channel and consisting of 3-dimensional matrix with dimensions $n \times 12 \times 2$ (number of packets x data blocks per packet x 2 values per each channel per each data block) was initialized. The

hexadecimal values in data packets were converted into decimal format, which gives the azimuth angle values, time of flight distance values and calibrated reflectivity values. The elevation angles were fixed for a given channel. Data corresponding to the appropriate channel (values 0-15) were stored in the each of the cells in the *distance* cell structure array. The distances, azimuth angles and elevation angles mapped the environment in spherical coordinates which were converted into Cartesian coordinates (X, Y, Z) using the following formulae:

$$X_n = r_n \cos(\omega_n) \sin(\alpha_n) \quad \textbf{Equation 3-1}$$

$$Y_n = r_n \cos(\omega_n) \cos(\alpha_n) \quad \textbf{Equation 3-2}$$

$$Z_n = r_n \sin(\omega_n) \quad \textbf{Equation 3-3}$$

Where

r_n = Distance of sample number 'n' (mm)

ω_n = Elevation angle of sample number 'n' (°)

α_n = Azimuth angle of sample number (°)

n = Sample number

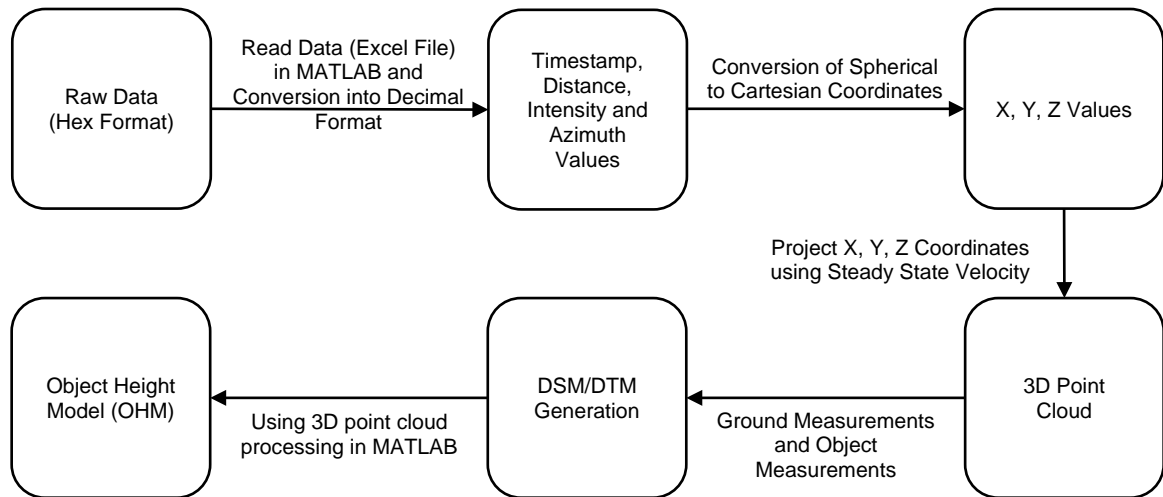


Figure 3-3: Data processing flow diagram showing the order of steps performed to acquire the object height model.

Since data logging was synchronized with the motion of the system, the first timestamp and last timestamps represented the beginning and end of the motion profile, respectively. The LiDAR position was determined using the timestamp and basic kinematic equations. Only data collected during the steady-state portion of each test were used in analysis. The resulting steady-state data was split into two data sets, one for forward motion and another for backward motion. Cartesian coordinates obtained were projected into 3D space in Z axis (direction of travel, Figure 2-6) using the steady state velocity by using the following formulae:

$$Z_{nt} = r_n \sin(\omega_n) + v \times (t - t_0) \quad \text{Equation 3-4}$$

$$Z_{nt} = r_n \sin(\omega_n) - v \times (t - t_0) \quad \text{Equation 3-5}$$

Where

Z_{nt} = Z coordinate of sample number 'n' at time 't' (mm)

r_n = Distance of sample number 'n' (mm)

ω_n = Elevation angle of sample number 'n' (°)

v = Steady state velocity (m/s)

t = instantaneous timestamp (s)

t_0 = initial timestamp (s)

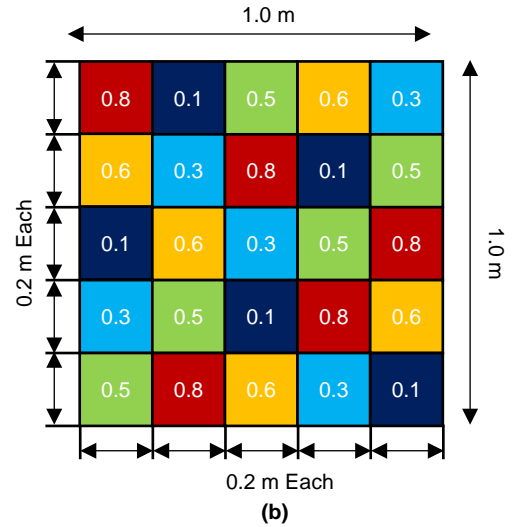
Equation 3-4 refers to the projection of Z coordinate in forward motion and Equation 3-5 refers to the projection of z coordinate in backward motion. Thus, the local 3D point cloud relative to the linear motion system was obtained after projecting the LiDAR data into 3D space. Refer to Appendix D.1 for the data processing of raw LiDAR data.

3.3.3 *Test Target Experiment*

A test target shown in Figure 3-4a was used to validate the LiDAR sensor when translating on the linear motion system at varying velocities. The target consisted of 25 anodized aluminum tubes of 5 different heights pseudo-randomly arranged in the order as shown in Figure 3-4b, where no height was repeated in a single row or column. The tubes were bolted onto a 1 m x 1 m x 1.9 cm medium-density fiberboard (MDF) base using countersunk machine screws. A magnet embedded in a 3D printed plastic mount was attached to the top of each tube. Sheet metal steel plates (0.2 m x 0.2 m x 0.3 cm) were centered on each magnetic mount. The plates were painted flat black on one side and flat white on reverse side to provide two relative intensity values when recording LiDAR data. Refer to Appendix E for CAD models for the test target.



(a)



(b)

**Figure 3-4: (a) Orientation and placement of the test target underneath the LiDAR
(b) Physical layout showing the arrangement of heights of the test target.**

The test target was located in the middle of the frame and the sides of the MDF base were aligned parallel to X and Z axes. The LiDAR sensor translated above and across the LiDAR test target. The LiDAR sensor scanned the test target while translating forward and backward for a single velocity profile. LiDAR data were collected in five different velocity profiles and replicated three times. Each replication was split into two different scans, one for forward motion and one for backward motion (5x3x2). The entire experiment was repeated for the white and black side of the target plates. The ground (i.e. floor) measurements were taken prior to target measurements at a lowest velocity profile. Velocities within each replication were pre-determined randomly and the same order was applied when scanning white and black targets.

Table 3-1 shows the order in which data were collected for each target color. Velocities within each replication were pre-determined randomly and the same order was applied when scanning white and black targets.

Table 3-1: Data Collection procedure of the LiDAR with the order of velocities

Velocity (m/s)	Replication Number
1.0	1
0.1	1
0.5	1
1.5	1
2.2	1
0.1	2
1.0	2
2.2	2
0.5	2
1.5	2
2.2	3
1.5	3
1.0	3
0.1	3
0.5	3

3.3.4 Point Cloud Processing of the Test Target

The DTM was obtained separately for both white and black targets immediately before scanning the targets. For obtaining the average height of the ground, a region of interest (ROI) with a dimension of ± 0.1 m in the y-axis at the expected ground level was applied to filter out extraneous points. The points lying inside were found and the point cloud corresponding to the ROI where the target would be located was obtained. The average height (y-coordinate) was found and used as a constant height DTM as the ground was relatively smooth.

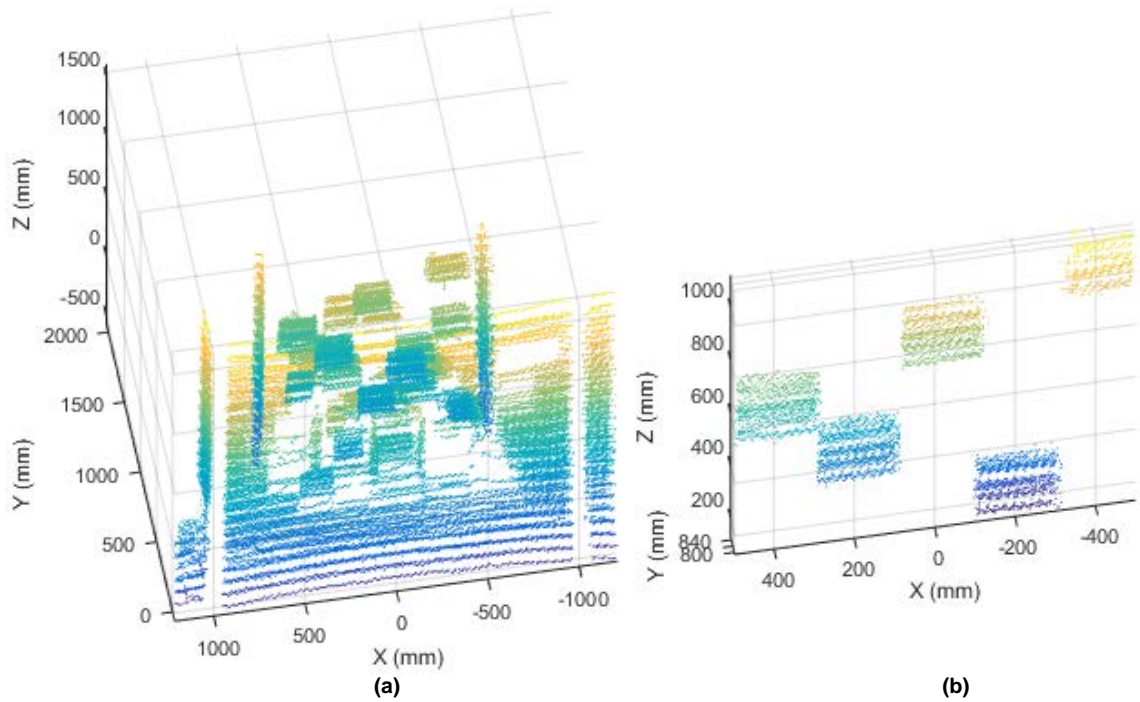


Figure 3-5: LiDAR point cloud of (a) Test target at a velocity of 1.0 m/s (OHM) and (b) Extracted 0.8 m targets at a velocity of 1.0 m/s.

For the test target measurements, both the forward and backward motion point cloud of the test target (Figure 3-5a), which was the resulting OHM, were generated from the constant height DTM and DSMs. There were two MATLAB scripts written for the forward and backward scans to obtain height measurements of the targets. Figure 3-5b shows the targets at 0.8 m extracted from the OHM using a ± 0.05 m ROI in the Y axis centered at a height of 0.8 m above the ground level. The points lying inside were found again and the point cloud containing the points, which were the 0.8 m targets, was obtained. From the point cloud the limits of the Z-axis were found for that particular ROI and the lower limit was selected which was termed as the Z-coordinate of the target. The Z-coordinate was used to find the location of other targets in Z-axis as the position of the targets were known relative to each other in the Z-axis.

Twenty-five different ROI's were defined for each of the 25 targets to extract points associated with each target. For a three-dimensional ROI corresponding to a particular plate, the y-axis range was a total of 0.15 m for the 0.1 m, 0.3 m and 0.8 m height targets, and 0.125 m for the 0.5 m and 0.6 m height targets. The decrease in the y-axis range for 0.5 m and 0.6 m targets was to avoid the overlap of the corresponding ROIs due to the smaller separation distance. The ROI range of x-axis and z-axis for each target was ± 0.1 m centered at the approximate center location of each target. This provided a large buffer around each ROI to ensure points from adjacent targets were not accidentally associated with a given target due to potential misalignment between the linear motion system and the test target.

The points lying inside each ROI were extracted and associated with the corresponding target. In addition to retaining the raw height measurements within a ROI for statistical analysis, attributes computed from the points in each ROI included estimated average height, standard deviation in height, number of points on a particular target (point density), missing targets (targets where there were less than 10 points on them due to high speed or FOV obstruction due to adjacent targets), and the average intensity. All processed data were organized into a single Excel file where each spreadsheet consisted of one replication with five different velocity profiles and twenty-five different targets of five different heights. Refer to Appendix D.2 for feature extraction of test target from point cloud.

3.4 Results and Discussion

3.4.1 LiDAR Firmware Version

Drift in the average distance to a stationary object was encountered when initially testing LiDAR, which was in excess of 10 cm from the time LiDAR was powered on until reaching

a steady-state value. Drift was determined to be due to changes in the internal temperature of the LiDAR and was mitigated through a firmware update. The maximum variability in measurements to the ground surface were less than 0.5 cm when using the updated firmware, which was well within the accuracy specifications of the LiDAR. Figure 3-6 illustrates the distance to the ground surface measured by the LiDAR as a function of LiDAR temperature. Note that newer versions of the firmware were available but not used in this experiment due to the need to return the sensor to the manufacturer for update beyond version 3.0.24.1.

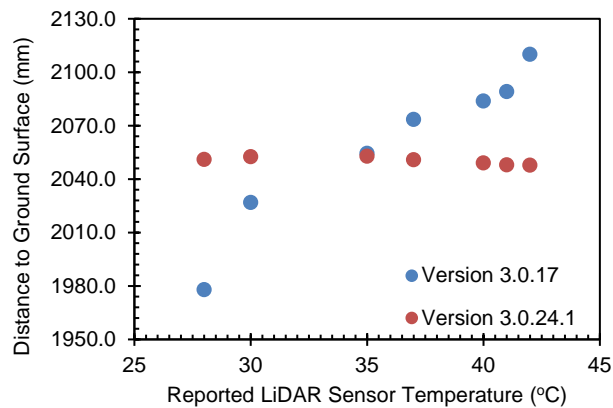


Figure 3-6: Temperature response of the LiDAR sensor as exhibited by measured distance the ground surface at varying LIDAR sensor temperatures.

3.4.2 Test Target Experiment

The purpose of this experiment was to study the accuracy and precision of a LiDAR sensor when measuring a predefined test target and assess its performance for different velocities and target heights. The estimated height of a single target from the LiDAR was calculated as the average height of all the raw data points within the boundary of the target. Target height error was calculated by subtracting the actual height from the estimated

height. Figure 3-7 shows the height measurement error results as box and whisker plots for both height and velocity when measuring the white and black targets.

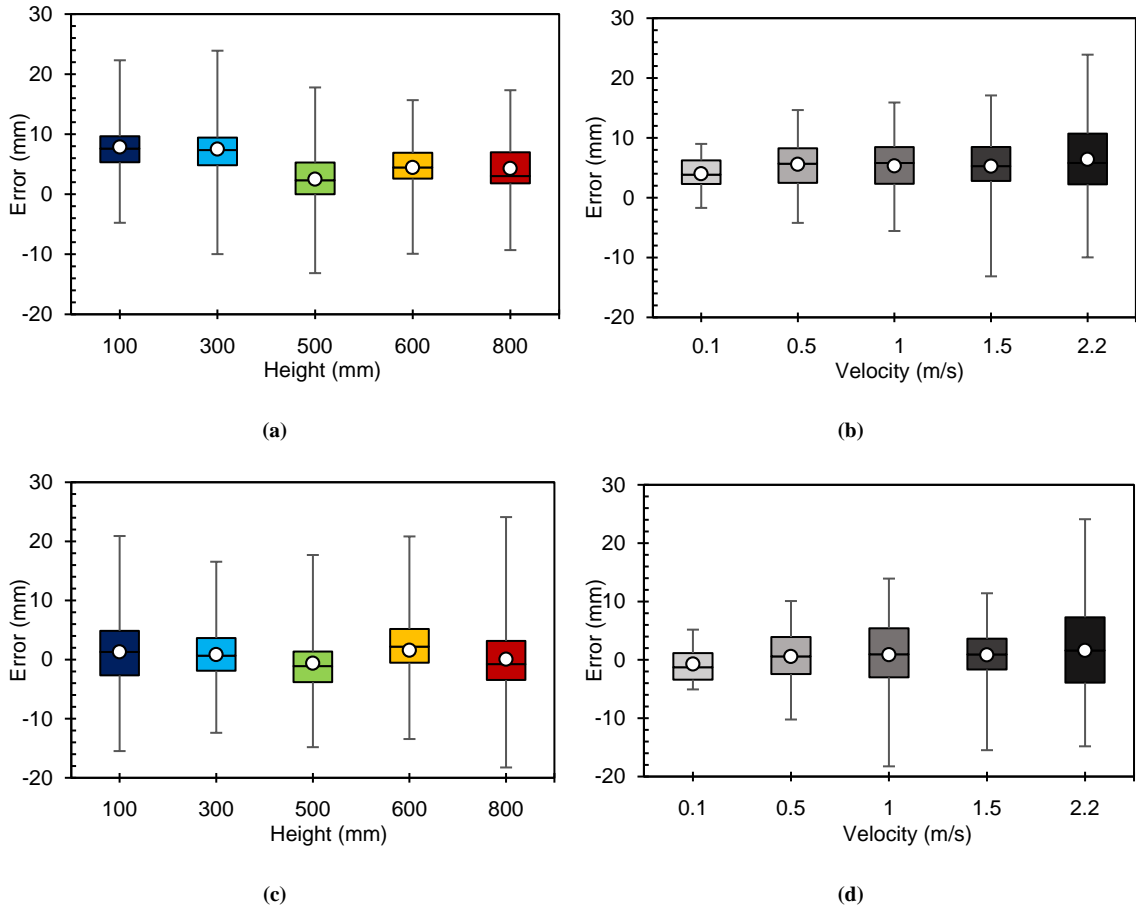


Figure 3-7: Box and whisker plots of estimated height error for (a) white targets vs. actual height (b) white targets vs. actual velocity (c) black targets vs. actual height and (d) black targets vs. actual velocity.

For example, in the Figure 3-7a, the red box and whisker plot shows the distribution of error for the 0.8 m targets across different replications, target numbers and velocities for the white targets. The box shows the 25% to 75% quartile values with the midline as the median of the data and the circular dot as the mean of the data. The whiskers show the minimum and maximum values of the data. From the box and whisker plots of the estimated height error, it was visually interpreted that error was more consistent between

target heights than LiDAR sensor velocity. As the velocity increases the width of the error distribution increased (higher kurtosis). There was also an increase in mean error as the velocity increased but difficult to interpret visually.

Table 3-2 provides the height measurement error values averaged across all replications for given actual target heights and LiDAR sensor velocities. The average height measurement error across the whole experiment was determined to be 5.3 mm for the white targets and 0.6 mm for the black targets. This result was particularly interesting given that the white targets had an average relative intensity of 88.2 while the black targets had an average relative intensity of 5.4. While only two target types are likely not enough to quantify a trend in error due to emissivity, it appears to contribute to error, albeit less than the specified accuracy of the LiDAR sensor.

Table 3-2: Average error values of the white and black targets at different velocities and heights

Average Height Measurement Error White Targets (mm)						Average Height Measurement Error Black Targets (mm)				
Actual Height (m)	Velocity (m/s)					Velocity (m/s)				
	0.1	0.5	1.0	1.5	2.2	0.1	0.5	1.0	1.5	2.2
0.1	6.1	8.4	7.5	7.5	9.7	-0.8	1.0	2.8	1.1	2.5
0.3	5.8	7.9	7.7	7.3	9.0	-0.9	1.0	0.7	1.2	2.2
0.5	1.0	2.7	1.8	2.2	4.7	-2.0	-0.2	-0.5	-0.6	0.3
0.6	3.7	4.4	5.5	4.5	4.0	0.9	1.3	1.7	2.1	1.8
0.8	3.3	4.6	4.1	4.8	4.6	-0.9	-0.2	-0.2	0.4	0.9

The standard deviation of a single target was calculated as the standard deviation of the heights of all the points inside the target boundary. Figure 3-8 shows the height measurement variability results as box and whisker plots for both height and velocity when measuring the white and black targets.

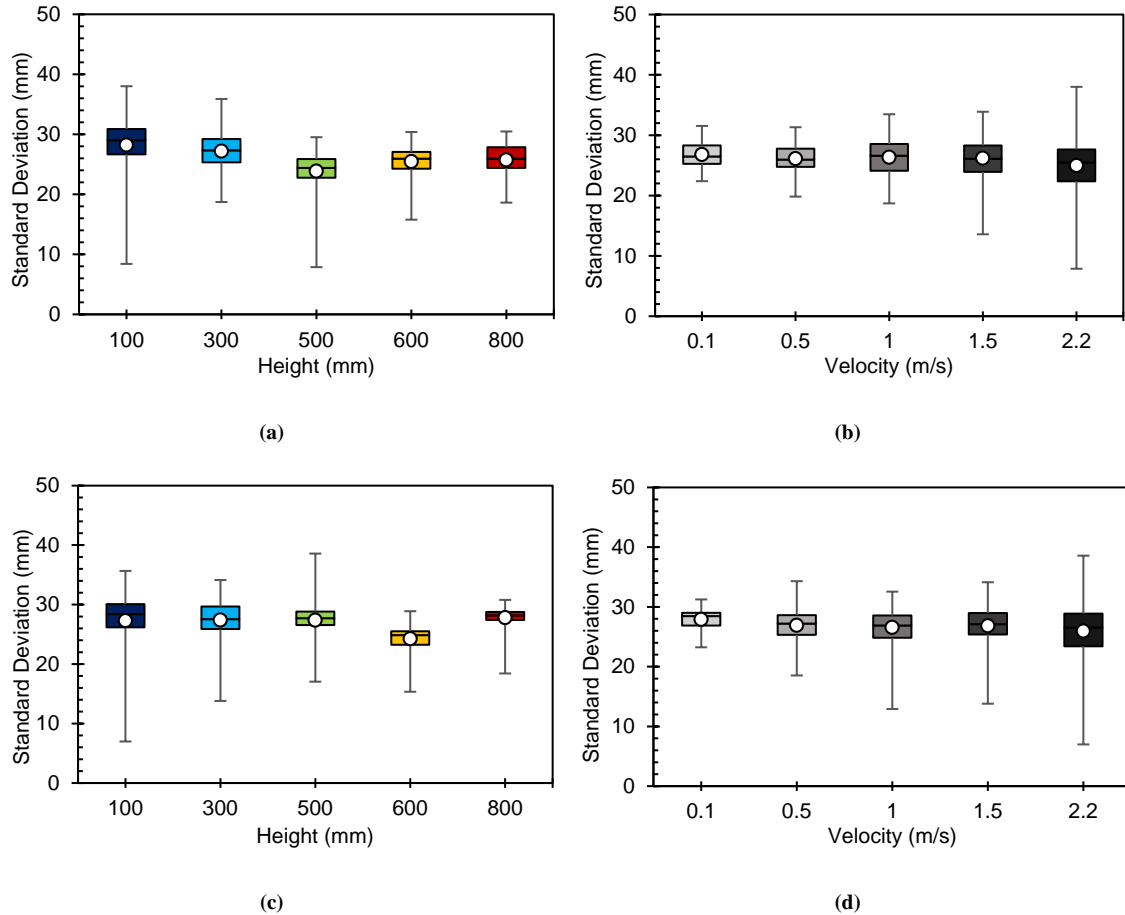


Figure 3-8: Box and whisker plots of standard deviation for (a) White targets vs. actual height (b) White targets vs. actual velocity (c) Black targets vs. actual height (d) Black targets vs. actual velocity.

From the box and whisker plots of standard deviation, it was visually interpreted that the standard deviation values were consistent with height with exception of the 0.5 m targets in Figure 3-8a and 0.6 m targets in Figure 3-8c. Standard deviation values were also spread out more on both the targets as the velocity increases (higher kurtosis). In contrast to error values, the average standard deviation decreased as the LiDAR sensor velocity increased. Several explanations for this result may exist but the most plausible is that variability in height measurements for a given target (i.e., noise) tends to be smaller when fewer measurements are present over a shorter period. In other words, translating a LiDAR sensor

over a target slowly to capture a denser point cloud may increase the noise present due to external factors not controlled for in this experiment.

Table 3-3 provides the standard deviation of the estimated height averaged across all targets and replications for different actual heights and velocities. The average standard deviation across the whole experiment was 26.0 mm for the white targets and 26.9 mm for the black targets.

Table 3-3: Average standard deviation values of the white and black targets at different heights and velocities

Average Standard Deviation White Targets (mm)						Average Standard Deviation Black Targets (mm)				
Actual Height (m)	Velocity (m/s)					Velocity (m/s)				
	0.1	0.5	1.0	1.5	2.2	0.1	0.5	1.0	1.5	2.2
0.1	28.7	28.1	28.6	28.6	27.8	29.4	27.4	26.2	27.6	25.8
0.3	27.5	26.9	27.5	27.4	27.2	29.1	27.6	27.2	27.2	28.1
0.5	24.4	24.0	24.8	24.2	20.6	28.1	27.2	27.6	27.4	26.4
0.6	25.2	25.4	26.0	25.9	24.8	24.9	24.4	24.2	24.1	23.6
0.8	25.7	25.9	25.1	25.8	24.9	28.4	28.1	27.8	28.1	26.5

Point density and the intensity values were some of the other important parameters of the LiDAR data studied. The point density of a single target was calculated as the number of points recorded within the boundary of the target. Figure 3-9a illustrates the point density for the white targets averaged across all targets and replications for different heights versus the LiDAR sensor velocity. As velocity increased, point density decreased following a power series. Figure 3-9b illustrates the point density for the white targets averaged across all the targets and replications for different LiDAR sensor velocities versus the actual target height. Point density increased linearly with increase in height at a given velocity. The linear model resulted in a poorer fit for height than the power series model did for velocity due to the physical structure of the targets. Higher targets obstruct the line of sight to lower targets, and the pseudo-random distribution of target heights is what caused the deviations

between the linear model and point density for varying height. The black targets exhibited similar point density results to the white targets.

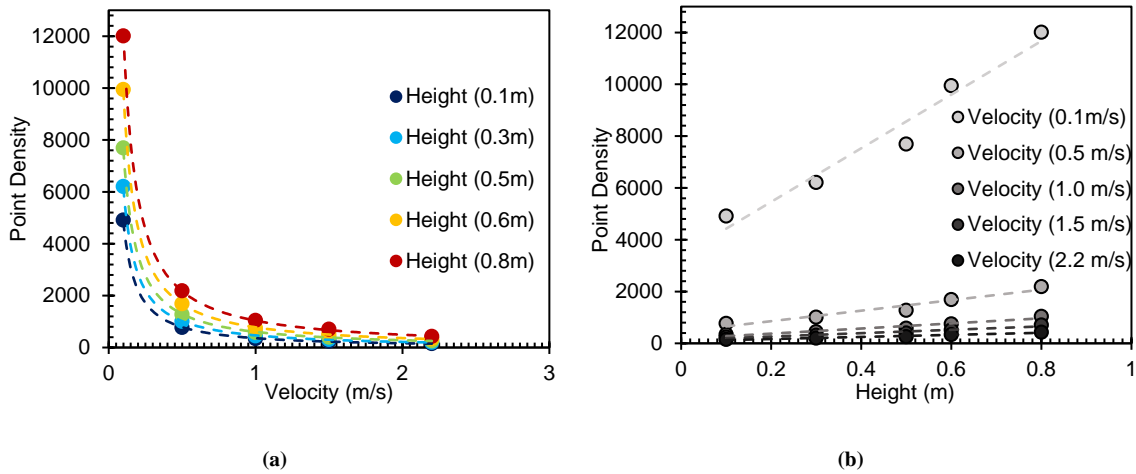


Figure 3-9: (a) Relationship showing the correlation of point density with velocity at different heights (b) Relationship showing the correlation of point density with height at different velocities.

The intensity of a single target was calculated as the average intensity of all the points within the boundary of the target. Table 3-4 provides the intensity values averaged across all the targets and replications for different velocity profiles and different heights. The average relative intensity value across the whole experiment was 88.2 for the white targets and 5.4 for the black targets and did not substantially deviate with change in velocity and actual height. The average intensity values agreed with the LiDAR specifications, which were intensities of 100 for perfectly white and 0 for perfectly black.

Table 3-4: Average intensity values of white and black targets at different heights and velocities

Actual Height (m)	Average Intensity White Targets					Average Intensity Black Targets				
	Velocity (m/s)					Velocity (m/s)				
	0.1	0.5	1.0	1.5	2.2	0.1	0.5	1.0	1.5	2.2
0.1	84.7	85.2	84.8	84.9	84.9	5.9	6.1	6.0	6.6	6.3
0.3	89.5	90.7	90.6	90.4	88.5	5.3	5.5	5.5	5.5	5.4
0.5	88.7	90.5	90.5	90.2	89.8	5.4	5.6	5.7	5.6	5.5
0.6	88.9	89.6	88.5	89.0	88.9	5.3	5.4	5.4	5.4	5.3
0.8	86.8	87.3	88.1	88.1	85.5	4.6	4.7	4.7	4.7	4.8

3.4.3 Statistical Analysis of the Test Target

Statistical analysis was performed to determine if significant differences between errors and standard deviations existed for different heights and velocities. Refer to Appendix G for the statistical analysis software used in this chapter. The raw data was fed into a SAS script (9.4, SAS, Cary, NC) and was sorted by target height, replication, velocity and target number and averaged across all the data points. The height measurement error was calculated as the difference of estimated and actual target height. Then the height measurement error was modeled with four classes consisting of velocity, target number, replication and actual height. The replication and target number were considered as the random blocks and the target-to-target variability was included. The height measurement error was estimated using actual height, velocity and interaction between actual height and velocity. Table 3-5 provides the results of the statistical analysis of error prediction of the white targets.

Table 3-5: Statistical analysis of the white targets: (a) Error model (b) Velocity

tukey grouping table

TYPE III Tests of Fixed Effects					Tukey-Kramer Grouping for Velocity Least Squares Means (Alpha=0.05)			
Effect	Num DF	Den DF	F Value	Pr > F	LS-means with the same letter are not significantly different			
Actual Height	4	20	2.93	0.0464	Velocity (m/s)	Estimate (mm)	Tukey Grouping	
Velocity	4	20	4.06	0.0143	2.2	6.4141	A	
Velocity*Actual	16	680	1.36	0.1567	0.5	5.5810	B	A
					1.0	5.3101	B	A
					1.5	5.2413	B	A
					0.1	3.9931	B	

(a)

(b)

The p-value of the interaction factors (Table 3-5a) was ≥ 0.05 , so the interaction factor was considered insignificant. The velocity has a p-value < 0.05 making it a significant factor affecting measurement error. The p-value of the height factor was slightly less than 0.05, indicating weak significance. The Tukey grouping table (Table 3-5b) was studied for different velocities and their error estimates, and used to detect which values or group of values in a particular factor was causing it to be a significant factor. It was found that the error estimates of 0.1 and 2.2 m/s velocities were significantly different and the error estimate increased as the velocity increased. The Tukey grouping of the height (not shown) did not exhibit any significantly different errors in heights, ruling out height as the significant factor affecting the error of the white targets.

The same statistical analysis was repeated for the black targets. Table 3-6 provides the summary of the statistical analysis of error prediction of the black targets.

Table 3-6: Statistical analysis of the black targets: (a) Error model (b) Velocity

tukey grouping table

TYPE III Tests of Fixed Effects				
Effect	Num DF	Den DF	F Value	Pr > F
Actual Height	4	20	0.42	0.7894
Velocity	4	20	4.33	0.0110
Velocity*Actual	16	680	0.79	0.7020

(a)

Tukey-Kramer Grouping for Velocity Least Squares Means (Alpha=0.05)			
LS-means with the same letter are not significantly different			
Velocity (m/s)	Estimate (mm)	Tukey Grouping	
2.2	1.6134		A
0.5	0.8787	B	A
1.0	0.8347	B	A
1.5	0.5747	B	A
0.1	-0.7177	B	

(b)

The p-values for both the interaction and actual height was > 0.05 (Table 3-6a) making them both insignificant factors. The velocity was again a significant factor (p-value < 0.05) affecting the error. The Tukey grouping table (Table 3-6b) was again studied for different velocities and their error estimates were similar to the white targets. There were significant differences between slowest and fastest velocity profiles and the error estimate increased as the velocity increased. The slowest velocity profile has a negative error estimate inferring that it underestimated the heights of the targets whereas the faster velocities have a positive error estimate by overestimating the heights of the black targets.

The standard deviation for both the targets were modeled similar to the error. Table 3-7 refers to the statistical analysis of standard deviation prediction of the white targets.

Table 3-7: Statistical analysis of the white targets: (a) Standard deviation model (b)

Velocity tukey grouping table (c) Height tukey grouping table

TYPE III Tests of Fixed Effects					Tukey-Kramer Grouping for Velocity Least Squares Means (Alpha=0.05)			Tukey-Kramer Grouping for Height Least Squares Means (Alpha=0.05)		
Effect	Num DF	Den DF	F Value	Pr > F	LS-means with the same letter are not significantly different			LS-means with the same letter are not significantly different		
Actual Height	4	20	3.18	0.0357						
Velocity	4	20	10.80	<0.0001	Velocity (m/s)	Estimate (mm)	Tukey Grouping	Height (mm)	Estimate (mm)	Tukey Grouping
Velocity * Actual Height	16	680	1.10	0.3489	0.1	26.8011	A	0.1	28.2014	A
					1	26.3524	A	0.3	27.1905	B A
					1.5	26.1795	A	0.8	25.7138	B A
					0.5	26.1131	A	0.6	25.4516	B A
					2.2	24.9717	B	0.5	23.8606	B

The p-value for the interaction was > 0.05 , making it insignificant. The p-values for the velocities and heights were < 0.05 making them significant factors affecting the standard deviation of measured height. The Tukey grouping tables (Table 3-7b, Table 3-7c) were studied for different velocities/heights and their standard deviation estimates. From the Tukey grouping table of velocity (Table 3-7b), the standard deviation of the slowest velocity was significantly different from the faster ones and the estimate was decreased on an average as the velocity increased. Less point density at faster velocity profiles was the likely reason behind the reduction in standard deviation. From the Tukey grouping table of height (Table 3-7c), the standard deviation estimate of the 0.5 m targets were significantly different than the other heights causing the target height as one of the significant factors of standard deviation. It was proved by the fact that after excluding the 0.5 m target height data the height factor became insignificant.

The same statistical analysis on standard deviation was repeated for the black targets. Table 3-8 summarizes the statistical analysis of standard deviation prediction of the black target.

Table 3-8: Statistical Analysis of the black targets: (a) Standard deviation model (b) Velocity tukey grouping table (c) Height tukey grouping table

TYPE III Tests of Fixed Effects					Tukey-Kramer Grouping for Velocity Least Squares Means (Alpha=0.05)			Tukey-Kramer Grouping for Height Least Squares Means (Alpha=0.05)		
Effect	Num DF	Den DF	F Value	Pr > F	LS-means with the same letter are not significantly different			LS-means with the same letter are not significantly different		
Actual Height	4	20	3.07	0.0401						
Velocity	4	20	11.06	<0.000	Velocity (m/s)	Estimate (mm)	Tukey Grouping	Height (mm)	Estimate (mm)	Tukey Grouping
			1		0.1	27.8919	A	800	27.7959	A
Velocity *Actual Height	16	680	1.21	0.2533	0.5	26.8911	B	300	27.3925	B
					1.5	26.8210	C	500	27.3618	B
					1.0	26.5436	C	100	27.2752	B
					2.2	25.9273	C	600	24.2494	B

The p-value for interaction was > 0.05 making it insignificant. The p-values for the velocities and heights were significant (< 0.05) making them a significant factor affecting the standard deviation. The Tukey grouping tables (Table 3-8b, Table 3-8c) were studied for different velocities, heights and their standard deviation estimates. From the Tukey grouping table of velocity (Table 3-8b), the standard deviation estimate of the slower velocities were significantly different from the faster ones and the estimate decreased as the velocity increased similar to the white targets. From the Tukey grouping table of height (Table 3-8c) the standard deviation estimate of 0.6 m targets were significantly different than the other heights, resulting in target height as one of the significant factors of standard deviation. It was also shown by the fact that after excluding the 0.6 m target height data the actual height factor became highly insignificant.

Target height was a significant factor only affecting the standard deviation in height measurements. The ROI was less in height (y-coordinate) for 0.5 m and 0.6 m targets than the other targets due to being 0.1 m apart. Increasing ROI for the both targets will result in

overlap of both ROI, generating inaccurate measurements. The decreased ROI for both the heights had an impact on their standard deviations causing them to be a significant factor affecting the standard deviation.

Finally, a t-test was conducted for the errors between white and black targets across different replications, velocities and heights. Table 3-9 summarizes t-test results of the errors between the white and black targets.

Table 3-9: T-test of the errors between white and black targets

Equality of Variances				
Method	Num DF	Den DF	F Value	Pr > F
Folded F	749	749	1.15	0.0524
Test Statistics				
Method	Variances	DF	t Value	Pr > t
Pooled	Equal	1498	-19.06	<0.0001
Satterthwaite	Unequal	1490.5	-19.06	<0.0001

The equality of variances came out to be insignificant concluding that both the while and black targets have an equal variance or distribution of errors. As the variances was determined to be insignificant, the pooled statistic was given the priority over Satterthwhite, and was significant as the p-value was < 0.05.

3.5 Conclusions

A linear motion test fixture was used to control the dynamics of a LiDAR sensor. A test target was fabricated to determine the effects of target height and LiDAR velocity on the accuracy and precision of height measurements. Results showed that height measurement error increased as the velocity increased, concluding that accuracy decreases as the velocity increases. The variability of error in height measurements also increased as the velocity of

the LiDAR increased. Although the statistical analysis showed a significant difference between the faster and slower velocity profiles, the difference was approximately 1 mm over the range of target heights and LiDAR velocities tested. The standard deviation estimate followed an opposite trend, as the velocity increased the standard deviation decreased by approximately 1 mm over the range of target heights and LiDAR velocities tested. Statistical analysis showed a difference in standard deviation of height measurements between faster and slower velocity profiles.

In total, these results conclude that the small changes in target height and LiDAR velocity will affect the accuracy and precision of LiDAR measurements. The effect is small and may not be substantial for agricultural applications, where other sources of error, such as moving crop canopies or error in resolving position of the sensor are more likely to dominate overall measurement error. The velocity of the LiDAR will be a tradeoff variable with lower velocities having higher point densities, higher variability, and higher post-processing times, and with higher velocities having lower point densities, lower variability, and lower post-processing time.

CHAPTER 4: PHYSICAL MODELING OF ALFALFA USING LIDAR INTEGRATED ONTO A LINEAR MOTION TEST FIXTURE

4.1 Summary

To summarize the chapter, the linear motion test fixture from CHAPTER 2 was placed over a quadrat (1 m square frame) to scan the alfalfa crop. Two alfalfa plots were scanned at five different velocities and 3 replications of each. Post processing of the raw data and point cloud processing was done to extract mean height, max and min height, point density, difference between max and min height, percent of points more than half the mean height and less than 25% of the mean height. Reference data were obtained through photogrammetry and field measurements. Insufficient alfalfa plots were scanned which prevented any statistical analysis from being used to compare the different methods. However, the comparison between LiDAR and photogrammetric data showed some promising results which may be further replicated in the future.

4.2 Introduction

Alfalfa is a type of forage crop which grows in most parts of the world. It requires warmer temperate climates for optimum growth. Fully grown alfalfa reach canopy heights between 0.6 m and 1.0 m. Alfalfa has a high nutritional value and is one of the most cultivated forage crop in the world (Radović et al., 2009). Alfalfa is typically harvested three to four times a year and usually rotated with other plants. Alfalfa is commonly a uniform and dense crop grown in large quantity, which makes it difficult to extract physical properties at field scales.

Modeling spatial variability in crops is essential for precision management as summarized in CHAPTER 1. This study deals with the computation of point cloud of the

alfalfa crop from the LiDAR data and processing the point cloud to obtain the physical characteristics of the alfalfa. The results obtained from the LiDAR will be compared with the field based and photogrammetric measurements. Photogrammetry is the process of computing spatial measurements from a three dimensional model derived from a series of overlapping photographs taken from different viewing angles. There are many studies on the use of photogrammetry to extract various crop parameters and researchers have found that the results obtained can achieve spatial resolution at the centimeter level (Colomina and Molina, 2014). Grenzdörffer, 2014 conducted a study on using photogrammetry to determine the heights of oilseed rape, corn, and wheat crops, and found a high correlation ($R^2 > 0.9$) between the photogrammetric and field measured data. Balenović et al., 2015 also estimated the mean tree height of different types of forest stands with an error of 1.5% on average and concluded that photogrammetry is a viable option to find the heights of forest stands.

Both LiDAR and photogrammetry are rapidly advancing technologies and have their own set of advantages and disadvantages. There are notable differences between LiDAR and photogrammetry in terms of how the resulting data can be used to obtaining crop parameters, in addition to environmental considerations required for optimum performance. LiDAR provides positional, spectral and echo information of objects with high resolution, with a focus on more of the geometric information while photogrammetry focuses more on spectral information (typically red, green, and blue wavelengths for visible imagery) of the object that is being scanned. Although LiDAR has more spatial accuracy, the 3D images from photogrammetry are easier to interpret by humans and can extract

other information like the texture and multichannel reflectance (Nex and Rinaudo, 2011; Mesas-Carrascosa et al., 2012).

There have been many studies regarding the comparison between the LiDAR and photogrammetry, and also the possibility of integrating both technologies to achieve higher resolution as they have complementary benefits. Nex and Rinaudo, 2011 conducted a study on using a LiDAR-derived DSM with a photogrammetric-derived DTM to obtain an OHM used to extract features of a building. In contrast St-Onge et al., 2008 conducted a study on a canopy height model (CHM) using a LiDAR-derived DTM with a photogrammetric-derived DSM to measure the canopy height, but found out that they achieved lower resolution than using the LiDAR alone to derive the CHM. More research is needed to see if there are benefits in integrating both the techniques when making spatial and spectral measurements.

The objective of this chapter was to collect preliminary data of alfalfa using the LiDAR sensor when integrated onto the linear motion test fixture. The LiDAR data will also be compared to field and photogrammetric data. The specific objectives were as follows:

- 1) Collect the LiDAR data from alfalfa at varying forage quality factors of height and density
- 2) Determine the alfalfa physical parameters by 3D processing the data
- 3) Correlate the LiDAR measurements with the field-based measurements and photogrammetric measurements

4.3 Materials and Methods

4.3.1 Test Setup

This study was conducted in at the C. Oran Little Research Center (Woodford County Farm) in Versailles, KY. The alfalfa was about 0.5 m to 0.8 m tall and were scheduled to be harvested within a few days. A quadrat made of 1-1/2 PVC Schedule 40 pipe of dimensions 1.0 m x 1.0 m x 1.0 m was used as the study area for alfalfa. The quadrat was placed on the alfalfa crop as shown in the Figure 4-1. The nominal outside diameter of the PVC pipes were 1.9 in (48.3 mm), which will be used in the point cloud processing of alfalfa to identify the quadrat.



Figure 4-1: Quadrat of dimensions 1.0 m x 1.0 m x 1.0 m placed in an alfalfa field.

The linear motion test fixture with the LiDAR sensor integrated onto it was placed on top of the quadrat as shown in the Figure 4-2 to scan the alfalfa inside the quadrat.



Figure 4-2: Modeling the alfalfa using the linear motion system over a quadrat

The LiDAR was run at 5 different velocity profiles with 3 replications as shown in the Table 3-1. Two plots with a high percentage of lodged plants were studied using the LiDAR sensor. The plots were also studied using a UAS (Phantom 4, DJI) and photogrammetry software (Pix4Dmapper Pro, Pix4D).

4.3.2 Point Cloud Processing of Alfalfa

Raw data of the alfalfa crop obtained from the LiDAR were processed similar to the method described in Section 3.3.2 to obtain the canopy height model (CHM) of alfalfa. The

ground plane was not scanned so the DTM was measured manually. A distance of 2087.0 mm from the LiDAR was taken as the constant height DTM for both the plots.

A region of interest (ROI) with dimensions of ± 0.1 m centered at height of 1.0 m in the y-axis was applied for over the entire range of the xz-axis to extract the points relating to the quadrat frame as it was positioned at a height of 1.0 m above the ground. The points lying inside were found and were attributed to the quadrat. A predefined function in MATLAB *pcdenoise* was applied to the quadrat point cloud to remove noise and to extract just the quadrat and remove any outliers. The limits of the boundary of the quadrat was found in the xz-axis. The alfalfa inside the quadrat was only of the interest to the study, so new limits in the xz-axis were defined for the alfalfa crop which were obtained from the limits of the boundary of the quadrat by adding the dimension of the outer diameter of 48.3 mm to the smaller limit and by subtracting it from the larger limit of the xz-axis to only include the alfalfa crop inside the test target. A new region of interest (ROI) with a dimension of 1.0 m in the y-axis and with the new limits in the xz-axis was applied. The points lying inside the ROI were found and were attributed to the alfalfa crop. Figure 4-3 shows point clouds of the full LiDAR scan, quadrat, and the alfalfa inside the quadrat.

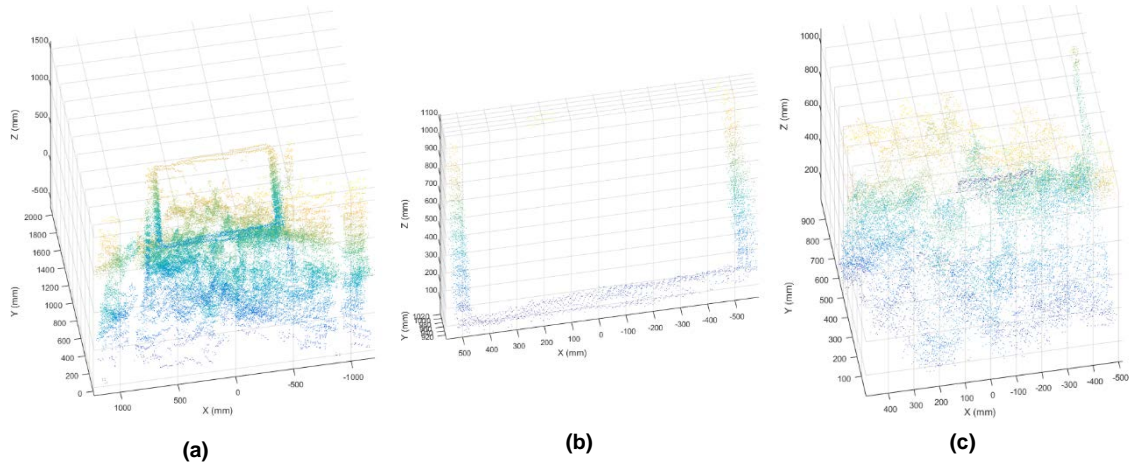


Figure 4-3: Points cloud of (a) Alfalfa and the quadrat (b) Quadrat (c) Alfalfa inside the quadrat

For the height distribution of the alfalfa, the mean of all the points was taken as the mean height of the alfalfa. The maximum and the minimum heights of the all the points in the y-axis, difference between the max and min height, point density, percentage of points more than half the mean height, and percentage of points less than 25% of the mean height among all the points were computed. Volume was also one of the desired physical properties of a crop. This study also attempts to compute the total volume occupied by the alfalfa crop. From the volume and the density of the alfalfa, the biomass within the quadrat can be predicted. The volume of the alfalfa crop was computed from four different methods in this study:

- 1) First Method (Block Method): The whole area in the xz-axis was divided into 100 x 100 blocks. All the points of the alfalfa crop inside the quadrat was distributed into those 100 x 100 blocks. The volume of each block was found from its area in its xz-axis multiplied by its average height computed for all the points inside the corresponding block in the y-axis. The volume of all the blocks were added to give the final volume of the alfalfa.

- 2) Octree Method: A MATLAB function Octree (Sven, 2013) from the MATLAB file exchange libraries was used for this method. The whole alfalfa point cloud was divided into octrees (analogous to quadtrees in 2D volume). The volume from each of the octree blocks in their smallest stage was added to give the final volume of the alfalfa.
- 3) Alpha Shape Method: A predefined MATLAB function *alphashape* was used for this method which computes the bounding volume that envelops the 3D point cloud of the alfalfa. The bounding volume was computed and attributed to the final volume of alfalfa.
- 4) Cube Method: This was a rudimentary/crude method to compute the volume. The limits of the boundary of the alfalfa point cloud was found and multiplied together as $V = \text{Length} \times \text{Width} \times \text{Height}$ to give the final volume of alfalfa.

Apart from these methods, the average maximum and minimum heights of the points in an approximate 50 mm x 50 mm square were also computed by using the block method and finding the average heights of all the 5 x 5 adjacent blocks. This was done two ways:

- 1) Side by Side: The whole 100 x 100 blocks were divided into 20 x 20 blocks
- 2) Corresponding: Each combination of 5 x 5 squares combination was taken to yield 96 x 96 blocks.

Due to large percentage of lodged plants, the height of alfalfa computed only gives the apparent height of alfalfa instead of the true height of alfalfa. This study also attempts to compute the true height from the apparent height. This study provides only a preliminary technique to compute the true height and the method used may not be broadly applicable. The idea behind the algorithm was that lodged plants have more point density than the

surrounding plants. To accomplish this, the height and point density values from each of the 100 x 100 blocks was found. Mean and standard deviation of the point density of the blocks were computed. If any block had a point density greater than the mean point density plus two standard deviations, then the block was termed to have lodged plants. A sine of the average lodged angle of 30° was applied over the heights of those lodged plants and the mean, max and min height of all the plants was computed.

One of the other concerns was the presence of large number of ground points in the alfalfa point cloud, so a threshold was applied to differentiate between ground and points of alfalfa. The threshold applied in this study was 200 mm. Any points which have a height less than 200 mm were considered ground points and not included in the statistical analysis of alfalfa. The mean height of alfalfa was computed again after applying the threshold. All parameters were written to an Excel file consisting of two spreadsheets, with each sheet corresponding to a separate alfalfa plot. Each spreadsheet contained six replications each plot grouped for five different velocity profiles. Refer to Appendix F.1 for software regarding the feature extraction to model the alfalfa.

The photogrammetry method was effectively a turn-key process, for which the results of only the second plot were obtained. The video from the digital camera for the second plot was “stitched” using a photogrammetric software to obtain the point cloud for the alfalfa. The point cloud was imported in a MATLAB script to extract similar parameters as the LiDAR data.

4.3.3 Field Measurements

Field measurements were taken with the help of a carbon fiber GPS pole (Trimble, Inc.). Field measurements were taken at four to five different places within the quadrat. The

measurements were also taken by straightening lodged plants and measuring their height normal to the ground. The average of these four to five measurements was taken as the mean height of the alfalfa and the max was taken as the maximum height of alfalfa.

4.4 Results and Discussion

4.4.1 Alfalfa Physical Parameters

Average of the six replications for physical parameters of alfalfa obtained through the point cloud processing was calculated and were shown below in Table 4-1 and Table 4-2.

Table 4-1: Average physical properties across six replications of alfalfa for plot 1 at different velocities

Physical Parameters	0.1 m/s	0.5 m/s	1.0 m/s	1.5 m/s	2.2 m/s
Mean Height (mm)	420.35	385.93	359.64	357.38	329.17
Max Height of all Points (mm)	1000.00	999.56	999.32	957.91	841.95
Min Height of all Points (mm)	0.02	0.10	0.40	0.35	0.45
Max - Min (mm)	999.98	999.46	998.92	957.55	841.50
Total Samples	333245.67	38228.17	16081.50	11056.17	6516.33
More than Half Height (%)	34.19	29.23	24.43	26.74	29.46
Thinness (%)	4.48	4.54	4.89	4.84	5.04
Max Height in 50 mm x 50 mm (Side by Side) (mm)	778.02	712.82	701.04	619.87	428.06
Min Height 50 mm x 50 mm (Side by Side) (mm)	38.99	18.66	6.50	3.48	1.64
Max Height in 50 mm x 50 mm (Corresponding) (mm)	792.82	741.77	717.19	660.37	481.43
Min Height in 50 mm x 50 mm (Corresponding) (mm)	36.60	15.24	3.12	1.65	0.00
Volume by Block method (mm ³)	5.32E+08	3.18E+08	2.11E+08	1.78E+08	1.15E+08
Volume by Octree method (mm ³)	6.98E+08	4.03E+08	3.88E+08	3.13E+08	2.31E+08
Volume by Alpha shape (mm ³)	3.26E+08	2.30E+08	2.26E+08	2.27E+08	1.83E+08
Volume by Cube method (mm ³)	1.48E+09	9.77E+08	8.77E+08	8.36E+08	6.61E+08
Mean Height Lodged (mm)	476.83	452.66	415.01	414.96	363.33
Max Height Lodged (mm)	1467.62	1402.25	1391.05	1174.87	799.82
Min Height Lodged (mm)	38.99	18.66	6.50	3.48	1.64
Mean Height Threshold (mm)	485.77	448.60	425.88	423.39	392.97

Table 4-2: Average physical properties across six replications of alfalfa for plot 2 at different velocities

Physical Parameters	0.1 m/s	0.5 m/s	1.0 m/s	1.5 m/s	2.2 m/s
Mean Height (mm)	484.76	444.85	452.73	436.92	441.17
Max Height of all Points (mm)	999.99	999.88	999.47	895.87	868.34
Min Height of all Points (mm)	0.05	0.22	4.46	3.49	25.82
Max - Min (mm)	999.95	999.66	995.01	892.38	842.52
Total Samples	302215.00	43068.83	19336.50	13664.50	7652.83
More than Half Height (%)	40.03	35.59	36.18	49.82	51.56
Thinness (%)	2.86	2.88	2.38	2.54	2.17
Max Height in 50 mm x 50 mm (Side by Side) (mm)	754.57	679.67	643.26	636.45	550.70
Min Height 50 mm x 50 mm (Side by Side) (mm)	27.48	22.42	8.14	3.52	0.00
Max Height in 50 mm x 50 mm (Corresponding) (mm)	853.54	750.15	757.28	715.89	610.74
Min Height in 50 mm x 50 mm (Corresponding) (mm)	20.53	19.02	3.84	0.64	0.00
Volume by Block method (mm ³)	5.22E+08	3.94E+08	3.33E+08	2.79E+08	1.65E+08
Volume by Octree method (mm ³)	7.46E+08	4.65E+08	4.46E+08	3.85E+08	3.04E+08
Volume by Alpha shape (mm ³)	2.34E+08	2.34E+08	2.64E+08	2.71E+08	2.16E+08
Volume by Cube method (mm ³)	1.21E+09	9.95E+08	1.01E+09	9.09E+08	7.47E+08
Mean Height Lodged (mm)	538.35	470.78	490.63	470.76	484.47
Max Height Lodged (mm)	1480.07	1270.40	1271.92	1200.34	1023.05
Min Height Lodged (mm)	27.48	22.42	8.14	3.52	0.00
Mean Height Threshold (mm)	513.08	470.56	475.76	461.13	463.85

4.4.2 Comparison with Field Measurements and Photogrammetric Measurements

The field measurements were shown in the Table 4-3.

Table 4-3: Field based measurements for plots 1 and 2

Physical Parameters	Plot 1	Plot 2
Mean Height (mm)	840	830
Max Height of all Points (mm)	980	900

The mean height from the plots 1 and 2 were close to 800 mm whereas the mean height from the plots 1 and 2 from the LiDAR sensor varied from 330 mm to 480 mm which were far from the field measurements. One of the reasons behind this was most of the individual plants were lodged and only four to five hand measurements were taken at a single plot.

From Table 4-1 and Table 4-2, the mean height of the alfalfa after applying the algorithms to straighten the lodged plants and to remove the ground related points were far greater than the normal, but still there was a lot of error remains between field and LiDAR measurements which was mostly due to very coarse sampling of alfalfa crop with very few field measurements.

The photogrammetric measurements are shown in the Table 4-4.

Table 4-4: Alfalfa physical parameters obtained from the photogrammetry

Physical Parameters	
Mean Height (mm)	585.30
Max Height of all Points (mm)	998.17
Min Height of all Points (mm)	259.74
Max - Min (mm)	738.43
Total Samples	26756
More than Half Height (%)	37.82
Thinness (%)	0

The mean height obtained from photogrammetry was 585.30 mm while the mean height from LiDAR was about 450 mm – a difference of 130 mm. One of the reasons for this deviation was the minimum height, which was 259 mm when using photogrammetry and nearly 0 mm when using LiDAR. This led to the threshold, and after it was applied the mean height went to about 470 mm. There was still about 100 mm difference between both methods. Figure 4-4 shows the histograms of heights from LiDAR at a steady-state velocity of 1.0 m/s after applying threshold and photogrammetry.

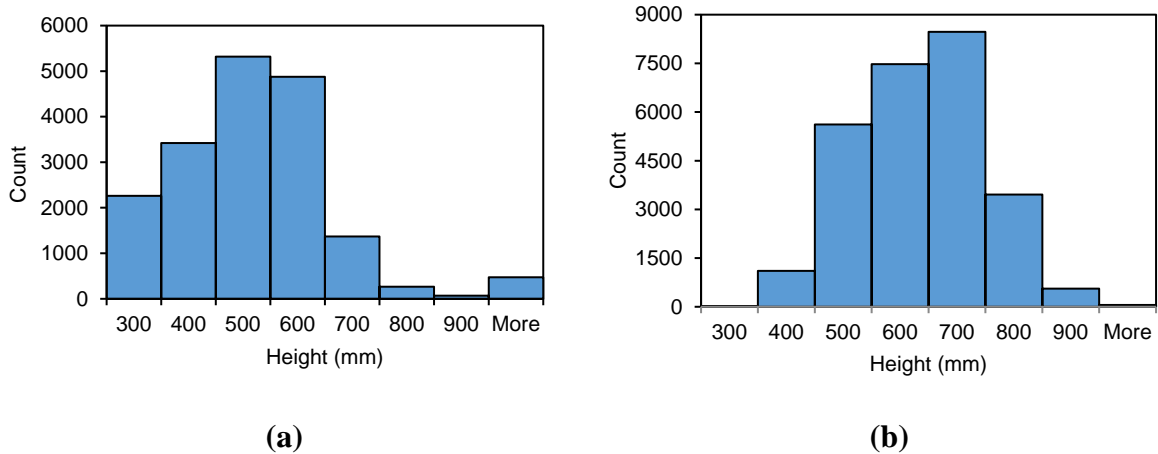


Figure 4-4: Histograms of heights of alfalfa from (a) LiDAR at a steady-state velocity of 1.0 m/s and (b) Photogrammetry

From both the histograms there is clear offset of about 100 mm between the two methods. From a single plot, the conclusion was really difficult to reach without any replications from photogrammetric data.

4.5 Conclusions

This chapter introduces the concept of modeling the physical structure of alfalfa using LiDAR and comparing the results to similar data collected using photogrammetry. The chapter also describes the software used to process the raw data from the LiDAR to obtain the physical properties of the alfalfa crop. In this study, point cloud processing was applied to extract the height, volume, and different other statistical parameters. Similar properties were also obtained through photogrammetry and the field measurements.

While the limited data collected show some indication that similar results were possible, far more sets of 3D scans are needed to make any conclusions. A single plot will not be sufficient to compare and do statistical differences between different measurements. There was an offset between LiDAR and photogrammetry measurements which was difficult to

explain without replications. This study also concluded that multiple sampling velocities may not be necessary because there was little difference in sampled parameters except between the extreme velocities. Scanning the alfalfa at just one velocity with 3 replications should be sufficient for future work.

CHAPTER 5: SUMMARY AND CONCLUSIONS

5.1 Summary of Work

In the first chapter, LARS and LiDAR were introduced. In the second chapter, a linear motion test fixture was developed to control the dynamics of the 3D LiDAR and constrain it to a one-dimensional path parallel to the ground to remove any uncertainties in the velocity and position of the LiDAR, which may induce inaccuracies in the projection of its measurements. The LiDAR was attached to a carriage assembly that translates back and forth via force supplied by the timing belt. The linear motion test fixture was validated for displacement, steady state velocity and frame movement using a total station.

In the third chapter, software was developed to control the linear motion system and record the LiDAR data in the background parsing it in a CSV file. A test target containing metal plates of five different heights was built for validating the LiDAR. The LiDAR was translated across in five different velocity profiles. The raw data was processed to obtain different statistical parameters for each target plate for different velocity profiles. Generalized linear mixed models were fitted with the error and standard deviation as the response and velocity, actual height, and their interaction as the fixed effects to determine if there were significant differences in error and standard deviation for different velocities and heights.

In the fourth chapter, the concept of modeling the alfalfa crop using the linear motion test fixture was introduced. Two plots of alfalfa crop were scanned at five different velocities with three replications of each velocity. Post processing of the raw data and point cloud processing was done to extract important statistical parameters and compare it with the alfalfa data obtained through photogrammetry and field measurements.

5.2 Conclusions

In the second chapter, the results concluded the effectiveness of the test fixture for eliminating most of the uncertainty present in a traditional test fixture used for remote sensing.

In the third chapter, Results showed that velocity is a significant factor affecting the accuracy and standard deviation. The error estimate was higher for faster velocities compared to slower velocities. In contrast, the standard deviation of estimated height was lower for faster velocities although by a smaller margin.

In the fourth chapter, the comparison between LiDAR and other methods concluded that there should be more tests and plots to be modeled to do any statistical analysis to observe meaningful differences.

5.3 Future Work

Future work consists of testing more plots with the LiDAR to model the physical structure of alfalfa. Instead of going to the field, it may be more efficient to plant alfalfa in trays in the lab to remove external factors not controlled in this study. The other main concern is with stalk lodging, which made point cloud data difficult to compare to hand measurements. In the future a more refined technique should be developed to lodging and account for its impact on height measurements.

CHAPTER 6: APPENDICIES

Appendix A. Linear Motion System CAD Models

A.1 Linear Frame

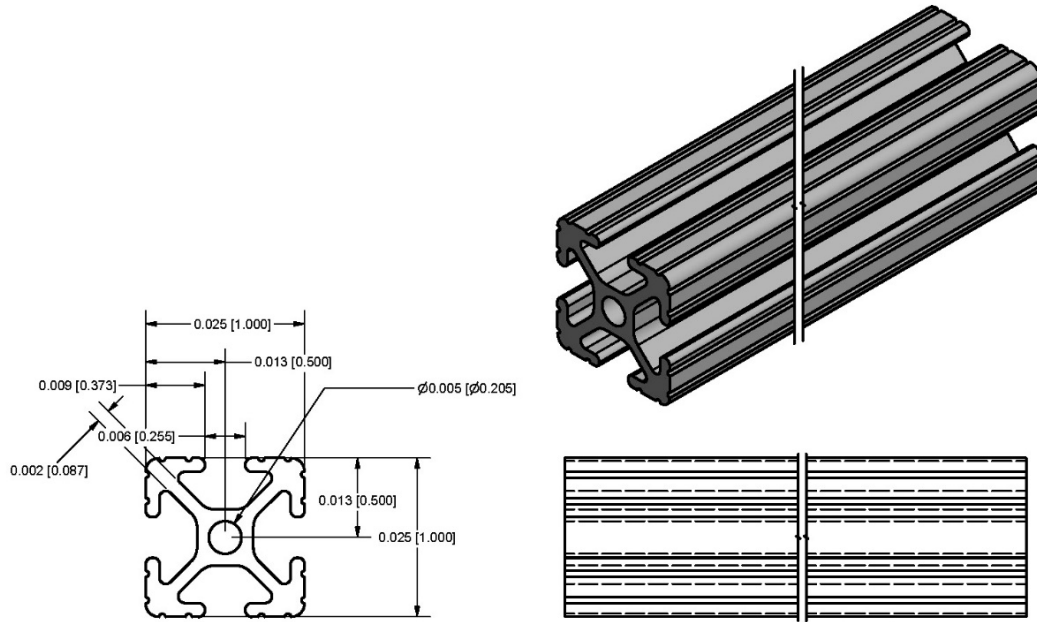


Figure 6-1: Cross sectional area of T-slotted aluminum (1010, 80/20 Inc.).

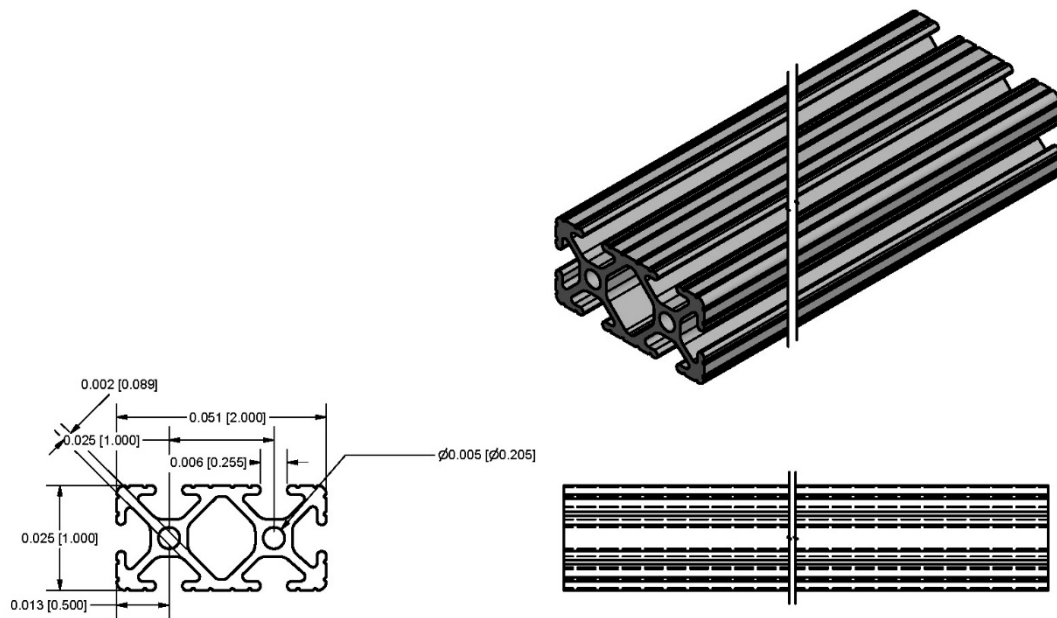


Figure 6-2: Cross sectional area of T-slotted aluminum (1020, 80/20 Inc.).

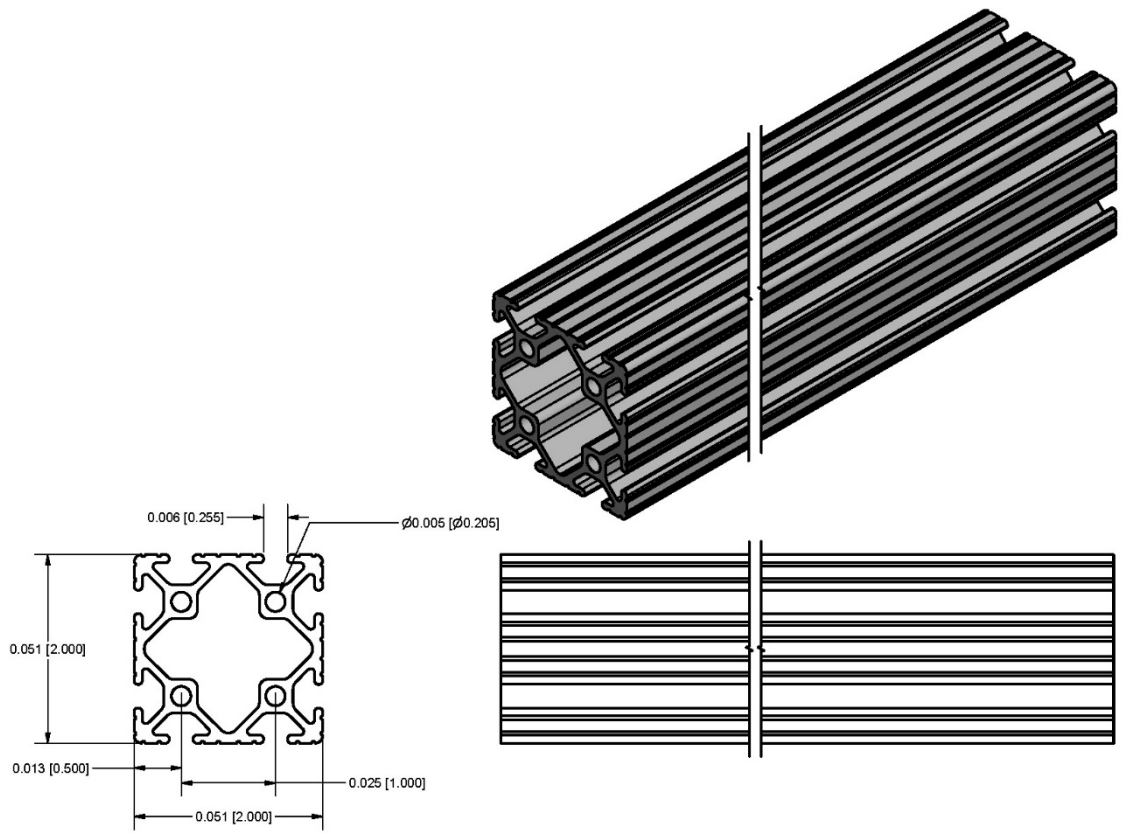


Figure 6-3: Cross sectional area of T-slotted aluminum (2020, 80/20 Inc.).

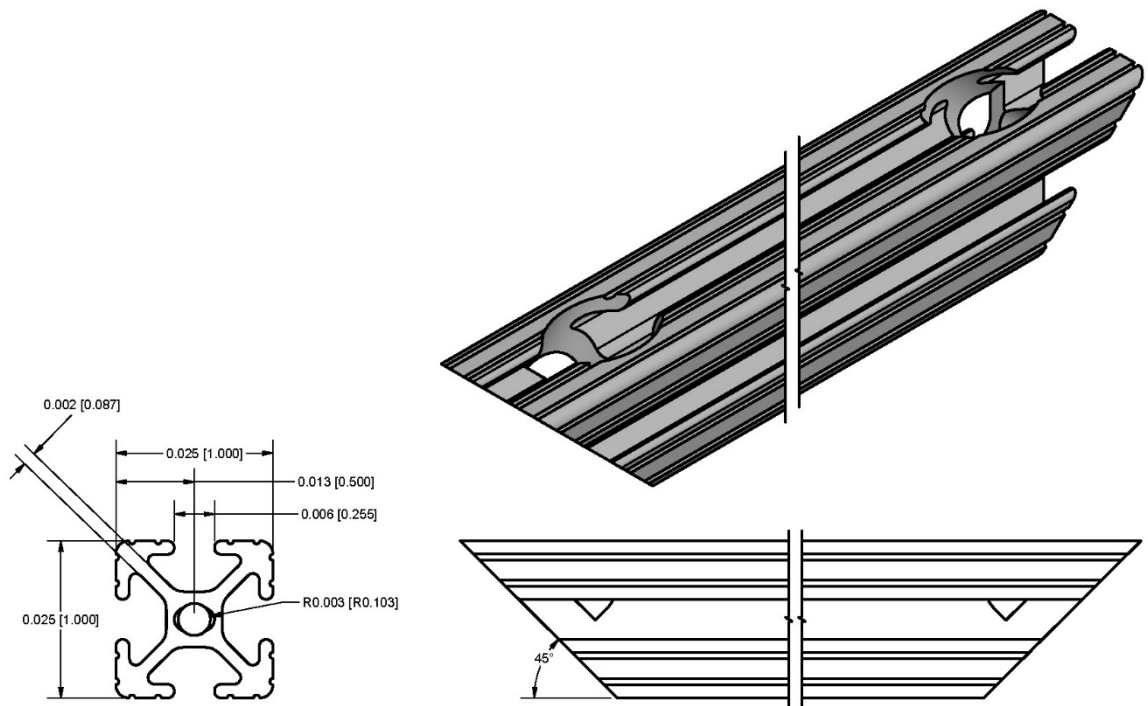


Figure 6-4: Cross sectional area of T-slotted aluminum (b1010, 80/20 Inc.).

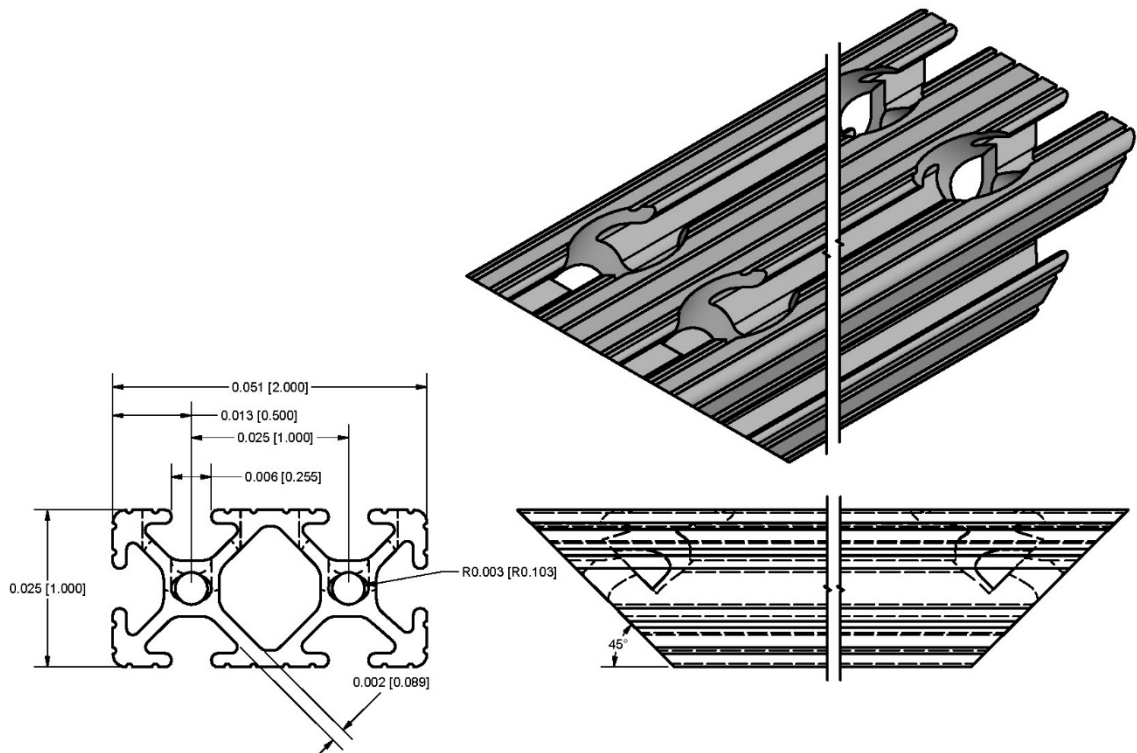


Figure 6-5: Cross sectional area of T-slotted aluminum (b1020, 80/20 Inc.).

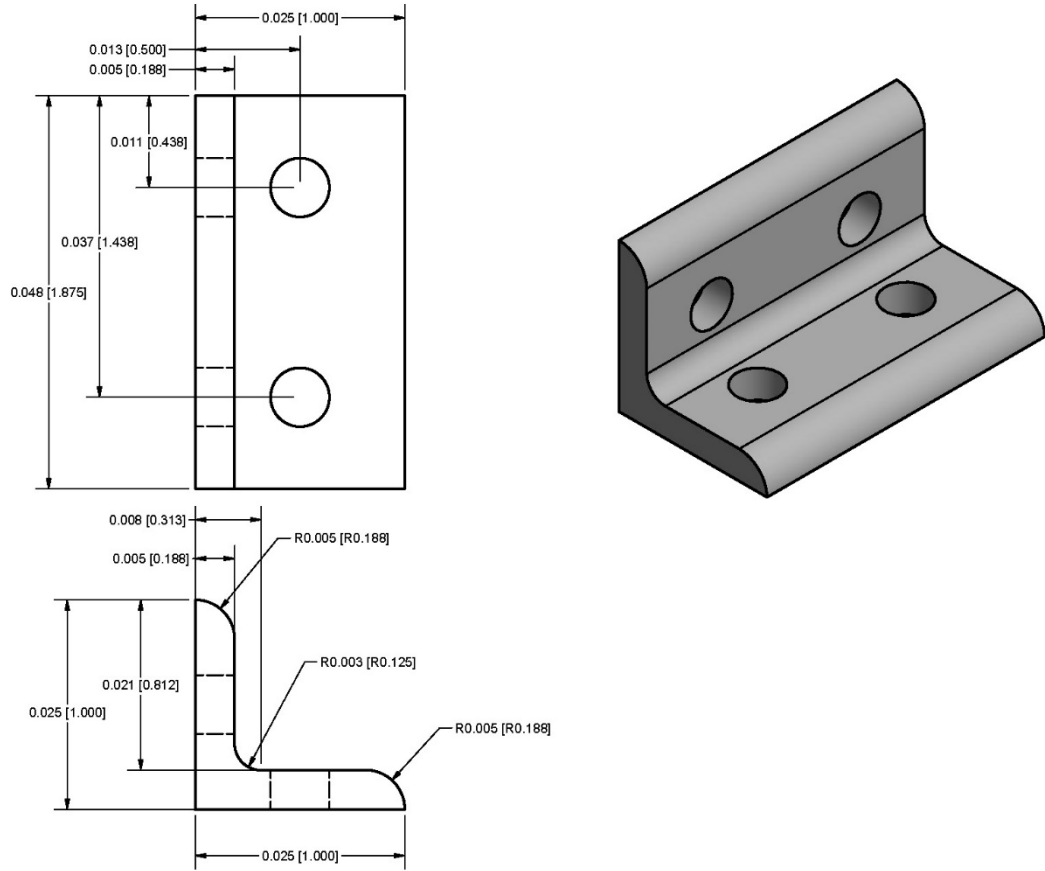


Figure 6-6: A 2 x 2 bent bracket used for connecting two T-slotted aluminum joints.

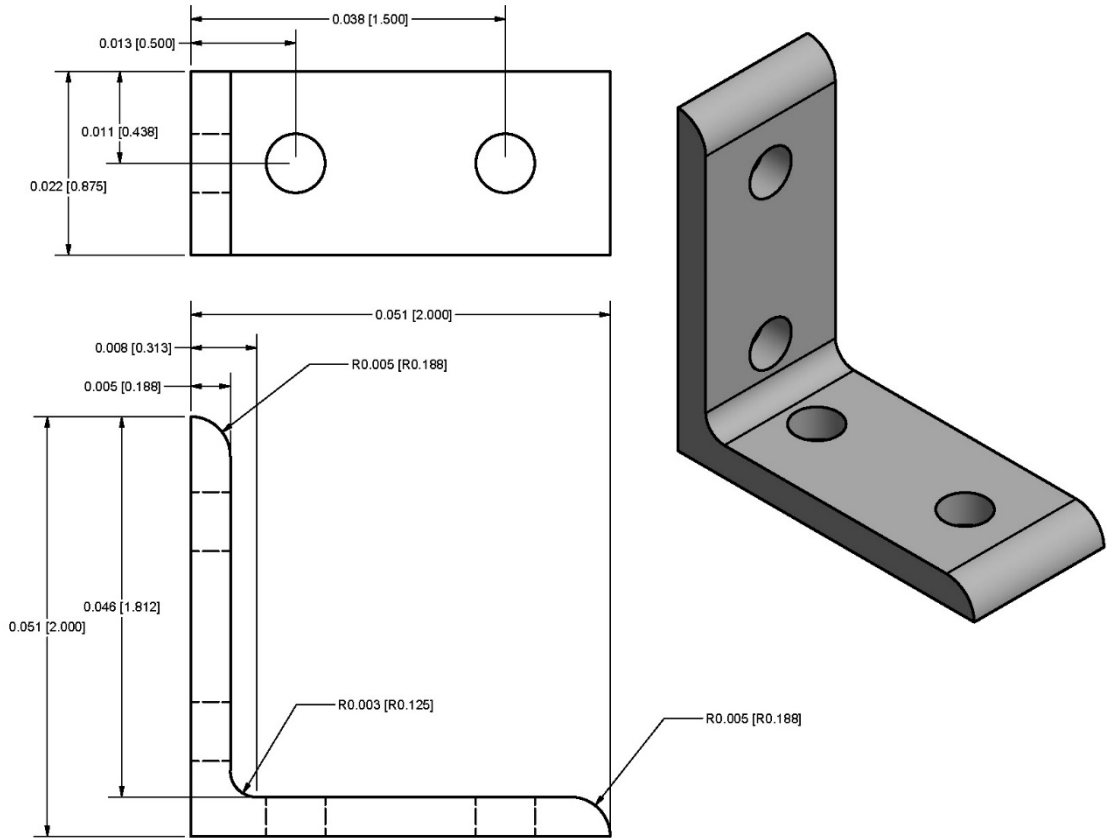


Figure 6-7: A 1 x 1 bent bracket used for connecting two T-slotted aluminum joints.

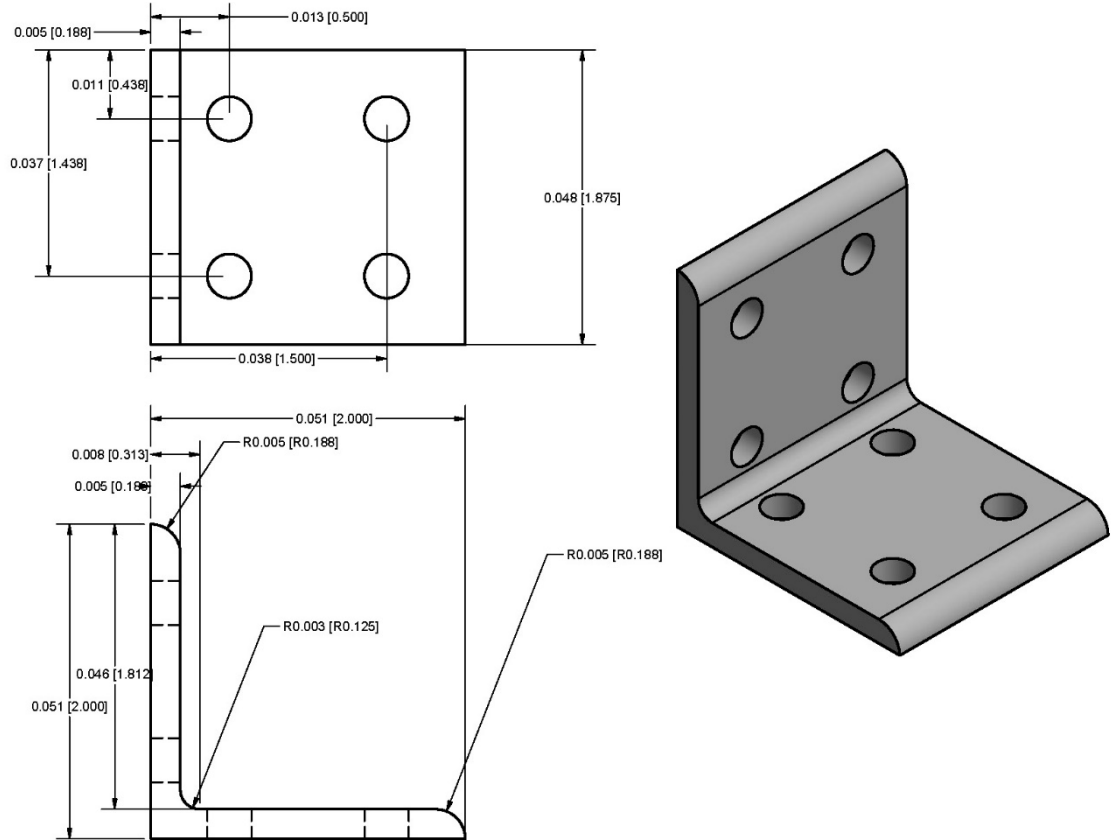


Figure 6-8: A 4 x 4 bent bracket used for connecting two T-slotted aluminum joints.

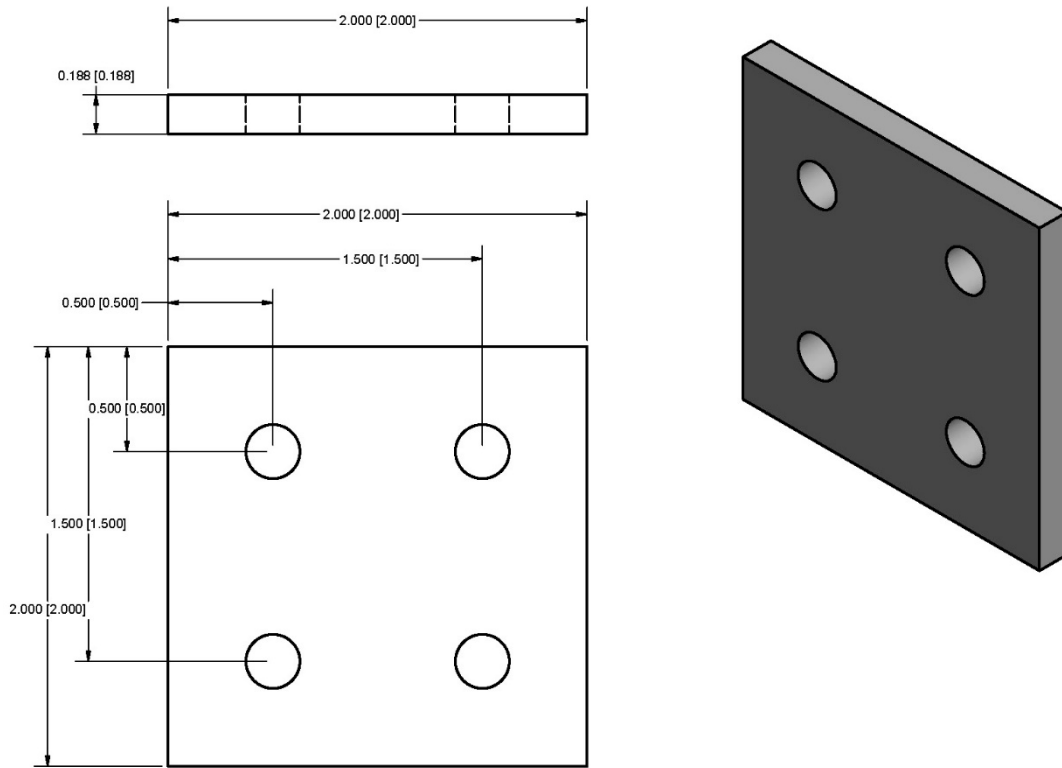


Figure 6-9: A 2 x 2 flat bracket used for connecting two T-slotted aluminum joints.

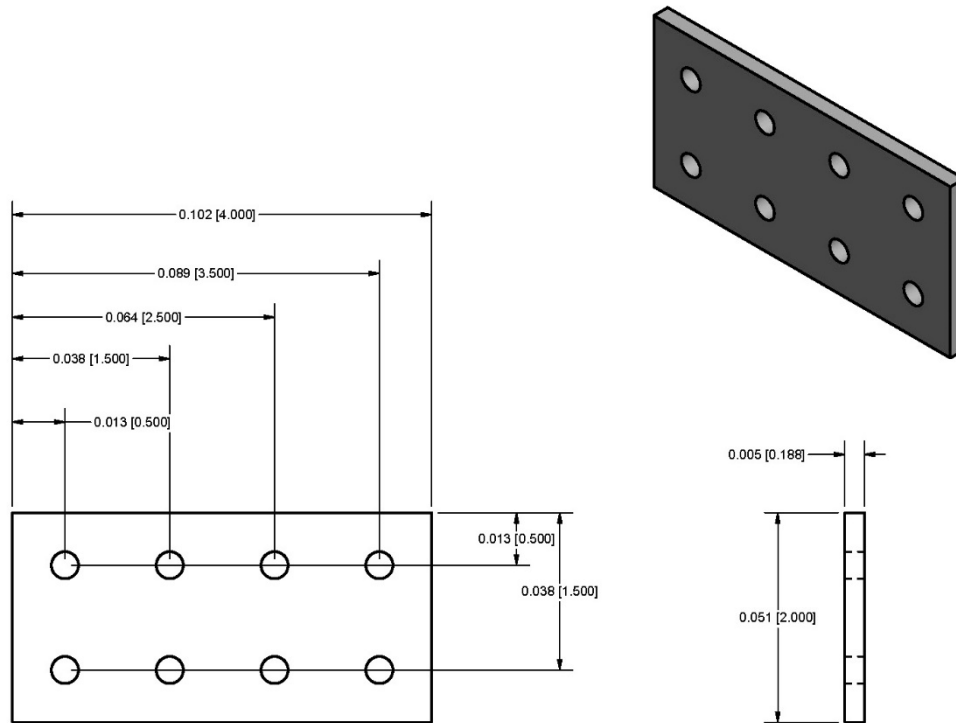


Figure 6-10: A 4 x 2 flat bracket used for connecting two T-slotted aluminum joints.

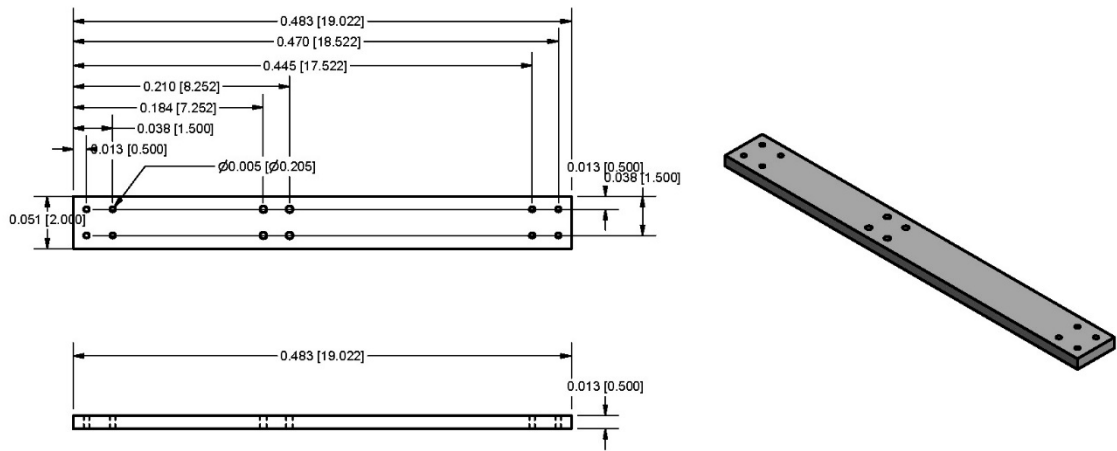


Figure 6-11: Brace on one end of the top assembly to join two linear rails together.

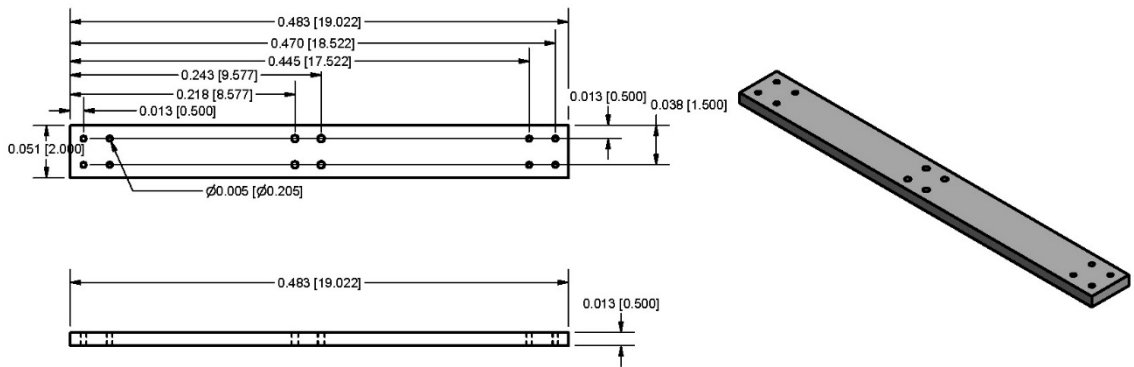


Figure 6-12: Brace on one end of the top assembly to join two linear rails together.

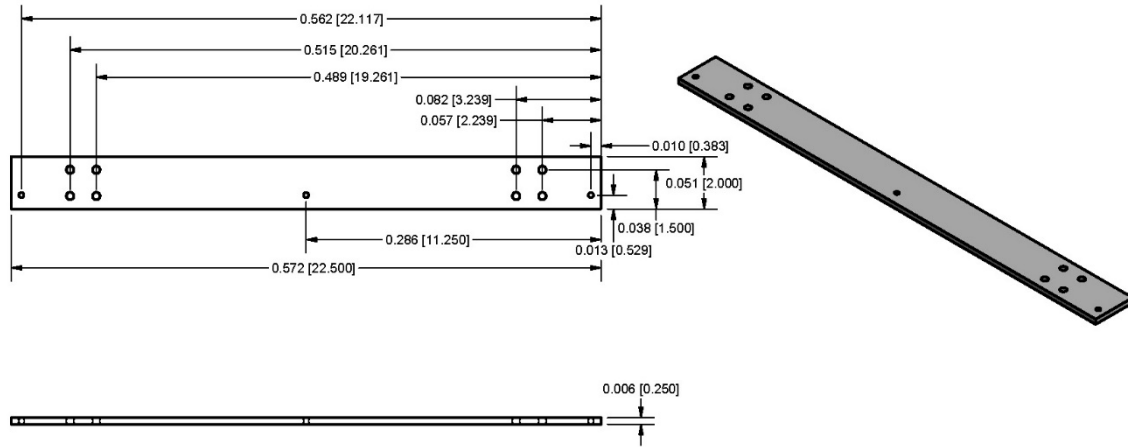


Figure 6-13: Bracket for integrating the motion control plastic box to the top assembly.

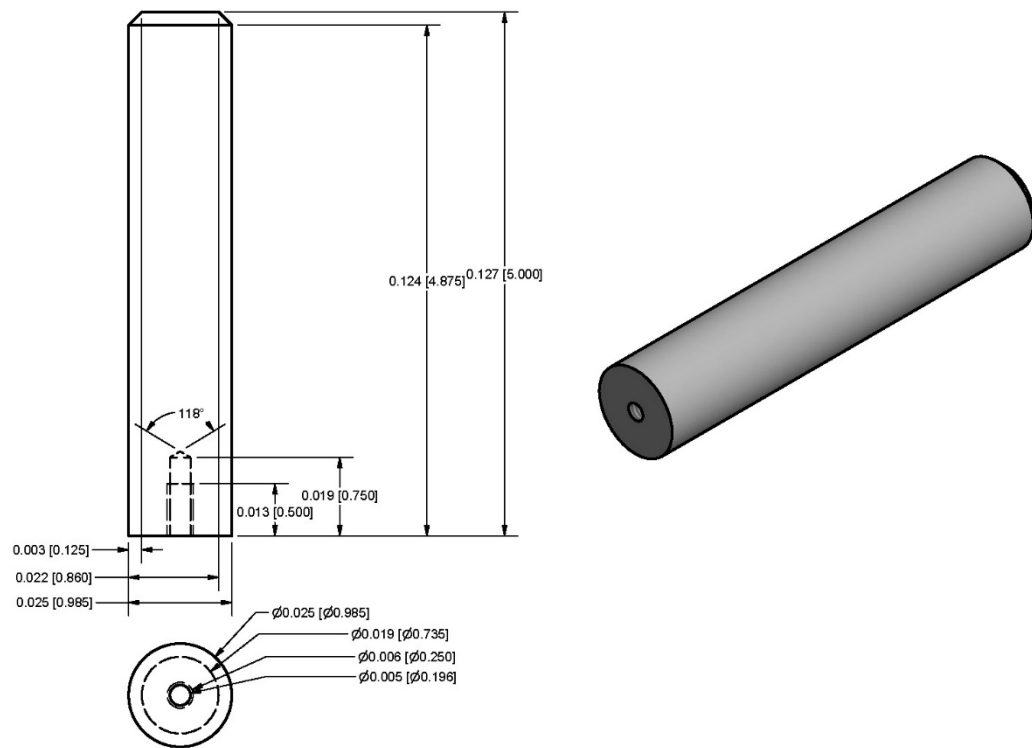


Figure 6-14: Sliding joint that connects the top and frame assembly.

A.2 Carriage Assembly

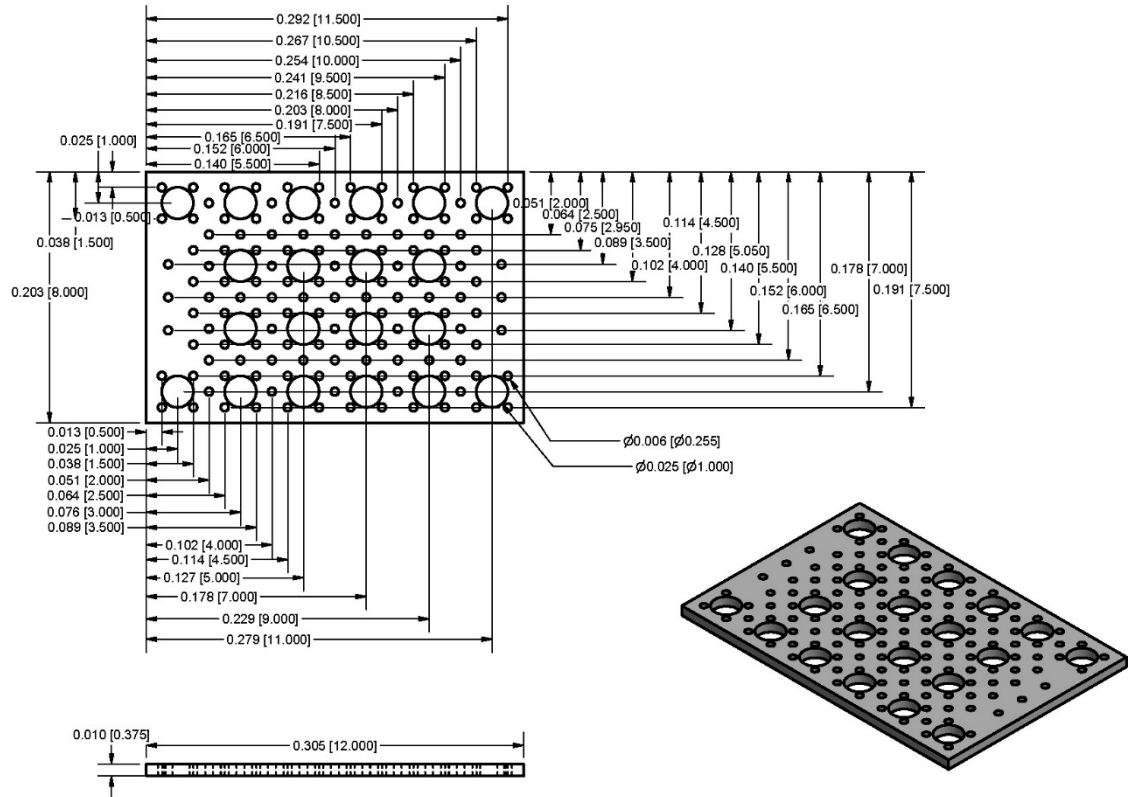


Figure 6-15: Multipurpose mount to integrate different sensors and instruments.

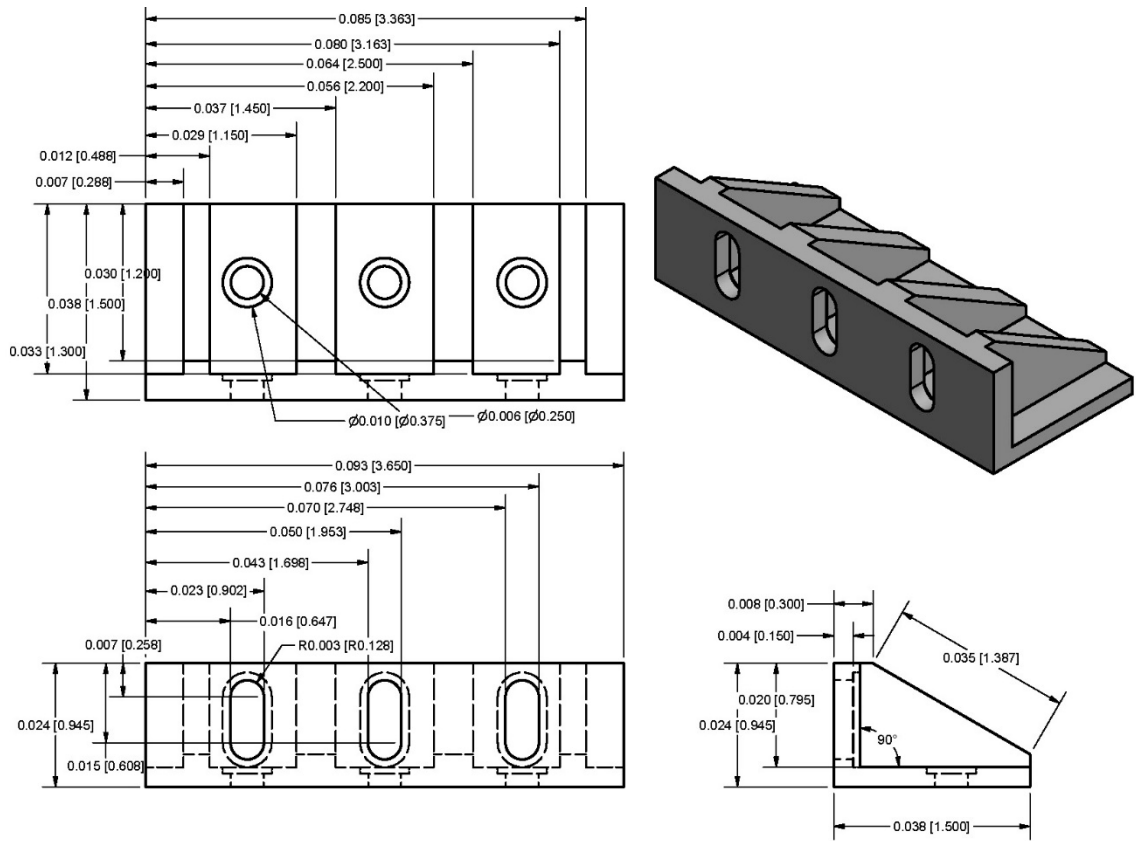


Figure 6-16: Base mount to secure multipurpose mount to the carriage rail.

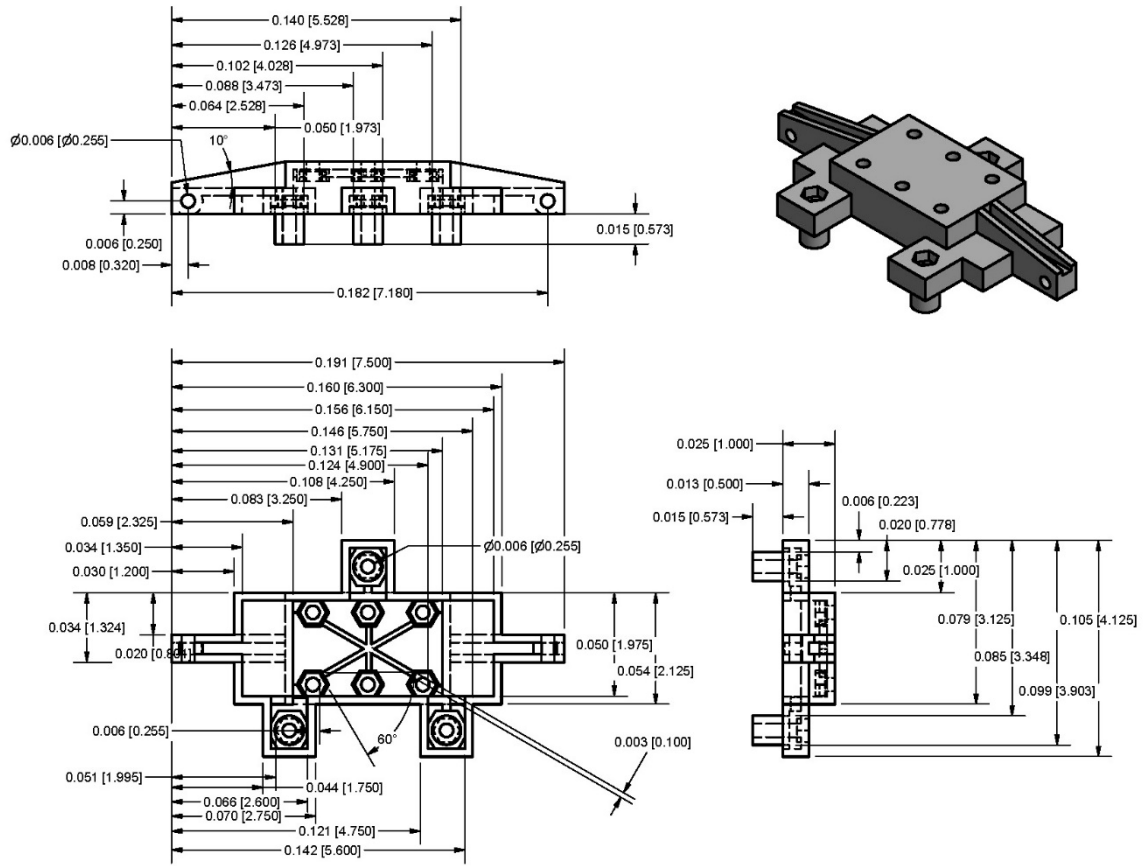


Figure 6-17: Carriage rail to translate back and forth on the linear rail.

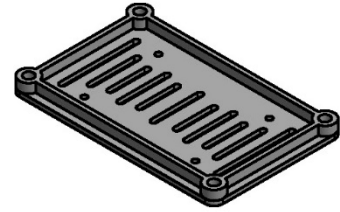
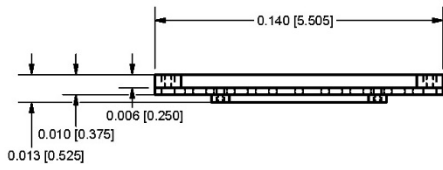
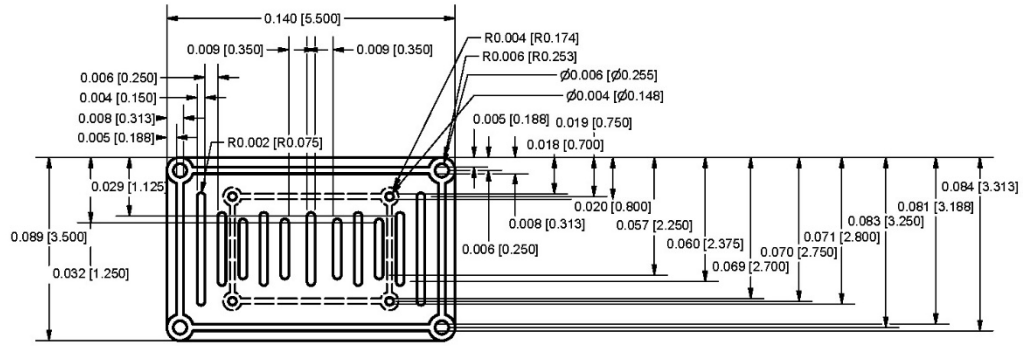


Figure 6-18: Multipurpose mount to secure LiDAR assembly to the multipurpose mount.

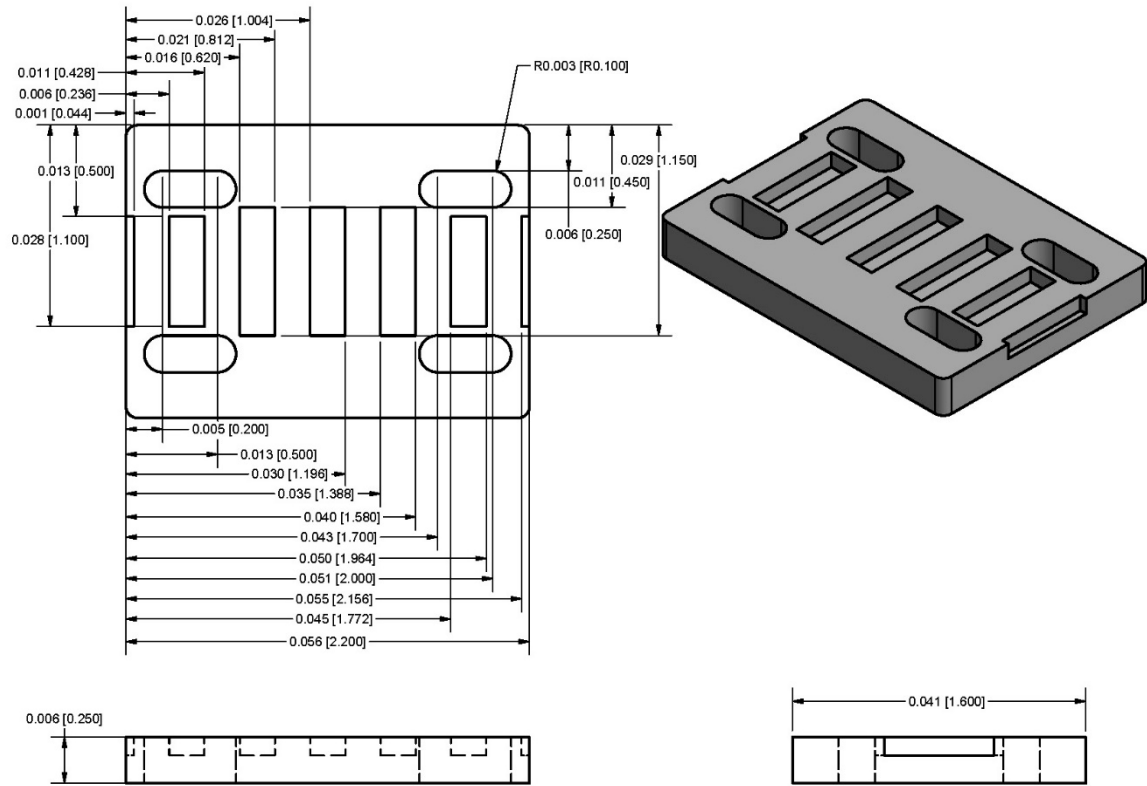


Figure 6-19: Upper part of the belt mount to transfer the force from timing belt to the carriage assembly.

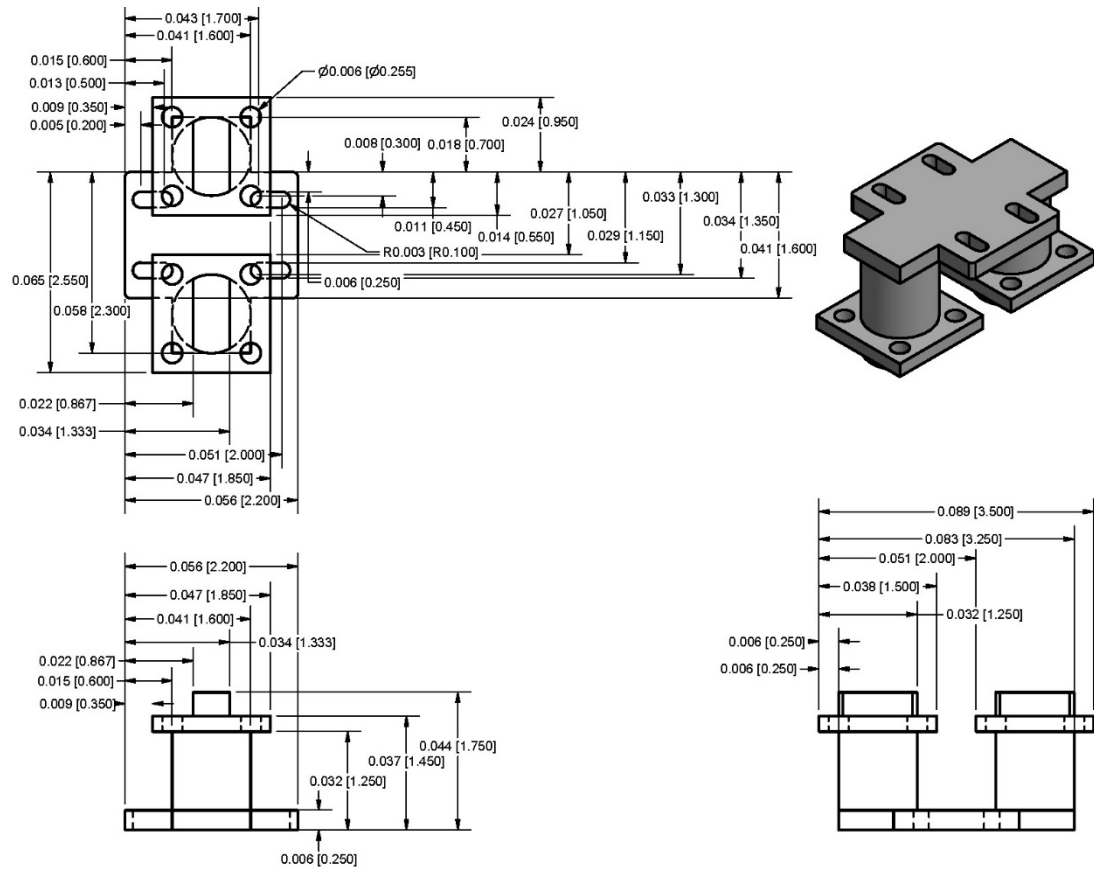


Figure 6-20: Lower part of the belt mount to transfer the force from timing belt to the carriage assembly.

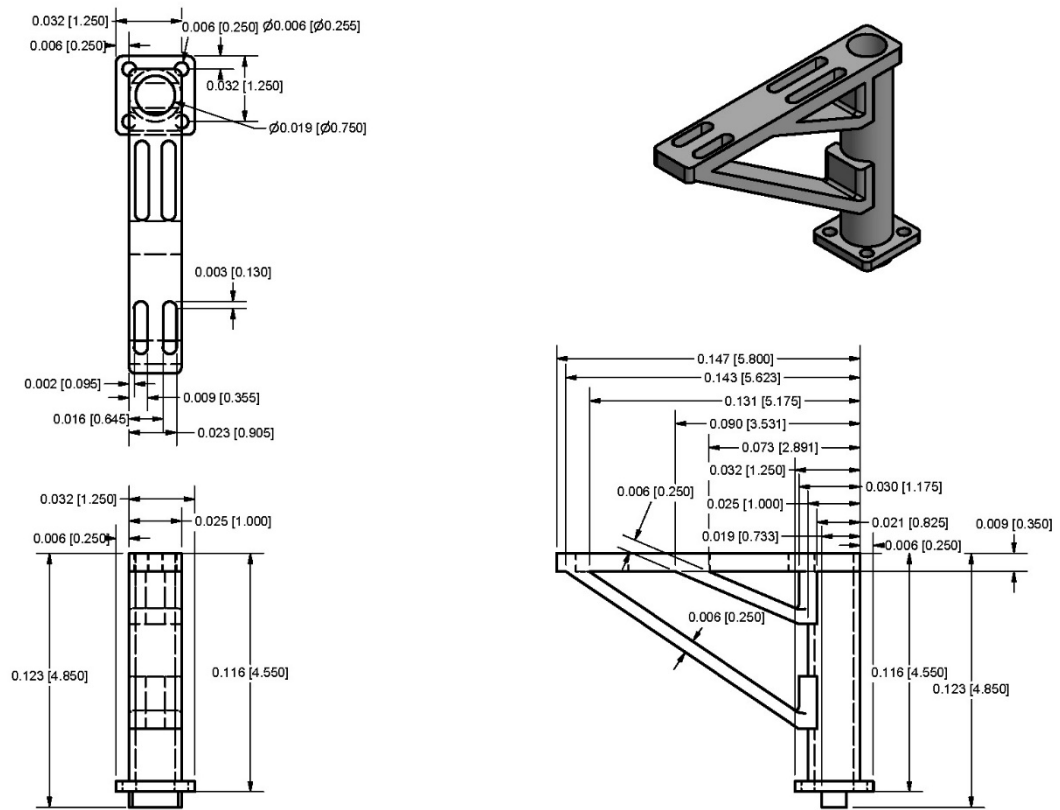


Figure 6-21: Cable carrier mount to secure the flexible carrier to it to translate along with the carriage assembly.

A.2.1 Wheel Assembly

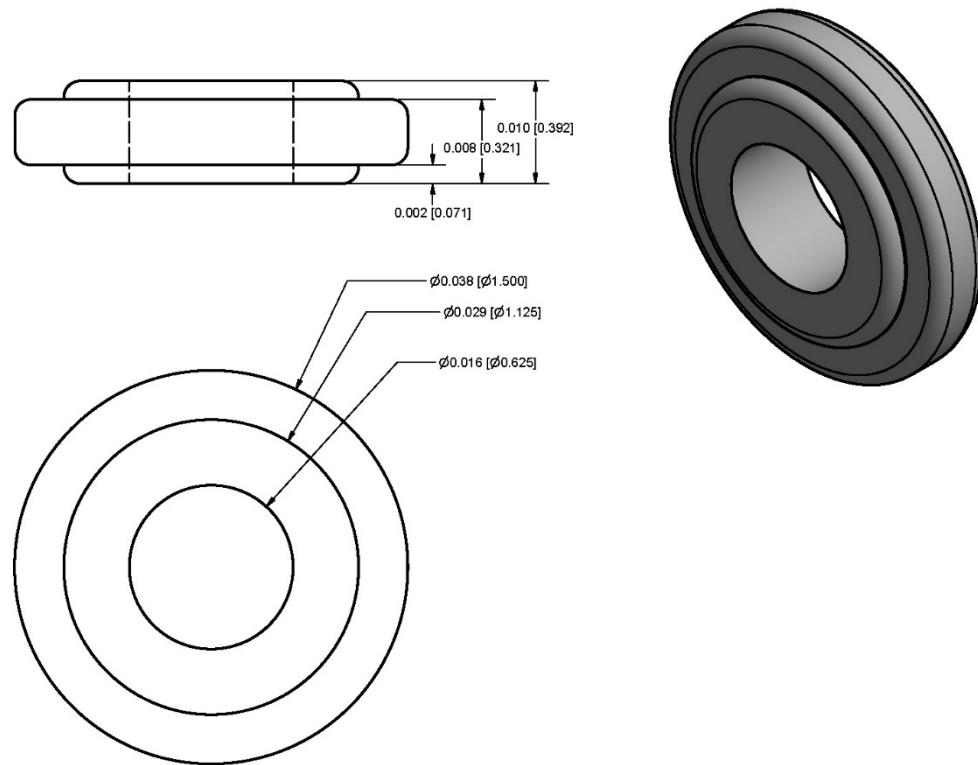


Figure 6-22: Wheel designed to translate along the indentation of the linear rail.

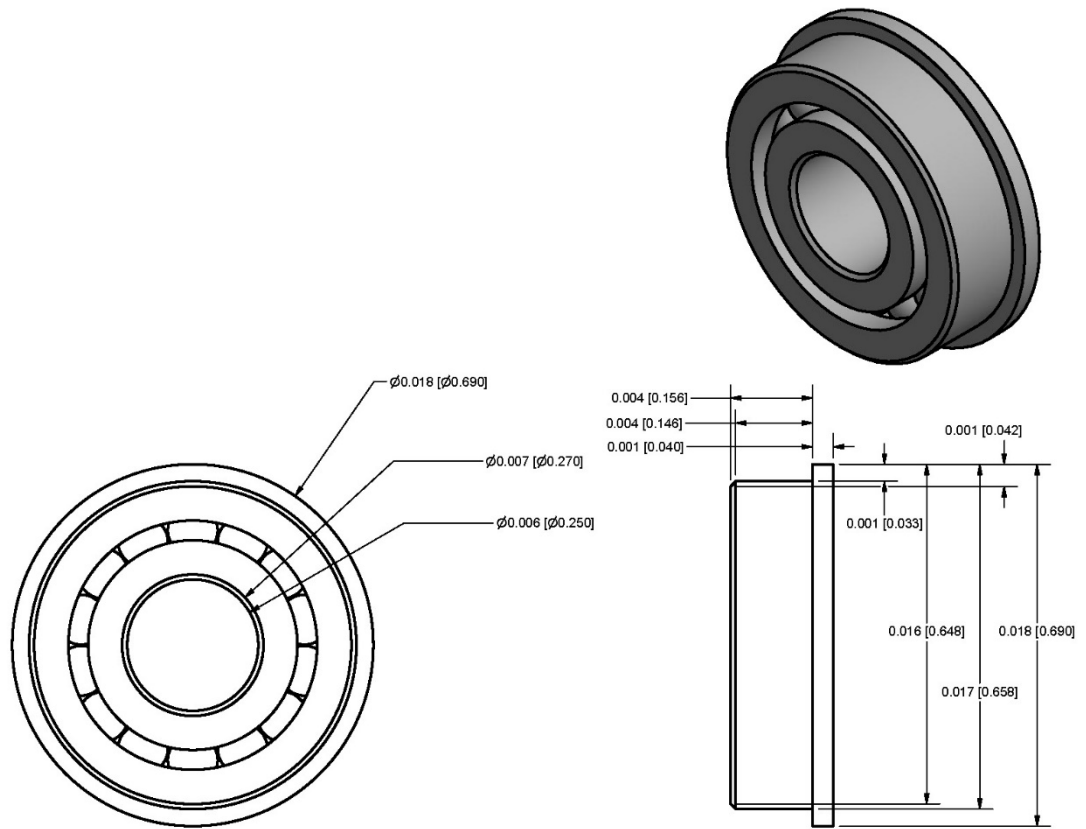


Figure 6-23: Ball bearing on either side of the wheel to ensure uniform distribution of force on the wheel.

A.3 LiDAR Assembly

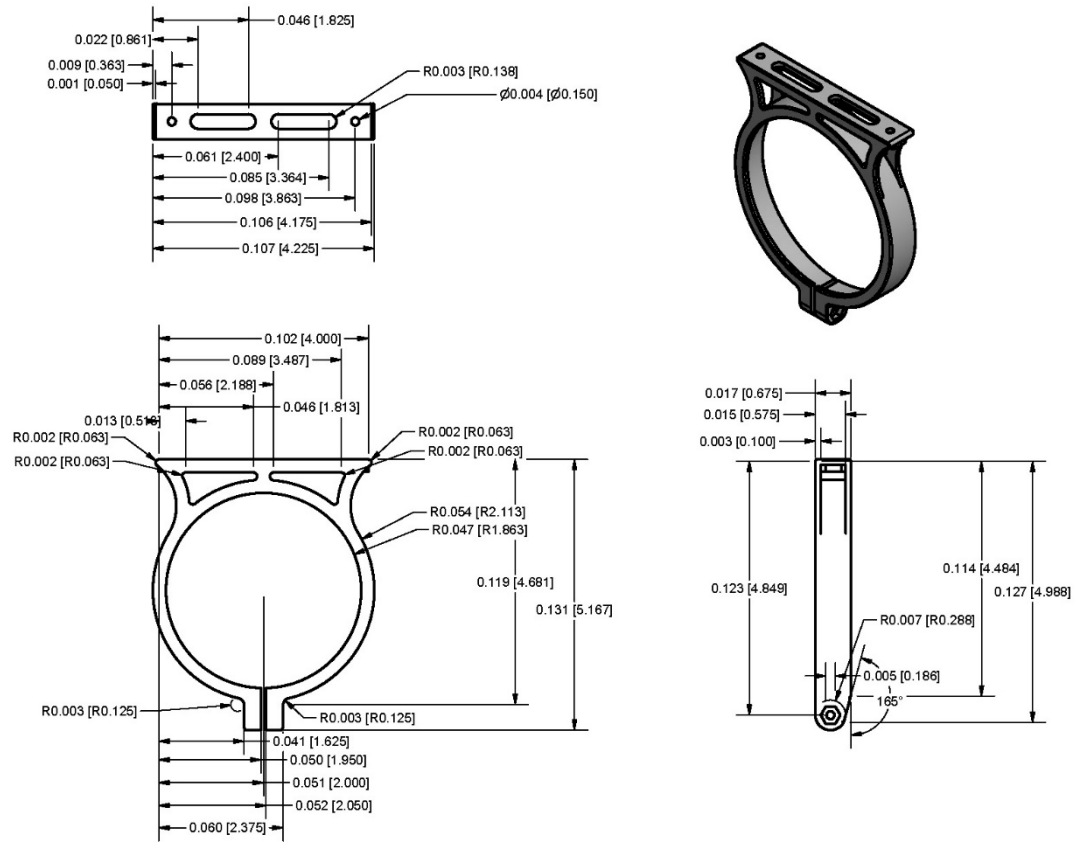


Figure 6-24: Top clamp to secure the LiDAR

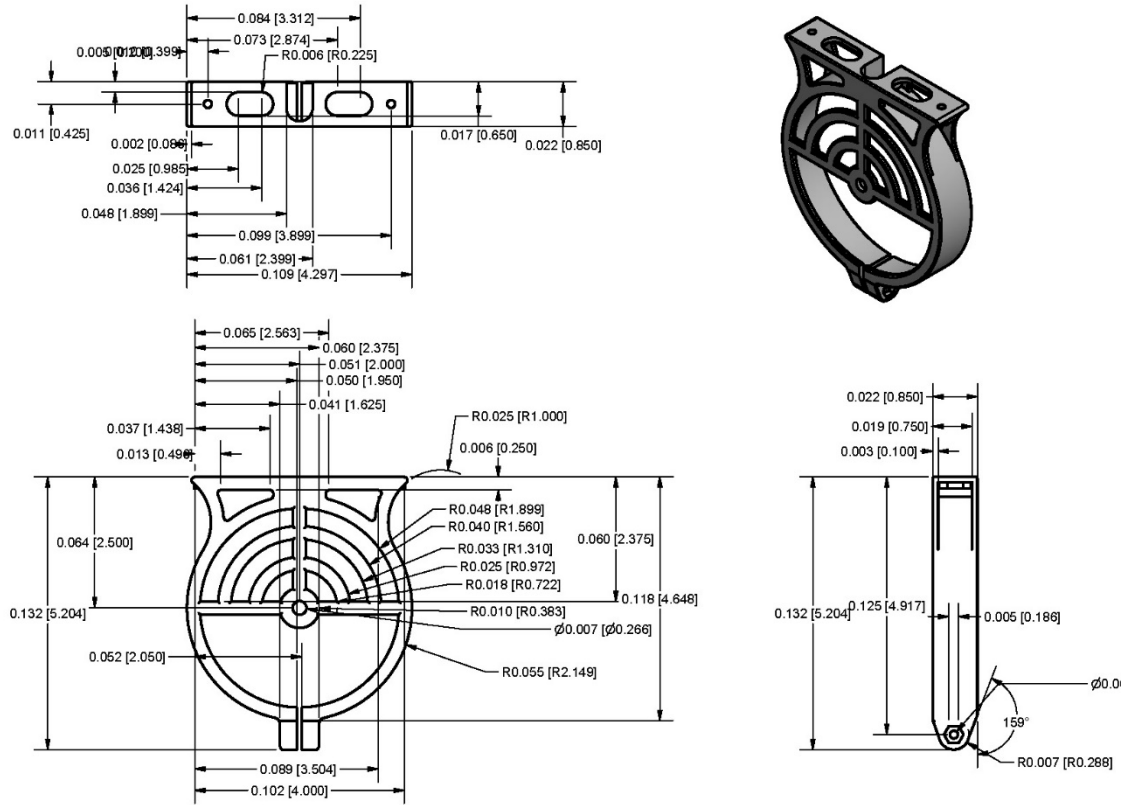


Figure 6-25: Bottom clamp to secure the LiDAR

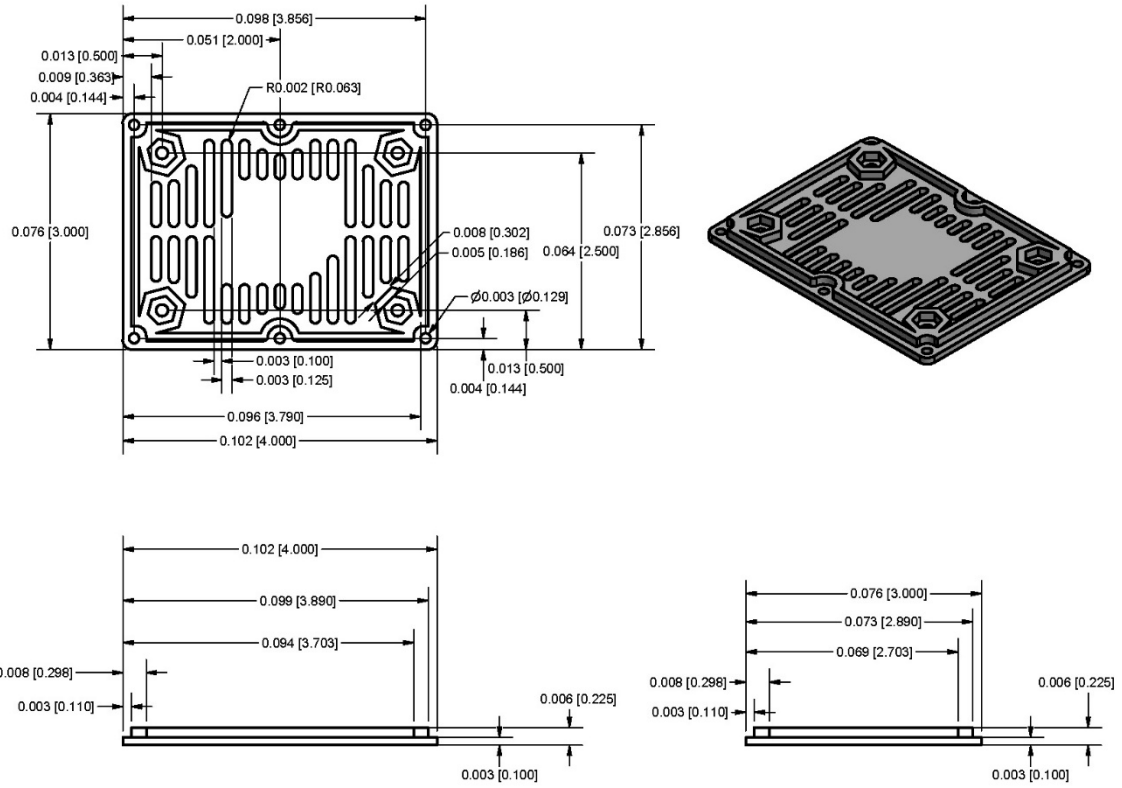


Figure 6-26: Top bracket to enclose the LiDAR assembly

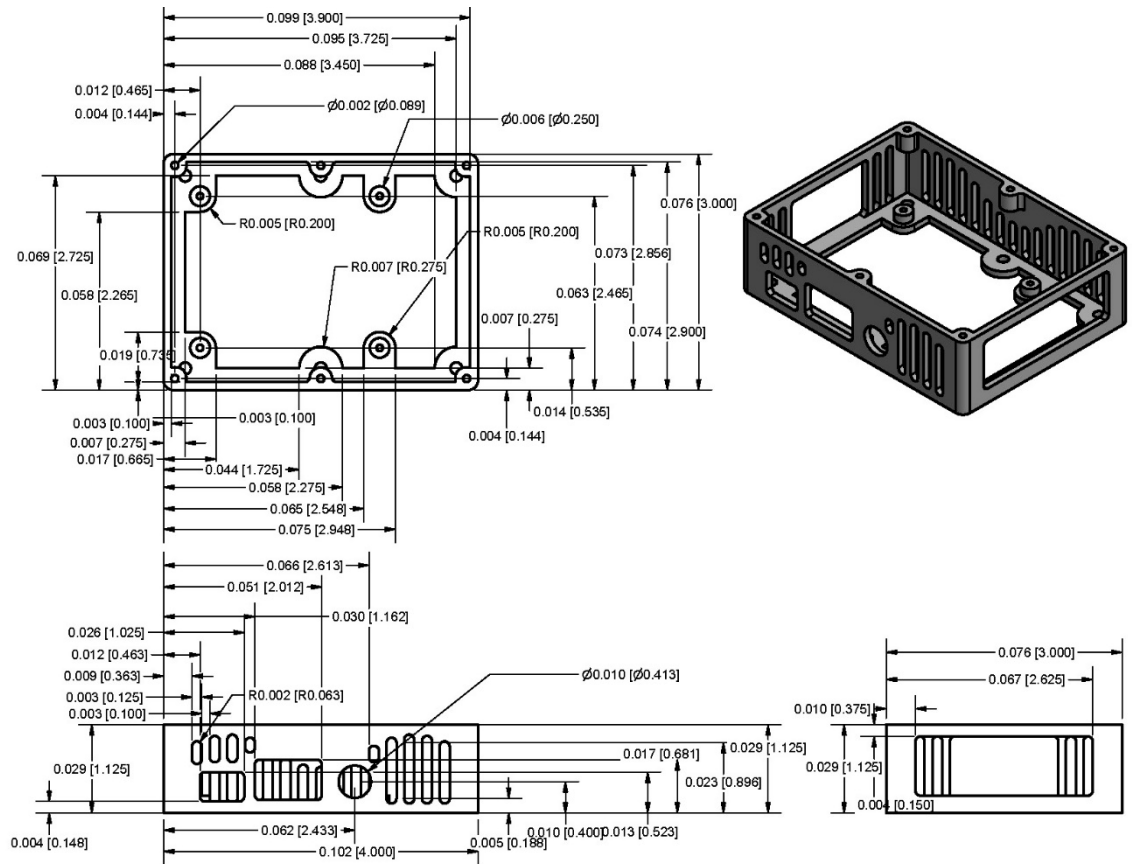


Figure 6-27: Bottom part of the LiDAR assembly to enclose it.

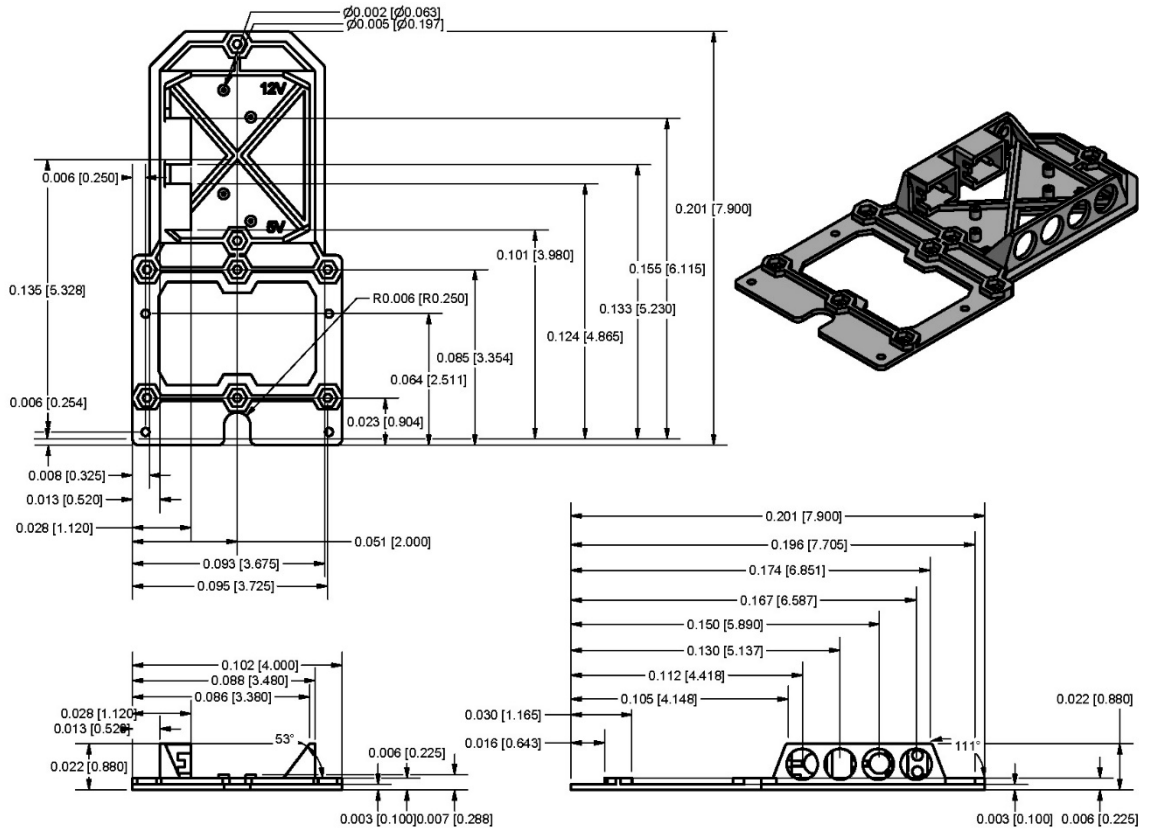


Figure 6-28: Mount for attaching all the instrumentation in LiDAR assembly.

A.4 Motion Control

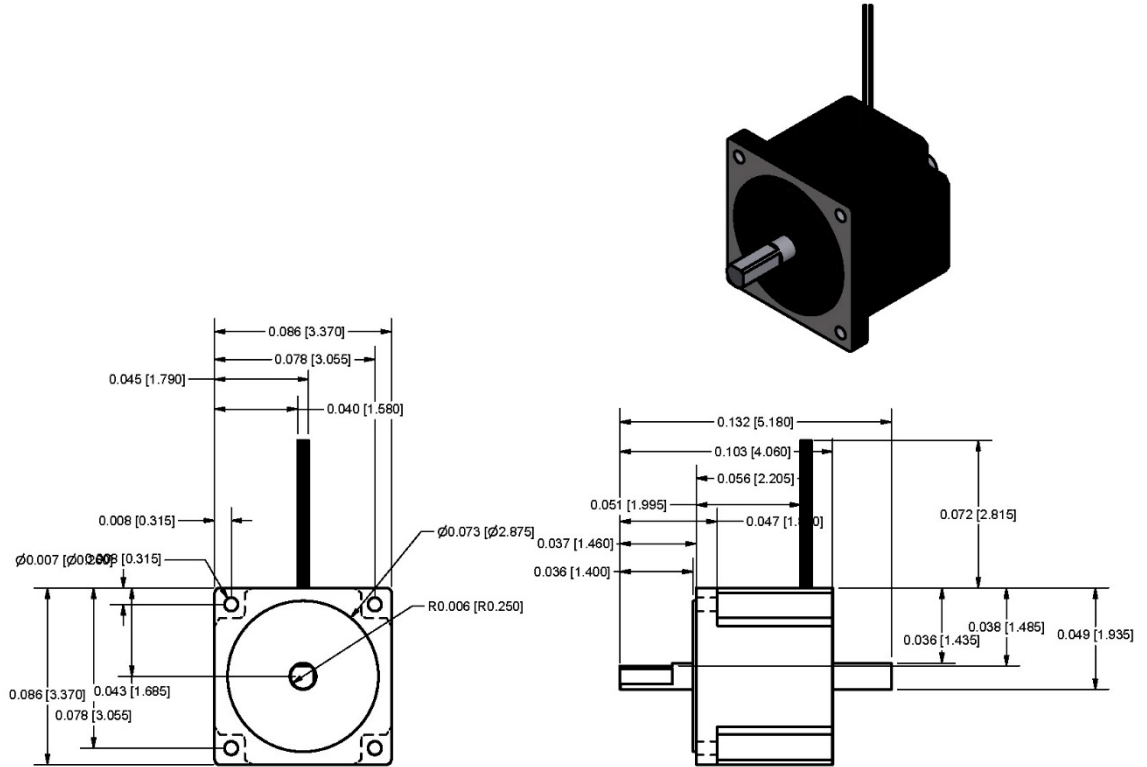


Figure 6-29: Stepper motor (STP-MTRH-34066D, AutomationDirect).

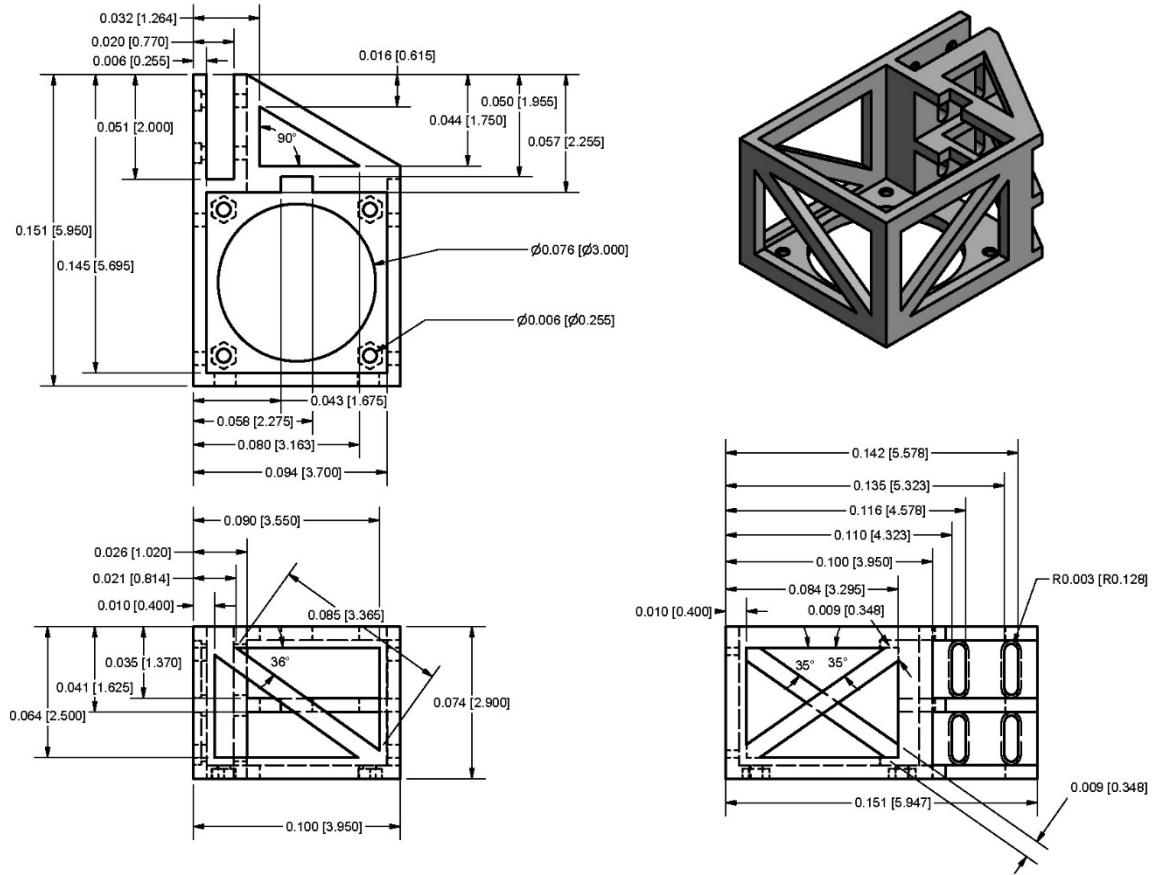


Figure 6-30: Motor bracket for securing the stepper motor to the frame.

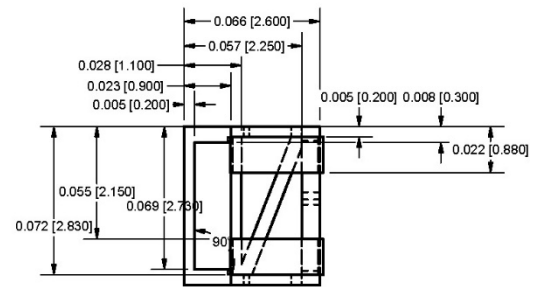
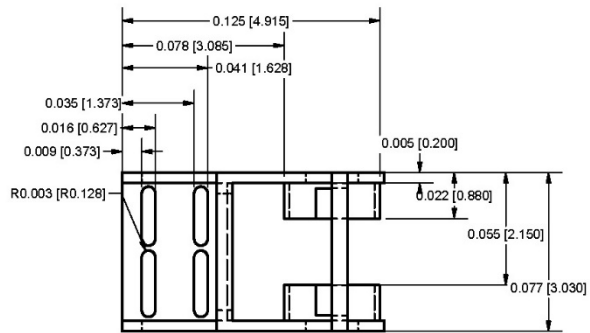
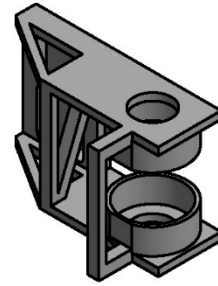
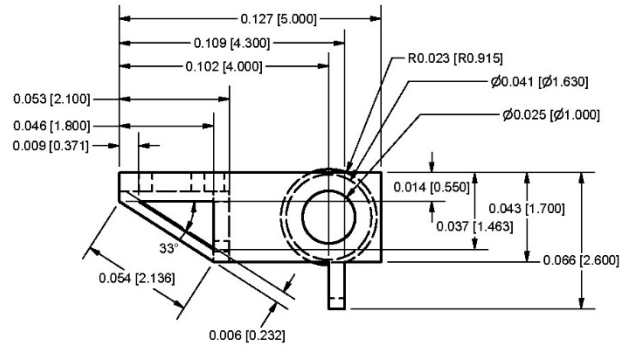


Figure 6-31: Pulley bracket for securing the pulley to the frame.

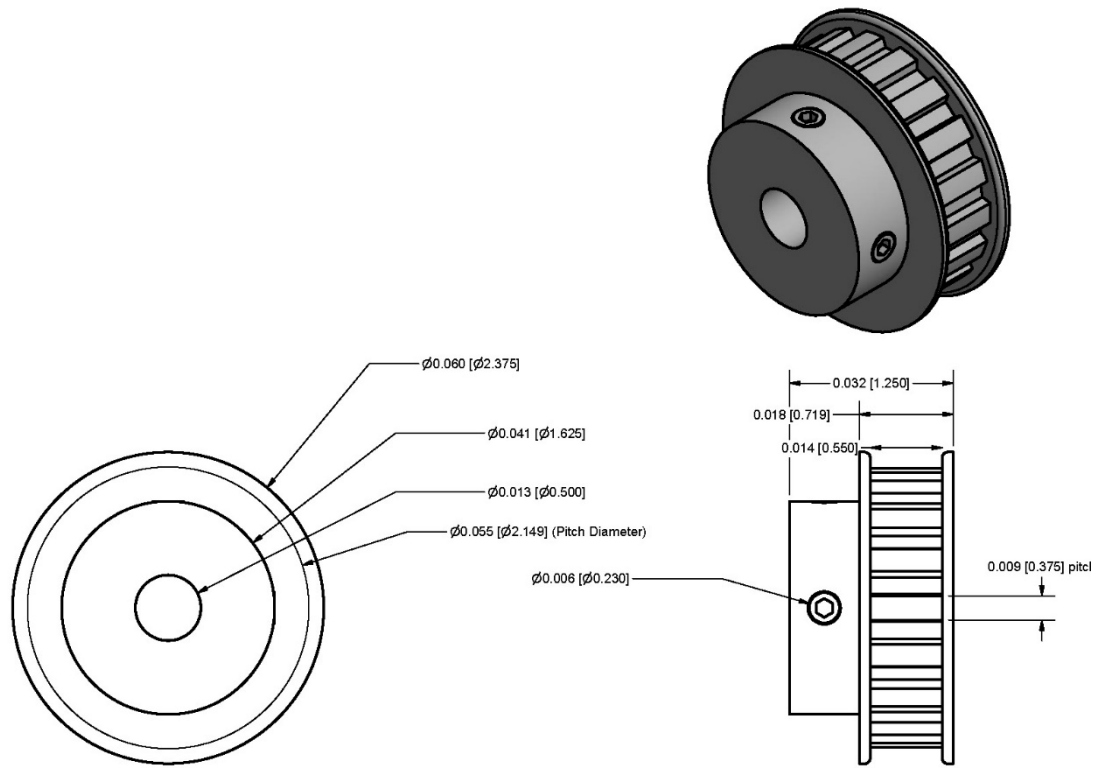


Figure 6-32: Timing belt pulley to transfer the rotary motion from the stepper motor to the linear motion of the system.

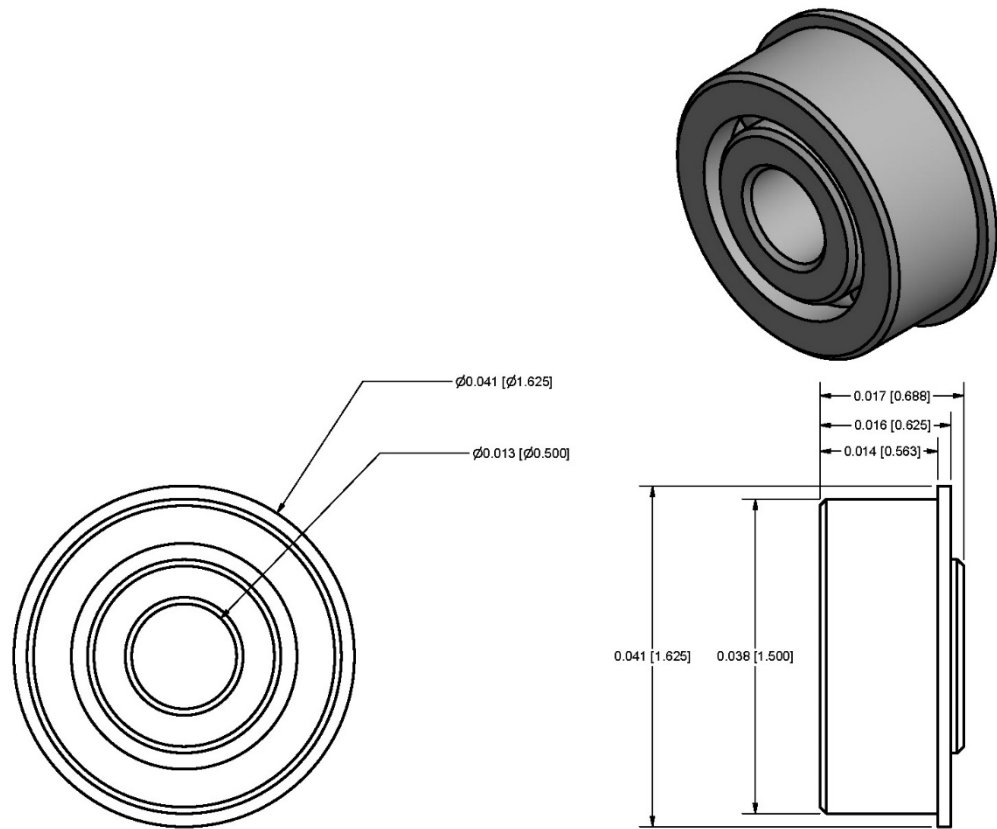


Figure 6-33: Bearings on either side of the pulley to ensure uniform distribution of force on the pulley

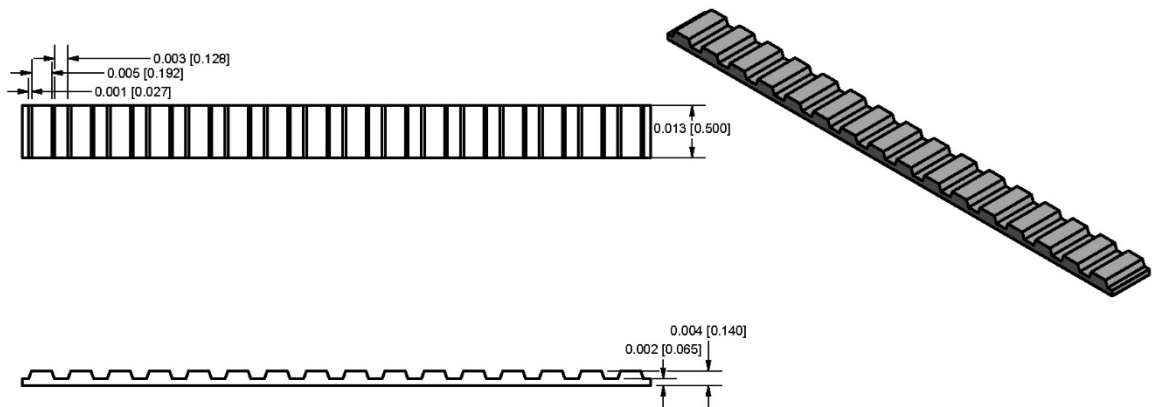


Figure 6-34: Cross sectional view of L-series timing belt.

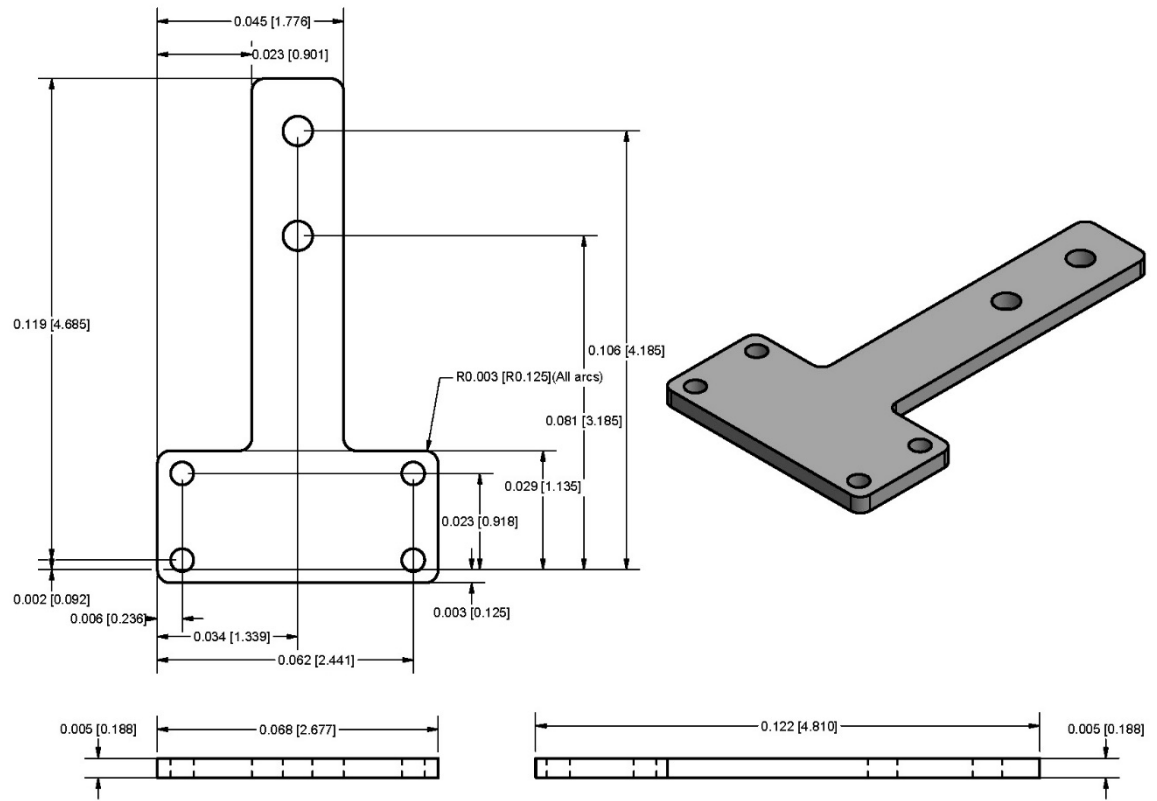


Figure 6-35: Limit switch bracket for securing limit switches to the frame.

A.5 Linear Motion Test Fixture Full Assembly

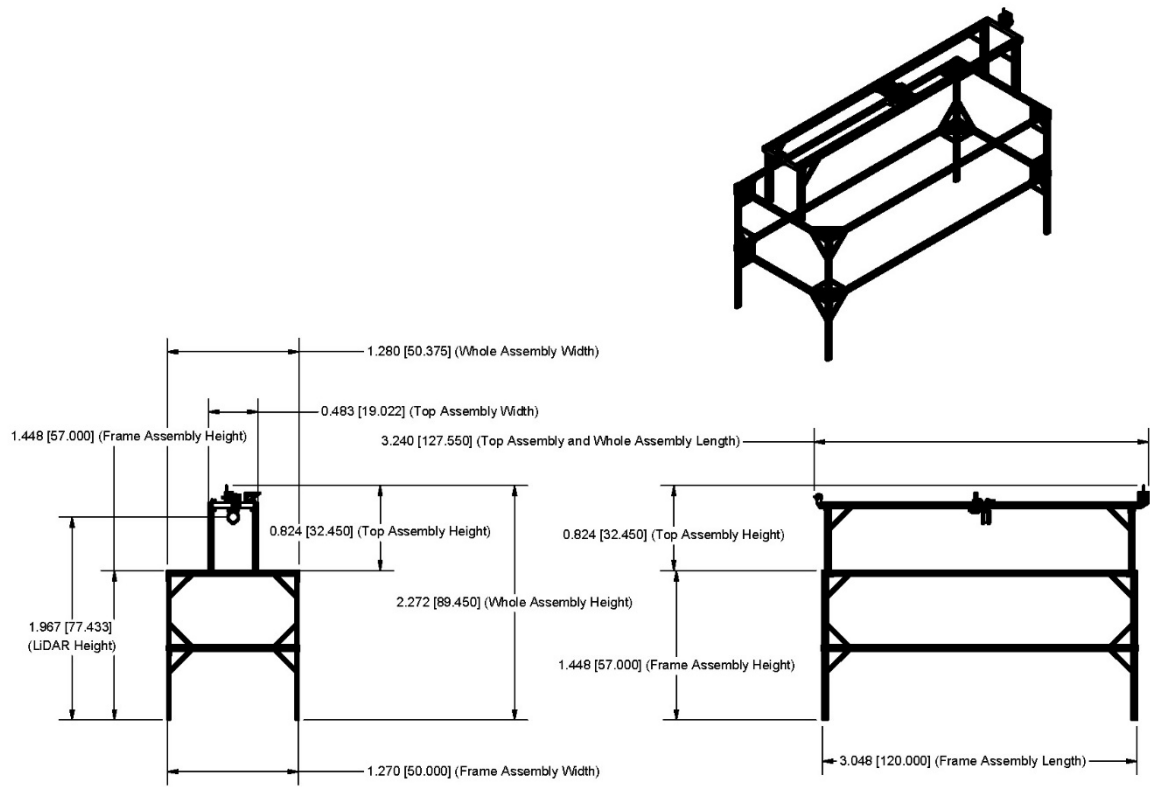


Figure 6-36: Linear motion system with the dimensions of the top and frame assembly satisfying the design constraints.

Appendix B. Linear Motion Test Fixture Wiring Diagram

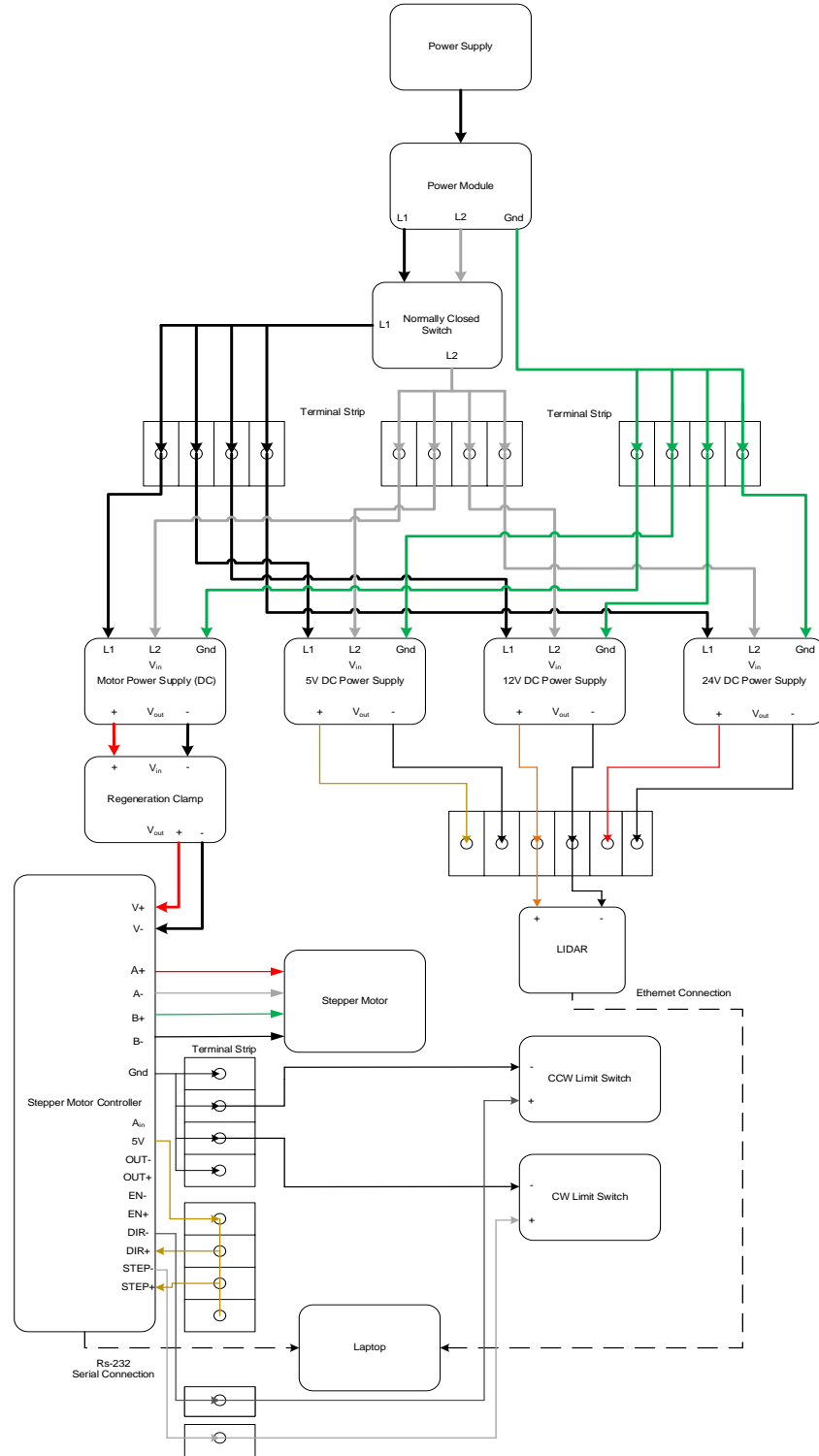


Figure 6-37: Wiring schematic for the motion control hardware.

Appendix C. Linear Motion Control Software and LiDAR Data Collection Software

```
1. #!/usr/bin/env python
2. # -*- coding: utf-8 -*-
3.
4. """ lidar_final.py: Used to control a stepper motor and log the data from VLP-
5. 16 LiDAR synchronously
6. Description:
7.     1) Controls the stepper motor using a serial port by sending the req
8.     uired messages
9.     2) Logs the data from VLP-
10.    16 LiDAR by connecting to a UDP port (HOST = "192.168.1.77" #host address, Port
11.    = 2368)
12.    3) Synchronizes the data of LiDAR to the motion of stepper motor by
13.    using its position
14.    Author: Saket Dasika
15.    Date created: 10/25/2017
16.    Date last modified: 06/06/2018
17.    Python Version: 2.7
18. """
19.
20. Stepper motor functions
21. ACx - Acceleration of x rev/sec/sec
22. DEX - Deceleration of x rev/sec/sec
23. VEx - Deceleration of x rev/sec
24. DI/DI-x - Move x rev in clockwise direction/counter clockwise direction
25. FP- Moves the stepper motor in the desired velocity profile
26. DL1 - Activating Limit Switches
27. """
28. #importing required modules
29. import socket #for udp socket communication to log the data from lidar
30. import time #for creating a file with timestamp
31. import Tkinter #for gui
32. import ttk #for creating comboboxes
33. from Tkinter import *
34. import serial #for serial communication to control stepper motor
35. global ser #defining a serial port variable named 'ser'
36. #defining serial port parameters
37. port = "COM3" #COM port
38. baud = 9600 #baud rate
39. ser = serial.Serial(port,baud)
40. ser.bytesize = serial.EIGHTBITS #number of bits per bytes
41. ser.parity = serial.PARITY_NONE #set parity check: no parity
42. ser.stopbits = serial.STOPBITS_ONE #number of stop bits
43. ser.timeout = 0 #block read
44. import threading #for starting a parallel thread
45. global a,b,c,x,y,z,f,g,h,stopclick,stoplidar #defining global variables
46.
47. """
48. Creating a gui with three frames columnwise and placing the comboxes,buttons and
49. labels
50. on these three columns
51. """
52. class Application(Frame): #defining a class
53.
54.     def __init__(self, parent):
55.         Frame.__init__(self, parent)
56.         self.initialize()
```

```

52.
53.     def initialize(self):
54.         self.grid() #initializing a grid
55.         #dividing grid into three rectangular frames
56.         rightframe = Frame(root) #right frame
57.         middleframe = Frame(root) #middleframe
58.         leftframe = Frame(root) #left frame
59.         #positioning three vertical frames in the grid
60.         rightframe.grid(column=2,row=0,padx=90)
61.         middleframe.grid(column=1,row=0)
62.         leftframe.grid(column=0, row=0,padx=20)
63.         root.resizable(False, False) #gui size got locked
64. #row - refers to which row
65. #column - refers to which coloumn
66. #pad and ipad - inner and outer padding (size and size between)
67. #x, y - x and y direction
68.
69.         #creating four buttons and attributing it to corresponding four commands
70.         # positioning and sizing it
71.         button1 = Button(rightframe,text = "Connect",font = "Times 20",command=s
self.onbutton1click) #button connect
72.         button1.grid(column=0,row=0,pady=10,ipadx=40,ipady=3)
73.         button2 = Button(rightframe,text = "Start Position",font = "Times 20",co
mmand=self.onbutton2click) #button start position
74.         button2.grid(column=0,row=1,pady=10,ipadx=40,ipady=3)
75.         button3 = Button(rightframe,text = "Run",font = "Times 20",command=self.
onbutton3click) #button run
76.         button3.grid(column=0,row=2,pady=10,ipadx=40,ipady=3)
77.         button4 = Button(rightframe,text = "Disconnect",font = "Times 20",comman
d=self.onbutton4click) #button disconnect
78.         button4.grid(column=0,row=3,pady=10,ipadx=40,ipady=3)
79.         #creating four labels for velocity profile values
80.         #positioning and sizing it
81.         label1 = Label(leftframe, anchor="w",fg="black",text="Acceleration (m/s2
)",font = "Times 20") #label acceleration
82.         label1.grid(column=0,row=0,pady=10,ipadx=40,ipady=3)
83.         label2 = Label(leftframe,anchor="w",fg="black",text="Velocity (m/s)",fon
t = "Times 20") #label velocity
84.         label2.grid(column=0,row=1,pady=10,ipadx=40,ipady=3)
85.         label3 = Label(leftframe,anchor="w",fg="black",text="Deceleration (m/s2
)",font = "Times 20") #label deceleration
86.         label3.grid(column=0,row=2,pady=10,ipadx=40,ipady=3)
87.         label4 = Label(leftframe,anchor="w",fg="black",text="Displacement (m)",f
ont = "Times 20") #label displacement
88.         label4.grid(column=0,row=3,pady=10,ipadx=40,ipady=3)
89.         #creating four comboboxesfor selecting values corresponding to the label
beside and attributing a variable to it
90.         #four comboboxes for each of the acc,vel,dec,dis
91.         #defining a function to it
92.         #sizing and positing it
93.         self.cbsymbol1 = ttk.Combobox(middleframe, textvariable = variable1,font
= "Times 20")
94.         self.cbsymbol1.bind("<Return>", self.cbsymbol1_onEnter)
95.         self.cbsymbol1.bind("<<ComboboxSelected>>", self.cbsymbol1_onEnter) #fun
ction define
96.         self.cbsymbol1['values'] = ('0.1','0.2','0.3','0.4','0.5','0.6','0.7','0
.8','0.9','1.0','1.1','1.2','1.3','1.4','1.5','1.6','1.7','1.8','1.9','2.0')
97.         self.cbsymbol1.grid(row=0,pady=10,ipadx=40,ipady=3,column=0)
98.

```

```

99.         self.cbsymbol2 = ttk.Combobox(middleframe, textvariable= variable2,font
    = "Times 20")
100.             self.cbsymbol2.bind("<Return>",self.cbsymbol2_onEnter)
101.             self.cbsymbol2.bind("<<ComboboxSelected>>",self.cbsymbol2_onEnte
r) #function define
102.             self.cbsymbol2['values'] = ('0.1','0.2','0.3','0.4','0.5','0.6',
'0.7','0.8','0.9','1.0','1.1','1.2','1.3','1.4','1.5','1.6','1.7','1.8','1.9','2
.0')
103.             self.cbsymbol2.grid(row=1,pady=10,ipadx=40,ipady=3,column=0)
104.
105.             self.cbsymbol3 = ttk.Combobox(middleframe,textvariable= variable
3,font = "Times 20")
106.             self.cbsymbol3.bind("<Return>",self.cbsymbol3_onEnter)
107.             self.cbsymbol3.bind("<<ComboboxSelected>>",self.cbsymbol3_onEnte
r) #function define
108.             self.cbsymbol3['values'] = ('0.1','0.2','0.3','0.4','0.5','0.6',
'0.7','0.8','0.9','1.0','1.1','1.2','1.3','1.4','1.5','1.6','1.7','1.8','1.9','2
.0')
109.             self.cbsymbol3.grid(row=2,pady=10,ipadx=40,ipady=3,column=0)
110.
111.             self.cbsymbol4 = ttk.Combobox(middleframe, textvariable=variable
4,font = "Times 20")
112.             self.cbsymbol4.bind("<Return>",self.cbsymbol4_onEnter)
113.             self.cbsymbol4.bind("<<ComboboxSelected>>",self.cbsymbol4_onEnte
r) #function define
114.             self.cbsymbol4['values'] = ('0.1','0.2','0.3','0.4','0.5','0.6',
'0.7','0.8','0.9','1.0','1.1','1.2','1.3','1.4','1.5','1.6','1.7','1.8','1.9','2
.0')
115.             self.cbsymbol4.grid(row=3,pady=10,ipadx=40,ipady=3,column=0)
116.
117.             #below four functions are for comboboxes for getting the value wh
en entered and assigning it to a global variable
118.             #combobox has also option to take any value between 0 to end by t
yping and entering it
119.             def cbsymbol1_onEnter(self,event):
120.                 global mytext1
121.                 mytext1 = variable1.get() #getting the value from combobox when
selected
122.                 vals = self.cbsymbol1.cget('values')
123.                 self.cbsymbol1.select_range(0,END)
124.                 return 'break'
125.             def cbsymbol2_onEnter(self,event):
126.                 global mytext2
127.                 mytext2 = variable2.get() #getting the value from combobox when sele
cted
128.                 vals = self.cbsymbol2.cget('values')
129.                 self.cbsymbol2.select_range(0,END)
130.                 return 'break'
131.
132.             def cbsymbol3_onEnter(self,event):
133.                 global mytext3
134.                 mytext3 = variable3.get() #getting the value from combobox when sele
cted
135.                 vals = self.cbsymbol3.cget('values')
136.                 self.cbsymbol3.select_range(0,END)
137.                 return 'break'
138.
139.
140.             def cbsymbol4_onEnter(self,event):
141.                 global mytext4

```

```

142.         mytext4 = variable4.get() #getting the value from combobox when sele
        cted
143.         vals = self.cbsymbol4.cget('values')
144.         self.cbsymbol4.select_range(0,END)
145.         return 'break'
146.
147.         def onbutton1click(self): #if connect is pushed then it will open the se
        rial port and connects to the motor
148.             ser.write("SCy.d.d..`'.....v.;...p.....Z...`...X.X.2...
        ....N 1.....N 8R.....2.(....2..v.q.....d...v...K.....p.xcustom motor
        .....v...a...W.....ZSTPMTRH34066D!." + "\r\n") #serial command written to
        initialize the motor
149.             ser.write("HR" + "\r\n")
150.
151.         else:
152.             ser.open()
153.             ser.write("SCy.d.d..`'.....v.;...p.....Z...`...X.X.2..
        .....N 1.....N 8R.....2.(....2..v.q.....d...v...K.....p.xcustom motor
        .....v...a...W.....ZSTPMTRH34066D!." + "\r\n")
154.             ser.write("HR" + "\r\n")
155.
156.         def onbutton2click(self): #'start position' is used to start the sys
        tem at the same poosition
157.             ser.write("DI-200" + "\r\n")
158.             ser.write("VE0.5" + "\r\n")
159.             ser.write("SH2L" + "\r\n") # it will comeback and hit the limit
        switch
160.             ser.write("SP0" + "\r\n") #It will make the position of limit sw
        itch as origin
161.             ser.write("FP11676" + "\r\n") #it will move 0.1m forward for off
        set
162.
163.         """
164.         The lidar() function connects to an ip address and port and starts loggi
        ng the data when the linear motion starts and
165.         stops when it reaches back using event 'stopclick'
166.         """
167.         def lidar(self): #lidar data collection
168.             global mytext1,mytext2,mytext3,mytext4
169.             global stopclick
170.             HOST = "192.168.1.77" #host address
171.             PORT = 2368 #port
172.             timestr = time.strftime("%H%M%S-%m%d%Y") #timestring formatted
173.             file_object_UDP = open('C:/Data/' + timestr + ' ' + mytext1 + '
        ' + mytext2 + ' ' + mytext3 + ' ' + mytext4 + ' ' + 'gr' + '.csv', 'a') #filenam
        e with time and velocity profile values
174.             soc = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) #creating
        a socket
175.             soc.bind((HOST, PORT)) #socket binds to the port
176.             file_object_UDP.write('TimeStamp' + ',' + 'distance' + ',' + myt
        ext1 + ',' + mytext2 + ',' + mytext3 + ',' + mytext4 + '\n') #it writes intially
        all the velcoity profile paramters in the first row
177.             def UDP_parsing(data,file_obj): #parse the incoming data
178.                 timestamp = data[1200:1204] #timestamp data
179.                 timeresult = int(timestamp[3].encode('hex'),16)*16777216+int
        (timestamp[2].encode('hex'),16)*65536+int(timestamp[1].encode('hex'),16)*256+int
        (timestamp[0].encode('hex'),16)#micro seconds
180.                 t = (float(timeresult)/float(1000000)) #seconds past the hou
        r
181.                 file_obj.write(str(t) + ',' +str(data.encode('hex')) + '\n')
        #it writes in hex format

```

```

182.         while True:
183.             try:
184.                 data = soc.recv(2048)
185.                 if len(data) > 0:
186.                     UDP_parsing(data,file_object_UDP) #if data is coming
187.                     parse it
188.             except Exception, e:
189.                 print dir(e), e.message, e.__class__.__name__
190.                 traceback.print_exc(e)
191.                 if stopclick == True: #stopclick is used to stop the data l
192.                     ogging when it reached it start position again
193.                     stopclick = False #stopclick is reverted back to false
194.
195.                 file_object_UDP.close() #closes the file
196.                 break
197.             """
198.             The onbutton3click() function sends the command to the stepper motor fo
199.             r moving the motor
200.             in the desired velocity profile and starts a background thread
201.             """
202.             def onbutton3click(self): #run button #all commands below are ahead
203.             y defined for the stepper motor
204.                 global mytext1,mytext2,mytext3,mytext4
205.                 global stopclick
206.                 stopclick = False #initially stopclick = flase
207.                 stoplidar = '' #initially empty string
208.                 ser.flushInput() #flush input buffer, discarding all its conten
209.                 ts
210.                 ser.flushOutput() #flush output buffer, aborting current output
211.
212.                 a = (float(mytext4) * 5.832 * 20000) #converting m to revolution
213.                 s of stepper motor
214.                 b = int(11676 + a) #offset
215.                 c = str(b)
216.                 x = (float(mytext1) * 5.832) #converting all the m/s to surestep
217.                 pro language (rev/sec)
218.                 y = (float(mytext2) * 5.832)
219.                 z = (float(mytext3) * 5.832)
220.                 f = str(x)
221.                 g = str(y)
222.                 h = str(z)
223.                 t = threading.Thread(target=self.lidar, args=()) #parallel threa
224.                 ding
225.                 t.start() #start the thread
226.                 r = 3600*((float(mytext1)*2+float(mytext2)*float(mytext2))/(flea
227.                 t(mytext1)*float(mytext2))) #just to check the document size if we get all the d
228.                 ata
229.                 l = str(r)
230.                 print(l+"KB") #printing that on screen
231.                 ser.write("DL1" + "\r\n")#activating limit switches on either di
232.                 rections
233.                 ser.write("AC" + f + "\r\n") #ac - acceleration #writing all the
234.                 velocity profile to stepper motor
235.                 ser.write("VE" + g + "\r\n") #ve-velocity
236.                 ser.write("DE" + h + "\r\n") #de - deceleration
237.                 ser.write("FP" + c + "\r\n") #displacement
238.                 ser.write("AC" + f + "\r\n")
239.                 ser.write("VE" + g + "\r\n")
240.                 ser.write("DE" + h + "\r\n")
241.                 ser.write("FP11676" + "\r\n") #it goes forward and comes back to
242.                 starring position, starting position = 11676

```



```

228.             while True:
229.                 stoplidar = ser.readline() #reads the incoming position value by writing a line of code
230.                 ser.write("IP" + "\r\n") #by writing IP to serial port it sends out the immediate position of stepper motor
231.                 if stoplidar.find("IP=11676")!=-1: #if it reaches the starting position again then break and stop the data logging by making stopclick true
232.                     break
233.                 stopclick = True
234.
235.             def onbutton4click(self):
236.                 ser.close() #disconnect will close the serial port
237.
238.
239.         if __name__ == "__main__":
240.             root = Tk() #creates a window which doesn't show up
241.             root.geometry('{}x{}'.format(1100,400)) #size of the gui
242.             root.title('Linear Motion System') #title
243.             variable1 = StringVar(root) # 4 variables created for four comboboxes
244.             variable2 = StringVar(root)
245.             variable3 = StringVar(root)
246.             variable4 = StringVar(root)
247.             app = Application(root) #creates the application
248.             root.mainloop() #looping indefinitely

```

Appendix D. LiDAR Data Post Processing for Test Target

D.1 Point Cloud from Raw LiDAR Data

The MATLAB script used to obtain a 3D point cloud from raw LiDAR data by converting hexadecimal to decimal data and using velocity values to project into local 3D coordinate frame of linear motion system. The script can be used for any other object with minor changes in ground height section

D.1.1 Authorship Information

- Author : Saket Dasika
- Last Modified: 06/06/2018
- Version: R2016a

D.1.2 Automating Files and Preallocation

```
clearvars -except meanheightgrwh meanheightgrbl i fName; %clear the workspace
except the mean height of ground value
clc; %clear the command window
%file = 'C:\Users\sda273\OneDrive\Data\135515-01262018 0.3 0.5 0.3 2.0 wh
repl.xlsx';
%file=fNames{i}; %automating the files
[num,text,row] = xlsread(file);% Read from excel
Timestamp = zeros(length(row),1); %create Timestamp for preallocation
azimuth = zeros(length(row),24); %create azimuth for preallocation
azimuth1 = zeros(length(row),24); %create azimuth1(copy of azimuth) for
preallocation (used for finding missing azimuth values)
distance{1,16} = []; %create distance 1*16 cell for preallocation
intensity{1,16} = [];%create intensity 1*16 cell for preallocation
for i=1:16
    distance{i} = zeros(length(row),12,2);
    intensity{i} = zeros(length(row),12,2);
end
```

D.1.3 Height of Ground

```
%getting the file name and see if it is white side or black side and then
%use the corresponding ground
[filepath,name,ext] = fileparts(file);
whit = 'wh';
blac = 'bl';
if isempty(strfind(name, whit))==0
    meanheightgr = meanheightgrwh;
else if isempty(strfind(name, blac))==0
    meanheightgr = meanheightgrbl;
end
end
```

D.1.4 Velocity Profile Calculations

```
acc = num(1,3);%acceleration
vel = num(1,4);%steady state velocity
dec = num(1,5); %deceleration
dis = num(1,6);%displacement in one direction
%A1-A6 are indices of the timestamp when they reach the different aspects
%of velocity profile
A1 = zeros(length(raw),1); %preallocation A1
A2 = zeros(length(raw),1); %preallocation A2
A3 = zeros(length(raw),1); %preallocation A3
A4 = zeros(length(raw),1); %preallocation A4
A5 = zeros(length(raw),1); %preallocation A5
A6 = zeros(length(raw),1); %preallocation A6
acctime = vel/acc; %time to accelerate
accdist = (acc*acctime*acctime)/2; %distance moved while accelerating
dectime = vel/dec; %time to deceleration
decdist = ((vel*dectime) - (dec*dectime*dectime)/2); %distance moved while
deceleration
stedist = 2 - accdist - decdist; %calculate steady state distance
stetime = (dis-accdist-decdist)/vel; %time in steady state velocity
totaltime = (acctime+stetime+dectime)*2; %total time taken

for i = 2:length(raw)
    Timestamp(i) = num(i,1); %getting all the timestamp values
end

try
    for i = 2:length(raw)
        A1(i,1) = Timestamp(i) - Timestamp(2);
    end %finding the differences between corresponding timestamp and the
starting
catch
end

I1 = find(A1>acctime); %find when it reaches steady state velocity
a1 = I1(1); %finding the index/packet number of it

try
    for i = a1:length(raw)
        A2(i,1) = Timestamp(i) - Timestamp(a1);
    end %finding the differences between corresponding timestamp and the
starting
catch
end

I2 = find(A2>stetime); %find when it starts to decelerate
a2 = I2(1); %finding the index/packet number of it

try
    for i = a2:length(raw)
        A3(i,1) = Timestamp(i) - Timestamp(a2);
```

```

    end %finding the differences between corresponding timestamp and the
starting
catch
end
I3 = find(A3>dectime); %find when it stops in forward motion
a3 = I3(1); %finding the index/packet number of it

try
    for i = a3:length(raw)
        A4(i,1) = Timestamp(i) - Timestamp(a3);
    end %finding the differences between corresponding timestamp and the
starting
catch
end
I4 = find(A4>acctime); %find when it reaches steady state velocity in backward
motion
a4 = I4(1); %finding the index/packet number of it

try
    for i = a4:length(raw)
        A5(i,1) = Timestamp(i) - Timestamp(a4);
    end %finding the differences between corresponding timestamp and the
starting
catch
end
I5 = find(A5>stetime); %find when it starts to declerate in backward motion
a5 = I5(1); %finding the index/packet number of it

try
    for i = a5:length(raw)
        A6(i,1) = Timestamp(i) - Timestamp(a5);
    end %finding the differences between corresponding timestamp and the
starting
catch
end
I6 = find(A6>dectime);%find when it stops in backward motion
try
    a6 = I6(1); %finding the index/packet number of it, Try statement is used
because sometimes it may not be there
catch
end

```

D.1.5 Hexadecimal to Decimal Conversion

```

try
    for i = 2:length(raw)
        for j = 0:200:2200
            %finding azimuth angles which are spaced apart 200 characters in
between
            %azimuth1 - used for finding the missing azimuth

```

```

        azimuth(i, (2*(j/200))+1) =
hex2dec([raw{i,2}(1,7+j),raw{i,2}(1,8+j),raw{i,2}(1,5+j),raw{i,2}(1,6+j)])/100;
        azimuthl(i, (2*(j/200))+1) =
hex2dec([raw{i,2}(1,7+j),raw{i,2}(1,8+j),raw{i,2}(1,5+j),raw{i,2}(1,6+j)])/100;
        for k = 0:96:96
            %distance values calculated for all 16 channels for all 12 data
            blocks spaced 200 characters in between(j) and
            %2 times which are spaced 96 characters in between(k)
            distance{1}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+11+k),raw{i,2}(1, j+12+k),raw{i,2}(1, j+9+k),raw{i,2}(1, j+1
0+k)])*2;
            distance{2}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+17+k),raw{i,2}(1, j+18+k),raw{i,2}(1, j+15+k),raw{i,2}(1, j+
16+k)])*2;
            distance{3}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+23+k),raw{i,2}(1, j+24+k),raw{i,2}(1, j+21+k),raw{i,2}(1, j+
22+k)])*2;
            distance{4}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+29+k),raw{i,2}(1, j+30+k),raw{i,2}(1, j+27+k),raw{i,2}(1, j+
28+k)])*2;
            distance{5}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+35+k),raw{i,2}(1, j+36+k),raw{i,2}(1, j+33+k),raw{i,2}(1, j+
34+k)])*2;
            distance{6}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+41+k),raw{i,2}(1, j+42+k),raw{i,2}(1, j+39+k),raw{i,2}(1, j+
40+k)])*2;
            distance{7}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+47+k),raw{i,2}(1, j+48+k),raw{i,2}(1, j+45+k),raw{i,2}(1, j+
46+k)])*2;
            distance{8}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+53+k),raw{i,2}(1, j+54+k),raw{i,2}(1, j+51+k),raw{i,2}(1, j+
52+k)])*2;
            distance{9}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+59+k),raw{i,2}(1, j+60+k),raw{i,2}(1, j+57+k),raw{i,2}(1, j+
58+k)])*2;
            distance{10}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+65+k),raw{i,2}(1, j+66+k),raw{i,2}(1, j+63+k),raw{i,2}(1, j+
64+k)])*2;
            distance{11}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+71+k),raw{i,2}(1, j+72+k),raw{i,2}(1, j+69+k),raw{i,2}(1, j+
70+k)])*2;
            distance{12}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+77+k),raw{i,2}(1, j+78+k),raw{i,2}(1, j+75+k),raw{i,2}(1, j+
76+k)])*2;
            distance{13}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+83+k),raw{i,2}(1, j+84+k),raw{i,2}(1, j+81+k),raw{i,2}(1, j+
82+k)])*2;
            distance{14}(i, (j/200)+1, (k/96)+1) =
hex2dec([raw{i,2}(1, j+89+k),raw{i,2}(1, j+90+k),raw{i,2}(1, j+87+k),raw{i,2}(1, j+
88+k)])*2;

```

```

        distance{15}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+95+k),raw{i,2}(1,j+96+k),raw{i,2}(1,j+93+k),raw{i,2}(1,j+
94+k)])*2;
        distance{16}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+101+k),raw{i,2}(1,j+102+k),raw{i,2}(1,j+99+k),raw{i,2}(1,
j+100+k)])*2;

        %intensity values calculated for all 16 channels for all 12
data blocks spaced 200 characters in between(j) and
        %2 times which are spaced 96 characters in between(k)
        intensity{1}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+13+k),raw{i,2}(1,j+14+k)]);
        intensity{2}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+19+k),raw{i,2}(1,j+20+k)]);
        intensity{3}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+25+k),raw{i,2}(1,j+26+k)]);
        intensity{4}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+31+k),raw{i,2}(1,j+32+k)]);
        intensity{5}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+37+k),raw{i,2}(1,j+38+k)]);
        intensity{6}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+43+k),raw{i,2}(1,j+44+k)]);
        intensity{7}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+49+k),raw{i,2}(1,j+50+k)]);
        intensity{8}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+55+k),raw{i,2}(1,j+56+k)]);
        intensity{9}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+61+k),raw{i,2}(1,j+62+k)]);
        intensity{10}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+67+k),raw{i,2}(1,j+68+k)]);
        intensity{11}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+73+k),raw{i,2}(1,j+74+k)]);
        intensity{12}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+79+k),raw{i,2}(1,j+80+k)]);
        intensity{13}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+85+k),raw{i,2}(1,j+86+k)]);
        intensity{14}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+91+k),raw{i,2}(1,j+92+k)]);
        intensity{15}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+97+k),raw{i,2}(1,j+98+k)]);
        intensity{16}(i,(j/200)+1,(k/96)+1) =
hex2dec([raw{i,2}(1,j+103+k),raw{i,2}(1,j+104+k)]);
        end
    end
end
catch
end

```

D.1.6 Finding Missing Azimuth Values

```
try
    for i = 2:length(row) %finding missing azimuth values using azimuthl
        for j = 0:10
            if (azimuth(i,(2*j)+3) < azimuth(i,(2*j)+1)) %checking for reverted
back angles
                azimuthl(i,(2*j)+3) = azimuthl(i,(2*j)+3) + 360; %adding 360
degrees to the reverted back angles
            end
            azimuth(i,(2*j)+2) = azimuth(i,(2*j)+1) + ((azimuthl(i,(2*j)+3)-
azimuth(i,(2*j)+1))/2); %interpolating the missing ones
            if ( azimuth(i,(2*j)+2)>=360) %checking if they are bigger than 360
degrees
                azimuth(i,(2*j)+2) = azimuth(i,(2*j)+2) - 360; %reverting back
the angles by subtracting 360 degrees
            end
        end
        azimuth(i,24) = (azimuth(i,23)) + (azimuth(i,23) - azimuth(i,21))/2;
%last azimuth angle
        if ( azimuth(i,24)>=360) %checking if they are bigger than 360 degrees
            azimuth(i,24) = azimuth(i,24) - 360; %reverting back the angles by
subtracting 360 degrees
        end
    end
catch
end
```

D.1.7 Conversion of Polar Coordinates to Cartesian Coordinates

```
%preallocating for forward motion
Xforw = zeros(a2-a1+1,384);
Yforw = zeros(a2-a1+1,384);
Zforw = zeros(a2-a1+1,384);
inforw = zeros(a2-a1+1,384);
timeoffsetforw = zeros(a2-a1+1,384); %timeoffset for firing lasers

%preallocating for backward motion
Xback = zeros(a5-a4+1,384);
Yback = zeros(a5-a4+1,384);
Zback = zeros(a5-a4+1,384);
inback = zeros(a5-a4+1,384);
timeoffsetback = zeros(a5-a4+1,384); %timeoffset for firing lasers

lasertime = 2.304*10^-6; %time between firing two adjacent lasers in same
sequence
firingtime = 55.296*10^-6; %time between whole sequence
%timeoffset for laser = sequence index*firingtime + laserindex *
%lasertime
```

```

for i = a1:a2%adding Z value because the system moves in positive Z direction
    %two times for two azimuth angles and two types of laser ids(odd and even)
    for j = 0:200:2200
        if (azimuth(i,2*(j/200)+1)>30 && azimuth(i,2*(j/200)+1)<330) % clipping
            the values below mentioned angles, only looking between -30 to 30 degrees
                for k =1:16
                    Xforw(i-(a1-1),((j/200))*32+(k)) = NaN;
                    Yforw(i-(a1-1),((j/200))*32+(k)) = NaN;
                    Zforw(i-(a1-1),((j/200))*32+(k)) = NaN;
                    inforw(i-(a1-1),((j/200))*32+(k)) = NaN;
                end
            else
                %finding X,Y,Z for even channels 0,2,4.. and converting polar to
                %cartesian coordinates
                for k=1:2:16
                    Xforw(i-(a1-1),((j/200))*32+(k)) =
                    distance{k}(i,(j/200)+1,1)*cosd(k-16)*sind(azimuth(i,2*(j/200)+1));
                    Yforw(i-(a1-1),((j/200))*32+(k)) = meanheightgr-
                    (distance{k}(i,(j/200)+1,1)*cosd(k-16)*cosd(azimuth(i,2*(j/200)+1)));
                    timeoffsetforw(i-(a1-1),((j/200))*32+(k)) = (j/100)*firingtime
                    + (k-1)*lasertime; %timeoffset for laser
                    Zforw(i-(a1-1),((j/200))*32+(k)) =
                    distance{k}(i,(j/200)+1,1)*sind(k-16) + (vel*1000*(Timestamp(i) +
                    timeoffsetforw(i-(a1-1),((j/200))*32+(k)) - Timestamp(a1)));
                    inforw(i-(a1-1),((j/200))*32+(k)) =
                    intensity{k}(i,(j/200)+1,1);
                end
                for k =2:2:16
                    %finding X,Y,Z for odd channels 1,3,5.. and converting polar to
                    %cartesian coordinates
                    Xforw(i-(a1-1),((j/200))*32+(k)) =
                    distance{k}(i,(j/200)+1,1)*cosd(k-1)*sind(azimuth(i,2*(j/200)+1));
                    Yforw(i-(a1-1),((j/200))*32+(k)) = meanheightgr-
                    (distance{k}(i,(j/200)+1,1)*cosd(k-1)*cosd(azimuth(i,2*(j/200)+1)));
                    timeoffsetforw(i-(a1-1),((j/200))*32+(k)) = (j/100)*firingtime
                    + (k-1)*lasertime; %timeoffset for laser
                    Zforw(i-(a1-1),((j/200))*32+(k)) =
                    distance{k}(i,(j/200)+1,1)*sind(k-1) + (vel*1000*(Timestamp(i) +
                    timeoffsetforw(i-(a1-1),((j/200))*32+(k)) - Timestamp(a1)));
                    inforw(i-(a1-1),((j/200))*32+(k)) =
                    intensity{k}(i,(j/200)+1,1);
                end
            end
        end

        if (azimuth(i,2*(j/200)+2)>30 && azimuth(i,2*(j/200)+2)<330) % clipping
            the values below mentioned angles, only looking between -30 to 30 degrees
                for k = 17:32
                    Xforw(i-(a1-1),((j/200))*32+(k)) = NaN;
                    Yforw(i-(a1-1),((j/200))*32+(k)) = NaN;
                    Zforw(i-(a1-1),((j/200))*32+(k)) = NaN;
                end
            end
        end
    end
end

```



```

        inforw(i-(a1-1),((j/200))*32+(k)) = NaN;
    end
else
    %finding X,Y,Z for even channels 0,2,4.. and converting polar to
    %cartesian coordinates
    for k = 17:2:31
        Xforw(i-(a1-1),((j/200))*32+(k)) = distance{k-
16}(i,(j/200)+1,2)*cosd(k-32)*sind(azimuth(i,2*(j/200)+2));
        Yforw(i-(a1-1),((j/200))*32+(k)) = meanheightgr-(distance{k-
16}(i,(j/200)+1,2)*cosd(k-32)*cosd(azimuth(i,2*(j/200)+2)));
        timeoffsetforw(i-(a1-1),((j/200))*32+(k)) =
((j/100)+1)*firingtime + (k-17)*lasertime; %timeoffset for laser
        Zforw(i-(a1-1),((j/200))*32+(k)) = distance{k-
16}(i,(j/200)+1,2)*sind(k-32) + (vel*1000*(Timestamp(i) + timeoffsetforw(i-
(a1-1),((j/200))*32+(k)) - Timestamp(a1)));
        inforw(i-(a1-1),((j/200))*32+(k)) = intensity{k-
16}(i,(j/200)+1,2);
    end
    for k=18:2:32
        %finding X,Y,Z for odd channels 1,3,5.. and converting polar to
        %cartesian coordinates
        Xforw(i-(a1-1),((j/200))*32+(k)) = distance{k-
16}(i,(j/200)+1,2)*cosd(k-17)*sind(azimuth(i,2*(j/200)+2));
        Yforw(i-(a1-1),((j/200))*32+(k)) = meanheightgr-(distance{k-
16}(i,(j/200)+1,2)*cosd(k-17)*cosd(azimuth(i,2*(j/200)+2)));
        timeoffsetforw(i-(a1-1),((j/200))*32+(k)) =
((j/100)+1)*firingtime + (k-17)*lasertime; %timeoffset for laser
        Zforw(i-(a1-1),((j/200))*32+(k)) = distance{k-
16}(i,(j/200)+1,2)*sind(k-17) + (vel*1000*(Timestamp(i) + timeoffsetforw(i-
(a1-1),((j/200))*32+(k)) - Timestamp(a1)));
        inforw(i-(a1-1),((j/200))*32+(k)) = intensity{k-
16}(i,(j/200)+1,2);
    end
end
end
end
for i = a4:a5 %subtracting Z value because the system moves in negative Z
direction
    for j = 0:200:2200
        if (azimuth(i,2*(j/200)+1)>30 && azimuth(i,2*(j/200)+1)<330) % clipping
the values below mentioned angles, only looking between -30 to 30 degrees
            for k =1:16
                Xback(i-(a4-1),((j/200))*32+(k)) = NaN;
                Yback(i-(a4-1),((j/200))*32+(k)) = NaN;
                Zback(i-(a4-1),((j/200))*32+(k)) = NaN;
                inback(i-(a4-1),((j/200))*32+(k)) = NaN;
            end
        else
            for k=1:2:16
                %finding X,Y,Z for even channels 0,2,4.. and converting polar
to

```

```

        %cartesian coordinates
        Xback(i-(a4-1),((j/200))*32+(k)) =
distance{k}(i,(j/200)+1,1)*cosd(k-16)*sind(azimuth(i,2*(j/200)+1));
        Yback(i-(a4-1),((j/200))*32+(k)) = meanheightgr-
(distance{k}(i,(j/200)+1,1)*cosd(k-16)*cosd(azimuth(i,2*(j/200)+1)));
        timeoffsetback(i-(a1-1),((j/200))*32+(k)) = (j/100)*firingtime
+ (k-1)*lasertime; %timeoffset for laser
        Zback(i-(a4-1),((j/200))*32+(k)) =
distance{k}(i,(j/200)+1,1)*sind(k-16) - (vel*1000*(Timestamp(i) +
timeoffsetback(i-(a1-1),((j/200))*32+(k)) - Timestamp(a4)));
        inback(i-(a4-1),((j/200))*32+(k)) =
intensity{k}(i,(j/200)+1,1);
    end
    for k = 2:2:16
        %finding X,Y,Z for odd channels 1,3,5... and converting polar
to
        %cartesian coordinates
        Xback(i-(a4-1),((j/200))*32+(k)) =
distance{k}(i,(j/200)+1,1)*cosd(k-1)*sind(azimuth(i,2*(j/200)+1));
        Yback(i-(a4-1),((j/200))*32+(k)) = meanheightgr-
(distance{k}(i,(j/200)+1,1)*cosd(k-1)*cosd(azimuth(i,2*(j/200)+1)));
        timeoffsetback(i-(a1-1),((j/200))*32+(k)) = (j/100)*firingtime
+ (k-1)*lasertime; %timeoffset for laser
        Zback(i-(a4-1),((j/200))*32+(k)) =
distance{k}(i,(j/200)+1,1)*sind(k-1) - (vel*1000*(Timestamp(i) +
timeoffsetback(i-(a1-1),((j/200))*32+(k)) - Timestamp(a4)));
        inback(i-(a4-1),((j/200))*32+(k)) =
intensity{k}(i,(j/200)+1,1);
    end
end

    if (azimuth(i,2*(j/200)+2)>30 && azimuth(i,2*(j/200)+2)<330) % clipping
the values below mentioned angles, only looking between -30 to 30 degrees
        for k = 17:32
            Xback(i-(a4-1),((j/200))*32+(k)) = NaN;
            Yback(i-(a4-1),((j/200))*32+(k)) = NaN;
            Zback(i-(a4-1),((j/200))*32+(k)) = NaN;
            inback(i-(a4-1),((j/200))*32+(k)) = NaN;
        end
    else
        for k = 17:2:31
            %finding X,Y,Z for even channels 0,2,4..and converting polar to
%cartesian coordinates
            Xback(i-(a4-1),((j/200))*32+(k)) = distance{k-
16}(i,(j/200)+1,2)*cosd(k-32)*sind(azimuth(i,2*(j/200)+2));
            Yback(i-(a4-1),((j/200))*32+(k)) = meanheightgr-(distance{k-
16}(i,(j/200)+1,2)*cosd(k-32)*cosd(azimuth(i,2*(j/200)+2)));
            timeoffsetback(i-(a1-1),((j/200))*32+(k)) =
((j/100)+1)*firingtime + (k-17)*lasertime; %timeoffset for laser

```

```

        Zback(i-(a4-1),((j/200))*32+(k)) = distance{k-
16}(i,(j/200)+1,2)*sind(k-32) - (vel*1000*(Timestamp(i) + timeoffsetback(i-(a1-
1),((j/200))*32+(k)) - Timestamp(a4)));
        inback(i-(a4-1),((j/200))*32+(k)) = intensity{k-
16}(i,(j/200)+1,2);
        end
        for k=18:2:32
            %finding X,Y,Z for odd channels 1,3,5.. and converting polar to
            %cartesian coordinates
            Xback(i-(a4-1),((j/200))*32+(k)) = distance{k-
16}(i,(j/200)+1,2)*cosd(k-17)*sind(azimuth(i,2*(j/200)+2));
            Yback(i-(a4-1),((j/200))*32+(k)) = meanheightgr-(distance{k-
16}(i,(j/200)+1,2)*cosd(k-17)*cosd(azimuth(i,2*(j/200)+2)));
            timeoffsetback(i-(a1-1),((j/200))*32+(k)) =
((j/100)+1)*firingtime + (k-17)*lasertime; %timeoffset for laser
            Zback(i-(a4-1),((j/200))*32+(k)) = distance{k-
16}(i,(j/200)+1,2)*sind(k-17) - (vel*1000*(Timestamp(i) + timeoffsetback(i-(a1-
1),((j/200))*32+(k)) - Timestamp(a4)));
            inback(i-(a4-1),((j/200))*32+(k)) = intensity{k-
16}(i,(j/200)+1,2);
        end
    end
end
end
end

```

D.1.8 Defining the Point Cloud

```

Total_intensity_forw = mean(mean(inforw, 'omitnan'), 'omitnan'); %finding the
mean intensity in forward direction
Total_intensity_back = mean(mean(inback, 'omitnan'), 'omitnan'); %finding the
mean intensity in backward direction

ptCloudforw = pointCloud([Xforw(:),Yforw(:),Zforw(:)]); %pointcloud in forward
direction (positive z direction)
ptCloudback = pointCloud([Xback(:),Yback(:),Zback(:)]); %pointcloud in backward
direction (negative z direction)

ptCloudinten_forw = pointCloud([inforw(:),inforw(:),inforw(:)]); %intensity
cloud in forward direction
ptCloudinten_back = pointCloud([inback(:),inback(:),inback(:)]); %intensity
cloud in backward direction

[ptCloudforwup,inlierIndices1] = pcdnoise(ptCloudforw); %denoising forward
point cloud
[ptCloudbackup,inlierIndices2] = pcdnoise(ptCloudback); %denoising backward
point cloud
ptCloudinfo = select(ptCloudinten_forw,inlierIndices1); %removing the same
indices for intensity cloud as point cloud
ptCloudinbade = select(ptCloudinten_back,inlierIndices2); %removing the same
indices for intensity cloud as point cloud

```

```

%plotting all the point clouds
figure(1)
pcshow(ptCloudforw);
title('Forward Motion');
xlabel('X');
ylabel('Y');
zlabel('Z');

figure(2)
pcshow(ptCloudback);
title('Backward Motion');
xlabel('X');
ylabel('Y');
zlabel('Z');

%plotting those denoised ones
figure(3)
pcshow(ptCloudforwup);
title('Forward Motion(updated)');
xlabel('X');
ylabel('Y');
zlabel('Z');
% zoom(0.7); %adding zoom if required

figure(4)
pcshow(ptCloudbackup);
title('Backward Motion(updated)');
xlabel('X');
ylabel('Y');
zlabel('Z');

```

D.2 Feature Extraction from Point Cloud

The MATLAB script was used to extract heights of each point, mean height, standard deviation, mean intensity, point density on a single test target from the point cloud of the whole test target

D.2.1 Authorship Information

- Author : Saket Dasika
- Last Modified: 06/07/2018
- Version: R2016a

D.2.2 Obtaining the Z Coordinate of the Whole Test target

```

%the z coordinate is the lower limit of the range of z axis of 800mm plates
%which can be used to find the location of other plates as the position of
%the plates was known relative to each other in z axis

```

```

clearvars -except meanheightgrwh meanheightgrbl ptCloudforwup ptCloudbackup
ptCloudinfode ptCloudinbade Total_intensity_forw Total_intensity_back name
fNames
clc;
roi = [-500 500;750 850;0 2000]; %roi for the 800 mm targets
indices = findPointsInROI(ptCloudforwup,roi); %finding the inlier points
valout = select(ptCloudforwup,indices);
%the pointcloudforwup was changed to ptcloudbackup for backward motion
%plotting the 800 mm targets
figure(1)
pcshow(valout)
title('Validation_8(Plates)')

zcoordinate = valout.ZLimits(1); %finding the z coodinate which can be then
used to find the roi's for all the other height targets

```

D.2.3 Roi's for the Test Targets

```

%finding the point clouds from those indcies which corresponds to the 25
%targets
%roix_y corresponds to roi for plate numberx and height y
%example roi3_6 corresponds to plate 3 in 600mm plates
%plate numbering can be any order

roi1_8 = [-500 -300;750 900;zcoordinate+800 zcoordinate+1000];
roi2_8 = [-300 -100;750 900;zcoordinate zcoordinate+200];
roi3_8 = [-100 100;750 900;zcoordinate+600 zcoordinate+800];
roi4_8 = [100 300;750 900;zcoordinate+200 zcoordinate+400];
roi5_8 = [300 500;750 900;zcoordinate+400 zcoordinate+600];

indices1_8 = findPointsInROI(ptCloudforwup,roi1_8);
indices2_8 = findPointsInROI(ptCloudforwup,roi2_8);
indices3_8 = findPointsInROI(ptCloudforwup,roi3_8);
indices4_8 = findPointsInROI(ptCloudforwup,roi4_8);
indices5_8 = findPointsInROI(ptCloudforwup,roi5_8);

valdeout1_8 = select(ptCloudforwup,indices1_8);
valdeout2_8 = select(ptCloudforwup,indices2_8);
valdeout3_8 = select(ptCloudforwup,indices3_8);
valdeout4_8 = select(ptCloudforwup,indices4_8);
valdeout5_8 = select(ptCloudforwup,indices5_8);

roi1_6 = [-500 -300;575 700;zcoordinate+600 zcoordinate+800];
roi2_6 = [-300 -100;575 700;zcoordinate+400 zcoordinate+600];
roi3_6 = [-100 100;575 700;zcoordinate zcoordinate+200];
roi4_6 = [100 300;575 700;zcoordinate+800 zcoordinate+1000];
roi5_6 = [300 500;575 700;zcoordinate+200 zcoordinate+400];

indices1_6 = findPointsInROI(ptCloudforwup,roi1_6);
indices2_6 = findPointsInROI(ptCloudforwup,roi2_6);

```

```

indices3_6 = findPointsInROI(ptCloudforwup,roi3_6);
indices4_6 = findPointsInROI(ptCloudforwup,roi4_6);
indices5_6 = findPointsInROI(ptCloudforwup,roi5_6);

valdeout1_6 = select(ptCloudforwup,indices1_6);
valdeout2_6 = select(ptCloudforwup,indices2_6);
valdeout3_6 = select(ptCloudforwup,indices3_6);
valdeout4_6 = select(ptCloudforwup,indices4_6);
valdeout5_6 = select(ptCloudforwup,indices5_6);

roi1_5 = [-500 -300;450 575;zcoordinate zcoordinate+200];
roi2_5 = [-300 -100;450 575;zcoordinate+200 zcoordinate+400];
roi3_5 = [-100 100;450 575;zcoordinate+800 zcoordinate+1000];
roi4_5 = [100 300;450 575;zcoordinate+400 zcoordinate+600];
roi5_5 = [300 500;450 575;zcoordinate+600 zcoordinate+800];

indices1_5 = findPointsInROI(ptCloudforwup,roi1_5);
indices2_5 = findPointsInROI(ptCloudforwup,roi2_5);
indices3_5 = findPointsInROI(ptCloudforwup,roi3_5);
indices4_5 = findPointsInROI(ptCloudforwup,roi4_5);
indices5_5 = findPointsInROI(ptCloudforwup,roi5_5);

valdeout1_5 = select(ptCloudforwup,indices1_5);
valdeout2_5 = select(ptCloudforwup,indices2_5);
valdeout3_5 = select(ptCloudforwup,indices3_5);
valdeout4_5 = select(ptCloudforwup,indices4_5);
valdeout5_5 = select(ptCloudforwup,indices5_5);

roi1_3 = [-500 -300;250 400;zcoordinate+200 zcoordinate+400];
roi2_3 = [-300 -100;250 400;zcoordinate+600 zcoordinate+800];
roi3_3 = [-100 100;250 400;zcoordinate+400 zcoordinate+600];
roi4_3 = [100 300;250 400;zcoordinate zcoordinate+200];
roi5_3 = [300 500;250 400;zcoordinate+800 zcoordinate+1000];

indices1_3 = findPointsInROI(ptCloudforwup,roi1_3);
indices2_3 = findPointsInROI(ptCloudforwup,roi2_3);
indices3_3 = findPointsInROI(ptCloudforwup,roi3_3);
indices4_3 = findPointsInROI(ptCloudforwup,roi4_3);
indices5_3 = findPointsInROI(ptCloudforwup,roi5_3);

valdeout1_3 = select(ptCloudforwup,indices1_3);
valdeout2_3 = select(ptCloudforwup,indices2_3);
valdeout3_3 = select(ptCloudforwup,indices3_3);
valdeout4_3 = select(ptCloudforwup,indices4_3);
valdeout5_3 = select(ptCloudforwup,indices5_3);

roi1_1 = [-500 -300;50 200;zcoordinate+400 zcoordinate+600];
roi2_1 = [-300 -100;50 200;zcoordinate+800 zcoordinate+1000];
roi3_1 = [-100 100;50 200;zcoordinate+200 zcoordinate+400];
roi4_1 = [100 300;50 200;zcoordinate+600 zcoordinate+800];
roi5_1 = [300 500;50 200;zcoordinate zcoordinate+200];

```

```

indices1_1 = findPointsInROI(ptCloudforwup,roi1_1);
indices2_1 = findPointsInROI(ptCloudforwup,roi2_1);
indices3_1 = findPointsInROI(ptCloudforwup,roi3_1);
indices4_1 = findPointsInROI(ptCloudforwup,roi4_1);
indices5_1 = findPointsInROI(ptCloudforwup,roi5_1);

valdeout1_1 = select(ptCloudforwup,indices1_1);
valdeout2_1 = select(ptCloudforwup,indices2_1);
valdeout3_1 = select(ptCloudforwup,indices3_1);
valdeout4_1 = select(ptCloudforwup,indices4_1);
valdeout5_1 = select(ptCloudforwup,indices5_1);

%plotting all these targets
figure(3)
subplot(1,5,1)
pcshow(valdeout1_8)
subplot(1,5,2)
pcshow(valdeout2_8)
subplot(1,5,3)
pcshow(valdeout3_8)
subplot(1,5,4)
pcshow(valdeout4_8)
subplot(1,5,5)
pcshow(valdeout5_8)

figure(4)
subplot(1,5,1)
pcshow(valdeout1_6)
subplot(1,5,2)
pcshow(valdeout2_6)
subplot(1,5,3)
pcshow(valdeout3_6)
subplot(1,5,4)
pcshow(valdeout4_6)
subplot(1,5,5)
pcshow(valdeout5_6)

figure(5)
subplot(1,5,1)
pcshow(valdeout1_5)
subplot(1,5,2)
pcshow(valdeout2_5)
subplot(1,5,3)
pcshow(valdeout3_5)
subplot(1,5,4)
pcshow(valdeout4_5)
subplot(1,5,5)
pcshow(valdeout5_5)

```

```

figure(6)
subplot(1,5,1)
pcshow(valdeout1_3)
subplot(1,5,2)
pcshow(valdeout2_3)
subplot(1,5,3)
pcshow(valdeout3_3)
subplot(1,5,4)
pcshow(valdeout4_3)
subplot(1,5,5)
pcshow(valdeout5_3)

```

```

figure(7)
subplot(1,5,1)
pcshow(valdeout1_1)
subplot(1,5,2)
pcshow(valdeout2_1)
subplot(1,5,3)
pcshow(valdeout3_1)
subplot(1,5,4)
pcshow(valdeout4_1)
subplot(1,5,5)
pcshow(valdeout5_1)

```

D.2.4 Feature Extraction

Function 'meanheight' was used to compute the paramters mentioned above for all the 25 point clouds corresponding to the tesr targets for more details refer to the mean height function

```

[meanheight1_8,count1_8,tcount1_8,stdeviation1_8,meanintensity1_8,heightm1_8] =
meanheight(valdeout1_8,ptCloudinfode,indices1_8);

[meanheight2_8,count2_8,tcount2_8,stdeviation2_8,meanintensity2_8,heightm2_8] =
meanheight(valdeout2_8,ptCloudinfode,indices2_8);

[meanheight3_8,count3_8,tcount3_8,stdeviation3_8,meanintensity3_8,heightm3_8] =
meanheight(valdeout3_8,ptCloudinfode,indices3_8);

[meanheight4_8,count4_8,tcount4_8,stdeviation4_8,meanintensity4_8,heightm4_8] =
meanheight(valdeout4_8,ptCloudinfode,indices4_8);

[meanheight5_8,count5_8,tcount5_8,stdeviation5_8,meanintensity5_8,heightm5_8] =
meanheight(valdeout5_8,ptCloudinfode,indices5_8);

[meanheight1_6,count1_6,tcount1_6,stdeviation1_6,meanintensity1_6,heightm1_6] =
meanheight(valdeout1_6,ptCloudinfode,indices1_6);

```



```

[meanheight2_6,count2_6,tcount2_6,stdeviation2_6,meanintensity2_6,heightm2_6] =
meanheight(valdeout2_6,ptCloudinfode,indices2_6);

[meanheight3_6,count3_6,tcount3_6,stdeviation3_6,meanintensity3_6,heightm3_6] =
meanheight(valdeout3_6,ptCloudinfode,indices3_6);

[meanheight4_6,count4_6,tcount4_6,stdeviation4_6,meanintensity4_6,heightm4_6] =
meanheight(valdeout4_6,ptCloudinfode,indices4_6);

[meanheight5_6,count5_6,tcount5_6,stdeviation5_6,meanintensity5_6,heightm5_6] =
meanheight(valdeout5_6,ptCloudinfode,indices5_6);

[meanheight1_5,count1_5,tcount1_5,stdeviation1_5,meanintensity1_5,heightm1_5] =
meanheight(valdeout1_5,ptCloudinfode,indices1_5);

[meanheight2_5,count2_5,tcount2_5,stdeviation2_5,meanintensity2_5,heightm2_5] =
meanheight(valdeout2_5,ptCloudinfode,indices2_5);

[meanheight3_5,count3_5,tcount3_5,stdeviation3_5,meanintensity3_5,heightm3_5] =
meanheight(valdeout3_5,ptCloudinfode,indices3_5);

[meanheight4_5,count4_5,tcount4_5,stdeviation4_5,meanintensity4_5,heightm4_5] =
meanheight(valdeout4_5,ptCloudinfode,indices4_5);

[meanheight5_5,count5_5,tcount5_5,stdeviation5_5,meanintensity5_5,heightm5_5] =
meanheight(valdeout5_5,ptCloudinfode,indices5_5);

[meanheight1_3,count1_3,tcount1_3,stdeviation1_3,meanintensity1_3,heightm1_3] =
meanheight(valdeout1_3,ptCloudinfode,indices1_3);

[meanheight2_3,count2_3,tcount2_3,stdeviation2_3,meanintensity2_3,heightm2_3] =
meanheight(valdeout2_3,ptCloudinfode,indices2_3);

[meanheight3_3,count3_3,tcount3_3,stdeviation3_3,meanintensity3_3,heightm3_3] =
meanheight(valdeout3_3,ptCloudinfode,indices3_3);

[meanheight4_3,count4_3,tcount4_3,stdeviation4_3,meanintensity4_3,heightm4_3] =
meanheight(valdeout4_3,ptCloudinfode,indices4_3);

[meanheight5_3,count5_3,tcount5_3,stdeviation5_3,meanintensity5_3,heightm5_3] =
meanheight(valdeout5_3,ptCloudinfode,indices5_3);

```

```

[meanheight1_1,count1_1,tcount1_1,stdeviation1_1,meanintensity1_1,heightm1_1] =
meanheight(valdeout1_1,ptCloudinfode,indices1_1);

[meanheight2_1,count2_1,tcount2_1,stdeviation2_1,meanintensity2_1,heightm2_1] =
meanheight(valdeout2_1,ptCloudinfode,indices2_1);

[meanheight3_1,count3_1,tcount3_1,stdeviation3_1,meanintensity3_1,heightm3_1] =
meanheight(valdeout3_1,ptCloudinfode,indices3_1);

[meanheight4_1,count4_1,tcount4_1,stdeviation4_1,meanintensity4_1,heightm4_1] =
meanheight(valdeout4_1,ptCloudinfode,indices4_1);

[meanheight5_1,count5_1,tcount5_1,stdeviation5_1,meanintensity5_1,heightm5_1] =
meanheight(valdeout5_1,ptCloudinfode,indices5_1);

%finding the mean height of all the 5 targets of the same height
meanheight_8 =
((meanheight1_8+meanheight2_8+meanheight3_8+meanheight4_8+meanheight5_8)/(count
1_8+count2_8+count3_8+count4_8+count5_8));
meanheight_6 =
((meanheight1_6+meanheight2_6+meanheight3_6+meanheight4_6+meanheight5_6)/(count
1_6+count2_6+count3_6+count4_6+count5_6));
meanheight_5 =
((meanheight1_5+meanheight2_5+meanheight3_5+meanheight4_5+meanheight5_5)/(count
1_5+count2_5+count3_5+count4_5+count5_5));
meanheight_3 =
((meanheight1_3+meanheight2_3+meanheight3_3+meanheight4_3+meanheight5_3)/(count
1_3+count2_3+count3_3+count4_3+count5_3));
meanheight_1 =
((meanheight1_1+meanheight2_1+meanheight3_1+meanheight4_1+meanheight5_1)/(count
1_1+count2_1+count3_1+count4_1+count5_1));

%finding the mean point density of all the 5 targets of the same height
tcount_8 =
((tcount1_8+tcount2_8+tcount3_8+tcount4_8+tcount5_8)/(count1_8+count2_8+count3_
8+count4_8+count5_8));
tcount_6 =
((tcount1_6+tcount2_6+tcount3_6+tcount4_6+tcount5_6)/(count1_6+count2_6+count3_
6+count4_6+count5_6));
tcount_5 =
((tcount1_5+tcount2_5+tcount3_5+tcount4_5+tcount5_5)/(count1_5+count2_5+count3_
5+count4_5+count5_5));
tcount_3 =
((tcount1_3+tcount2_3+tcount3_3+tcount4_3+tcount5_3)/(count1_3+count2_3+count3_
3+count4_3+count5_3));
tcount_1 =
((tcount1_1+tcount2_1+tcount3_1+tcount4_1+tcount5_1)/(count1_1+count2_1+count3_
1+count4_1+count5_1));

```

```

%finding the mean standard deviation of all the 5 targets of the same height
stdeviation_8 =
((stdeviation1_8+stdeviation2_8+stdeviation3_8+stdeviation4_8+stdeviation5_8)/
(count1_8+count2_8+count3_8+count4_8+count5_8));
stdeviation_6 =
((stdeviation1_6+stdeviation2_6+stdeviation3_6+stdeviation4_6+stdeviation5_6)/
(count1_6+count2_6+count3_6+count4_6+count5_6));
stdeviation_5 =
((stdeviation1_5+stdeviation2_5+stdeviation3_5+stdeviation4_5+stdeviation5_5)/
(count1_5+count2_5+count3_5+count4_5+count5_5));
stdeviation_3 =
((stdeviation1_3+stdeviation2_3+stdeviation3_3+stdeviation4_3+stdeviation5_3)/
(count1_3+count2_3+count3_3+count4_3+count5_3));
stdeviation_1 =
((stdeviation1_1+stdeviation2_1+stdeviation3_1+stdeviation4_1+stdeviation5_1)/
(count1_1+count2_1+count3_1+count4_1+count5_1));

%finding the mean intensity of all the 5 targets of the same height
meanintensity_8 =
((meanintensity1_8+meanintensity2_8+meanintensity3_8+meanintensity4_8+meaninten
sity5_8)/(count1_8+count2_8+count3_8+count4_8+count5_8));
meanintensity_6 =
((meanintensity1_6+meanintensity2_6+meanintensity3_6+meanintensity4_6+meaninten
sity5_6)/(count1_6+count2_6+count3_6+count4_6+count5_6));
meanintensity_5 =
((meanintensity1_5+meanintensity2_5+meanintensity3_5+meanintensity4_5+meaninten
sity5_5)/(count1_5+count2_5+count3_5+count4_5+count5_5));
meanintensity_3 =
((meanintensity1_3+meanintensity2_3+meanintensity3_3+meanintensity4_3+meaninten
sity5_3)/(count1_3+count2_3+count3_3+count4_3+count5_3));
meanintensity_1 =
((meanintensity1_1+meanintensity2_1+meanintensity3_1+meanintensity4_1+meaninten
sity5_1)/(count1_1+count2_1+count3_1+count4_1+count5_1));

%no of missing targets
a5 = 5-(count1_8+count2_8+count3_8+count4_8+count5_8);
a4 = 5-(count1_6+count2_6+count3_6+count4_6+count5_6);
a3 = 5-(count1_5+count2_5+count3_5+count4_5+count5_5);
a2 = 5-(count1_3+count2_3+count3_3+count4_3+count5_3);
a1 = 5-(count1_1+count2_1+count3_1+count4_1+count5_1);

```

D.2.5 Sort and Write to Excel File

```

%putting everything in the excel file
A1 = [meanheight1_1 meanheight2_1 meanheight3_1 meanheight4_1 meanheight5_1
meanheight1_3 meanheight2_3 meanheight3_3 meanheight4_3 meanheight5_3
meanheight1_5 meanheight2_5 meanheight3_5 meanheight4_5 meanheight5_5
meanheight1_6 meanheight2_6 meanheight3_6 meanheight4_6 meanheight5_6
meanheight1_8 meanheight2_8 meanheight3_8 meanheight4_8 meanheight5_8];

```

```

A2 = [stdeviation1_1 stdeviation2_1 stdeviation3_1 stdeviation4_1
stdeviation5_1 stdeviation1_3 stdeviation2_3 stdeviation_3 stdeviation4_3
stdeviation5_3 stdeviation1_5 stdeviation2_5 stdeviation3_5 stdeviation4_5
stdeviation5_5 stdeviation1_6 stdeviation2_6 stdeviation3_6 stdeviation4_6
stdeviation5_6 stdeviation1_8 stdeviation2_8 stdeviation3_8 stdeviation4_8
stdeviation5_8];
A3 = [meanintensity1_1 meanintensity2_1 meanintensity3_1 meanintensity4_1
meanintensity5_1 meanintensity1_3 meanintensity2_3 meanintensity3_3
meanintensity4_3 meanintensity5_3 meanintensity1_5 meanintensity2_5
meanintensity3_5 meanintensity4_5 meanintensity5_5 meanintensity1_6
meanintensity2_6 meanintensity3_6 meanintensity4_6 meanintensity5_6
meanintensity1_8 meanintensity2_8 meanintensity3_8 meanintensity4_8
meanintensity5_8];
A4 = [tcount1_1 tcount2_1 tcount3_1 tcount4_1 tcount5_1 tcount1_3 tcount2_3
tcount3_3 tcount4_3 tcount5_3 tcount1_5 tcount2_5 tcount3_5 tcount4_5 tcount5_5
tcount1_6 tcount2_6 tcount3_6 tcount4_6 tcount5_6 tcount1_8 tcount2_8 tcount3_8
tcount4_8 tcount5_8];
A5 = {heightm1_1 heightm2_1 heightm3_1 heightm4_1 heightm5_1 heightm1_3
heightm2_3 heightm3_3 heightm4_3 heightm5_3 heightm1_5 heightm2_5 heightm3_5
heightm4_5 heightm5_5 heightm1_6 heightm2_6 heightm3_6 heightm4_6 heightm5_6
heightm1_8 heightm2_8 heightm3_8 heightm4_8 heightm5_8};
A = [A1;A2;A3;A4];

%converting the raw heights into columns with different row numbers to put them
in excel file
for i=1:25
    A5mod = A5{i};
    for row = 1 : size(A5mod, 1)
        ca{row,i} = A5mod(row);
    end
end

%It checks for differnet velocites and puts them in a particular placel
% for example if the velocity is 1.0 m/s then it starts putting from BB 10
% for the average values and it puts the raw height data values from BB 14
string1 = '0.1';
string2 = '0.5';
string3 = '1.0';
string4 = '1.5';
string5 = '2.2';

if isempty(strfind(name, string1))==0
    place = 'B';
else if isempty(strfind(name, string2))==0
    place = 'AB';
    else if isempty(strfind(name, string3))==0
        place='BB';
        else if isempty(strfind(name, string4))==0
            place='CB';
            else if isempty(strfind(name, string5))==0
                place = 'DB';

```

```

        end
    end
end
end
placeex = strcat(place,'10');
placeex2 = strcat(place,'14');

% If it is black then it will start from sheet number from 15 to 17 for 3
replications
% if it is white then it will start from sheet number 9 to 11 for 3
replications

string6 = 'bl';
string7 = 'wh';
string8 = 'repl';
string9 = 'rep2';
string10 = 'rep3';
if isempty(strfind(name, string6))==0
    sheet = 15;
else if isempty(strfind(name, string7))==0
    sheet=9;
    end
end
if isempty(strfind(name, string8))==0
    sheetmod = 0;
else if isempty(strfind(name, string9))==0
    sheetmod=1;
    else if isempty(strfind(name, string10))==0
        sheetmod=2;
    end
end
end
sheetex = sheet+sheetmod;
% writing to excel
xlswrite('C:\Users\sda273\OneDrive\Documents\masters thesis
files\LiDAR_Data.xlsx',A,sheetex,placeex)
xlswrite('C:\Users\sda273\OneDrive\Documents\masters thesis
files\LiDAR_Data.xlsx',ca,sheetex,placeex2)

```

D.3 Mean Height

The function 'meanheight' was used to compute the raw heights of all the points,mean height,standard deviation,mean intensity,missing targets for the point clouds of the individual targets

D.3.1 Authorship Information

- Author : Saket Dasika
- Last Modified: 06/06/2018

- Version: R2016a

D.3.2 Code for the Function

```

function [y,l,tcount,stdeviation,meanintensity,heightm] =
meanheight(ptcloud,ptcloudl,indices)
x = zeros(1,101); %preallocating x and z coordinates for creating a xz block to
divide the target volume over 10,000 blocks (100*100 squares)
z = zeros(1,101);
height = zeros(100,100); %preallocating the heights of each block
count = zeros(100,100); %preallocating the pointdensity of each block

try
    x(1) = ptcloud.XLimits(1); %getting the left limit of x and z of the point
cloud of the ground
    z(1) = ptcloud.ZLimits(1);
    xyz = ptcloud.Location; %getting all the points of the pointcloud
    heightm = zeros(100,100,length(xyz)); % preallocation heights of all data
points
    for i=2:101
        x(i) = ptcloud.XLimits(1)+((ptcloud.XLimits(2)-
ptcloud.XLimits(1))/100)*i); %splitting the x and z into 100*100 squares
        z(i) = ptcloud.ZLimits(1)+((ptcloud.ZLimits(2)-
ptcloud.ZLimits(1))/100)*i);
        end

        %for those 100x100 squares we check to see if each point lies in that
square, if it lies then it belongs to only that square
        for i = 1:100
            for j=1:100
                for k = 1:length(xyz)
                    if xyz(k,1)>=x(i) && xyz(k,1)<x(i+1) && xyz(k,3)>=z(j) &&
xyz(k,3)<z(j+1)
                        heightm(i,j,k) = xyz(k,2)-25.9; %finding the heights of all
the data points over the target for each square (25.9 is the offset of the MDF
)
                        height(i,j) = height(i,j) + xyz(k,2)-25.9; %finding the
average height of each square
                        count(i,j)=count(i,j)+1; %finding the number of points in
each square
                    end
                end
            end
            if count(i,j)==0 %if points are zero then height is zero
                height(i,j) = 0;
            end
        end
    end
    y = sum(sum(height))/sum(sum(count)); %finding the total mean height
    valinout = select(ptcloudl,indices); %finding the intensity values for the
data points in the target
catch

```

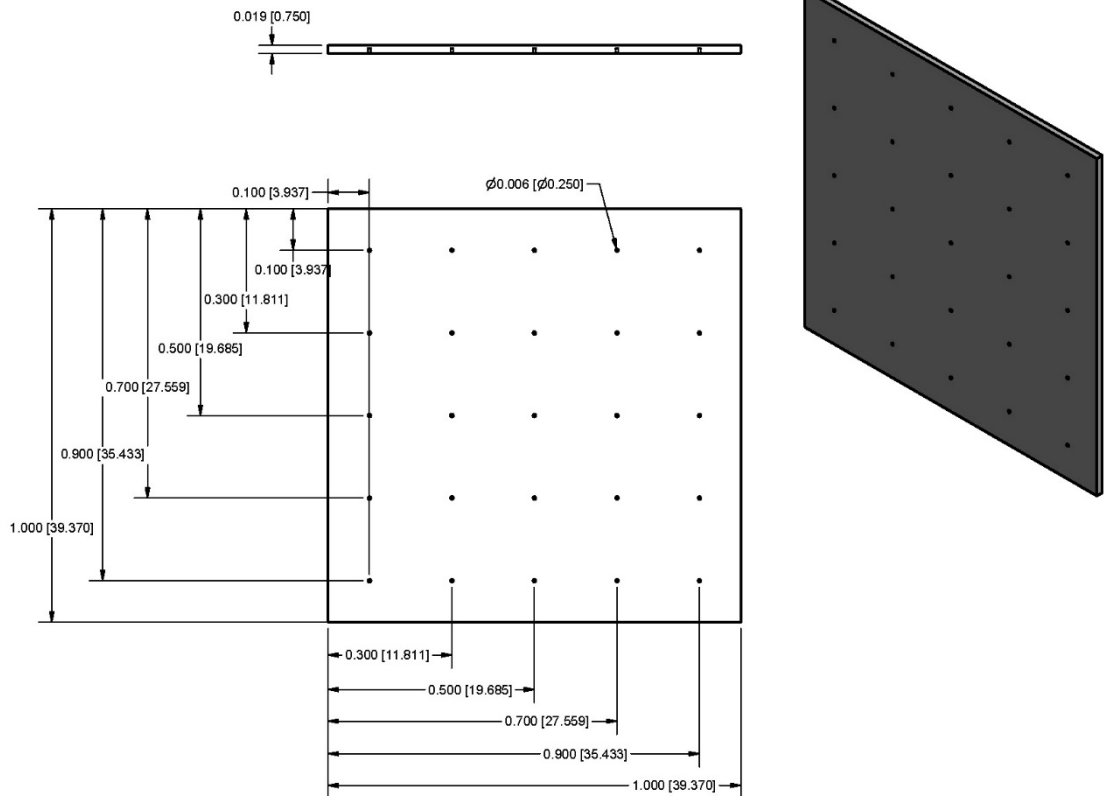
```

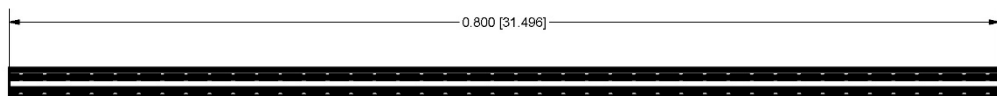
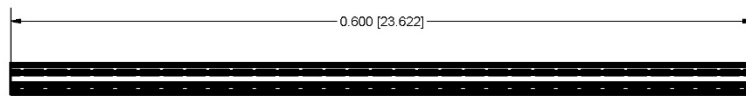
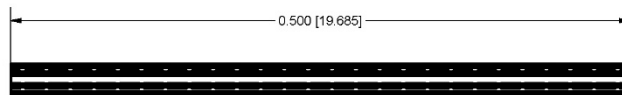
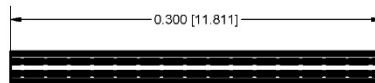
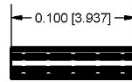
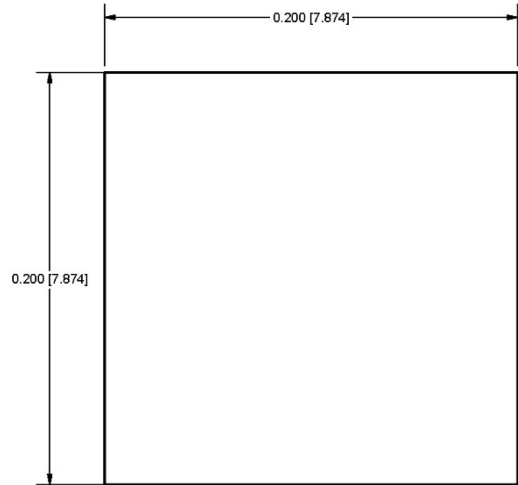
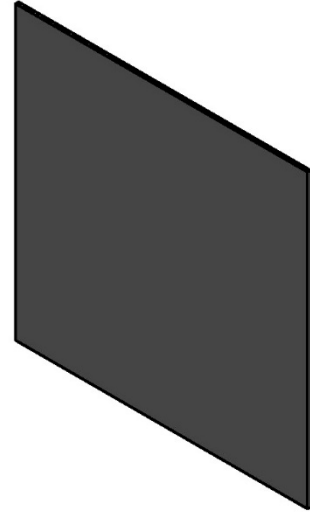
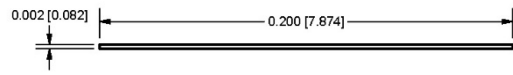
end

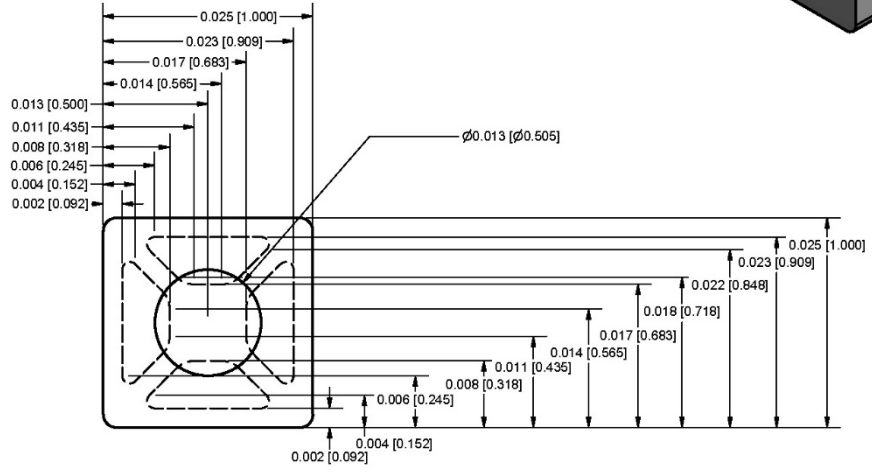
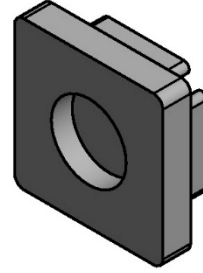
heightm(heightm==0) = []; %removing the zero values in the raw heights
heightm = heightm(:); %converting them into a single column
if sum(sum(count))<10 || isnan(y) || y == 0 %if there are less than 10 points
in a single target then the data is insufficient
    %and all the variables are zero
    l = 0;
    y = 0;
    tcount = 0;
    stdeviation = 0;
    meanintensity = 0;
else
    l = 1;
    y = sum(sum(height))/sum(sum(count)); %mean height
    tcount = sum(sum(count)); %total number of points
    stdeviation = std(heightm); %mean standard deviation
    meanintensity = mean(valinout.Location(:,1)); %mean intensity over the
target
end

```

Appendix E. LiDAR Test Target CAD Models







Appendix F. LiDAR Post Processing for Alfalfa

F.1 Feature Extraction from Point Cloud of Alfalfa

The MATLAB script was used to extract statistical parameters of the physical properties of the alfalfa plant

F.1.1 Contents

- [Authorship Information](#)
- [Limits of the Quadrat and Extraction of Point Cloud of Alfalfa](#)
- [Cube Method for finding the volume](#)
- [alphashape Method](#)
- [Block Method](#)
- [Octree Method](#)
- [Finding the min and max heights in n x n square](#)
- [Statistical distribution](#)
- [Lodged Plants](#)
- [Applying Threshold](#)
- [Sort and Write to Excel File](#)

F.1.2 Authorship Information

- Author : Saket Dasika
- Last Modified: 06/17/2018
- Version: R2016a

F.1.3 Limits of the Quadrat and Extraction of Point Cloud of Alfalfa

```
%this section was used to find the limits of boundary of quadrat and from
%that extract the alfalfa
clc;
clearvars -except ptCloudforwup ptCloudbackup ptCloudinfode ptCloudinbade name
fNames
xrange = ptCloudforwup.XLimits;%limits of the whole point cloud
yrange = ptCloudforwup.YLimits;
zrange = ptCloudforwup.ZLimits;

roiframe = [xrange(1) xrange(2);900 1100; zrange(1) zrange(2)];%extracting the
quadrat which is at a height of 1.0 m
frameindices = findPointsInROI(ptCloudforwup, roiframe);
frame = select(ptCloudforwup, frameindices);%quadrat point cloud
frame = pcdenoise(frame, 'NumNeighbors',100);%denoising to remove outliers
%plotting them
figure(1)
pcshow(ptCloudforwup)
figure(2)
pcshow(frame)

xrangefr = frame.XLimits;%limits of the boundary of quadrat
```

```

yrangefr = frame.YLimits;
zrangefr = frame.ZLimits;
widthframe = 48.3;%width of the quadrat frame used
% we are only interestedn in inside of quadrat so the width is added to the
% outer limits to obtain the inner limits of the quadrat
xrangefr(1) = xrangefr(1) + widthframe;
xrangefr(2) = xrangefr(2) - widthframe;
zrangefr(1) = zrangefr(1) + widthframe;
zrangefr(2) = zrangefr(2) - widthframe;
%check to see if the distance between inner limits is close to 1000
if xrangefr(2) - xrangefr(1) > 1025
    xrangefr(1) = xrangefr(1) + (0.5*widthframe);
    xrangefr(2) = xrangefr(2) - (0.5*widthframe);
end
if zrangefr(2) - zrangefr(1) > 1025
    zrangefr(1) = zrangefr(1) + (0.5*widthframe);
    zrangefr(2) = zrangefr(2) - (0.5*widthframe);
end

%extrcating alfalfa which is less than 1.0 m
roialfalfa = [xrangefr(1) xrangefr(2);0 1000;zrangefr(1) zrangefr(2)];
alfalfaindices = findPointsInROI(ptCloudforwup, roialfalfa);
alfalfa = select(ptCloudforwup, alfalfaindices);
alfalfa = pcdenoise(alfalfa,'NumNeighbors',500);%denoising to remove outliers
xyzalfa = alfalfa.Location;
xalfa = xyzalfa(:,1);
yalfa = xyzalfa(:,2);
zalfa = xyzalfa(:,3);
figure(3) %plotting the alfalfa point cloud
pcshow(alfalfa)
xlabel('X axis')
ylabel('Y axis')
zlabel('Z axis')

```

F.1.4 Cube Method for finding the volume

It's just taking the ROI as a whole cube and taking the cube volume as $V = abc$;

```

volumecube = (alfalfa.XLimits(2)-alfalfa.XLimits(1))*(alfalfa.YLimits(2)-
alfalfa.YLimits(1))*(alfalfa.ZLimits(2)-alfalfa.ZLimits(1));

```

F.1.5 alphashape Method

```

shp = alphaShape(xalfa,yalfa,zalfa);

figure(4)

plot(shp)

```

```
volumeshp = volume(shp);
```

F.1.6 Block Method

Dividing the xz plane to 10000 blocks and for each block the average height and volume is found All the volumes are added to give the total volume and the average is taken of all the average heights of each block to get the average height of the total area

```
block = 100; %100 x 100 = 10000
% preallocating the limits of each block
x = zeros(1,block+1);
z = zeros(1,block+1);
y = zeros(block,block);
count = zeros(block,block);
missing = zeros(block,block);

x(1) = alfalfa.XLimits(1);
z(1) = alfalfa.ZLimits(1);
volumemine=0;
height = zeros(block,block);

% finding the limits of each block
for i=2:block+1
    x(i) = alfalfa.XLimits(1)+(((alfalfa.XLimits(2)-
alfalfa.XLimits(1))/block)*i);
    z(i) = alfalfa.ZLimits(1)+(((alfalfa.ZLimits(2)-
alfalfa.ZLimits(1))/block)*i);
end

% check to see for each point if it is inside those limits
for i = 1:block
    for j=1:block
        for k = 1:length(xyzalfa)
            if xyzalfa(k,1)>=x(i) && xyzalfa(k,1)<x(i+1) && xyzalfa(k,3)>=z(j)
&& xyzalfa(k,3)<z(j+1)
                y(i,j) = y(i,j) + xyzalfa(k,2);
                height(i,j) = height(i,j) + xyzalfa(k,2);
                count(i,j)=count(i,j)+1;
            end
        end
        if count(i,j)==0
            y(i,j)=0;
            height(i,j) = 0;
            missing(i,j) = 0;

        else
            y(i,j) = y(i,j)/count(i,j);
            missing(i,j) = 1;
        end
    end
end
```

```

    end
end

meanheight = sum(sum(height))/sum(sum(count)); % mean height of alfalfa

for i = 1:block
    for j=1:block
        volumemine = volumemine+((x(i+1)-x(i))*y(i,j)*(z(i+1)-z(i))); %total
        volume by block method by adding each block volume
    end
end
end

```

F.1.7 Octree Method

Used a function OcTree developed by Sven (Copyright (c) 2013, Sven) in Matlab Libraries

```

OT=OcTree(xyzalfa,'style','weighted');
OT.shrink;
figure
boxH = OT.plot;
cols = lines(OT.BinCount);
doplot3 = @(p,varargin)plot3(p(:,1),p(:,2),p(:,3),varargin{:});
for i = 1:OT.BinCount
    set(boxH(i),'Color',cols(i,:), 'LineWidth', 1+OT.BinDepths(i))
    doplot3(xyzalfa(OT.PointBins==i,:), '.', 'Color',cols(i,:))
end
xlabel('x')
ylabel('y')
zlabel('z')
axis image, view(3)
volumeoctree = 0;
depthmax = max(OT.BinDepths);
L1=find(OT.BinDepths==depthmax);
b1=L1(1);
for m = b1:length(OT.BinBoundaries)
    volumeoctree = volumeoctree+ ((OT.BinBoundaries(m,4)-
OT.BinBoundaries(m,1))*(OT.BinBoundaries(m,5)-
OT.BinBoundaries(m,2))*(OT.BinBoundaries(m,6)-OT.BinBoundaries(m,3)));
    %volume by octree method
end
end

```

F.1.8 Finding the min and max heights in n x n square

```

% first method : side by side
% averaging over 25 blocks
space = 5; % n x n square
iterations = round(block/space);
heightm = zeros(iterations,iterations);%p[reallocating

```

```

countm = zeros(iterations,iterations);

for i = 1:iterations
    for j = 1:iterations
        heightm(i,j) = (sum(sum(y((i-1)*(space)+1:i*(space),(j-1)*(space)+1:j*(space))))/(space*space));
        countm(i,j) = (sum(sum(count((i-1)*(space)+1:i*(space),(j-1)*(space)+1:j*(space))))/(space*space));
    end
end

maxheightm = max(heightm(:));
minheightm = min(heightm(:));
maxcountm = max(countm(:));
mincountm = min(countm(:));

% second method: corresponding method
% averaging over twenty blocks
iteration = block-space+1;
heighti = zeros(iteration,iteration);
counti = zeros(iteration,iteration);

for i = 1:iteration
    for j = 1: iteration
        heighti(i,j) = (sum(sum(y(i:i+(space-1),j:j+(space-1))))/(space*space));
        counti(i,j) = (sum(sum(count(i:i+(space-1),j:j+(space-1))))/(space*space));
    end
end

maxheighti = max(heighti(:));
minheighti = min(heighti(:));
maxcounti = max(counti(:));
mincounti = min(counti(:));

```

F.1.9 Statistical distribution

```

maxheightalfa = max(yalfa);%max of all y coordinates
minheightalfa = min(yalfa);%min of all y coordinates
heights = maxheightalfa - minheightalfa;%max -min
totalsamples = alfa.Count;%total samples
morethanhalfheight = sum(yalfa>(heights/2 + minheightalfa))/totalsamples;
%percent of samples more than half the height
thinnessalfa = sum(yalfa<(mean(yalfa)-minheightalfa)*.25)/totalsamples;
%percent of samples less than quarter of mean height

```

F.1.10 Lodged Plants

```
% If the point density in a certian block is more than mean plus two
% standard deviations then it contains lodged plants
heightlodged = height;
ylodged = y;
stdev = std(countm(:)); %standard deviation
meancount = mean(countm(:));
countmax = meancount + 2*(stdev);
lodged = zeros(iterations,iterations);
heightlodgedm = zeros(iterations,iterations);
angle_lodged = 30; %angle of lodged plants with the ground

for i = 1:iterations
    for j = 1:iterations
        if countm(i,j)>=countmax
            lodged(i,j) = 1;
        else
            lodged(i,j) = 0;
        end
    end
end

% the points in lodged blocks will be projected with sin of the angle
% lodged to have true height of alfalfa
for i = 1:block
    for j = 1:block
        if lodged(ceil(i/5),ceil(j/5)) == 0;
            heightlodged(i,j) = heightlodged(i,j);
        else
            heightlodged(i,j) = heightlodged(i,j)/sind(angle_lodged);
        end
        if count(i,j)==0
            ylodged(i,j)= 0;
            heightlodged(i,j) = 0;
        else
            ylodged(i,j) = heightlodged(i,j)/count(i,j);
        end
    end
end

meanheight_lodged = sum(sum(heightlodged))/sum(sum(count)); % mean height after
applying the lodged plants algorithm

for i = 1:iterations
    for j = 1:iterations
        heightlodgedm(i,j) = (sum(sum(ylodged((i-1)*(space)+1:i*(space),(j-
1)*(space)+1:j*(space))))/(space*space));
    end
end
```

```

end

maxheightlodged = max(heightlodgedm(:)); %max height after lodged plants
algorithm
minheightlodged = min(heightlodgedm(:)); %min height after lodged plants
algorithm

```

F.1.11 Applying Threshold

```

% A threshold is applied to remove any ground points and 200 mm was used as
% threshold
xyzalfanew = xyzalfa(xyzalfa(:,2)>=200,:);
xalfanew = xyzalfanew(:,1);
yalfanew = xyzalfanew(:,2);
zalfanew = xyzalfanew(:,3);
alfalfanew = pointCloud([xalfanew,yalfanew,zalfanew]);
figure
pcshow(alfalfanew)
xlabel('X (mm)')
ylabel('Y (mm)')
zlabel('Z (mm)')
meanheightthreshold = mean(xyzalfanew(:,2)); % finding the mean height after
applying threshold

```

F.1.12 Sort and Write to Excel File

```

%putting everything in the excel file
A =
[meanheight;maxheightm;minheightm;maxheighti;minheighti;volumemine;volumeoctree
;volumeshp;volumecube;meanheight_lodged;maxheightlodged;minheightlodged;maxheig
htalfa;minheightalfa;heights;totalsamples;morethanhalfheight;thinnessalfa;meanh
eightthreshold];
%It checks for differnet velocites and puts them in a particular placel
string1 = '0.1';
string2 = '0.5';
string3 = '1.0';
string4 = '1.5';
string5 = '2.2';

string6 = 'rep1';
string7 = 'rep2';
string8 = 'rep3';

if isempty(strfind(name, string1))== 0
    place = 2;
else if isempty(strfind(name, string2))==0
    place = 9;
    else if isempty(strfind(name, string3))==0
        place = 16;

```



```

        else if isempty(strfind(name, string4))==0
            place = 23;
        else if isempty(strfind(name, string5))==0
            place = 30;

            end
        end
    end
end

if isempty(strfind(name, string6))== 0
    placemod = 0;
else if isempty(strfind(name, string7))==0
    placemod = 1;
    else if isempty(strfind(name, string8))==0
        placemod= 2;
        end
    end
end

placeex = place+placemod;
plac = xlcolumnletter(placeex); %custom function written to convert column
number to a letter for excel
plac = strcat(plac,'10');

% If it is pl1 then it will put in sheet 1
% if it is pl2 then it will put in sheet 2
string6 = 'pl1';
string7 = 'pl2';
if isempty(strfind(name, string6))==0
    sheet = 1;
else if isempty(strfind(name, string7))==0
    sheet = 2;
    end
end

%writing to excel
xlswrite('C:\Users\sda273\Documents\Alflafa_Data_new.xlsx',A,sheet,plac)

```

Appendix G. LiDAR Data Statistical Analysis Software

```

/*****
Filename: plate_analysis_final
Author: Surya Saket Dasika
Last Modified: 06/07/2018
Version: 9.4
The SAS Script is used to find the statistical differences of errors and standard deviations of black and
white targets for different velocities and heights. The script can be used to find the statistical differences
of errors between black and white targets
*****/
/* Importing data - remember to change filepath */

```

```

proc import out = black
datafile = "C:\Users\sda273\OneDrive\Documents\masters thesis files\Stats_Data_Black_updated.csv"
dbms = csv replace;
getnames = yes;
run;
/* Make column names */
data black;
set black;
rename _ = replication VAR2 = plot_no VAR3 = actual_height VAR4 = velocity
Var5 = estimated_height;
run;
/* Averaging over heights on same plate */
/* To average we need to sort the data */
proc sort data = black;
by replication plot_no actual_height velocity;
run;
/* Also looked at standard deviation of the measurements on a plate */
proc means data = black noprint;
by replication plot_no actual_height velocity; /* average by these factors */
var estimated_height; /* Variable to be averaged over */
output out = black_analysis mean = estimated_height std = sd_height; /* Output this to a data set
called black_analysis */
run;
/* Calculating the diff of estimated from actual */
data black_analysis;
set black_analysis;
height_diff = estimated_height - actual_height; /* Actual difference */
col = "Black";
run;
/* Analysis on height differences */
/* histogram of response */
proc sgplot data = black_analysis;
histogram height_diff;
run;
/* Boxplots of differences by actual height and velocity */
proc sgplot data = black_analysis;
vbox height_diff / group = actual_height;
run;
proc sgplot data = black_analysis;
vbox height_diff / group = velocity;
run;
/* Analysis of absolute height differences by actual_height and velocity */
proc glimmix data = black_analysis ;
*where velocity ^= -;
class velocity replication plot_no actual_height; /* Class variables */
model height_diff = actual_height velocity actual_height*velocity ; /* predict distance by
actual_height, velocity, and actual_height*velocity */
random replication velocity(replication) plot_no(actual_height); /* Random blocks
(or replications),also there is plate to plate variability that should be included */
lsmeans actual_height*velocity;
lsmeans velocity actual_height / diff adjust = tukey lines; /* Comparing velocities */
ods output lsmeans = fitblacks;
run;

```

```

/*plotting for the interaction between velocity and height*/
proc sort data = fitblacks; by velocity actual_height; run;
proc sgplot data = fitblacks;
where velocity > 0;
where actual_height > 0 ;
scatter x = velocity y = estimate / markercharattrs = (color=blue);
series x = velocity y = estimate / group = actual_height lineattrs = (pattern = 2 thickness = 2);
run;
/* Standard deviation of height measurements */
proc sgplot data = black_analysis;
histogram sd_height;
run;
/* Boxplots of differences by actual height and velocity */
proc sgplot data = black_analysis;
vbox sd_height/ group = actual_height;
run;
proc sgplot data = black_analysis;
vbox sd_height / group = velocity;
run;
/* Analysis of height differences by actual_height and velocity */
proc glimmix data = black_analysis;
where actual_height ^= 600;
class velocity replication plot_no actual_height; /* Class variables */
model sd_height = actual_height velocity actual_height*velocity ; /* predict distance by
actual_height, velocity, and actual_height*velocity */
random replication velocity(replication) plot_no(actual_height) ; /* Random blocks
(or replications),also there is plate to plate variability that should be included */
lsmeans actual_height*velocity;
lsmeans velocity actual_height / diff adjust = tukey lines; /* Comparing velocities */
ods output lsmeans = fitblacked;
run;
/*plotting for the interaction between velocity and height*/
proc sort data = fitblacked; by velocity actual_height; run;
proc sgplot data = fitblacked;
where velocity > 0;
where actual_height > 0 ;
scatter x = velocity y = estimate / markercharattrs = (color=blue);
series x = velocity y = estimate / group = actual_height lineattrs = (pattern = 2 thickness = 2);
run;
/* same analysis for white plates */
proc import out = white
datafile = "C:\Users\sda273\OneDrive\Documents\masters thesis files\Stats_Data_White_updated.csv"
dbms = csv replace;
getnames = yes;
run;
/* Make column names */
data white;
set white;
rename _ = replication VAR2 = plot_no VAR3 = actual_height VAR4 = velocity
Var5 = estimated_height;
run;
/* Averaging over heights on same plate */
/* To average we need to sort the data */

```

```

proc sort data = white;
by replication plot_no actual_height velocity;
run;
/* Also looked at standard deviation of the measurements on a plate */
proc means data = white noprint;
by replication plot_no actual_height velocity; /* average by these factors */
var estimated_height; /* Variable to be averaged over */
output out = white_analysis mean = estimated_height std = sd_height; /* Output this to a
data set called black_analysis */
run;
/* Calculating the diff of estimated from actual */
data white_analysis;
set white_analysis;
height_diff = estimated_height - actual_height; /* Actual difference */
col = "White";
run;
/* Analysis on height differences*/
/* histogram of response */
proc sgplot data = white_analysis;
histogram height_diff;
run;
/* Boxplots of differences by actual height and velocity */
proc sgplot data = white_analysis;
vbox height_diff / group = actual_height;
run;
proc sgplot data = white_analysis;
vbox height_diff / group = velocity;
run;
/* Analysis of absolute height differences by actual_height and velocity */
proc glimmix data = white_analysis ;
*where velocity ^= 1.5;
*where actual_height ^= 500;
class velocity replication plot_no actual_height; /* Class variables */
model height_diff = actual_height velocity actual_height*velocity ; /* predict distance by
actual_height, velocity, and actual_height*velocity */
random replication velocity(replication) plot_no(actual_height); /* Random blocks
(or replications),also there is plate to plate variability that should be included */
lsmeans actual_height*velocity;
lsmeans velocity actual_height / diff adjust = tukey lines; /* Comparing velocities */
ods output lsmeans = fitwhites;
run;
/*plotting for the interaction between velocity and height*/
proc sort data = fitwhites; by velocity actual_height; run;
proc sgplot data = fitwhites;
where velocity > 0;
where actual_height > 0 ;
scatter x = velocity y = estimate / markercharattrs = (color=blue);
series x = velocity y = estimate / group = actual_height lineattrs = (pattern = 2 thickness = 2);
run;
/* Standard deviation of height measurments */
proc sgplot data = white_analysis;
histogram sd_height;
run;

```

```

/* Boxplots of differences by actual height and velocity */
proc sgplot data = white_analysis;
vbox sd_height / group = actual_height;
run;
proc sgplot data = white_analysis;
vbox sd_height / group = velocity;
run;
/* Analysis of height differences by actual_height and velocity */
proc glimmix data = white_analysis;
*where actual_height ^= 500;
*where velocity ^= 2.2;
class velocity replication plot_no actual_height; /* Class variables */
model sd_height = actual_height velocity actual_height*velocity ; /* predict distance by
actual_height, velocity, and actual_height*velocity */
random replication velocity(replication) plot_no(actual_height) ; /* Random blocks
(or replications),also there is plate to plate variability that should be included */
lsmeans actual_height*velocity;
lsmeans velocity actual_height / diff adjust = tukey lines; /* Comparing velocities */
ods output lsmeans = fitwhited;
run;
/*plotting for the interaction between velocity and height*/
proc sort data = fitwhited; by velocity actual_height; run;
proc sgplot data = fitwhited;
where velocity > 0;
where actual_height > 0 ;
scatter x = velocity y = estimate / markercharattrs = (color=blue);
series x = velocity y = estimate / group = actual_height lineattrs = (pattern = 2 thickness = 2);
run;
/*t-test between white and black*/
data all_analysis;
set black_analysis white_analysis;
run;
proc ttest data = all_analysis;
var height_diff;
class col;
run;

```

REFERENCES

- Arnó, J., Escolà, A., Vallès, J. M., Llorens, J., Sanz, R., Masip, J., . . . Rosell-Polo, J. R. (2013). Leaf area index estimation in vineyards using a ground-based LiDAR scanner. *Precision Agriculture*, *14*(3), 290-306. <https://doi.org/10.1007/s11119-012-9295-0>
- Balenović, I., Seletković, A., Pernar, R., & Jazbec, A. (2015). *Estimation of the mean tree height of forest stands by photogrammetric measurement using digital aerial images of high spatial resolution.*
- Bonan, G. B. (1993). Importance of leaf area index and forest type when estimating photosynthesis in boreal forests. *Remote Sensing of Environment*, *43*(3), 303-314. [https://doi.org/10.1016/0034-4257\(93\)90072-6](https://doi.org/10.1016/0034-4257(93)90072-6)
- C. Swain, K., Jayasuriya, H., & Salokhe, V. (2018). *Low-Altitude Remote Sensing with Unmanned Radio-Controlled Helicopter Platforms: A Potential Substitution to Satellite-based Systems for Precision Agriculture Adoption under Farming Conditions in Developing Countries.*
- Colaço, A. F., Trevisan, R. G., Molin, J. P., Rosell-Polo, J. R., & Escolà, A. (2017). Orange tree canopy volume estimation by manual and LiDAR-based methods. *Advances in Animal Biosciences*, *8*(2), 477-480. <https://doi.org/10.1017/S2040470017001133>
- Colomina, I., & Molina, P. (2014). Unmanned aerial systems for photogrammetry and remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, *92*, 79-97. <https://doi.org/10.1016/j.isprsjprs.2014.02.013>
- Cui, Y., Zhao, K., Fan, W., & Xu, X. (2010). Using airborne LiDAR to retrieve crop structural parameters. *IEEE International Geoscience and Remote Sensing Symposium*.pp. 2107-2110 <https://doi.org/10.1109/IGARSS.2010.5650834>

Department of Defense, U. S. o. A. (2008). Global Positioning System Standard Positioning Service Performance Standard. *4th*

Dworak, V., Selbeck, J., & Ehlert, D. (2011). Ranging sensors for vehicle-based measurement of crop stand and orchard parameters: A review. *Transactions of the ASABE*, *54*(4), 1497. <https://doi.org/10.13031/2013.39013>

Estornell, J., Ruiz, L. A., Velázquez-Martí, B., & Fernández-Sarría, A. (2011). Estimation of shrub biomass by airborne LiDAR data in small forest stands. *Forest Ecology and Management*, *262*(9), 1697-1703. <https://doi.org/10.1016/j.foreco.2011.07.026>

Grenzdörffer, G. (2014). Crop height determination with UAS point clouds. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *40*(1), 135

Guo, T., Kujirai, T., & Watanabe, T. (2012). Mapping crop status from an unmanned aerial vehicle for precision agriculture applications. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, *XXXIX-B1*, 485-490. <https://doi.org/10.5194/isprsarchives-XXXIX-B1-485-2012>

Huang, Y., Thomson, S., Brand, H. J., & Reddy, K. N. (2016). *Development and evaluation of low-altitude remote sensing systems for crop production management* (Vol. 9).

Kelly, M., & Di Tommaso, S. (2015). Mapping forests with LiDAR provides flexible, accurate data with many uses. *California Agriculture*, *69*(1), 14-20. <http://dx.doi.org/10.3733/ca.v069n01p14>

Lamb, D. W., & Brown, R. B. (2001). PA—Precision Agriculture: Remote-Sensing and Mapping of Weeds in Crops. *Journal of Agricultural Engineering Research*, 78(2), 117-125. <https://doi.org/10.1006/jaer.2000.0630>

Lefsky, M. A., Cohen, W. B., Parker, G. G., & Harding, D. J. (2002). LiDAR remote sensing for ecosystem studies: LiDAR, an emerging remote sensing technology that directly measures the three-dimensional distribution of plant canopies, can accurately estimate vegetation structural attributes and should be of particular interest to forest, landscape, and global ecologists. *BioScience*, 52(1), 19-30. [http://dx.doi.org/10.1641/0006-3568\(2002\)052\[0019:LRSFES\]2.0.CO;2](http://dx.doi.org/10.1641/0006-3568(2002)052[0019:LRSFES]2.0.CO;2)

Lin, Y. (2015). LiDAR: An important tool for next-generation phenotyping technology of high potential for plant phenomics? *Computers and electronics in Agriculture*, 119, 61-73. <https://doi.org/10.1016/j.compag.2015.10.011>

Liu, X., & Zhang, Z. (2008). *LiDAR data reduction for efficient and high quality DEM generation* (Vol. 37).

Malveaux, C., G Hall, S., & Price, R. (2014). *Using drones in agriculture: Unmanned aerial systems for agricultural remote sensing applications*. 2014 Montreal, Quebec Canada July 13 – July 16, 2014, St. Joseph, MI. <http://elibrary.asabe.org/abstract.asp?aid=44960&t=5>

Mesas-Carrascosa, F. J., Castillejo-González, I. L., de la Orden, M. S., & Porras, A. G.-F. (2012). Combining LiDAR intensity with aerial camera data to discriminate agricultural land uses. *Computers and electronics in Agriculture*, 84, 36-46. <https://doi.org/10.1016/j.compag.2012.02.020>

Nex, F. C., & Rinaudo, F. (2011). LiDAR or Photogrammetry? Integration is the answer. *Rivista Italiana Di Telerilevamento*, 43(2), 107-121

Nie, S., Wang, C., Dong, P., Xi, X., Luo, S., & Zhou, H. (2016). Estimating leaf area index of maize using airborne discrete-return LiDAR data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(7), 3259-3266. <https://doi.org/10.1109/JSTARS.2016.2554619>

Pittman, J., Arnall, D., Interrante, S., Moffet, C., & Butler, T. (2015). Estimation of biomass and canopy height in bermudagrass, alfalfa, and wheat using ultrasonic, laser, and spectral sensors. *Sensors*, 15(2), 2920. <https://doi.org/10.3390/s150202920>

Radović, J., Sokolović, D., & Marković, J. (2009). Alfalfa-most important perennial forage legume in animal husbandry. *Biotechnology in Animal Husbandry*, 25(5-6-1), 465-475

Sanz-Cortiella, R., Llorens-Calveras, J., Escolà, A., Arnó-Satorra, J., Ribes-Dasi, M., Masip-Vilalta, J., . . . Rosell-Polo, J. R. (2011). Innovative LiDAR 3D dynamic measurement system to estimate fruit-tree leaf area. *Sensors*, 11(6), 5769. <https://doi.org/doi:10.3390/s110605769>

Selbeck, J., Dworak, V., & Ehlert, D. (2010). Testing a vehicle-based scanning lidar sensor for crop detection. *Canadian Journal of Remote Sensing*, 36(1), 24-35. <http://doi.org/10.5589/m10-022>

St-Onge, B., Vega, C., Fournier, R. A., & Hu, Y. (2008). Mapping canopy height using a combination of digital stereo-photogrammetry and lidar. *International Journal of Remote Sensing*, 29(11), 3343-3364. 10.1080/01431160701469040

Stamatiadis, S., Tsadilas, C., & Schepers, J. S. (2010). Ground-based canopy sensing for detecting effects of water stress in cotton. *Plant and Soil*, 331(1), 277-287.

<https://doi.org/10.1007/s11104-009-0252-2>

Sugiura, R., Noguchi, N., Ishii, K., & Terao, H. (2003). Development of Remote Sensing System using an Unmanned Helicopter (Part 1)

GIS Mapping for Agricultural Land Information. *Journal of the Japanese Society of Agricultural Machinery*, 65(1), 53-61. <http://doi.org/10.11357/jsam1937.65.53>

Sun, S., Li, C., Paterson, A. H., Jiang, Y., Xu, R., Robertson, J. S., . . . Chee, P. W. (2018). In-field high throughput phenotyping and cotton plant growth analysis using LiDAR. *Frontiers in Plant Science*, 9, 16. <https://doi.org/10.3389/fpls.2018.00016>

Sven. (2013). octree - partitioning 3D points into spatial subvolumes. *mathworks.com/MATLAB Central/File Exchange*

Vázquez-Arellano, M., Griepentrog, H., Reiser, D., & Paraforos, D. (2016). 3-D imaging systems for agricultural applications—A review. *Sensors*, 16(5), 618.

<https://doi.org/10.3390/s16050618>

Zhang, C., & Kovacs, J. M. (2012). The application of small unmanned aerial systems for precision agriculture: A review. *Precision Agriculture*, 13(6), 693-712.

<https://doi.org/10.1007/s11119-012-9274-5>

Zhang, L., & Grift, T. E. (2012). A LIDAR-based crop height measurement system for *Miscanthus giganteus*. *Computers and electronics in Agriculture*, 85, 70-76.

<https://doi.org/10.1016/j.compag.2012.04.001>

VITA

Surya Saket Dasika

PLACE OF BIRTH

- Eluru, India

EDUCATION

- B.Tech(Hons.). Agricultural and Food Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India, April 2016. Concentration: Control Systems and Robotics, Farm Machinery Systems Design. GPA: 8.29/10.00

PROFESSIONAL EXPERIENCE

- Graduate Research Assistant, Department of Biosystems and Agricultural Engineering, University of Kentucky, Lexington, Kentucky, August 2016 - Present. Advisor: Dr. Michael Sama
- Summer Intern, Department of Biosystems and Agricultural Engineering, University of Kentucky, Lexington, Kentucky, May 2015 - July 2015. Advisor: Dr. Michael Sama
- Summer Intern, Tractors and Farm Equipment Limited (TAFE), Chennai, India, May 2014 - June 2014. Advisor: Mr. Sreenivasulu Rao

PROFESSIONAL AND SCHOLASTIC ACHIEVEMENTS

- Ranked 3rd in my class in my undergrad in terms of GPA (April 2016)
- Ranked in top 1 % percentile in my entrance exam (IIT-JEE) among all the participants for my bachelor's degree
- KVPY Scholar, Indian Institute of Science, Bangalore, India

PROFESSIONAL PUBLICATIONS

- Sama, M.P., J.T. Evans, A.P. Turner, **S.S. Dasika**, “As-Applied Estimation of Volumetric Flow Rate from a Single Sprayer Nozzle Series using Water Sensitive Spray Cards”, Transactions of the ASABE. Vol. 59(3): 861-869, 2016.

PROFESSIONAL PRESENTATIONS

- **S. S. Dasika**, M. P. Sama, L. F. Pampolini, C. B. Good, J. S. Dvorak, “Performance Validation of a LiDAR Test Fixture for Remote Sensing in Agriculture”, AETC, Louisville, 2018.
- **S. S. Dasika**, M. P. Sama, L. F. Pampolini, C. B. Good, J. S. Dvorak, “A Linear Motion System for Remote Sensing Instrument Testing”, ASABE Annual International Meeting, Spokane, 2017.

PROFESSIONAL SOCIETIES

- American Society of Agricultural and Biological Engineers (ASABE)

PROFESSIONAL DEVELOPMENT

- ASABE ¼-Scale Tractor Student Design Competition. Peoria, IL, 2018.
- ASABE Agricultural Equipment Technology Conference. Louisville, KY, 2018.
- ASABE Annual International Meeting. Spokane, WA, 2017.
- CLOUD-MAP Flight Test Campaign. Stillwater, OK, 2017.
- ASABE ¼-Scale Tractor Student Design Competition. Peoria, IL, 2017.
- ASABE ¼-Scale Tractor Student Design Competition. Peoria, IL, 2015.

Surya Saket Dasika

July 20, 2018