VIABILITY AND PERFORMANCE OF INDOOR MAPPING USING THE VELODYNE VLP-16 LIDAR

by

ROHAN R PALEJA

A thesis submitted to the
School of Graduate Studies
Rutgers, The State University of New Jersey
In partial fulfillment of the requirements
For the degree of
Master of Science
Graduate Program in Mechanical and Aerospace Engineering
Written under the direction of
Professor Fransisco Javier Diez-Garias
And approved by

New Brunswick, New Jersey
October, 2018
ABSTRACT OF THE THESIS

Viability and Performance of Indoor Mapping using the Velodyne VLP-16 LiDAR

By ROHAN R PALEJA

Dissertation Director:
Professor Francisco Javier Diez-Garias

There is a growing demand for performing high precision tasks in indoor environments. Using a LiDAR to map the environment alongside a SLAM algorithm called LOAM, developed by Dr. Ji Zhang from CMU, environments can be mapped with low computational complexity. The use of this algorithm and sensor is tested in indoor environments to assess the performance and viability of indoor mapping with the Velodyne VLP-16 LiDAR. The experiment is tailored so that it mimics certain behaviors of a mobile robot in the hopes that the conclusion of this experiment can be generalized to mobile robots. Results of this experiment produced highly accurate clouds that were replicas to the real-world environment with an accuracy as high as 99.71%. Large indoor environments were also mapped (above 100 meters in length) with drift less than 1 meter in the best scenario. These results verify that accurate point cloud generation in indoor environments is viable and can be useful for mobile robots.
Acknowledgements

I would like to express deep gratitude to my advisor Professor Javier Diez-Garias for his constant leadership, loyalty, and confidence in me and this research effort. I would also like to thank Dr. Ji Zhang and Kaarta for their ROS package LOAM and use of their service free of charge.

I would also like to give thanks to my fellow student researchers Jubilee Prasad Rao, Marco Maia, Aristedes Costeas, Joao Serafini, and Andrii Hlyvko who assisted me with various miscellaneous tasks throughout the project. Lastly, I would like to thank my family for their continuous support and encouragement throughout my study.
# Table of Contents

Abstract ................................................................. ii

Acknowledgements .................................................... iii

List of Tables ............................................................ vi

List of Figures ........................................................... vii

1. Introduction ......................................................... 1
   1.1. Literature Review .............................................. 4
       1.1.1. Direct Monocular SLAM ................................. 5
       1.1.2. Direct Stereo Camera SLAM ............................. 6
       1.1.3. LiDAR-SLAM Point Clouds ............................... 7
           Google Cartographer ........................................ 7
           HDL Graph SLAM ............................................ 9
           LOAM: LiDAR Odometry and Mapping ....................... 10
       1.2. Theory of LiDAR Odometry and Mapping ................. 13

2. Experimental Setup-Instructional ................................. 16
   2.1. Equipment Details ........................................... 16
   2.2. Hardware Setup .............................................. 17
   2.3. Software Setup .............................................. 19
       2.3.1. Linux-Based ........................................... 20
           Package Information ...................................... 21
           Parameter Tuning ......................................... 26
           ROS Nodes ................................................ 27
       2.3.2. Windows-Based ......................................... 28
2.4. Experimental Procedure

2.5. Limitations

3. Results

3.1. Linux-Based LOAM

3.1.1. Mapping the Applied Fluids Laboratory, Moving to a Goal Position

3.1.2. Mapping the Applied Fluids Laboratory, Moving to a Goal Position and Back

3.1.3. Comparison Between Single Path and Traversed Path

3.1.4. Mapping the Eng. Hallway

3.1.5. Discussion on Point Cloud Noise and Error using Linux-Based LOAM

3.1.6. Analysis of Accuracy

Accuracy of the Applied Fluids Laboratory Point Clouds

Accuracy of the Eng. Hallway Point Clouds

3.2. Kaarta-Based LOAM

3.2.1. Mapping the Applied Fluids Laboratory, Moving to a Goal Position

3.2.2. Mapping the Applied Fluids Laboratory, Moving to a Goal Position and Back

3.2.3. Comparison Between Single Path and Traversed Path

3.2.4. Mapping the Eng. Hallway

3.2.5. Discussion on Point Cloud Noise using Kaarta LOAM

3.2.6. Analysis of Accuracy

Accuracy of the Applied Fluids Laboratory Point Clouds

Accuracy of Eng. Hallway Point Clouds

3.3. Post-Processing and Smoothing

3.4. Comparison Between Linux-Based LOAM and Kaarta-Based LOAM

4. Conclusions

4.1. Future Experiments

References
## List of Tables

3.1. Actual and Measured Distance to Objects in the Point Cloud, Traveling to Goal ................................................. 45

3.2. Actual and Measured Distance to Objects in the Point Cloud, Traveling to Goal and Back ........................................ 46

3.3. Hallway Experiments Accuracy ................................................. 47

3.4. Actual and Measured Distance to Objects in the Point Cloud, Traveling to Goal ................................................. 63

3.5. Actual and Measured Distance to Objects in the Point Cloud, Traveling to Goal and Back ........................................ 64

3.6. Hallway Accuracy ................................................................. 66
List of Figures

1.1. Usage of the Velodyne VLP-16 on Mobile Robots ........................................ 3
1.2. Sample Point Clouds ................................................................................... 4
1.3. Pathway from Feature Detection to Visual Words. Obtained from Mathworks. 5
1.4. LSD-SLAM Point Cloud Output [9] ............................................................... 6
1.5. LSD-Stereo-SLAM Point Cloud Output [6] .................................................... 6
1.7. HDL_Graph_SLAM Block Diagram [9] .......................................................... 9
1.8. HDL_Graph_SLAM Point Cloud Output [9] .................................................. 10
1.13. Selection of Feature and Edge Points [13] .................................................... 14
2.1. Velodyne VLP-16 LiDAR ............................................................................ 16
2.2. Sensor Plate for Mobile Robots ................................................................... 17
2.3. Wiring Diagram for Sensor Plate .................................................................. 19
2.5. Topics Displayed While Running the MPU-9250 .......................................... 23
2.6. Displays that Will be Open During Launch .................................................. 25
2.7. Depiction of all ROS Nodes during Experimentation ..................................... 27
2.8. VeloView Program Playing Recorded .pcap File ........................................... 29
2.9. Student Carrying the Sensor Plate During Experimentation with LOAM .... 31
2.10. Experimental Setup for Mapping the Eng. Hallway ..................................... 32
Chapter 1

Introduction

As the need for autonomous robots grows in industry, military, and other high-risk applications, the necessity of accurate localization and environment sensing is essential. Path planning in GPS-denied environments is nearly impossible or incredibly inefficient without a previously known map. After an accurate map of the environment is known, localization is a much easier task and thus, high precision operations can be conducted.

Since a map of the environment is clearly needed for most operations, sensors are required that can produce a high quality map. It is even more essential to not depend on sensors that are prone to loss of signal or high drift; these sensors include most GPS systems and cameras. Even sensors such as inertial measurement units and wheel encoders are prone to high drift if not merged with enough other data. GPS systems fail in most indoor environments and in situations where the GPS device cannot connect to at least 4 satellites. More so, a low-end GPS jammer device can easily disrupt satellite signals in a large area. Thus, GPS-based localization is impractical in indoor environments due to its unreliability and due to the attenuation of the GPS signal indoors. Cameras, on the other hand, are able to operate in most environments but are very dependent on lighting. In low-lit areas, cameras are known to fail. More so, camera localization depends on a feature rich environment and can be very computationally intensive and fail as environments contain less features.

Laser scanners on the other hand have numerous benefits when it comes to the SLAM (Simultaneous Localization and Mapping) problem. Several SLAM systems for mobile robots using a LiDAR have been proposed for use in indoor settings \[1\],\[2\]. While these methods are fabricated for indoor settings, they require the environment to be partially known before-hand and suffer from high computational complexity. An increase in computational complexity and thus, the computing resources required, causes the microcomputer on-board
the mobile robot to scale in size. It is important to note that Unmanned Aerial Vehicles (UAVs) have limited payload size and thus, limited memory. Thus, to ensure maximum usability of all mobile robots, it is necessary to use sensors and an algorithm with a strong balance between accuracy and computational complexity, where computational complexity limits how many sensors are being fused, and the number of processes the microcontroller can conduct.

Here, the reduction of payload and complexity is accomplished by the use of a smart algorithm alongside a high-performance LiDAR as the main sensor during SLAM. While the main sensor of this experiment is the LiDAR, analysis is performed with and without the assistance of an inertial measurement unit (IMU, which is combined into the odometry through the use of a Extended Kalman Filter), but this addition does not add significant computational complexity to the SLAM task.

To accomplish the SLAM task, the LiDAR must be able to sense the environment efficiently, solely recreating the common sensor "skirt" around mobile robots. In general, LiDARs sense by sending out powerful lasers, and use the time-velocity relationship to compute distances to objects. In the experiments and analysis presented below, the LiDAR used is the Velodyne VLP-16 Puck. Some special aspects of this LiDAR that make it ideal to be the sole sensor in SLAM application is its 360° horizontal field of view (FOV) and a 30° vertical FOV. Alongside its unique hardware with a light mass of 830 grams and low power consumption, its 100 meter range allows for sensing large areas. The VLP-16 can also generate up to 600,000 points per second leading to instantaneous static map generation. Some common uses of the Velodyne VLP-16 LiDAR on mobile robots are shown below.

The Pixforce LiDAR UAV carries the VLP-16 and can be used for mapping vegetation and capturing an aerial view of a city. One special aspect that aerial vehicles with a LiDAR can make use of is the dual return setting, which greatly increases its usability outdoors. Since light can penetrate multiple surfaces, several return lasers come from each beam and thus, sensing through surfaces and mapping regions such as the bottom of a small lake or the ground beneath a tree can be accomplished. This is another powerful aspect of LiDAR which shows its priority over a standard camera, which cannot see through a surface unless it is transparent. Several universities have made even more use of the return beam which
Figure 1.1: Usage of the Velodyne VLP-16 on Mobile Robots

has a measure of the intensity and have attempted recognition, successfully distinguishing between healthy vegetation and unhealthy vegetation among other tasks. Figure 1.1b corresponds to the Ford Autonomous Vehicle. The VLP-16 is used for localization in fusion with an IMU and wheel-encoders and ensures safe operation. Majority of the self-driving cars (excluding Tesla and a few others) use a LiDAR as their main sensor for obstacle detection and avoidance.

This paper seeks to test the performance of LiDAR SLAM in indoor environments that are feature-rich and feature-lax while assessing drift, noise, computational complexity, and overall accuracy. Tracking the trajectory of the mobile robot when creating the map is of great importance as this directly relates to the odometry; as point clouds and experiments are displayed, it is important to note that high performance tracking directly relates to a high-performance point cloud. More so, it is important to know the definition of a point cloud with focus on the relation between point cloud density and point cloud quality. Experimentation is presented that differs from previous experiments in that these experiments specifically show how LiDAR performs in indoor environments. More so, the experiments
Figure 1.2 refers to sample point cloud images. These provide a precursor for what can be expected from the VLP-16 when used alongside SLAM algorithms. Each of these point clouds can be considered of high performance as the environment is portrayed well and a human can distinctly identify certain aspects of the cloud. Note that each of these makes use of a different type of sensor fusion, SLAM algorithm, and mobile robot. At the heart of each of these algorithms is the process of scan matching, a technique to recover the relative position and orientation of two laser scans. The output of 3D laser scanner SLAM can then be compared to several other types of SLAM including monocular camera SLAM and stereo camera SLAM.

1.1 Literature Review

Vision-based SLAM systems commonly rely on visual feature identification making use of processes such as SURF or SIFT. Two standard types of vision-based SLAM will be presented below followed by a comparison to several SLAM algorithms for LiDAR.
1.1.1 Direct Monocular SLAM

For a more significant comparison (in terms of mapping size), Large-Scale Direct Monocular SLAM (LSD-SLAM) \([3]\) will be presented. This type of SLAM poses several benefits. The first of which is the very cheap cost of a monocular camera, which is the main sensor used in LSD-SLAM. This feature-based direct visual odometry (VO) uses image intensities, thus enabling the use of all information in the image. In this type of SLAM, accurate and fully dense depth maps are computed using pixel to pixel comparisons. While the sensor cost is reduced, this type of SLAM requires a state-of-the-art GPU to be able to run in real time.

This method uses a type of Graph SLAM where the global map is represented as a pose graph. As related features are captured in several frames, accumulated drift can be corrected \([4]\). Based on the Real-Time Appearance Based Mapping (RTAB-Map) memory management \([5]\), LSD-SLAM can run in real-time on a GPU. While memory management and loop closure play a large role in run time performance, the most impacting feature of a SLAM algorithm is how it detects features. Speeded of Robust features or SURF is used as the type of feature filter. This feature has a unique descriptor associated with it, which is a set of pixels that make a feature. These small sets of pixels have pixel intensities that can then be compared. Figure 1.3 displays how the features are extracted into visual words

![Figure 1.3: Pathway from Feature Detection to Visual Words. Obtained from Mathworks.](image)

that are unique words that can be associated to an image. Based on how many visual words images have in common, they can be considered in the same frame (loop closing).

This algorithm utilizes tracking, depth map estimation, and map optimization. The image tracking keeps track of new images with respect to the current frame, using the pose of the previous frame as initialization. The depth map estimation uses tracked frames to
refine the depth map. The depth map is refined by filtering over many per-pixel stereo comparisons. The last step of map optimization detects loop closures and minimizes drift. A sample point cloud output from LSD-SLAM is displayed in figure 1.4.

![Figure 1.4: LSD-SLAM Point Cloud Output](image)

On a large-scale, this generated point cloud seems to hold accuracy that is comparable to the sample point clouds in figure 1.2.

### 1.1.2 Direct Stereo Camera SLAM

This method of SLAM also has considerable benefits. Stereo cameras can output depth images, where monocular cameras must compute them. This reduces the complexity of the SLAM problem. Moreover, stereo cameras can still be found in a very low price range and with their light weight, are also suited for mobile robots. The difference between stereo and monocular SLAM is that stereo SLAM remove depth scale as a free parameter. Thus, it can be expected that this type of mapping is more fine than the previous. An even greater specialty of this method is the generation of temporal depth maps from the individual cameras to estimate translational motion with very high accuracy.

![Figure 1.5: LSD-Stereo-SLAM Point Cloud Output](image)
Again, this generated point cloud seems to hold accuracy that is comparable to the sample point clouds in figure 1.2. While both of the displayed methods seem to have high accuracy, these maps are very time consuming to generate. With sensor ranges of up to 10 meters for a stereo and monocular camera (with accuracy decreasing greatly as distance increasing), it can quickly be imagined that these point clouds took hours to generate. As this type of SLAM is dependent on light, computational power, and reference points, failure can quickly result; LiDAR SLAM solves some of these issues and allows for higher quality point clouds.

1.1.3 LiDAR-SLAM Point Clouds

Recently, LiDAR-SLAM has become very popular. As Velodyne has been releasing high-end LiDARs and the autonomous driving feat is approaching quickly, the price of LiDAR sensors have greatly decreased and high-end algorithms are being produced continuously. Leading companies such as Nvidia, Google, and Uber push the market development of these algorithms for ground-based robots, and companies like Real Earth, DJI, and Kaarta push the development of these algorithms for aerial robots. Traditional LiDAR approaches use scan-to-scan matching to compute relative pose changes [7]. This can quickly accumulate error if not accounted for within the system. Other methods use particle filters to maintain a representation of the pose. This can quickly lead to a massive amount of dimensions of the system and is unlikely perform well on the large-scale. Three different algorithms for LiDAR-SLAM will be presented below that avoid these common issues followed by a choice based on several aspects of the algorithm including performance, computational complexity, usability, and ease of deployment.

**Google Cartographer**

Cartographer is an open-source system developed by Google that provides real-time simultaneous localization and mapping in 2D and 3D environments across multiple platforms and sensor configurations [8]. This system achieves real-time mapping utilizing loop closure at 5 cm resolution. Loop closure is very important feature of traditional SLAM approach and greatly reduces noise and drift but adds much computational complexity. This topic will be
expanded on in the upcoming theory section. Deployment of this sensor is using a sensor equipped backpack with an output of generating a 2D grid map. Google Cartographer uses scan to submap scan matching and the error of the pose estimate in the world frame is expected to accumulate as the duration of the experiment increases.

To cope with the growing error, pose optimization is run at a regular interval. This pose optimization is accomplished by classifying a submap as completed and then, using it for loop closure. Thus, over time, as more submaps become complete, the pose will converge. With this type of loop closure, error will still accumulate but at a much slower rate.

![Figure 1.6: Google Cartographer 2D Contour Map Output](image)

Above is a sample of the output generated by Google Cartographer. As a summary, this type of SLAM is two-dimensional, combines scan-to-submap matching with loop closure detection and graph optimization. A microcomputer with a GPU is required to generate the finished map in real-time. Some key advantages of Cartographer is the easy data collection process (easy deployment), and can be setup for multiple sensor configurations with ease. Some disadvantages include that areas without loop closures will be very inaccurate, the output is a 2-D contour map, which when compared to a 3-D point cloud is considerably less significant, and that an inertial measurement unit is required.
HDL Graph SLAM

hd_graph_slam is an open source ROS package for real-time 3D SLAM using a 3D LiDAR. It is based on 3D graph SLAM with scan matching-based odometry estimation and loop detection [9]. It also utilizes floor plane detection to generate an environment. Thus, a moderate requirement for the algorithm is that the floor must be flat and while this is a limitation, this aspect tailors HDL graph SLAM for indoor environments.

GraphSLAM [10] solves the full or offline SLAM problem. It performs well in a non-static environment and can handle a large number of features. It uses a mixture of motion constraints and soft constraints to minimize the error of pose estimation. A few drawbacks is that this algorithm does require the environment to have a large number of features. Graph-SLAM is also prone to quickly grow in size and end with information matrices with over a million cells. As ongoing technology improves, systems can solve the Graph SLAM problem faster and thus, a large environment can be mapped with great accuracy.

This system is only allowed to work in real time with heavy downsampling. The scan received from the sensor is downsampled and than iterative scan matching is applied between consecutive frames to estimate the pose. Floors are detected using the RANSAC plane fitting method. Similar to Cartographer, the accumulated error from scan-matching is compensated by loop detection and graph optimization.

Figure 1.7: HDL_Graph_SLAM Block Diagram [9].

A sample output from this type of SLAM is shown in figure 1.8.

A point cloud of high quality for a very large area was produced. There seems to be some errors present in the data but the 2D contour is outlined well. More so, the trajectory seems to also be traced very cleanly. As this system works in real-time and without an IMU,
this is a better option than Google Cartographer. The added floor detection improves the results greatly. This said, an uneven floor may affect this experiment in a negative way. Some advantages of this algorithm is that it is open-source and well documented, works well in large-scale indoor environments, can work in real time, and functions with the Velodyne VLP-16 as the sole sensor. An added benefit is that this system has also been setup for an added GPS to ensure equal functionality in outdoor environments where there may be less features.

**LOAM: LiDAR Odometry and Mapping**

LiDAR odometry and mapping (LOAM) is an algorithm created by Ji Zhang and Sanjiv Singh at Carnegie Mellon University that is an ingenius method for real-time odometry and mapping [11],[12],[13]. It achieves low-drift and low computational complexity without the need for an IMU. The key novelty of this approach is the division of the SLAM problem into two smaller problems and thus requires two algorithms; one algorithm performs odometry at a high frequency to estimate the velocity of the LiDAR. The other runs at a much lower frequency and performs fine matching and registration of the point cloud. The most
significant aspect of this algorithm as a whole is that it does not use loop closure.

Similar to the HDL Graph SLAM method, the Velodyne can be used as the sole sensor for LOAM. This method works by looking for similar features in each successive frame and lining them up based on structure. Features of focus are shape edges, corners, and the intersection of planes. As the LiDAR has a wide FOV, some features may remain in frame for long periods of time and serve a great landmark in odometry. An additional part of this method is the correction due to motion distortion of the LiDAR as the system is moving and the LiDAR is rotating. According to Zhang, on a wide scale of 60 meters, this algorithm achieves about 1\% error. This combined with the fast computing makes this algorithm the most suitable for mobile robots. The lightness of the two sub-algorithms allows for much smaller microcomputers to be used and thus, allows for UAV deployment. More so, this system also has the capability to add an IMU to further reduce motion estimation drift.

Several universities have utilized this method for testing of different applications. [14] explores using LOAM on a UAV in indoor and outdoor settings to test the aerial performance. For indoor testing, the resolution was tested within one room where several objects were placed throughout the room; a drone then took off and moved across the room. After the mapping, the offset between the true distance between the placed objects became the error. Indoor settings achieved object dimension estimations with the errors below 6 centimeters.

[15] discusses solving the SLAM problem with focus on accounting for changes of altitude and the motion dynamics. It also takes into account the geometry of indoor environments, and discusses how the planar position of the UAV can be found efficiently from a scan matching algorithm. This paper also supports the usability of this algorithm as it focuses on how the loop closure is not necessary.

[16] discusses in depth the fusion between an inertial navigation system and a 2-D LiDAR with a focus on indoor mapping. This supports the idea that sensor errors will decrease with the merger of the IMU data using an EKF scheme. Thus, this is another precursor that a LiDAR-IMU integrated system will function well to a sub-meter accuracy.

[17] explores using LOAM on a UAV and integrating it with a camera and an IMU in an indoor setting. Consistent 3D maps were created with this system and thus, provides more
support for the choice of the LOAM algorithm. In this case, LOAM was computationally light enough for Camera SLAM to be run simultaneously alongside an EKF to combine the two estimations.

From references, results from the LOAM algorithm in several scenarios were impressive. A real-time result is displayed in figure 1.9 and the post-processed result 1.10.

Figure 1.9: Point Cloud Output from LOAM [11].

Figure 1.10: Post-processed Point Cloud Output from LOAM [11].

As both figures presented show, a large region can be mapped with great precision. Planar surfaces are mapped very well; a human can easily label different aspects of a point cloud from this image such as trees, buildings, etc. This is comparable to systems with loop closure and thus, the point clouds presented can be considered a very high performance and the LOAM algorithm is a suitable choice for mapping environments.
1.2 Theory of LiDAR Odometry and Mapping

As LOAM will be the method for SLAM used in this experiment, the theory of LOAM will be discussed here. Figure 1.11 depicts the software system and the split between the mapping and transform computation. The stages of in this block diagram are as follows: the point cloud registration refers to the points received by a laser scan, which in this step are saved into a tensor. After this the LiDAR odometry is computed assisting which both map generation and transform integration (which is a finer odometry).

Starting with feature analysis, both sub-algorithms extract feature points located on edges and planar surfaces, and merge the edge points to an edge line, and planar surfaces to planar surface patches. The geometric distributions of local point clusters have correspondences to other clusters that can be found by studying the associated eigenvalues and eigenvectors.

Plane and edge detection is done using the Random Sample Consensus (RANSAC) plane fitting method. It can be interpreted as an outlier detection method. Thus, the inliers or data that can be explained by some model parameters can be separated from the outliers. In 3 dimensions, a plane of best fit can be found, and the outliers may mark other surfaces that RANSAC can be performed on again, iteratively sectioning out all the planes in each laser scan. Precise tuning of the RANSAC algorithm is required for the best indoor quality; this will be discussed further in the parameter tuning section. After a full scan is separated into all its planes (inliers) and edges, a scan can be separated into 4 sub regions each with up to 2 edge points and 4 planar points.

After the identification of planes and edges, the iterative closest point (ICP) algorithm is used to find the transformation between frames. More specifically, Iterative closest point attempts to find the translation t and rotation R that minimizes the sum of the squared
error between two corresponding point sets.

Figure 1.12: Depiction of Iterative Closest Point [18].

Figure 1.12 displays how the ICP algorithm works. Based on a number of point clouds, the transformation can be identified and then the clouds can be fused to produce a single cloud. As it can be imagined, the higher the number of points, the more time it takes to perform this numerical computation and find the transformation between two clouds. For this to be done in real-time, the point cloud must be downsampled, using the method of voxel-grid downsampling. This method will merge each point with several points near it to greatly reduce the number of points while preserving much of the spatial information. Dr. Zhang introduces a very interesting method to further simplify the ICP task and make it more robust.

Figure 1.13: Selection of Feature and Edge Points [13].

Figure 1.13a shows the LiDAR along with two surface patches A and B. The surface
patch detected by A has some angle to the LiDAR alongside and some identifiable dimensions (plane width). Surface B is roughly parallel to the beam, and thus it is ignored and not selected as a feature point. Figure 1.13b shows two surface patches again. Point A is the boundary of an unseen region and Point B is a boundary of another surface. Both of these will be detected as edge points. As the LiDAR moves and the angle changes meaning that the occluded region of A will become available, A will then be detected as a planar point. Now instead of matching two point clouds, the transformations between sets of features and edge points can be determined. A quantitative description of the smoothness of a surface can be defined by equation 1 in [13]. At high $c$ values, points in the scan are labeled edge points. At $c$ values below the threshold, points are labeled planar points.

This greatly reduces the computation required to perform ICP. The last problem that must be solved is that the point clouds are of different time steps. This can be fixed by re-projecting the current point cloud to the next time step. Then, both point cloud $P_t$ and $P_{(t+1)}$ are of the same time step.

Figure 1.14: Reprojecting Point Cloud to the End of a Sweep [13].

Figure 1.14 shows the point cloud being reprojected onto the next time step. Then, these two point clouds can be used to estimate the motion.

The LiDAR mapping algorithm runs at a lower frequency than the odometry algorithm, and is only called once per sweep. As the pose transformation between two point clouds is known, the mapping algorithm matches and registers the point clouds in the world coordinates. As this will be done repetitively, the map will quickly grow upholding a strong accuracy due to the LiDAR odometry. Using Dr. Zhang’s methods, LOAM is more likely to converge to the true pose of the mobile robot, the true map of the environment, and be able to solve the SLAM problem in real-time.
Chapter 2

Experimental Setup-Instructional

This chapter will go over the equipment details, hardware setup, software setup, experimental procedure, and the limitations of this experiment. Recreation of the experiment should be possible after reading this section.

2.1 Equipment Details

Simultaneous Localization and Mapping (SLAM) requires for a set of sensors and connections to operate properly. One sensor in particular will be considered the main sensor and will produce data that can be simplified into features. The main sensor used in this experiment is the Velodyne VLP-16 LiDAR.

Figure 2.1: Velodyne VLP-16 LiDAR.
The Velodyne VLP-16 LiDAR, also known as the Puck, is a lightweight and high performance LiDAR created for mobile robots. With its previously mentioned field of view, and 16 spinning lasers with 100 m range and 3 cm accuracy, it is an ideal sensor for mapping. The low power consumption (8 Watts) of the Puck allows for it to be readily paired with the power distribution of the mobile robot and can easily be powered by a LiPo battery. A plate made of acrylic plastic to carry all the sensors to perform LOAM was created with the goal of placing on a mobile robot.

Figure 2.2: Sensor Plate for Mobile Robots.

Figure 2.2 shows the plate created containing the Velodyne Puck, the Velodyne interface box, an Arduino Mega 2560, a MPU-9250 Inertial Measurement Unit, a 4S 5200 mAh LiPo Battery, and a breadboard for routing connections. Note that the Puck is also elevated to ensure that the lasers have a clear view so that nothing on the plate will get sensed by the LiDAR. This plate is meant to be placed on the top of a mobile robot.

2.2 Hardware Setup

The Velodyne interface box is in charge of transducing the laser data and sending out packets through an Ethernet connection. These packets contain point cloud information including the points themselves, time of flight distance measurements, reflectivity measurements, and synchronized times steps. This box also supplies power to the LiDAR; As this box accepts a DC Voltage of 9-32, a DROK DC Voltage Regulator to output a constant 12 V from the
battery is required. More information about this can be seen in figure 2.3. Settings for the interface box can be toggled to induce special features such as dual return, output intensity, and change the spin to rates from 5-20 Hz. All of these settings are set to the default in this experiment. The output Ethernet connection from the box must attach to a laptop or a microcomputer such as a Nvidia Jetson TX2. Systems such as a Raspberry Pi may be able to run these computations but systems with higher processing units and GPU support are much better options.

The IMU while not needed assists in the LOAM algorithm by aiding in pose estimation. Since this algorithm is iterative (every 2 steps in sequence are compared), there is no drift in the pose estimation of the IMU since it is only estimating one value to the next. Using an Extended Kalman Filter to combine the pose estimation from the LiDAR odometry and the IMU allows for higher accuracy SLAM. More so, the addition of the IMU helps in the smoothing out velocity and angular velocity estimations, and allows for the mobile robot to make more erratic movements without causing failure in the SLAM process. The MPU-9250 was chosen due to its wide use and reliability. The Arduino Mega 2560 was used to interface with the IMU and output serial data to the microcomputer/laptop. The powering of the Arduino was done using the same battery and another DROK DC Voltage Regulator.

The 4S LiPo battery was chosen for its large size and capacity. While the system is not power intensive, the size of the battery can contribute to a longer period of mapping. More so, on lighter mobile robots, specifically drones, there is not much room to add multiple batteries. Thus, this same battery would be used to power the motors alongside another flight controller. A useful improvement to this setup by [14] is to stream the point data from the interface box to an off system processor through Wifi. While this places a limit on the range of the experiment, it can be useful to monitor data in real-time, and have a weight reduction due to an off-board microcomputer. The base wiring diagram for the system is shown in Figure 2.3.

The system shown contains three voltage regulators, one for the Arduino, one for the IMU, and one for the Velodyne Interface Box. The voltages that are given to each element are also labeled. It should be noted that while the interface box is protected against a short circuit, the IMU and Arduino are not. Thus, it is important to use regulators and
capacitors to avoid voltage spikes.

This plate is a very simple system where all elements are screwed down onto it. The VLP-16 is placed about 5 inches off the base of the plate. Mounting this system onto a ground robot can be done by placing it on top and then screwing the surface down. Mounting this system to a drone will require removing the sensors off the plate and reorganizing them onto the drone in an elegant fashion. Several key elements during deployment is to ensure that the LiDAR is only mapping the environment and is fairly stable.

2.3 Software Setup

This section discusses the instructions on how to set up LOAM on a Linux-Based system or a Windows-Based system. Steps considering how to install operating systems will not be discussed; These instructions are readily available online. The instructions presented below to setup the experiment can be used off a freshly installed OS.
2.3.1 Linux-Based

A Linux-Based system is any system that runs Linux as its base OS consisting of many different versions including Ubuntu, Red Hat, etc. This system and most experiments were conducted on a laptop running Ubuntu Xenial 14.04. This system was also tested on a Nvidia Jetson TX2 running Ubuntu 14.04 and there was little to no difference in setup. Here, the setup for a laptop running Ubuntu 14.04 will be discussed.

For the LiDAR sensor to communicate with the computer, there needs to be a communication platform that can facilitate a connection between multiple devices. Robot Operating System or ROS developed by Stanford University provides this service. While ROS is not an actual operating system in the traditional sense of process management and scheduling, it provides a communication layer very similar to those of standard operating systems. ROS is a widely used robotics development application where much software is open source and free to use. This includes several drivers, tools, libraries, and conventions that simplify the task of interacting with robotics. After Linux has been configured, ROS can be configured by first downloading it, than creating a working directory, and then installing the necessary packages. ROS.org specifies the steps for installing ROS very thoroughly. Go to http://wiki.ros.org/indigo/Installation/Ubuntu and follow the installation steps. Each of the commands listed must be entered into the terminal and ran sequentially. It is recommended to run a full Desktop Install so several of the precursors for LOAM are preinstalled. After this, it is recommended for the user to go through the https://wiki.ros.org/ROS/Tutorials to become familiar with the interface and commands.

After this, a working ROS directory can be created. This is a folder where all the ROS packages are linked to. Usually, this workspace is called catkin_ws. The code for initializing it is shown below.

```bash
mkdir -p ~/catkin_ws/src
```
```bash
cd ~/catkin_ws/
```
```bash
 catkin_make
```

Some of this syntax may be reviewed by searching ”terminal syntax” on Google and
clicking any of the top search results. The catkin workspace is a folder where the user modifies, builds, and installs catkin packages. As packages are compiled, the terminal will return feedback on the processes that are run. At this point the workspace is setup. It should be noted that ROS should be sourced at this step to ensure that all environment variables are added to the path and allows ROS to function properly. Alternatively, the command

```
source /devel/setup.bash
```

can be added to the bash, which will ensure every new terminal is setup for ROS.

**Package Information**

Several ROS packages must be installed for the whole SLAM system to work. The names of this packages are gesture_teleop, Velodyne_master, pcl_ros, rosserial, and loam_velodyne. These must be installed into the catkin workspace. rosserial and gesture_teleop work to interface the IMU with the Arduino and ROS. This means an Arduino ROS node will be used to acquire and publish sensor values to a ROS environment, and then other nodes can process it. The main communication between the PC and Arduino happens over UART. There is a dedicated protocol called ROS Serial, which can encode and decode ROS Serial messages. To install rosserial, use the command

```
sudo apt-get install ros-indigo-rosserial
sudo apt-get install ros-indigo-rosserial-arduino
```

These commands help to install the rosserial-arduino client package in ROS. Using this library, Arduino ROS nodes can be created that work like normal ROS nodes. After this, the Arduino IDE can be downloaded from [https://www.arduino.cc/en/Main/Software](https://www.arduino.cc/en/Main/Software). Choose the Linux 64/32-bit according to the OS configuration, and then, the command "arduno" can be run in the terminal. By default, the Arduino libraries folder will be located in /home/Arduino with the name "libraries"; here, all the header files for each package will be stored. Running the command,

```
rosrun rosserial_arduino make_libraries.py
```
will generate the ros_lib folder which will contain examples on ROS communication and allow for ROS communication. Any of the examples can now be run. To subscribe or publish to the Arduino board, the ROS serial server on the PC must be started. To do this, run these commands in the terminal

```
roscore
rosrun rosserial_python serial_node.py /dev/ttyACM0
```

The argument of the second command may have to change based on the port the Arduino is connected to. At this point, ROS and the Arduino should be communicating and running commands to view the ROS nodes or topic should confirm this.

To connect to the IMU, the MPU-9250 library must be downloaded. After moving the Arduino libraries folder mentioned previously in the terminal, the command

```
git clone https://github.com/sparkfun/MPU-9150_Breakout/tree/master/firmware
```

can be run to download the library for working with the IMU. Note that this library is meant for the MPU_9150 but works for the MPU_9250 as well. [19] provides some code for this IMU. At this point, this code should compile but not output any correct data.

![Flowchart of Arduino-ROS Node](image)

**Figure 2.4: Flowchart of Arduino-ROS Node [19].**

The flowchart in 2.4 depicts what the complete code will do. Later on, this will be configured to feed data into the LOAM package. More information about this code can be
found in [19]. After wiring the Arduino based on the wiring diagram given in figure 2.3, the commands to initialize the ROS serial server given above can be run in the terminal. To ensure that the IMU is working and outputting the correct data, the below topics should be displayed.

![Figure 2.5: Topics Displayed While Running the MPU-9250.](image)

Visualizing the IMU data and odometry can be done by running the command.

```bash
rosrun rviz rviz -f base_link
```

The next step is to convert the IMU data which is pitch, roll, and yaw into twist messages. This message type information can be viewed on [http://docs.ros.org/api/geometry_msgs/html/msg/Twist.html](http://docs.ros.org/api/geometry_msgs/html/msg/Twist.html). Twist messages contain the linear and angular velocity and can be integrated for odometry. This is the message type that the LOAM package accepts. [19] supplies this package alongside a python script that performs this conversion. It creates an Arduino-ROS node that is receiving IMU values and publishing the yaw, pitch, and roll as well as the transformation corresponding to the IMU movement as ROS topics. An explanation about each of these lines is given in [19]. This package comes with a launch file called gesture_teleop.launch that will launch this whole system. The gesture_teleop node will output the linear and angular velocities. This node name must be remapped for the LOAM package which inputs the name /imu/data. To do this, add the command

```xml
<remap from = "gesture_teleop" to = "/imu/data" />
```

Then to launch the program, type the command below into the terminal.

```bash
roslaunch gesture_teleop gesture_teleop.launch
```

After this, the packages Velodyne-master and pcl-ros can be installed by visiting ROS.org and manually downloading the packages from source and following the instructions given
there. The Velodyne-master package can be downloaded on [https://github.com/ros-drivers/velodyne](https://github.com/ros-drivers/velodyne) and then unzipped into the src folder inside the catkin workspace. The pcl_ros package can be found on [https://github.com/ros-perception/perception_pcl/tree/indigo-devel](https://github.com/ros-perception/perception_pcl/tree/indigo-devel) and unzipped into the same src folder. The package LOAM, is not available on ROS as the company Kaarta has patented that information. Several old copies are available; the base version used in this experiment can be obtained on [https://github.com/laboshinl/loam_velodyne](https://github.com/laboshinl/loam_velodyne). After this, the folder must be compiled to ensure there are no errors and all the dependencies are satisfied. Compiling can be done by running the commands,

```bash
cd ~/catkin_ws
catkin_make
```

This may take several minutes to complete. At this point, the terminal should be sourced again, and then LOAM is ready for use.

A set of commands that is extremely important is located in the ROSbag package. These commands contain tools for recording and playing back data from ROS topics. Thus, using this tactic, entire experiments can be saved. With this, the mapping process can be played back to the user post experiment. The user can subscribe to any published topics such as the `/velodyne_cloud_registered` topic that displays the current mapped point cloud. Commands such as `rosbag record` or `rosbag play` will record and play back the topics, respectively. Using a recorded file or a real-time feed, RVIZ can be used to view the point cloud. The LOAM folder contains a preconfigured RVIZ setup. A command line summary of the commands that must be run in order for this package to began are show below. Note, that everything must be wired properly and slight changes in arguments may have to be made based on the microcomputer. A common issue faced is that the velodyne interface box needs the ip address configured. The command ”`ifconfig`” can be use to configure this properly if an error is displayed.

```bash
roscore
roslaunch gesture_telelop gesture_teleop.launch
roslaunch velodyne_pointcloud VLP16_points.launch
```
roslaunch loam_velodyne hector_loam_velodyne.launch
rosbag record -o map /velodyne_cloud_registered

Each of these launch files must be run in a different terminal. The windows that will be displayed are shown below in figure 2.6. As these commands are run, warnings may be shown in yellow. In most cases, this can be ignored as they are just warnings. Errors shown in red should be taken into account and they will likely lead to experiment failure. From here, Rviz can be configured to display different things. During the experimentation, the static point cloud can be displayed, the generated map can be displayed, or other variant point cloud containing different features can be displayed. Note that the fixed frame is set to camera_init; this is just denoting the original system. Some of the odometry information is labeled by "camera" which may be misleading. Also, setting the point display to flat squares greatly helps the display quality of the cloud. After the points have been recorded, the Point Cloud Library (PCL) can be used to convert the bag file into a .pcd or .ply (point cloud) file. This will allow for the point cloud to be opened in the powerful software known as CloudCompare. An example of how to convert the bag file to a PCD file is shown below.

rosrun pcl_ros bag_to_pcd ~/catkin_ws/map /velodyne_cloud_registered
~/Desktop/Pointcloud1
ROS should return some feedback displaying the conversion taking place. Depending on the length of the bag file, multiple PCD files may be created, representing different regions of the map. Each of these files contain about 28,000 points and thus, for point cloud sizes containing 3,000,000 points (which can be generated in approximately 5 minutes), approximately 100 files will be generated. These can be imported simultaneously into CloudCompare and will line up as to display the same cloud seen in the Rviz display. Note that CloudCompare runs on Linux but does not have the PCD extension to add these point clouds. Thus, these files must be transferred over using a flash drive to a Windows OS or Mac OS computer. To upload these files to CloudCompare, use the add feature and select all the PCD files. After this, each of these files must be selected and then, must be merged together using the merge tool. Further use of the CloudCompare software will be discussed in the post-processing section.

**Parameter Tuning**

LOAM has many parameters that can be changed to improve map quality and change the performance. Many of these are combinations of changes from the github community (large amount of assistance and help from Yoshua Nava) that must be made to the base loam package. General changes include outputting much more progress reports to the terminal, printing out general messages, and commenting lines to add structure inside the package. The full package can be found on [https://github.com/rohanpaleja27/loam](https://github.com/rohanpaleja27/loam). The first large change comes by adding two libraries that greatly speed up computation: Eigen and Transform. Eigen contains several classes that contain new variable types that are great for computing subsequent transformations. Some methods have been replaced to include these new libraries. After this, several changes to parameters were made that helped with speed and performance. The first change was to change a parameter that made the point cloud publish less often. Thus, less computational effort is needed in each iteration. This makes the system more robust, as less data must be processed. Results show that the odometry input is better overall. After this, variables that detect divergence in the ICP process had
their thresholds lowered meaning less output would come to the screen, but outputs of the ICP will have much better quality. The number of iterations on the data has also increased, and thus, the system is more robust to turns.

**ROS Nodes**

As defined in the ROS wiki, a ROS node is an executable that can communicate with other nodes. Nodes can publish or subscribe to a topic to receive messages containing sensor data, outputs of an algorithm, etc. Topics are buses in which nodes exchange messages. Thus, if two nodes want to communicate, one would publish to a topic, and the other would subscribe to that same topic.

Thus, after launching the several launch files displayed above that launch and read IMU data, launch the velodyne driver, and launch the LOAM algorithm, nodes corresponding to each file will be initialized and begin publishing and reading to/from topics as they are programmed to do. Figure 2.7 depicting all the nodes and topics and their communications.

![Figure 2.7: Depiction of all ROS Nodes during Experimentation.](image)

Figure 2.7 displays topics as rectangles and nodes in circles. Titles of the package/node-manager will be displayed at the top of rectangles portraying that those topics or nodes are managed by this package. 28 topics and nodes are launched and required during this experiment. Some important ones will be discussed as follows.

Significant topics: the /imu/data topic contains unfiltered data regarding the roll, pitch, and yaw of the sensor placed on the PLA plate. Several topics display the Velodyne cloud, each displaying different types of information. /laser_cloud_flat and /laser_cloud_lessFlat
display the point cloud after the LOAM algorithm has performed feature recognition on it. Each of these depict the point cloud, while the non-plane data is filtered out. The less flat version has a lower threshold allowing for much more of the original point cloud to be filtered through. The `/velodyne_points` contains the cloud that is being streamed directly from the Velodyne VLP-16. The `/laser_cloud_registered` topic contains the full output of the experiment.

 Significant nodes: The `velodyne_nodelet_manager` is in charge of publishing the LiDAR data as its streamed in. This node is in charge of turning the packets (bare data) into usable point cloud data through the velodyne drivers. After this, the `/multiScanRegistration` node allows for handling new incoming clouds and registering them to assist in the upcoming steps of the LOAM cloud. The next node in the `laserOdometry` package named `/laserOdometry` is in charge of computing the laser odometry which outputs cloud features that are broken into deeper components. After this, the `/laserMapping` node is in charge of receiving the previously stated clouds, and combining the IMU data, to produce a final output of a high accuracy point cloud. The last node of `/transformMaintenance` is in charge of keeping track of the movement of the LiDAR.

 Each of these packages are split up in a way that each package handles a separate part of the LOAM process. The data is then combined using nodes that run scripts and a map is produced that can be viewed on Rviz and exported to CloudCompare.

### 2.3.2 Windows-Based

The windows-based setup uses the VeloView program to record LiDAR data, and the Kaarta online interface to stitch this recording together using LOAM. Any recent version of windows (7,8,10) can be used; Veloview is a program created by ParaView, a partner of Velodyne for easily streaming and viewing LiDAR data in real-time. It also displays distant measurements from the LiDAR as point cloud data, and supports custom color maps of multiple variables such as intensity, time, distance, azimuth, dual return, and laser id. This must be downloaded on [https://www.paraview.org/veloview](https://www.paraview.org/veloview). Point clouds can be recorded with the click on a button and stored as a .pcap file. This software also has the ability to also record GPS and IMU. A visualization of the VeloView interface is shown in figure 2.8.
VeloView is also supported on Mac and Linux and can be used, but multiple users have noted more driver issues and less features on those operating systems. Kaarta is a company that creates high quality 3-D models given point cloud data, created by the author of [13] Ji Zhang. Kaarta uses the same LOAM algorithm model, but it is much more highly tuned, and done in post-processing (takes about 1 hour to receive the stitched point cloud). Kaarta Cloud, more specifically, is a service where an uploaded point cloud is stitched together to produce a high quality .ply file. A Kaarta employee then looks at the data and analyzes it to give tips on improving data based on the application which can be highly helpful. After the point cloud has been recorded, uploads.kaarta.com provides an interface for uploading the .pcap file very easily. While Kaarta is a paid service, they offer several free uploads, and are even more lenient for its use in academia. While Kaarta accepts GPS and IMU data as well, the company is known for and has stated that high quality point clouds can perform without the need for this data due to their exceptional LiDAR odometry. While
not including this data may pose some constraints on the experiment, the computational complexity of this approach is certainly the least. The microcomputer has the simple role of saving point cloud data streamed from the Velodyne sensor. While this means that the LiDAR data cannot be used for navigation and avoidance (since maps are generated post-experiment), using LiDAR data for navigation and obstacle avoidance is generally much more expensive (both literally and computationally) than using a simple camera and a light or any other traditional method. The limitations induced by the lack of IMU data will be discussed more in the later subsection regarding limitations during the experiment. The outputs of Kaarta will be displayed in the results section.

2.4 Experimental Procedure

The primary indoor testing was conducted inside the Applied Fluids Laboratory (Eng. D124) at Rutgers University and the Eng. Laboratory Hallway. The sensor plate depicted in figure 2.2 is carried around the laboratory by a researcher while the Velodyne interface box is connected to a laptop. A depiction of this is shown in figure 2.9. The student carries the LiDAR far enough above his or her head to make sure that he or she is not being mapped during the experiment. After the LOAM algorithm is launched (Linux) or VeloView has begun recording (Windows), he or she slowly pitch and rolls the LiDAR before beginning movement. Then, the student proceeds to walk around the laboratory in a predefined path (based on the variable that will be tested) and makes sure to move as a mobile robot would do. In the case of a UAV, this experiment attempted to exhibit characteristics of a drone flying in altitude hold mode. This means that there will be very slight fluctuations in altitude. More so, this simulated a drone that would always fly front facing and move forward; the movements in the yaw direction were free while the pitch and roll directions had minimal motion. Overall, the experimentation conducted can best represent a drone moving at a gentle velocity in altitude hold mode.

Ground robots exhibit many of the same characteristics. When moving on a surface, the chassis of a ground vehicle tends to vary very little in acceleration; the robot usually moves forward facing and its pitch and roll are minimal. In general, the yaw motion is free and is very smooth. Thus, it can be theorized that this experiment represents a
general class of mobile robots fairly well. Some suggestions for data collection learned through experimentation and given by Kaarta include make sure to mount the sensor high enough to avoid sensor occlusion of the body, robot motion should be slow and smooth (Kaarta suggests under 20 mph), traversing the environment once slowly is better than performing multiple runs quickly, be careful of environment with very little distinct structure as odometry errors will arise.

The procedure for mapping the Eng. hallway was a slight variation of the method used in the laboratory. As this experiment was much longer, a cart was used to ease the strain carried on by the user shown in figure 2.10. In this experiment, a portable power system is connected to the Velodyne Puck to supply power. This is used to withhold constant power in a much longer experiment, and allow for multiple trials without recharging. This Velodyne Puck is held high (about 8 feet of the ground) while the cart is pushed around by another fellow researcher. The explanation of how this experiment mimics a mobile robot are the same.
The last step of experimental procedure is to assist in the point cloud analysis. To test point cloud accuracy, it must be compared to the actual map. While a standard layout may be available, exact dimensions of furniture and their distances to other furniture would take weeks to measure. A more clever method inspired by Derek Wolfe [14] is to place several objects around the mapping area with set distances and then compare the true distances to those of the point cloud (which can be measured using CloudCompare). Splitting the mapped trajectory into five or six regions each having two distinct objects placed a set distance apart, finding the percent error between the true distance and measured, and then averaging those percent errors allows users to find the true accuracy of LOAM.

2.5 Limitations

There are several limitations to this experiment that restrict experimental procedure and if not followed, can drastically change the accuracy of the generated point cloud. This section is developed with assistance from David Duggins from Kaarta, and [11] and [12]
by Dr. Zhang. As indoor environments tend to have symmetrical structure and the walls greatly constrain the view of the LiDAR, the rate of error growth after the odometry fails increases exponentially. A common misconception is that indoor environments have tons of features, and thus are easier to map while outdoor environments have less and thus, are harder. In the case of LiDARs, a 100m range is greatly shortened by the walls. Thus, while in outdoor environments, the LiDAR has a 100 m by 100 m range to look for features in indoor environments, the range is limited by how narrow the walls are. Due to this, David Duggins suggests that LOAM will begin to fail within the region of 15 m by 15 m for any office or hallway. As indoor environments become more structured and larger (such as theaters, school corridors, warehouses, and grocery stores), this number will grow. At a point, if the environment is large enough, there will be virtually no drift due to the number of features detected in each sweep. With the addition of the IMU as in the Linux-setup, this number grows greatly due to the odometry estimation becoming much more accurate as the IMU information is merged.

Turning rates (yaw rates) should also be limited to under 30 degrees/sec and even lower if the sensor is mapping at a tilt. Going above this value will lead to multiple planes in a point cloud being remapped and deteriorates the quality of the point cloud. If the pitch and roll are unconstrained (which is not true in this experimentation), tilting and turning should not be done at the same time. The limitations of the placement angle of the LiDAR can be displayed in figure 2.11a and 2.11b. Some limitations also arise from the sensor choice of
LiDAR. Sun glare when the sun shines directly or is reflected into the LiDAR can cause false points and give errors in the point cloud. While this can be easily identified and removed, this problem is not very impacting in closed indoor environments. Thus, areas near windows should be proceeded with caution or experiments should be conducted on overcast days. Reflective surfaces pose an issue with LiDARs. The scanner cannot distinguish between in-line and reflected data. Thus, LOAM will try to match the reflection with the actual scan data from that side of the room. Measures can be taken to clean up the data before LOAM produces the point cloud to ensure quality.
Chapter 3

Results

This chapter will go over the quality of each point cloud generated focusing on usability of the point cloud. The location and length of the experiment alongside the point cloud details and accuracy will be assessed. Accuracy of the point cloud is assessed through methods described in the experimental procedure chapter. To reiterate, the LiDAR is being carried at an elevation of 7.5 feet moving at a smooth velocity (both linear and angular). The goal of this experiment is to have this accuracy within one magnitude of the LiDAR’s resolution. Since the resolution is at 3 cm, the goal of this experiment is to have map resolution under 30 cm.

The two methods compared (Linux-Based and Kaarta-based) serve as a comparison between using an IMU alongside a self-tuned algorithm, versus the same SLAM process without an IMU but with high tuning, and much greater computing power (as this is done post-experiment).

3.1 Linux-Based LOAM

Using LOAM on Linux requires following a special and much more difficult procedure when compared to a Windows setup, as shown in the previous sections. To explain the process with brevity, each point cloud was produced by running the LOAM package on a laptop and conducting the experiment (moving along some trajectory with the LiDAR). After this, the recorded bag file was turned into a .pcd file, and then opened in CloudCompare for viewing. Very similar to the upcoming section, Kaarta-Based LOAM, the Applied Fluids Lab is mapped while traveling through the lab. It is then mapped again, while traversing the trajectory (going to the end of the lab and back). After this, the Eng. D-Hallway will be mapped.
While looking at the results, it is important to keep in mind that this system does involve an IMU, and uses this to make the odometry more accurate. Lastly, point clouds regarding the Applied Fluids Lab cannot be compared qualitatively between the Linux, and Windows version as renovations were performed between the experiment, changing some features of the experiment. However, the quantitative measurements are available and comparable as they make use of objects that remained through the renovation. It may also be noted that chronologically, the Kaarta-LOAM experiments and analysis were conducted earlier.

3.1.1 Mapping the Applied Fluids Laboratory, Moving to a Goal Position

The SLAM done in this experiment was done in D124, the Applied Fluids Laboratory using the Linux-LOAM method. Angular information using the current yaw, pitch, and roll of the IMU was used to assist the LiDAR in odometry. More specifically, it allowed for more influid movement, and faster/jerky turns.

Multiple runs are conducted for mapping the laboratory. The best of those experiments will be displayed below, but the accuracies presented are of an average trial. While the trajectories of the experiment are not shown in the point cloud, a sample image of the laboratory is shown in figure 3.1 that displays the trajectory alongside major objects that are measured to determine accuracy. This trajectory would allow the entire lab to be

![Image showing trajectory and important objects in D123.](image.png)

Figure 3.1: Image Visualizing Trajectory Experiment Trajectory and Important Objects in D123.
This experiment took about 15 seconds and generated 167 .pcd files totaling to 144 MB. In comparison, this is much heavier than the same mapping done using the Kaarta method. This point cloud contained 4,676,372 points. A display of the cloud contoured is shown in figure 3.2.

![LOAM Generated Point Cloud Contoured, Moving to Goal](image)

The color scheme used in cloud compare works to display objects in the best way for visualization. It displays objects of different heights in different colors, but as this is being viewed from a top view, color scheme does not provide much information. Thus, color scheme information other than that it plays a role in visual aesthetics should not play a large role in the qualitative assessment of this point cloud. The point cloud shown in figure 3.2 is contoured using CloudCompare to display the features of the laboratory better (the ceiling blocks some aspects of the cloud). Several features of this point cloud determine it to be of high quality; this includes the maintenance of the nearly perfect rectangular shape of the laboratory, the ease of identification of objects such as the planes hanging on the ceiling and structures in the room, a clear distinction between the free space and occupied space in the room. If a robot were to be given this information, high precision tasks could
be performed (disregarding dynamic properties in the environment). If one wishes to more closely examine this point cloud, or view a video containing different views of the point cloud, please access [https://github.com/rohanpaleja27](https://github.com/rohanpaleja27). A display taken from the video is shown in figure 3.3.

As both figure 3.3 and figure 3.2 show, several regions of the point cloud are less dense than others. This can be explained by the LOAM algorithm and experiment process. LOAM places high priority on plane and edge features, and thus, these are mapped with the highest quality. Objects inside the point cloud that have very high curvature are mapped poorly and will show a much less dense cloud. Partial regions of the laboratory are also blocked off by objects and may not be mapped to the fullest extent. If mapping were conducted longer, or if the resolution of LOAM was increased (note there will be a trade-off in computational complexity), these areas are likely to be more dense and the cloud will grow in quality.
3.1.2 Mapping the Applied Fluids Laboratory, Moving to a Goal Position and Back

Traversing an environment multiple times can give a robot a higher confidence of the environment. If there is drift or odometry error, traversing multiple times can lead to a decrease in point cloud quality. This experiment can be visualized by looking at figure 3.1 and imagining starting at one end of the trajectory, going to the other end, and coming back. This experiment had a duration of about 30 seconds and the generated point cloud had 8,022,251 points (288 files with a size of 248 MB). The result of this experiment is shown in figure 3.4. Note that the color scheme has changed; to restate, this should not play a role in the comparison between the point clouds in this subsection and the previous.

Figure 3.4: LOAM Generated Point Cloud Contoured, Moving to Goal and Back.

While an immediate response to a comparison of this image and figure 3.2 might generate the response that this point cloud is of low quality because it looks messier, the messiness is actually portraying an increase in definition. This is portrayed by comparing figures 3.3 and 3.5; the walls of the second point cloud are much sharper alongside the features of the desk space in the laboratory. The shelf in the corner of D124 (also displayed in figure ??) is
nearly perfect; if a robot were to have this cloud, even finer operations could be conducted. A slight overestimation may be stated that if this map was combined with RGB data, high-quality object recognition could be performed. The quantitative results in the upcoming section will support the previous statements.

In several areas of the point clouds displayed, there are several lines in parallel shown throughout the map; these refer to the ground. While a full conclusion cannot be made as to why the ground in only certain areas were mapped, the best conclusion is that those areas contain ground that is patterned. In general, the noise in this experiment is very minimal due to the filters installed in the LOAM package and the filters designed in the Velodyne interface box (in charge of transducing the data).

### 3.1.3 Comparison Between Single Path and Traversed Path

This section serves as a qualitative comparison between the point clouds displayed in the last two subsections. Both the experiments were conducted using the exact same procedure and were conducted on the same day. By comparing figures 3.2 and 3.3 to figures 3.4 and 3.5, it is found that both clouds are of high quality. There is no visible loss of odometry and
objects are easily recognizable. As previously stated, if the point cloud is of high quality, the trajectory tracking and thus, pose tracking of the robot can be assumed of high quality.

The second point cloud (goal and back) has a much higher point cloud density (nearly double) and much more definition when it comes to individual objects throughout the laboratory. Doubling the amount of points in an object greatly expands on one's ability to recognize it (can be thought of as related to an order higher than linear). Thus, both experiments produce usable point clouds with the second experiment producing higher quality results.

3.1.4 Mapping the Eng. Hallway

The goal of mapping the Eng. hallway was to estimate the drift in this experiment. As this experiment was conducted after the Kaarta-LOAM, this experiment mainly focused on how accurate one could get mapping of the hallway. This differed from the Kaarta section where both initial conditions and accuracy were considered. In difference with the accuracy measurements of the Kaarta LOAM section, the drift measurement of the hallway is done by using the CloudCompare point picker tool and measuring the difference in elevation between the ground planes of each side of the hallway. This was done at the top, middle, and bottom of the hallway (comparing left and right), and a range of error was computed. The worse of the three should be used as the true error, but the range will be listed in the accuracy to portray how the error changes throughout the experiment. Figure 3.6 displays the three sets of points used to find the previously stated range. Note that this experiment starts at location

![Diagram of the Eng. Hallway](image)
2 and loops till it reaches that area again. It is expected that locations 1 and 3 have similar error, and that location 2 will have the largest error (as the two measurement points are the farthest apart). Location 2 was chosen as the start point because this area had the highest amount of features initially and this proved to be an important aspect in map quality as shown in the Kaarta experimentation.

Some aspects of this experiment that differ from the experiment in D124 include a higher translating and turning speed, the presence of movement of people in the hallway, and the difference in the general environment (less features). One aspect of this experiment that may have affected the results (in a negative way) is the vibrations induced by the cart traveling on bumpy terrain. This led to both the Velodyne sensor and the IMU vibrating in several regions around the hallway. This experiment was conducted twice in the same manner; the experiment duration was approximately 180 seconds (including startup time) and 150 seconds. Both point cloud results will be presented simultaneously. For brevity, the first experiment in the hallway will be labeled hallway 1 and the second is labeled hallway 2. Hallway 1 contained 577 pcd files totaling 484 MB. Hallway 2 contained 565 files totaling 475 MB; these are very large file types and can portray the large sizes of point clouds in short durations. The results are displayed in figure 3.7. As both experiments are the same, the results displayed in this figure are expected to be very similar. Qualitatively, both of these clouds are of very high quality. Anyone familiar with the environment of the Rutgers D-Wing can easily state that this represents the hallway and can name different parts of this image. As a short survey, several students in the Applied Fluids Laboratory were asked to match each of these rooms with the true names of the rooms, and this was easily accomplished; about 8 out of 11 rooms in this image could be named from memory, and after taking a tour of the hall, each room was matched with its respective counterpart in the point clouds. A zoomed in view at an angle gives a deeper view of the quality of the point cloud as shown in figure 3.8. Several parts of this image depict the high quality. First, the separation between empty space and occupied space. Next, the stair case in the left region of the image is very easy to identify. After this, many windows and doors can be seen (depicted by rectangular holes in planes).
3.1.5 Discussion on Point Cloud Noise and Error using Linux-Based LOAM

Noise is very hard to distinguish from point cloud sparsity. As the algorithm and the interface box are set to reduce the amount of noise, majority of the data presented should be considered as noiseless. Regions where points are sparse should be considered those that have not been mapped clearly. In real-time, if one would like to reduce the amount of noise (even though it is minimal), the best way to accomplish this is by increasing the threshold for feature detection. This will cause a decrease in the number of points generated in the cloud, but everything displayed will have high confidence. CloudCompare also has some noise filters but this will be discussed in the Kaarta-LOAM section as this is done post-experiment and Linux-LOAM’s main use is its real time capabilities.

3.1.6 Analysis of Accuracy

The analysis of accuracy will be done on an average performance trial of mapping the Applied Fluids laboratory for the experiment where the LiDAR went from start to goal,
start to goal and back, and then on the two hallway experiments, hallway 1 and hallway 2. The quantitative analysis should serve to assist with the statements in the previous sections.

**Accuracy of the Applied Fluids Laboratory Point Clouds**

To measure the accuracy of the point clouds shown in figures 3.2 and 3.4, 6 aspects of each cloud will be measured in CloudCompare and compared to the real life distances. 2 of these aspects will test the accuracy in the x-axis, 2 in the y-axis, and 2 in the z-axis; as these objects are also spread out through the mapping process, together these aspects determine the true accuracy of the point cloud. The objects labeled in figure 3.1 depicts all the objects that will be measured and their general locations. The 6 aspects are as follows: the distance from the ground plane to plane-1, the ground plane to plane-2, the horizontal distance from shelf to desk, the width of the solder station, the length of the plane (nose to tail fin) and table length. The true measurements of each of these was achieved using a tape measure. Thus, slight human error should be taken into account when considering this experiment. Slight inaccuracy from selecting a different measurement point in CloudCompare to the exact real life point of measurement may also exist and produce some error. This is also considered to be minimal.

The goal of the experiment was to achieve accuracy within one magnitude of sensor
resolution meaning under 30 cm. Tables are presented below that depicts the accuracy of both experiments done in the Applied Fluids Laboratory.

<table>
<thead>
<tr>
<th></th>
<th>Ground Plane to Plane-1</th>
<th>Ground Plane to Plane-2</th>
<th>Horizontal Distance from Shelf to Desk</th>
<th>Width of Solder Station</th>
<th>Length of Plane (Nose to Tail Fin)</th>
<th>Table Length</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>2.844 m</td>
<td>2.571 m</td>
<td>1.447 m</td>
<td>1.778 m</td>
<td>1.575 m</td>
<td>1.365 m</td>
<td>*</td>
</tr>
<tr>
<td>Measured</td>
<td>2.911 m</td>
<td>2.484 m</td>
<td>1.375 m</td>
<td>1.815 m</td>
<td>1.502 m</td>
<td>1.403 m</td>
<td>*</td>
</tr>
<tr>
<td>Percent Error</td>
<td>2.355%</td>
<td>3.384%</td>
<td>4.976%</td>
<td>2.081%</td>
<td>4.635%</td>
<td>2.784%</td>
<td>3.369%</td>
</tr>
<tr>
<td>Absolute Error</td>
<td>0.067 m</td>
<td>0.087 m</td>
<td>0.072 m</td>
<td>0.037 m</td>
<td>0.073 m</td>
<td>0.038 m</td>
<td>0.051 m</td>
</tr>
<tr>
<td>% Allowable Error Relative to 30 cm</td>
<td>22.33%</td>
<td>29.00%</td>
<td>24.00%</td>
<td>12.33%</td>
<td>24.33%</td>
<td>12.67%</td>
<td>17.00%</td>
</tr>
</tbody>
</table>

Table 3.1: Actual and Measured Distance to Objects in the Point Cloud, Traveling to Goal

Comparing tables 3.1 and 3.2 a significant improvement comes from tracing back along a trajectory. Tracing back the experiment led to most accuracies within the bounds of accuracy of a static sensor. Both experiments are well within the bound of error of 30 centimeters. The first experiment has a cloud-to-world accuracy of 96.631% and the second experiment has a cloud-to-world accuracy of 99.71%. This reiterates the high quality of the point cloud and its usability. This also reiterates the point that even without loop closure, looping back on the data, which increases point density, leads to higher quality data and objects mapped more accurately.
Table 3.2: Actual and Measured Distance to Objects in the Point Cloud, Traveling to Goal and Back

Accuracy of the Eng. Hallway Point Clouds

As stated in [11], LOAM is expected to drift in very large areas. Without loop closure, this is to be expected; the hallway experiments are to test this drift. The method for finding accuracy is stated in the previous section and is done by comparing the height of the floor in 3 different regions. For a true hallway, the floor of the hallway should all be of the same elevation. 3.3 depicts that the hallway mapping was done with very little drift. As this was a region of low features, a region of higher features should have less drift. Thus, an estimation can be generated that if in a low feature region, approximately 1 meter of drift is generated for every 100 meter traveled, large experiments can be conducted producing clouds of high usability and quality.
Table 3.3: Hallway Experiments Accuracy

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Experiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Drift Range</td>
<td>0.15-.70 m</td>
</tr>
<tr>
<td>Percent</td>
<td>0.14-0.67%</td>
</tr>
<tr>
<td>Error</td>
<td></td>
</tr>
<tr>
<td>Compared</td>
<td></td>
</tr>
<tr>
<td>to</td>
<td></td>
</tr>
<tr>
<td>Hallway</td>
<td></td>
</tr>
<tr>
<td>Length of</td>
<td></td>
</tr>
<tr>
<td>104.08 m</td>
<td></td>
</tr>
</tbody>
</table>

3.2 Kaarta-Based LOAM

Each of the point cloud displayed below were developed by recording a .pcap file using VeloView and uploading this file to the Kaarta server. First, the Applied Fluids Lab is mapped while traveling through the lab. Then, the same Lab will be mapped going the same trajectory forward and back. Lastly, the Eng. D-Hallway will be mapped.

3.2.1 Mapping the Applied Fluids Laboratory, Moving to a Goal Position

The SLAM done in this experiment was done in the Applied Fluids Laboratory using the LOAM-Kaarta method. This laboratory is about 15 m x 15 m. Note that the LOAM-Kaarta method meant that there is no IMU attached and all the processing is done post experiment. Several runs of the same experiment were conducted; the best of those experiments will be displayed here but the accuracy will be an of an average quality run. This experiment to move to the goal position (which was determined by choosing a position where the path to that point would allow for the entire lab to be mapped) took about 15 seconds. This generated about a 34 Megabyte file that contains 3,049,345 points. A display of the generated cloud is shown below.

The base point cloud from a top view is a little hard to match with the real environment. Thus, the total quality of the point cloud cannot be determined by this image alone.
Though, from this image, some inaccuracies at the border of the cloud can be seen. An office wall outside the laboratory is also mapped through the door window. Furthermore, in comparison with other sensor SLAM methods, generating a cloud with this type of accuracy and density is impossible in 15 seconds. The speed at which LiDARs can map outperforms almost all sensors available. Several filters can be applied to this including filters to reduce noise and other smoothing filters; this will be discussed in the post-processing section. The ceiling of the point cloud can be cut using a simple segmentation feature, to provide a new point cloud shown in figure 3.10. Note that performing cuts, or any filters, causes in a large loss of points and thus, loss of information. While these filters are designed to withhold the majority of information, some information will always be lost. This contour was chosen to cut out most of the ceiling features. The remaining point cloud recognizes several features of the environment very well. To reiterate, color scheme should not play a role in qualitative analysis of the point cloud. A deeper view of this environment can be shown through figure 3.11. Several objects can be easily identified such as the plane hanging on the ceiling, tables around the laboratory, and the general structure of the laboratory. To repeat, the definition of a point cloud with high performance and good quality is one where a human is able to easily identify different portions of the laboratory. With this map, any future
mobile robot should be able to perform high performance localization. As the trajectory shows, much of the lab has not been explored and thus, regions that are more accurate and less accurate (bottom right quadrant) are displayed in the map. If a longer and more exploratory route was taken, the map would have much higher quality in the inaccurate areas and be more dense and complete. While there are 1000s of combinations of variations of the base experiment that can be conducted in this lab (examples include changing velocity of mapping, or height), here only one type of experiment has been conducted to test the viability of indoor mapping.

Both representations of the cloud above represent a 2-D contour as they are viewed from a top view. To grasp the full use of the point cloud, an angled view can be presented to highlight the height features. The angled view of the point cloud shown in figure 3.12 provides a display of the height of each feature. Features such as the plane can clearly be viewed as much higher than tables and objects on the ground. This type of information can be very valuable to a robot in tasks such as object retrieval.
3.2.2 Mapping the Applied Fluids Laboratory, Moving to a Goal Position and Back

In algorithms with loop closure, looping one’s route or seeing the same objects again leads to loop closure and thus, much higher accuracy. Looping generally leads to higher quality point clouds alongside more accurate trajectory estimation. Since LOAM does not make use of loop closure, an experiment was done to see if these aspects will still hold. In short, the question of: Can the odometry in LOAM manage to track the reverse in direction? and, Will the quality of the point cloud improve significantly? Note that if there is failure in odometry, quality will decrease rather than increase. The point cloud will first be analyzed in the same sense as the previous subsection and then, compared to the previous point cloud in the next subsection. Figure 3.13 is similar to that of 3.9. This experiment again was in D124 with a very similar trajectory to that of the previous experiment. The experiment had a duration of about 33 seconds (about double the previous) and 8,356,757 points (little more than double the previous). The point cloud size is about 97.33 Megabytes.

The surface visualized outside of the point cloud may look like a type of noise or a
Figure 3.12: Kaarta Generated Point Cloud: Angled View.

misregistration in odometry, but it is another office mapped through the window of the laboratory door. This portrays one of the great qualities of LiDARs are their ability to map far and through several surfaces. Figure 3.14 cuts out the ceiling to give the view of the inside of D124. The trajectory forward and back are very close. Thus, differentiation between which part of the trajectory line is forward versus which part of the line is the return is not of importance. Due to this trajectory being longer than the previous, the density of this cloud is much greater than the previous experiment. This results in a greater distinction between the noise and the important cloud data. Due to this greater point density, the edges of several structures in the laboratory are very precise. Objects are also easier to identify, while zooming in and analyzing the point cloud in CloudCompare. A generalization can be made that if something is easier for a human to identify, it should also be easier for a robot. Figure 3.15 depicts this. An small segment from figure 3.13 is shown here alongside an image of the objects actually shown in that picture. As the sole usage of a point cloud may not suffice for a robot to identify objects, prior information about the lab may aid in recognition. As this lab mainly focuses in drone creation and testing, a large X-shaped object can be identified as a drone, more specifically, the Naviator NV4 (this identification
is extensive and is due to familiarity, a robot would at most be able to identify this as a drone). The power boxes can also be visualized in both figures 3.15b and 3.15a, while this identification may be tough for a robot, with the actual image of the laboratory or some prior knowledge, a robot can identify this. The white block depicted in 3.15b on the left-hand side of the image is imaged very well in the point cloud. This is expected as LOAM is regarded highly in its detection of rectangular planes. Regardless of robot identification or robot usage (even though that is the main use of point clouds), aspects of this point cloud can easily be identified by a human and thus, this cloud is labeled of high quality.

As one closely familiar with the experiment and the details of the environment, separation between noise and smaller objects in the environment can be done. Depiction of the 2-D contour from above depicts more noise than is actually there. Panning and tilting in CloudCompare (if one would like to perform this action as well, the generated point clouds are available on [https://github.com/rohanpaleja27](https://github.com/rohanpaleja27)) can be done to assess different regions of the cloud. The main area of confusion is shown below in figure 3.16. These lines displayed seem to all be on the ground. While the ground in the area does have some texture, these lines do not follow this pattern; this area is near the region where the turn takes place, and this may be attributed to the speed at which the turn takes place. As stated previously, LOAM generally assumes velocities are smooth and large changes in velocity...
may cause failure within this algorithm. This reason that this area produced this type of texture was inconclusive.

### 3.2.3 Comparison Between Single Path and Traversed Path

The two experiments conducted were very similar in that they had very similar trajectories, conducted in the same room (within the same day) and were processed using the same method. As can be seen by comparing figures 3.10 and 3.14, the lines are much more distinct in figure 3.14; this can be attributed to the higher density. None of the object surfaces within the lab space are mapped in a way depicting loss of odometry. Loss of odometry usually results in planes remapped again and again over several spaces. The map where the trajectory is to the goal and back, contains objects that are much more easily identifiable. If the odometry and general mapping is highly accurate, it can be assumed that as the number of points grow, objects will be easier to recognize. Thus, qualitatively performing multiple runs over a segment proves to be beneficial for map quality.
3.2.4 Mapping the Eng. Hallway

Mapping the Eng. Hallway required a slightly different experimental setup as seen in figure 2.10. As stated previously, this is an estimation of drift; the expectations of the experiment is that if any drift is present, Instead of the outer region of the hallway being a perfect rectangle, there will be a slight disconnect between one of the edges.

Some significant aspects of the experiment in how it differed from the previous two include a much higher translating speed. A hallway is mapped that has several laboratories on both sides. As some of these laboratories have windows (one has windows on both sides), indoor areas alongside the general hallway shape will be mapped. Hallways also have less features and are much bigger. Being very long and narrow causes features to quickly lose visibility, causing loss of information. An excess loss of information is also attained due to the way LOAM chooses features.

This experiment was conducted three times, starting in different regions of the hallway. The first experiment was started in one of the four corners of the hallway. This area can be regarded as an area with very low features. The second experiment was started near to the corner, but in an area where a path to the exit of the building was visible. The last experiment was conducted in a region of the hallway with a large number of features during initialization. All three of these start locations are displayed in figure 3.17. Experiment duration of each experiment was 87 seconds, 95 seconds, and 57 seconds, respectively.
Figure 3.16: Segmented Point Cloud Depicting Region of Unknown Texture.

Figure 3.17: Simplified Depiction of Experiment Start Points in the Hallway.

Previous to the presentation of the results, several notes should be mentioned. First, the speed of the experiment should not affect the accuracy. LOAM has been tested on vehicles moving at around 20 mph, and thus, moving a cart at a much slower rate should not be a problem (as long as turns are smooth). The importance of initialization should also not be undervalued. The "strength" of the start point can determine the initial odometry error and long-term outcome. "Strength" refers to the amount of features that stay in view as the LiDAR moves and thus allows for higher accuracy odometry. If the environment initially has no distinct features, the mapping is likely to be very poor.

Figure 3.18 and 3.19 portray the results of each experiment. The point sizes of each
cloud are 12,447,722, 12,515,150, 9,960,342 with sizes of 601 megabytes, 603 megabytes, 481 megabytes, respectively. For brevity, throughout the pictures and rest of the explanation, the experiment starting from the corner is labeled experiment 1, the experiment starting next to the corner is experiment 2, and the experiment starting from the center of the hallway is experiment 3. See figure 3.17 for a more precise position of the starting points. As point clouds are very hard to visualize from any one angle, a top view and a side view are shown. Paying close attention to aspects of the cloud allows for several generalizations and trends to be deduced. Figure 3.18 depicts the top view of each point cloud from each experiment viewed in CloudCompare. Each point cloud has similar features in that several rooms are mapped alongside the hallway. The inside rooms that were mapped were of low quality, except for that the size of the rooms were measured accurately. One special note is that the machine shop, the large room on the right bottom of each image, had an opening to the outdoors; the LiDAR mapping in the hallway sent beams through the entire machine shop, outside the door, and sensed some vegetation (not depicted) that was about 100 meters away. Anyone with prior experience in the Eng. building would say that these contours are fairly accurate and do represent the environment well. On a repeated note, the speed at which this hallway was mapped is very impressive. As the dimensions of this hallway had to be compared to the measurements of the generated point cloud, the hallway had to be measured by hand, which took significantly longer than using this sensor to map the entire hallway. To get a better view of the point cloud, a side view of each experiment is shown in figure 3.19. These pictures gives a much more qualitative view of the point clouds. The accuracy of each cloud can clearly be seen. The odometry computed by LOAM made sure that the length of each hallway was almost exactly the same, but the measured height was not as clearly can be seen. Note that for Experiment 1 and 2, the ends of the trajectory are the points used to measure error. For experiment 3, the middle of the hallway is used to measure error as that is the starting point. Since Kaarta LOAM is conducted without an IMU, this error is very hard to fix. Think about the scenario where one starts moving down a hallway and then makes a turn. All references used to see how far he or she has gone are invisible but that distance can be stored in memory. As one turns again into a new portion of the hallway, he or she will know the direction turned but there is no way
Figure 3.18: Hallway Point Clouds Viewed from Top Position.
Figure 3.19: Hallway Point Clouds Viewed from Side Position.
to tell if he or she is moving upwards or downwards. Humans can use external sense such as stress on joints to determine if there is a ramp, but the sensor has no way of knowing unless there is some reference in the environment to help it. Since loop closure is not used in LOAM, if a reference goes out of view and comes back, it is not very useful. Strong reference features must remain in view or be continuously generated to ensure no drift in the vertical direction. On the plus side, due to the lightness of LOAM, sensor fusion is easily possible (any sensor that can measure altitude) to solve small issues like this. Thus, while the LOAM algorithm cannot be faulted for these errors, the size of these errors do tell something about the algorithm and experimentation procedure. The amount of features that the initialized position has correlates directly with the amount of drift. Figures 3.20, 3.21 and 3.23 show the trajectory of each experiment from a top view and a side view.

Figure 3.20: Hallway Trajectory from Experiment 1.

Note that all these trajectories are viewed for a similar angle. If one can place figure 3.17 on top of these images, one can clearly imagine the start of the hallway experiments. The goal is to have both lines as parallel as possible in the side view of the trajectory. This is because in truth the hallway is flat and is a perfect rectangle. It can easily be seen that the third case, where the experiment started in area with the largest numbers of features and thus, the most visibility, produced the best results. Experiment 1 has a very large amount of drift and immediately labels the point cloud as poor quality. Since most other variables
Figure 3.21: Hallway Trajectory from Experiment 2.
(a) Trajectory from Experiment 2.
(b) Trajectory Side View from Experiment 2.

Figure 3.22: Hallway Trajectory from Experiment 3.
(a) Trajectory from Experiment 3.
(b) Trajectory Side View from Experiment 3.
are controlled or have proved to be ineffective in changing the accuracy of the cloud (such as speed of translational motion), it can be concluded that the number of features in the start location directly determines the quality and drift of the point cloud. This conclusion can be generalized for point clouds mapped in regions with very low features (hallways, offices, etc.).

3.2.5 Discussion on Point Cloud Noise using Kaarta LOAM

Noise in general is very hard to distinguish in point clouds. During the viewing of point clouds in CloudCompare, there are several regions where points are floating in empty space. These could be objects that were viewed for a short amount of time, reflections, or noise. A generalization can be made that the environment is made up of all of these. As noise is a very old phenomena, many methods have been developed for removing large amounts of it, while keeping much data from the cloud. Using one of the most simple ways to remove noise, a nearest-neighbor grouping can be done that only keeps points with at least 5 points around it (also known as statistical outlier filter). This can be thought of as a simple low-pass filter and one that enhances the quality of planes and reduces the quality of objects and features that were minimally mapped. On most of the point clouds shown in this section, the noise filter removes about 1/4 of the points. Both of these images are very similar. The point size decreased from 12,447,72 to 9,118,571 while still maintaining all the important
features of the cloud. This filter can generally be used and tuned for environments. This type of filter can be used in post-processing to reduce the amount of noise and increase the quality of the point cloud.

3.2.6 Analysis of Accuracy

The analysis of accuracy will be done on an average performance trial of mapping the D124 laboratory for traveling to a goal, for traveling to a goal and back, and on each trial of the hallway experiment. This quantitative analysis should serve to assist with the statements in the previous sections.

Accuracy of the Applied Fluids Laboratory Point Clouds

For the accuracy of mapping the Applied Fluids Laboratory, distances to 6 objects will be measured in CloudCompare and this will be compared to the real life distances. These 6 objects are the same of that discussed in the accuracy of the Linux-LOAM. These 6 objects are in different regions of the map, and also represent different directions testing elevation accuracy, width accuracy, and length accuracy. The true measurements were taken with a standard tape measure. The distances measured in CloudCompare are done by selecting 2 points to where the true measurement was done. It should be noted that the ground plane is not mapped often in this image as the LiDAR was mapping at a certain height. Several times throughout the map the ground plane is captured. The set of points that have a minimum on the Z-axis should correspond to the ground points. Any point from this set can be used as a ground point. Using this alongside the distance tool, measurements to each object can be computed. These objects alongside the percent error are shown in the table below. A simple depiction of the laboratory is shown below. The stars represent the planes where vertical heights are measured. The lines represent horizontal distances that will be measured.

General human error can result from incorrectly matching the point selected in CloudCompare to the true measured point, as it is nearly impossible to get the point chosen to be the exact point measured. At most, this error would result in a slight excess error of ±1 cm. The accuracy testing results are displayed in Table 3.4. Since the initial goal of this
Figure 3.24: Depiction of Objects that will be Used to determine Accuracy.

<table>
<thead>
<tr>
<th></th>
<th>Ground Plane to Plane-1</th>
<th>Ground Plane to Plane-2</th>
<th>Horizontal Distance from Shelf to Desk</th>
<th>Width of Solder Station</th>
<th>Length of Plane (Nose to Tail Fin)</th>
<th>Table Length</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>2.844 m</td>
<td>2.571 m</td>
<td>1.447 m</td>
<td>1.778 m</td>
<td>1.575 m</td>
<td>1.365 m</td>
<td>*</td>
</tr>
<tr>
<td>Measured</td>
<td>2.803 m</td>
<td>2.546 m</td>
<td>1.369 m</td>
<td>1.846 m</td>
<td>1.518 m</td>
<td>1.267 m</td>
<td>*</td>
</tr>
<tr>
<td>Percent Error</td>
<td>1.441%</td>
<td>0.972%</td>
<td>5.39%</td>
<td>3.82%</td>
<td>3.61%</td>
<td>7.22%</td>
<td>4.22%</td>
</tr>
<tr>
<td>Absolute Error</td>
<td>0.031 m</td>
<td>0.025 m</td>
<td>.078 m</td>
<td>.068 m</td>
<td>.057 m</td>
<td>.098 m</td>
<td>.0595 m</td>
</tr>
<tr>
<td>% Allowable Error Relative to 30 cm</td>
<td>10.30%</td>
<td>8.33%</td>
<td>26.00%</td>
<td>22.66%</td>
<td>19.00%</td>
<td>32.67%</td>
<td>19.82%</td>
</tr>
</tbody>
</table>

Table 3.4: Actual and Measured Distance to Objects in the Point Cloud, Traveling to Goal
The experiment was to ensure that the error of the true point cloud was within one magnitude over the resolution. This meant that the % allowable error relative to 30 cm should be less than 100%. This number was found by taking the absolute error and dividing by 30 cm. Taking into account these 6 objects solely (a larger experiment could be conducted if need), the accuracy of the point cloud can be generalized to 95.78%. The allowable error is much within the bounds of planned error for this experiment. Next, the same analysis will be conducted for the mapping in D-124, where the LiDAR traveled to a goal position and back.

<table>
<thead>
<tr>
<th></th>
<th>Ground Plane to Plane-1</th>
<th>Ground Plane to Plane-2</th>
<th>Horizontal Distance from Shelf to Desk</th>
<th>Width of Solder Station</th>
<th>Length of Plane (Nose to Tail Fin)</th>
<th>Table Length</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>2.844 m</td>
<td>2.571 m</td>
<td>1.447 m</td>
<td>1.778 m</td>
<td>1.575 m</td>
<td>1.575</td>
<td>*</td>
</tr>
<tr>
<td>Measured</td>
<td>2.879 m</td>
<td>2.674 m</td>
<td>1.224 m</td>
<td>1.342 m</td>
<td>1.342 m</td>
<td>1.353</td>
<td>*</td>
</tr>
<tr>
<td>Percent Error</td>
<td>1.23%</td>
<td>4.01%</td>
<td>15.41%</td>
<td>20.75%</td>
<td>14.10%</td>
<td>.88%</td>
<td>9.40%</td>
</tr>
<tr>
<td>Absolute Error</td>
<td>0.035 m</td>
<td>0.103 m</td>
<td>0.223 m</td>
<td>0.369 m</td>
<td>0.222 m</td>
<td>.012</td>
<td>.1825 m</td>
</tr>
<tr>
<td>% Allowable Error</td>
<td>11.60%</td>
<td>34.30%</td>
<td>74.33%</td>
<td>123.00%</td>
<td>74.00%</td>
<td>4.00%</td>
<td>60.83%</td>
</tr>
</tbody>
</table>

Table 3.5: Actual and Measured Distance to Objects in the Point Cloud, Traveling to Goal and Back

From comparing 3.4 and 3.5, one can notice several trends that do not follow general opinion. The accuracy of the second experiment is 91.60%. While the allowable error of the
solder station is greater than the limitation, the average of the point cloud is still less than 100%. This shows that this point cloud is still of good quality. Surprisingly, this point cloud has accuracy less than the experiment traveling to the goal. While in usual cases, multiple runs greatly improves accuracy, in LOAM, multiple runs is not as significant because there is no loop closure. Kaarta LOAM has stated that going through the environment slowly is better than turning and doing multiple runs. To accompany this, LOAM is proven to be poor during turns and thus, a full 180 degree turn during the experiment may decrease the accuracy. These two reasons provide some explanation on why the second experiment had lower accuracy. It can also then be suggested, an increase in cloud density does not always improve cloud accuracy, if there is some loss in odometry.

**Accuracy of Eng. Hallway Point Clouds**

The previous accuracy analysis analyzed point cloud accuracy of mapping dimensions properly. This subsection tests the drift in large-scale areas with less features. As previously stated, the hallways should be a perfect rectangle. Thus, the drift can be measured by looking at the difference of the end points in the trajectory should in figures 3.20b, 3.21b, and 3.22b. The table below describes the error of each experiment in meters. Percent error cannot be computed as dividing by zero (the actual) would cause an error; thus, instead a relation is used to compare the measured drift versus the length of the trajectory.

As expected and can be seen qualitatively, the error drastically reduces as the initialization region has more features. Mapping a low-feature region that is 104.08 meters in length with errors under 3% in its best case represents a high quality point cloud. The Eng. hallway can be considered a region of very low feature when compared to D124 or most standard offices. Without color data, most rooms are similar; this implies that LOAM is able to generate point clouds for most environment.

### 3.3 Post-Processing and Smoothing

As noise filters have already been discussed in the previous section, this section focuses on the P.C.V. (Point Cloud Visualization) in CloudCompare. This works to simulate the
<table>
<thead>
<tr>
<th></th>
<th>Experiment 1</th>
<th>Experiment 2</th>
<th>Experiment 3</th>
<th>Hallway Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drift Compared to Hallway Length</td>
<td>14.378 m</td>
<td>3.523 m</td>
<td>1.362 m</td>
<td>104.08</td>
</tr>
<tr>
<td>Percent Error Compared to Hallway</td>
<td>13.31%</td>
<td>3.38%</td>
<td>1.30%</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 3.6: Hallway Accuracy

natural illumination of the scene. It cleans regions of the point cloud enhancing borders and plane-like features. This type of smoothing adds information to a 3-D model giving it a more realistic view and thus, making it easier to match to camera images. This can be done to every point cloud but takes several minutes as this tool is very computationally expensive. Performing segmenting tasks such as octree may allow for this type of smoothing and visualization tool to be conducted in real time but it is not recommended, and may not

Figure 3.25: P.C.V. Smoothing of a Point Cloud.
produce a large enough benefit versus the increase in computational complexity.

3.4 Comparison Between Linux-Based LOAM and Kaarta-Based LOAM

There are clearly key differences between the two methods. The Linux-LOAM method makes use of an IMU, generates a point cloud in real-time, is open-source, but takes much longer to setup and must be tuned. The Kaarta LOAM works very quickly as a file just must be recorded, but the point cloud will be generated after a period of time. This method also does not use an IMU so if the limitations of the experiment are not closely followed, drift and error are much more likely to occur.

Both of these methods produce clouds of high quality that a robot should be able to use to navigate the environment. If RGB data is available, object recognition should be capable by a robot using both these methods. An overall analysis of accuracy and point quality does determine that the Linux-LOAM performed better in the laboratory and hallway environments. This is due to its much higher accuracy and that the drift is much less. This can be attributed to the IMU and the LOAM parameter tuning stated in the previous sections.
Chapter 4
Conclusions

Using a Velodyne VLP-16 Puck in indoor environments that are feature-rich and feature-lax proved to be possible with the LOAM algorithm. Not only was mapping possible, highly accurate clouds were produced that were replicas to the real-world environment with an accuracy as high as 99.71% cloud-to-world. Large indoor environments were also mapped (above 100 meters in length) with drift less than 1 meter in the best scenario. These results also affirmed that LiDAR has considerable benefits over other sensors such as monocular and stereo cameras, and that LOAM has considerable advantages over other types of LiDAR SLAM.

Using a customized platform to mimic some behaviors of a mobile robot, the output of this experiment can be expanded to effectively prove that a mobile robot could use LOAM to generate a map of the environment. The high accuracy of the generated point clouds allowed for generalizations suggesting that this cloud could be used for navigation, obstacle avoidance, path planning, object detection, and generating useful information about the environment.

4.1 Future Experiments

The first general expansion of this experiment includes to move this system to a mobile robot, and test if the accuracies and precision remain; through the results of this experiment, there is a high probability that the results should remain similar. Environments like the hallway or laboratory should be able to mapped with high quality. Office environments (containing many cubicles) are likely to produce poor results as there are not many unique features around the room, and the narrow walkways attenuate much of the LiDAR’s range. Environments that have a large area (thus, allowing the LiDAR to use its full range) are
expected to perform very similarly. After experimentally identifying environments where this experiment produces high quality point clouds, tasks such as navigation and obstacle avoidance are capable and can be attempted.

While in this paper, LOAM is tested in two environments that are assumed to generalize over a set of environments, experimentation must still be conducted to precisely define this set of environments. Once this set is defined, and LOAM has been tested on a mobile robot, the task of indoor mapping (in this set of environments) can be conducted with ease by using the instructions presented throughout this paper.

A true experiment of the power and usability of LOAM would be to have one system (mobile robot) mapping the environment and streaming the data constantly to several robots. Since these robots have an initial map of the environment, they can use a simpler sensor configuration (IR and sonar sensors placed around the robot) to navigate the world by landmark planning. Using this method, high precision tasks can be accomplished by using many simple robots and one mobile robot running LOAM and streaming data.
References


