USDOT Region V Regional University Transportation Center Final Report

NEXTRANS Project No. 134PY2.1

Stationary LiDAR for Traffic and Safety Applications – Vehicles Interpretation and Tracking

Andrew P Tarko
Professor of Civil Engineering
Purdue University

Kartik B. Ariyur
Assistant Professor of Mechanical Engineering
Purdue University

Mario A. Romero
Research Scientist
Purdue University

Vamsi Krishna Bandaru
Graduate Research Assistant
Purdue University

Cheng Liu
Graduate Research Assistant
Purdue University

DISCLAIMER

Funding for this research was provided by the NEXTRANS Center, Purdue University under Grant No. DTRT07-G-005 of the U.S. Department of Transportation, Research and Innovative Technology Administration (RITA), University Transportation Centers Program. The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

ABSTRACT

The goal of the T-Scan project is to develop a data processing module for a novel LiDAR-based traffic scanner to collect highly accurate microscopic traffic data at road intersections. T-Scan uses Light Detection and Ranging (LiDAR) technology that can detect and track various types of road users, including buses, cars, pedestrians, and bicycles; and, unlike video detection, it does not experience the well-known occlusion problem. Moreover, LiDAR data has a one-to-one correspondence with the physical world, which makes it possible in principle to produce the positions and velocities of road users in real-time as needed for traffic and safety applications, with the errors of estimation dependent only on the resolution and accuracy of the LiDAR sensor. We faced two major challenges to this goal after integration of the sensing, data collection, and processing components: 1) the reduction of resolution farther from the sensor because of fewer light rays per unit area and reflections that do not return to the sensor and 1) the-wind induced oscillations of the sensor position. We solved the latter problem through installation of an inertial sensor atop the LiDAR to calibrate its motion. We are developing several approaches to solve the first problem, including video integration. A lot of our code is already open-source-based, and we have found from the computational requirements for various algorithms that the use of a standard GPU for processing should permit real-time running of our algorithms in the data processing module.

System Integration: The system we integrated consists of a LiDAR sensor installed on a pneumatic 42-foot telescoping mast. The sensor head rotates 900 times per minute, which results in 1.3 million data points per second. Data collected over a period of several hours to several days is stored in high-capacity devices. The final stage of integration and hardware testing was performed as a part of this project. A customized aluminum base physically integrates the mast, pan/tilt mechanism, LiDAR head, and two video surveillance cameras to provide a human-friendly data format for supervision and evaluation. The T-Scan collects 3D data with 64 laser sensors arranged vertically in a fan-like manner. The spherical data (range, vertical angle, and horizontal angle) are first converted to more convenient XYZ Cartesian data. The XYZ data points are then interpreted as moving objects and the background.

LiDAR Self Calibration: Using a LiDAR sensor outdoors on top of a mast and/or vehicle exposes it to wind-induced vibration. Displacement of the LiDAR leads to a change in the reference frame from which the measurements are taken; as a result, any further analysis on that data will be erroneous. In this report, methods are presented for establishing a global frame of reference and mapping the sensor data to the established global frame, thereby

negating the errors and effects of the environment. The first method uses an Inertial Measurement Unit (IMU) to track the motion of the LiDAR at high frequency and compensate for it in real-time, using a Kalman filter based on a simple model of fluid-structure vibration. The second method attempts to perform this compensation through detection of invariant planes and determining the translation and rotation of the sensor with respect to those planes from frame to frame and then converting all the data to a global stationary coordinate system. This method is generalizable to the detection and tracking of moving objects as well if there is enough sensor resolution. In our experiments we used both natural background features and fixed markers to attempt this self-calibration. The next step is to identify moving objects and track their path. We have developed a method of identifying the background; and when this background is subtracted from the data, all that remains is the moving objects. In the next steps, we will identify individual moving objects such as vehicles and pedestrians. Once the trajectory of each moving vehicle is identified, the results will be stored in SSAM format for future use.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

In spite of the remarkable progress in traffic measurement technology, a large variety of traffic measurements, ranging from simple counting of turning vehicles to complex counting of dangerous traffic interactions, require observers. The existing methods of counting traffic conflicts deploy surveillance cameras that are inexpensive but cannot deliver accurate measurements due to the intrinsic weakness of 2D data. These systems still require involvement of human observers in various stages of data processing. The automated video camera-based methods are inaccurate while the video camera-based methods with involvement of humans are expensive and inefficient for large measurement efforts.

The recently emerging LiDAR technology found applications for land surveying (NOAA, 2013), 3D objects modeling, autonomous vehicle control (Cheung, 2007), as well as other application areas. With the introduction of relatively inexpensive LiDAR units (Velodyne, 2014), the area of traffic measurement might finally be given technology that allows measuring the position and dimension of objects in a 3D space with a one-to-one correspondence between the measurement and the measured objects.

The LiDAR sensors are used as mobile sensors in surveying and in autonomous Google vehicles but have not been considered as stationary units for acquiring locations of vehicles in traffic streams. The locations of points on objects' surfaces sampled at the sufficient frequency could be used to identify individual vehicles, to measure their size, and to track their positions in the 3D space. If these measurements are sufficiently accurate, then they could be further processed to estimate the spatial and temporal proximity of vehicles to other moving and stationary objects – the fundamental quantities of surrogate measures of safety (Tarko, 2012; Johansson and Rootzen, 2014).

The recent considerable progress in the surrogate measures of safety (Tarko et al., 2009) based on microscopic measurements of vehicle motions refueled an interest in this alternative approach to measuring safety. The traditional crash-based methods cannot meet the growing demand for rapid safety estimation in the current reality of fast technological changes in vehicles and the supplementary components of road infrastructure.

The in-vehicle systems measure the vehicle motion and attempt to identify and count dangerous events based on the observable driver reaction or rapid changes in speed or direction of the vehicle (Dozza and Gonzales, 2013; Wu and Jovanis; 2013). The identification of such events appears to be quite difficult and results in many false positives or false negatives. The primary source of this difficulty is the missing environmental context that could be provided if the proximity of other vehicles or objects could be measured. Measuring the spatial or temporary proximity between objects offers the opportunity for proper interpretation of the traffic interaction and for evaluating the "riskiness" of the event. The proximity data can be analyzed with the extreme-value statistical methods leading to estimation of the risk of collision given the frequency and severity of vehicle interactions (Tarko, 2012). The critical condition of making this exciting perspective a reality is developing a practical traffic measurement method.

This report presents a research project that is the first step towards evaluating the feasibility and developing a practical tool of Traffic Scanning (TScan) for counting turning vehicles at intersections, measuring traffic interactions for the purpose of safety estimation, and conducting other traffic studies. The presented first phase of this research includes: 1) the integration and evaluation of a data acquisition system, 2) the development of basic pre-processing functions for data reduction, storing, and retrieval, 3) recognition and extraction of the fixed background, 4) correction of the measurements for the sensor motion, and 5) development of a concept for tracking and classifying moving objects.

The report is organized into eight sections, which include an introduction; the scope and focus of the current phase; integrating and testing the measuring equipment (TScan hardware prototype for research purposes); collecting the first data and developing basic per-processing functions; developing and evaluating the LiDAR motion with a separate Inertial Motion Unit (IMU), developing and evaluating a self-calibration method without the use of an IMU; developing a concept for a method of tracking and classifying objects; and finally, presenting the summary and conclusions from the reported first phase.

2 Research Scope and Focus of the Current Phase

The final objective of the entire research effort is to develop a data acquisition and processing system, called TScan, which is inexpensive, provides results with the required accuracy, and is applicable at most intersections and various weather and traffic conditions. The system will produce results in the form of time-stamped positions of moving objects (vehicles, bicycle, and pedestrians). These objects can be classified and their dimensions measured. The produced files could be further processed with various engineering applications. Eventually, two applications will be developed for the purpose of the system's evaluation:

(1) counting vehicles by turning movements at intersections, and
(2) counting traffic conflicts and measuring their severity at intersections.

The final research objective is ambitious and will require a considerable effort and time. The entire effort is organized into phases that form individual research projects with well-defined scopes, objectives, and clear connections between phases. To accomplish the envisioned research objectives, three phases are necessary:

Phase 1: Development of a data pre-processing module.
Phase 2: Development of a vehicle classification and tracking module.
Phase 3: Development of selected engineering applications.

All three phases include the concept of the component and its development and its testing and evaluation to confirm that the outcomes follow the concept and the results satisfy the quality requirements.

The planned activities and results of the first phase include:

1. Initial concept of the system.
2. Integrating and testing the measuring equipment.
3. Preliminary data collection and analysis (sensor features, data management, and data quality analysis).
4. Developing and evaluating the LiDAR stabilization component.
5. Developing and evaluating a preliminary tracking and self-calibration module.
6. Revising the concept of vehicle tracking and classification (if needed).
7. Writing a final research report.

Due to the novelty of the sensor and the proposed product, learning both the features of the new LiDAR sensor and the properties of the collected 3D data are expected. The research will be advanced with possible iterations to the extent allowed by the resources**.** The components of the developed pre-processing module will be evaluated to learn the properties of the measurements and their derivatives. The results of this learning process and evaluation of the intermediate outcomes of the research could lead to revising the original concept of the system.

The research is being conducted with the Purdue Traffic Mobile Laboratory, a dedicated van equipped with the required instruments for collecting, processing, and storing data. This hardware prototype allows continuous supervision of the data collection process and easy transfer of the collected data to the computational facilities of the Center for Road Safety and Purdue University for testing and analysis. The concept of the unit and its functions for implementation is presented in the next section of this report.

3 System Concept

This section describes the overall concept of the data acquisition, pre-processing and interpretation system and the user-oriented specifications.

The proposed system must be useful for collecting traffic data at compact and medium size intersections and relatively short road segments in normal and adverse weather during daylight and night-time conditions. The outcome of the data pre-processing includes the tracking and classification information for all the individual objects in the field view of the system. This information should include the type, dimensions, and positions of the moving objects at 10-15 instants per second. The classification and tracking information should be sufficient for successful application to the typical traffic analysis, including speed studies, counting turning vehicles, gap acceptance studies, measuring saturation flows, and counting traffic conflicts and measuring their severity.

The format of the outcome of the classification and tracking should follow the SSAM format for simulated traffic with modifications that reflect different sources (measurement vs. simulation). For instance the "wire-based" motion of vehicles in typical simulation is replaced with the realistic two-dimensional motion on a (x, y) plane in the real world. Large intersections with wide medians and channelization may require two or more measurement units, in which case, a single unit should generate results that can be integrated from all the units for successful tracking and identification of moving objects within the entire intersection.

The data pre-processing and storing must be executed in real-time. The classification and tracking objects also should be executed in real time if possible. Specialized processors for matrix operations and parallel computing should be considered to speed up the calculations. This is an important aspect of the system design and development due to the massive amount of data collected during a period that could be as long as one week. The excessive time that may be needed for data processing after collecting the data could undermine the practicality of the system. Although the current research is focused on the fundamental considerations of signal processing for objects classification and tracking, the computation load and processing efficiency must be addressed in the early stage of developing methods, algorithms, and computer codes.

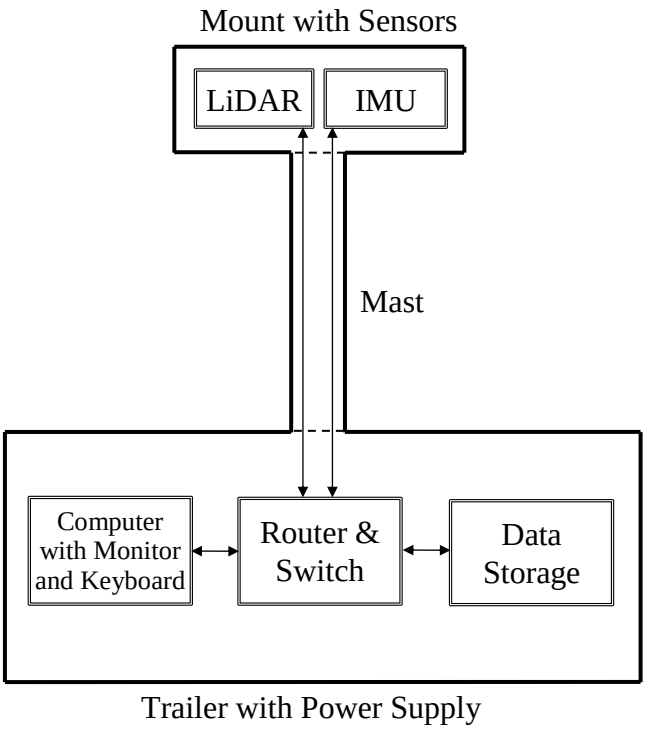


Figure 3.1 A general concept of the portable LiDAR-based system TScan

The system hardware includes four main components:

1. a trailer with power supply,
2. a telescoping mast,
3. sensors installed on a mount, and
4. a computer with communication component and data storage.

The trailer has stabilizing legs that allow transportation of the unit to the data collection site and setting the equipment in a suitable position and provides sufficient physical support and protection against tampering. The system should be able to generate its own continuous power supply with an option of using electric power if available at the site. The pneumatic telescoping mast raises the sensors on a mount with pant/tilt mechanism at a desired elevation. The mast includes an air compressor and a locking mechanism to keep the mast unfolded without running a compressor.

A low-end LiDAR sensor, such as the HDL-64E by Velodyne Acoustics, Inc., should be used to reduce the cost of the unit, but it should provide sufficiently dense and accurate data. A set of

Inertia Measuring Units (IMU) will be integrated with the primary sensor to adjust collected data for the sensor motion.

The computer with a monitor and a keyboard is used by a user to supervise setting of the data acquisition component. The computer also facilitates communication between the units, data pre-processing, and data storage during an unsupervised data collection period. The data storage should be sufficiently large to allow continuous data collection for at least one week.

In the most desirable design, all the data reduction, pre-processing, objects classification and tracking are conducted in real time. The product of the data collection and real-time processing is stored in files in the SSAM-like format, which is typically used in computer micro-simulation with transportation applications.

This file will contain input data for engineering software applications, thus it will serve an interface between the system and user applications. User applications are run in an office environment after the data collection completion and retrieval from the data storage unit.

4 Prototype Hardware Integration and Testing

The first step for this project was to integrate and test the prototype TScan hardware installed on the Purdue Mobile Traffic Laboratory (MTL). The next step was to understand how the LiDAR sensor worked as well as how all the hardware performed together as a system.

4.1 Current Setup

The Purdue MTL is owned by the School of Civil Engineering and used and maintained by the Center for Road Safety. The MTL has been used in several research projects involving video technologies for data collection and processing.

The MTL was built within a Chevy Express 3500 van and is equipped with a 42-foot pneumatic mast that can be operated from inside the van. After the last hardware upgrade, the MTL includes a LiDAR HDL-64E manufactured by the LiDAR Division of Velodyne Acoustics, Inc.. a pan-tilt base that holds the LiDAR, two pan-tilt-zoom IP dome cameras, two flat screen monitors, a computer, an eight-channel video recorder, and the one gigabit (GB) Ethernet network shown in Figure 4.1.

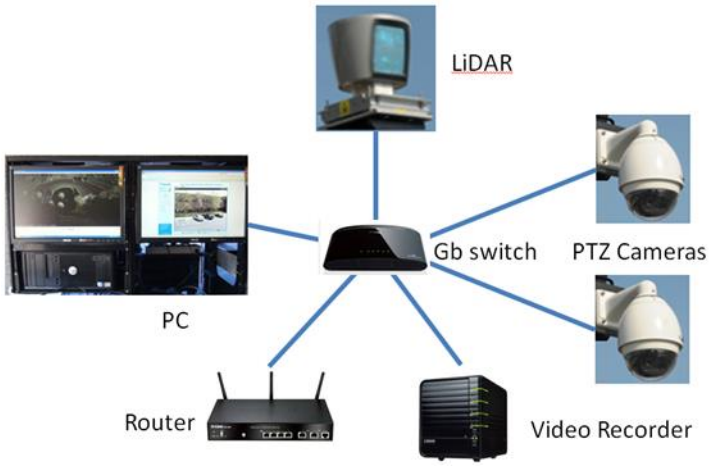


Figure 4.1 Network configuration

The equipment is powered by a heavy-duty inverter, with two heavy-duty batteries that are charged using the van engine. The major equipment is rack-mounted for safe transportation.

A customized aluminum base was made to support the cameras, the pan-tilt base, and the LiDAR on top of the pneumatic mast. It was designed to allow different configurations as shown in Figure 4.2.

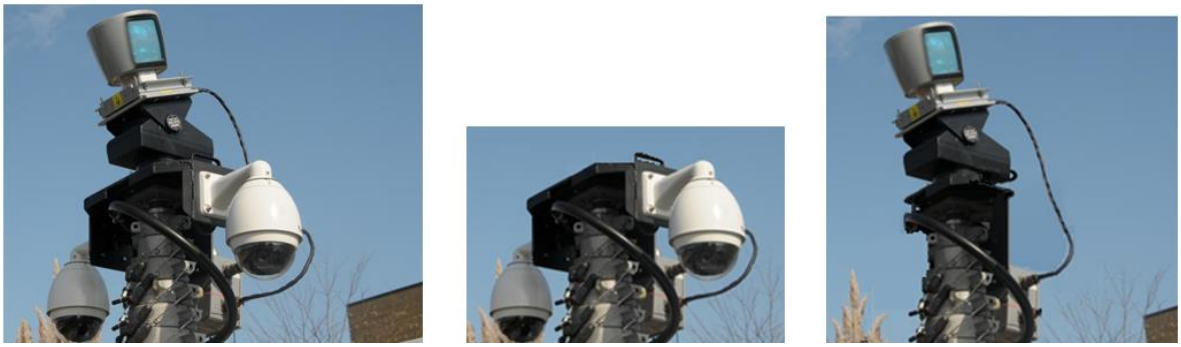


Figure 4.2 MLT sensors configurations

Video

MTL sensors include two IP digital cameras with pan, tilt, and zooming (PTZ) capabilities. The new cameras can be adjusted and controlled using any Internet browser, which simplifies the video control panel and the connections from the van to the top of the mast. Only power and k5 cables for Ethernet communications are required. Most importantly, the new sensors allow synchronized recording of video material in a digital format from the two cameras. The digital format eliminates the digitizing error from video processing. The recording function is executed through an autonomous recording system to reduce the computational load of the MTL computer.

Communication system

A one GB Ethernet-based network was designed for the MLT for system control and for managing video and LiDAR data transfer between the data acquisition, processing, and storage units. This type of network has capacity sufficient to handle collecting both the video and LiDAR data simultaneously. Nevertheless, the network was designed to operate as two separate sub-networks to avoid any potential interference between the LiDAR and the video systems in case of a conflict between the communication protocols of the video and laser technologies.

LiDAR

The LiDAR position can be adjusted before data collection at a site with the pan/tilt (PT) base to which the LiDAR head is attached. The PT base is controlled from inside the van with a convenient control panel. The PT base is attached to the mast top directly or through a

customized mount which also allows attaching the PTZ cameras. The mast mount was designed to ensure large and unobstructed fields of view for the LiDAR and the two surveillance cameras.

Any LiDAR data capture application can be used to manage the LiDAR data, but the software provided by Velodyne has visualization capabilities that help properly position the LiDAR sensor before data collection.

4.2 Hardware Testing

A step-by-step testing protocol was developed in order to ensure that each system component was properly installed as well as to guarantee the system integration. The protocol includes testing the power supply, communication, video and LiDAR components and the entire system integration.

Power

The first step was to test the cables with a short circuit tester. Once the cables were checked, the three required levels of voltage were tested at their sources:

1. 110 VAC to power up the computer, the two monitors, the video recorder, the router and GB switch;
2. 24 VAC used by the video cameras; and
3. 12 VDC used by the LiDAR and the PT base.

Once the output voltages were tested, the equipment was connected and tested for its basic operations.

PT base

The control cable for the PT base was tested to ensure the proper connections. Then, the PT base operation was tested using both the console control and the remote control to check the rotation angles, the end switches, and the rotation direction.

Network

The communication cables were tested using direct connection to a laptop. After direct communication was established, the switch and router were installed. Then, the communication to each element separately was tested again.

The required network was set up, which included assigning the IP addresses, setting the video recorder and cameras, and checking whether the supervising computer recognized these settings.

A turn-on sequence procedure was established during the network testing. The video systems must be turned on before the LiDAR due to the broadcast communication system it uses.

Video

The camera movements are controlled through the supervising software installed on the computer, and the cameras' control console therefore is no longer needed. The cameras' panning, tilting, and optical zooming were tested with the cameras' software and the video recorder software, both of which were installed on the system computer.

Finally, the functions of the video data recording and extraction were tested.

LiDAR

As soon as the LiDAR is powered up, it automatically sends broadcast messages through the network every 10 blocks of data. These LiDAR messages may interfere with the hand-shaking protocols between various hardware components, thus the LiDAR unit must be powered up last. After powering up all other system components, the LiDAR was turned on twice: 1) when directly connected to the computer with the vendor's software and 2) when connected through the Ethernet network.

5 Initial Data Collection, Pre-processing, and Testing

5.1 Velodyne HDL-64E Sensor

The Velodyne HDL-64E laser sensor used in the prototype unit is a compact sensor pod with 64 laser line scanners. The line scanners produce 64 laser beams arranged vertically inside a 27^{o} vertical angle. The unit rotates to give a full 360^{o} by 26.8^{o} field of view (FOV). The Velodyne sensor was used in autonomous vehicle applications, such as the DARPA Grand Challenge (Cheung, 2007). An overview of the specifications for the Velodyne scanner is provided in Table 5.1 (Velodyne, 2014).

Table 5.1 Manufacturer specifications for the HDL-64E S2 Scanner

Sensors	64 lasers
	360˚(horizontal) by 26.8˚ (vertical) FOV
	Range: 50 m (10% reflectivity) 120 m (80%)
	1.5 cm range accuracy (1 sigma)
	0.09° Horizontal Encoder Resolution
	>1.3333 MHz
Lasers	Class 1
	905 nm wavelength
	5 nanosecond pulse
	2.0 mrad beam divergence

The HDL-64E spins at rates ranging from 300 RPM (5 Hz) to 900 RPM (15 Hz). The default is 600 RPM (10 Hz). Changing the spin rate does not change the data rate. The unit sends out the same number of packets (at a rate of 1.3 million data points per second) regardless of the spin rate. The image resolution will increase or decrease depending on the rotation speed.

5.2 Collecting data with HDL-64E

The HDL-64E data are presented as distances and intensities only. The connection between the LiDAR and the computer is similar to a two-way LAN setup. The LiDAR consistently sends messages with a fixed IP source and destination addresses. The data collected is packaged in a format called .pcap.

Structure of HDL-64E Data

The HDL-64E outputs UDP Ethernet packets. Each packet contains a data payload of 1,206 bytes that consists of 12 blocks of 100-byte firing data followed by six bytes at the end of each packet that contains a spin counter and firmware version information. Each packet can be for either the 32 upper or 32 lower laser banks (called laser block).

The packet format is as follows:

2 bytes of header info

This header indicates whether the packet is for the upper block or the lower block. The upper block will have a header of 0xEEFF and the lower block will have a header of 0xDDFF[1].

2 bytes of rotational info

This is an integer between 0 and 35,999; dividing this number by 100 gives values in degrees.

32 laser return info of 3 bytes each

Each return contains

2 bytes of distance info, in .2 centimeter increments.

1 byte of intensity info. 0 - 255, with 255 being the most intense return.

A zero return indicates no return up to 65 meters.

6 status bytes

These status bytes alternate between packets. The end of the packet will show either:

- Reading showing the internal temperature of the unit. You will see a "DegC" ASCII string as the last four bytes of the packet. The two bytes before this string are the thermistor's reading in C in hex 8.8 format. This is in "big endian format" (i.e., the byte immediately preceding the DegC text is the whole degrees, and the byte preceding that is the fraction of a degree in 1/256 increments).

[1] The hex values shown in packages are in inverted order; therefore, upper block indicator EE FF will be shown as FF EE in Wireshark (**Error! Reference source not found.**), which is the case for all other values

Therefore, if you see the reading “c0 1a,” the temperature of the thermistor is 26.75 degrees C.

- Or, the version number of the firmware in ASCII character format "Vn.n," where n.n is the version number (i.e., "1.5").

In summary, the total bytes per packet of data is

$$1206 = 12 \times \left(2 + 2 + 32 \times \left(2 + 1\right)\right) + 6$$

For reference, an image depicting the data structure of a packet is shown in the Figure 5.1.

Using Velodyne's Software

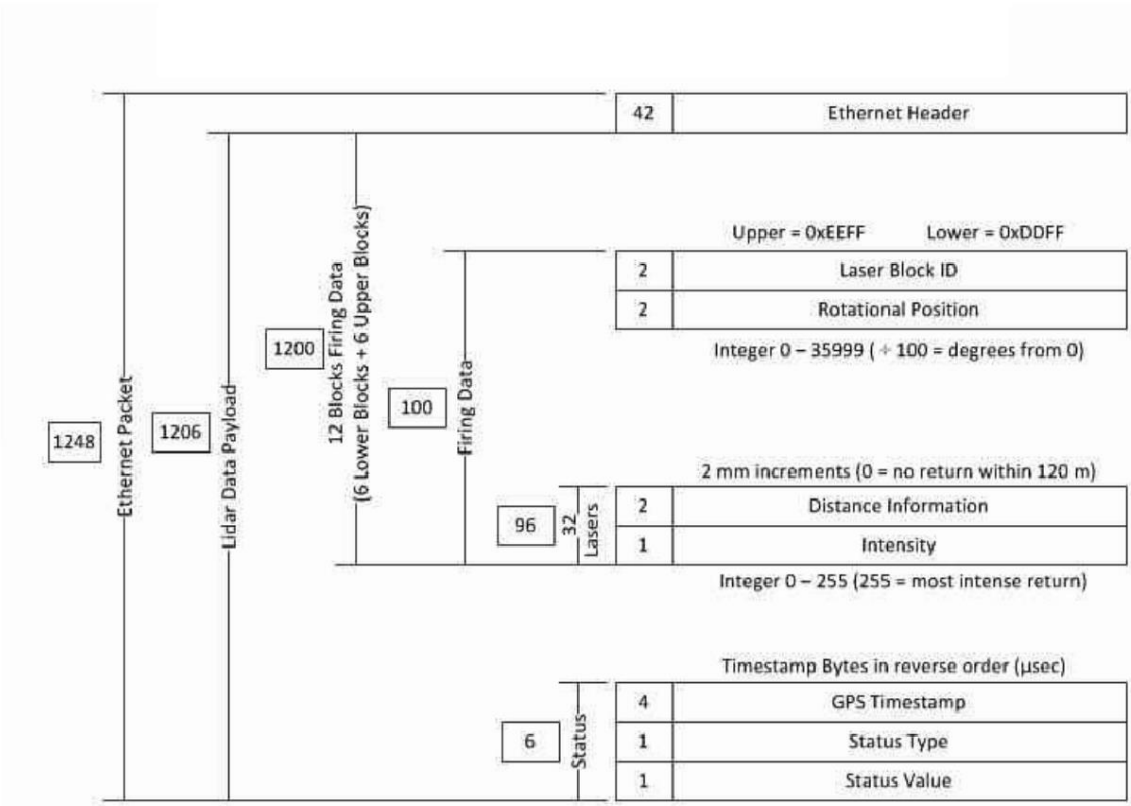
Velodyne provides software called Digital Sensor Recorder (DSR), whose installer files are on the CD that came with the unit. DSR is a 3-D point cloud visualization software program designed for use with the HDL-64E. DSR reads in the packets from the HDL-64E unit, performs the necessary calculations to plot the points presented in 3-D space, and plots the points on the viewer screen.

There are several disadvantages of using DSR to collect data. First, it collects all the data in a single .pcap file. If there is an interruption, all the collected data are lost. Second, DSR saves all the data sent from the LiDAR for full 360^{o} field of view. In fact, a reduced horizontal field of view (i.e., normally < 270 degrees) is needed in the considered application. If only useful readings are collected and stored while other readings are discarded, the processing speed and storage efficiency will be improved.

DSR is not an open source tool that can be modified and used for advanced analysis and development. Instead, open-source software should be used to collect data supplemented with MATLAB® and C++ to further process the collected data.

Wireshark and its filters

The DSR provided by Velodyne for capturing the LiDAR data on the host computer was replaced with open-source software Wireshark, which is a packet analyzer used for network troubleshooting, analysis, software and communications protocol development, and education. An example user interface of Wireshark is presented in Figure 5.2. There are some options provided by Wireshark that are useful when attempting to record data from the sensor. The various capture options are shown in Figure 5.3.



Status Type Rotates Through				Status Value Example		
Hex	ASCII	String	Description	Hex	ASCII	
48	72	H	Hours	15	21	21 Hours (9 pm)
4D	77	M	Minutes	3B	19	19 Minutes
53	83	S	Seconds	37	55	55 Seconds
44	68	D	Date	0F	15	15th day of the Month
4E	78	N	Month	0C	12	12th Month (December)
59	89	Y	Year	0A	10	2010
47	71	G	GPS	50	86	P = GPS has sync signal
54	84	T	Temperature °C	1B	27	Unit at 27°C
56	86	V	Firmware Version	E7	N/A	Version e7
31	49	1	Calibration & Unit Parameters (refer to sheet 2 of Datagram diagram)			
32	50	2				
33	51	3				
34	52	4				
35	53	5				
36	54	6				
37	55	7				

GPS Status Values		
Hex	ASCII	String and Definition
41	65	A = GPS receiving both sync signal and NMEA time command record
56	86	V = GPS receiving NMEA time command record only
50	80	P = GPS receiving sync signal only
00	00	0 = GPS not connected

LEGEND

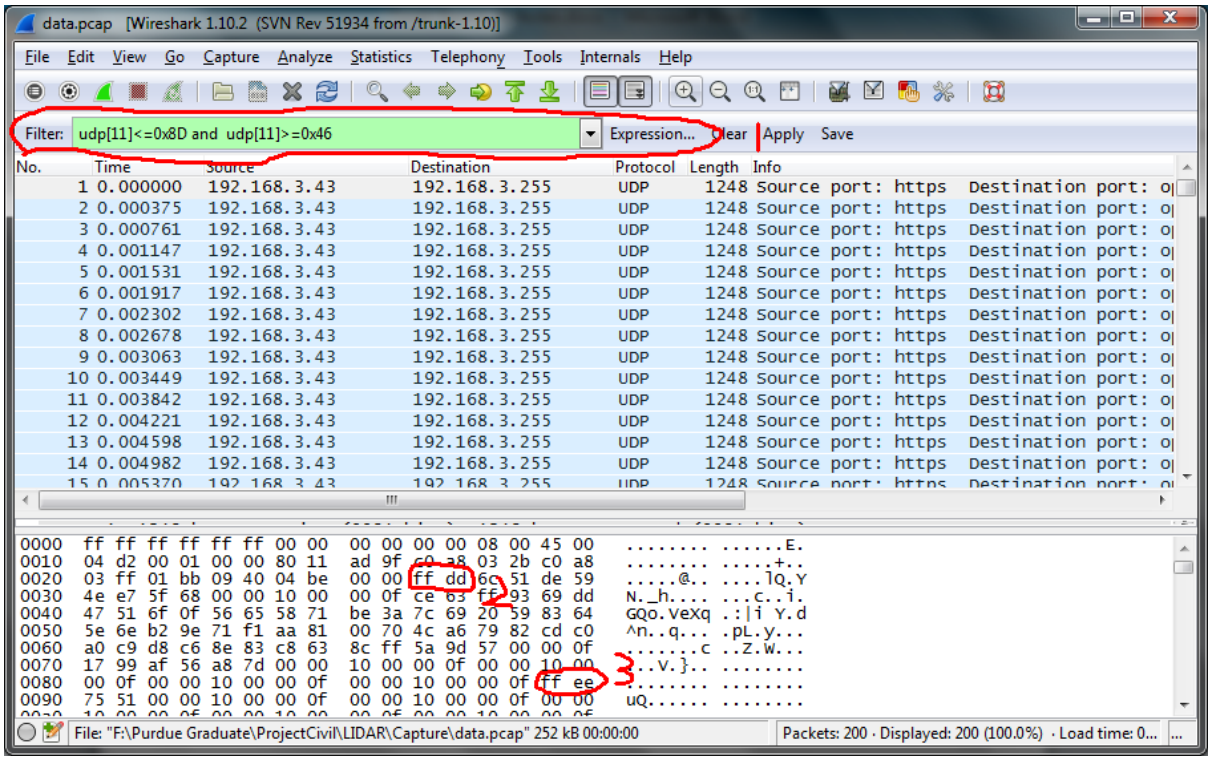
Bytes

32 bit unsigned integer
GPS Timestamp Example
92 18 52 D6

Reverse Order
D6 52 18 92

Convert to Decimal
3595704466 µsec
or
3595.704466 sec

Figure 5.1 User Datagram Protocol (UDP) Ethernet Packet Format: HLD-64E

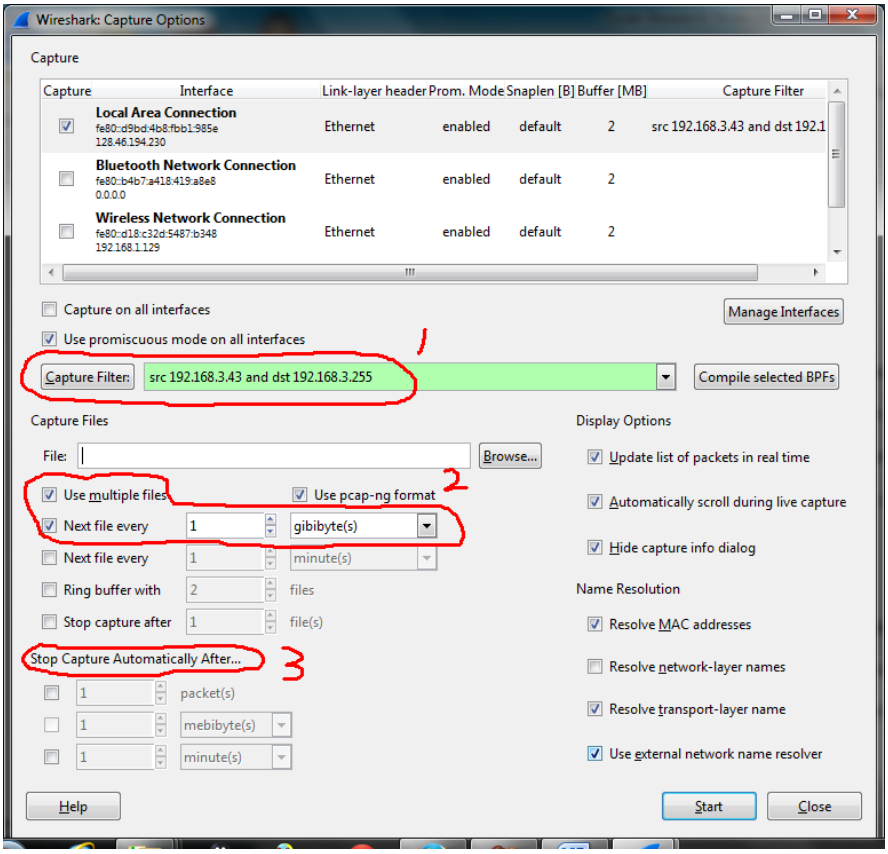


Notes: 1. Display filter; 2. Example data messages “DDFF” to determine the lower block; and 3. “EEFF” for the upper block)

Figure 5.2 Interface after opening a .pcap

The first set of options marked *Capture* in Figure 5.3 allows the user to choose the network interface from which the data will be collected. In our case, the LiDAR communicates with the host computer via an Ethernet; hence, that option is chosen (Figure 5.3). Also there is an option to mention the source IP and destination IP to filter out only the messages sent from the LiDAR (source) to the host computer (destination). This is useful where multiple devices are trying to communicate with a host computer in the same network. For example, in Figure 5.3 the capture filter is: src 192.168.3.43 and dst 192.168.3.255, which tells Wireshark to collect only data sent from 192.168.3.43 (LiDAR) to 192.168.3.255 (host computer) and to ignore any other device that may be sending data to the host computer via the Ethernet.

The second set of options is provided under the name *Capture Files*. Apart from specifying the location where the captured data must be recorded, there is a way to automatically split data into small .pcap packages by changing the “use multiple file” setting shown in Figure 5.3 and labeled as number 2.



Notes: 1. Capture filter; 2. Setting for how to split packages; and 3. Setting to terminate recording

Figure 5.3 Capture options in Wireshark

LiDAR readings do not always need to be captured from the entire 360° FOV and are only needed from a specified FOV defined via a horizontal angle. Wireshark has two kinds of filter applications: 1) the capture filter that is used for real time collection, and 2) the display filter that is used for post-processing. When the capture filter is chosen, the data that do not meet the criteria are discarded and not recorded at all. On the other hand, when the display filter is used, the entire incoming data are recorded and additional files are created that contain the data that meet the display filter criteria.

A description sentence entered by the user (similar to the "if" in programming languages) sets the criteria for collecting data. Choosing a specific FOV requires knowing the structure of the data packages described earlier in this section. Specifically, the angle information is needed to set the filter. We know that for a single package of 1,206 bytes, there are some bytes (two

bytes of rotational info) at fixed locations to indicate the angle position of the LiDAR. The bytes are at:

$$[(100 \cdot i)+3 \text{ and } (100 \cdot i)+4], \text{ where } i \in [0,11]$$

The value after the Hex to Dec transformation is between 0 and 35,999, and dividing that by hundred gives a value between 0 and 360. Considering the footnote, it is better to use only [100*i*+4] to determine the values because Wireshark cannot conduct Hex to Dec calculations.

From 0x00 to 0xFF, there are 256 values. We now assume all [100**i*+3] are 00. Table 5.2 lists the possible values of [100**i*+4,100**i*+3].

Table 5.2 Possible values of angles that may be present in the data from the LiDAR

Hex	[00,00]	[01,00]	[02,00]	…….	[8c,00]
Dec	0	256	512	…….	35840
Angle (^{o})	0	2.56	5.12	…….	358.40

We can use 2.56 degrees as the interval to select desired angle ranges. For example, if we would like an FOV from 0 to 120, first we need to transform the values to the nearest but larger range that can be divided by 2.56. Then, that value is multiplied by 100 and transformed to hex values

[0,120]→[0,120.32]→[0,12032]→[0x0000,0x2F00]

Take the first 1 byte of hex value and transform it back to decimal.

[0x00,0x2f]→[0,47]

Now we can construct a filter to select UDP packages that have the 4th byte inside the range of 0-47.

Filter: udp[11]>=0 and udp[11]<=47

After applying this filter, Wireshark will only collect or show the packages (depending on the filter chosen) when the LiDAR sensor is directed at the angle between 0^o and 120.32^o.

5.3 Background and Moving Objects Separation in Spherical Coordinates

A statistical analysis of the distance distribution for each laser-horizontal angle reading over a period of time is done to check if it is possible to distinguish between moving objects and the background based on the distance only.

If one assumes that the center of the LiDAR sensor is not moving (or its motion is negligible), then fixed background objects should remain at the same distance from the sensor in all frames. These objects include buildings, the road plane, and the vehicles parked during the data collection period. Continuous readings in the same direction may be a mixture of measurements of the background and of moving objects if moving objects are expected. In most such cases, a two-modal distance distribution is expected with much lower dispersion around a longer modal value representing the distance to the background. It will be very useful to find a distance threshold separating the background measurements from the moving object measurements. The moving objects include vehicles, pedestrians, and sometimes trees and lightweight objects moving in the wind.

We applied two steps to separate the background from the moving objects. In the first step, the LiDAR data were collected for several minutes in the .pcap file. The distance readings were grouped by similar directions in cells arranged in an array, and the distance distribution was analyzed for each cell. For example, the distance readings of laser block 7 at rotation angles between 62^o and 63^o are grouped in cell DistArray[7, 63]. If no moving objects were present in the [7, 63] direction during the data collection, then the distance readings in this cell would be distributed tightly around a single modal value.

Considering a possible measurement disturbance or slight motion of the sensor, a normal-like distribution might be expected for directions without moving objects. Our observations indicated that such directions could be identified. There are also directions where the distance readings exhibit two-modal distributions as expected when moving objects are temporarily present. It also should be noted that in the case of considerable motion of the sensor, the nominally-fixed direction by the sensor number and the horizontal angle is in fact not steady. A moving laser beam may hit different objects in different frames, leading to complex distance distributions.

A DistArray cell is considered a background cell if:

- at least 50 readings to provide data is sufficient for the analysis, and
- The distance reading range (maximum – minimum) is lower than two meters.

When tested on a sample data set of 23,040 cells, 7,752 cells were empty and 11,540 cells were classified as background only cells (76% of all the non-empty cells) and the other 3,748 cells were classified as cells with moving objects. Background cells are expected to be normally distributed but cells with moving objects are expected to be at least bi-modally distributed. In some cases, the distance difference between the background and objects is large enough to separate them; but in other cases, the objects are too close to the background and separation is much more difficult. A separation between the background and moving objects can be done by fitting two normal distributions. If the difference is too small, the points of the moving objects can be ignored. Example distributions are presented in Figure 5.4. For each cell, we estimated the mean and the variance.

In the second step, the data collected beyond the period of the initial several minutes were analyzed based on the distance distributions. If the distance was lower than the object threshold (mean – (n × standard deviation)), then the data point could confidently be classified as belonging to a moving object in front of the background. This classification allows reducing the number of points for further processing by focusing only on object readings (Figure 5.5).

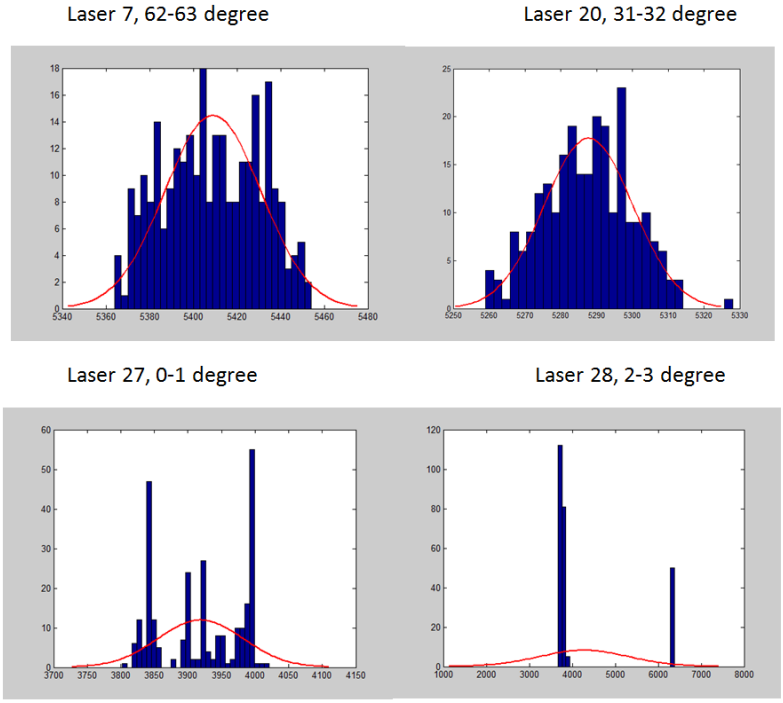


Figure 5.4 Sample data from spherical analysis

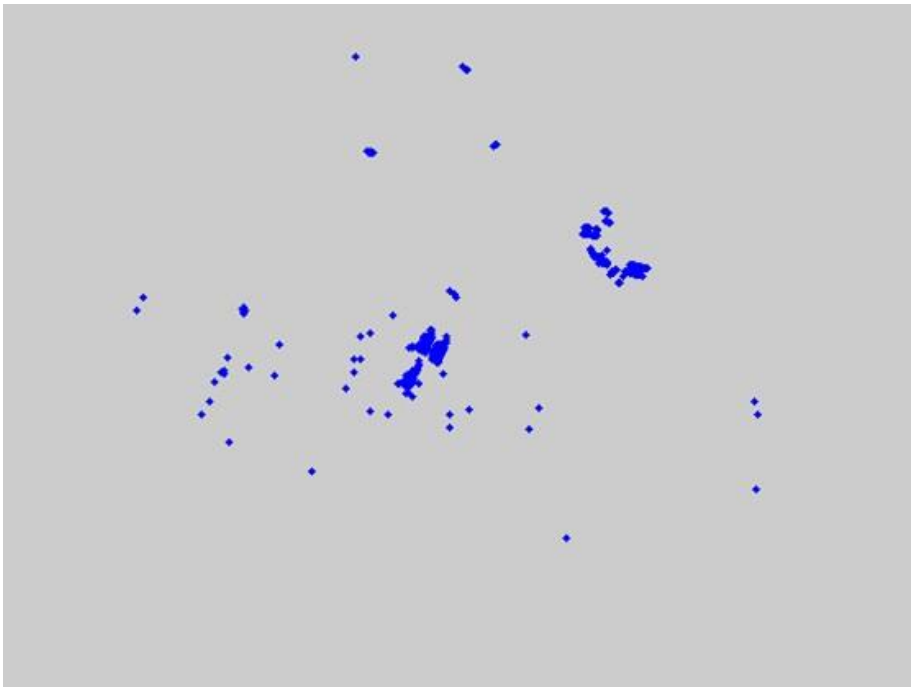


Figure 5.5 Example output of a frame with spherical analysis (blue points are assumed to be moving objects and isolated points are noises)

5.4 Transformation from Spherical to Cartesian Coordinates

The HDL-64E sensor collects three items of information for each reading: the rotation (horizontal) angle, the distance, and the intensity measured in the 64 laser blocks. The vertical angle is fixed and determined by the laser beam ID. In addition, a set of correction parameters specific for a sensor unit is provided by Velodyne.

The vertical and horizontal angles of individual beams and their corresponding distance readings can be viewed as measurements in a 3D system of spherical coordinates. Although the spherical coordinate system is efficient for storing data, it prohibits utilization of analysis techniques developed for the XYZ Cartesian coordinate system. The angles, distances, and calibration parameters are used to convert the spherical measurements into the corresponding XYZ orthogonal measurements.

Each HDL-64E laser is fixed with respect to the vertical angle and offset to the rotational index data provided in each packet. For each data point issued by the HDL-64E, rotational and horizontal correction factors must be applied to determine the point's location in 3-D space

referred to by the return. Each HDL-64E unit comes with its own unique XML file, called db.XML, that was generated as a result of the calibration performed at Velodyne's factory and uniquely stored in its CD disk. We need this XML file to calculate points accurately. The XML file also holds the key to interpreting the packet data for users whot wish to create their own interpretation and plotting routines. The db.XML file contains 64 sets of five adjustment values needed for the coordinate transformation. These adjustments are described in Table 5.3.

Table 5.3 Sensor calibration variables

Correction Type	Variable	Description
Rotational	β_i	This parameter is the rotational correction angle for each laser, as viewed from the back of the unit. Positive factors rotate to the left, and negative values rotate to the right.
Vertical	δ_i	This parameter is the vertical correction angle for each laser, as viewed from the back of the unit. Positive values have the laser pointing up, and negative values have the laser pointing down.
Distance	D_o^i	Each laser has its own unique distance due to minor variations in the parts used to construct the laser. This correction factor, in centimeters, accounts for this variance. This number should be directly added to the distance value read in the packet.
Vertical Offset	V_o^i	This value represents the height of each laser as measured from the bottom of the base. It is a fixed value for all upper block lasers and a different fixed value for all lower block lasers.
Horizontal Offset	H_o^i	This value represents the horizontal offset of each laser as viewed from the back of the laser. It is a constant positive or negative value for all lasers.

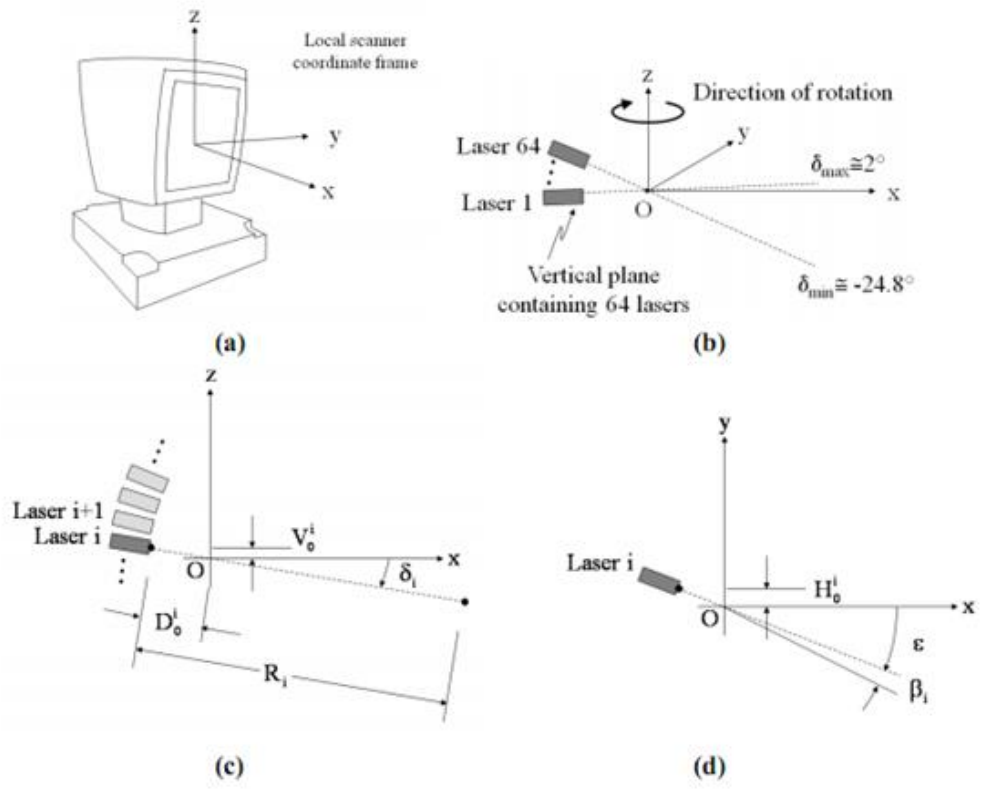


Figure 5.6 (a) sensor frame axes, (b) sensor layout, (c) scanner parameters in vertical plane, and (d) scanner parameters in horizontal plane

The transformation method used in our case was described in Glennie and Lichti (2010). The model is illustrated in Figure 5.6. The computation of the local scanner coordinates (x, y, z) for laser block i in the Velodyne LiDAR scanner is given by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (s^i \times R_i + D_o^i) \times \cos(\delta_i) \times \sin(\varepsilon - \beta_i) - \mathrm{H}_o^i \times \cos(\varepsilon - \beta_i) \\ (s^i \times R_i + D_o^i) \times \cos(\delta_i) \times \cos(\varepsilon - \beta_i) + \mathrm{H}_o^i \times \sin(\varepsilon - \beta_i) \\ (s^i \times R_i + D_o^i) \times \sin(\delta_i) + \mathrm{V}_o^i \end{bmatrix} \quad \textbf{Equation 5.1}$$

Where:

s^i is the distance scale factor for the laser i;

R_i is the raw distance measurement from lase i;

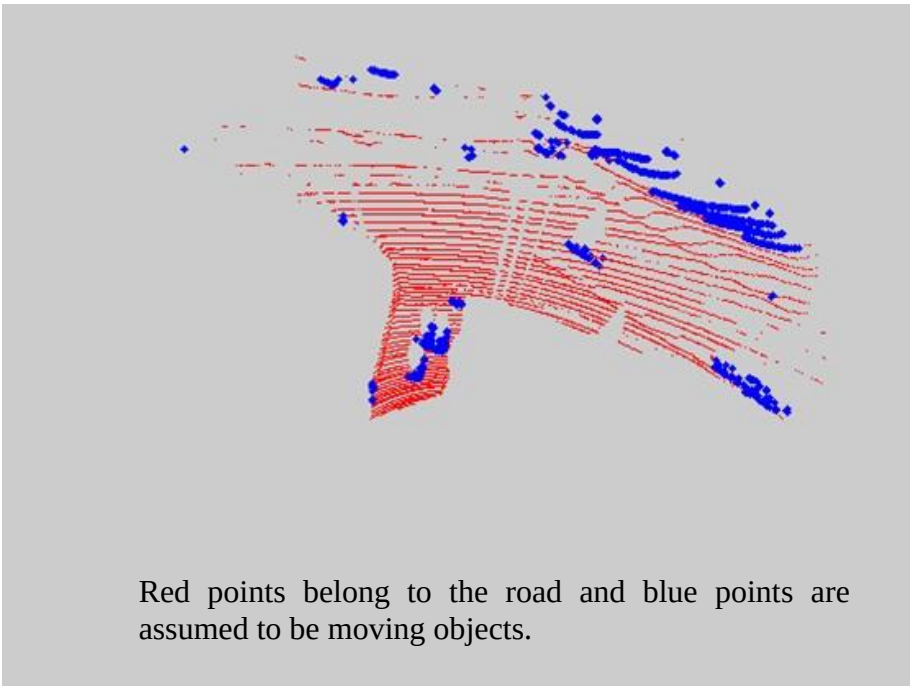
ε is the encoder angle measurement;

D_o^i, δ_i, β_i, H_o^i and V_o^i parameters pertaining to laser i as explained in Table 5.3.

5.5 Data Analysis in Cartesian Coordinates

At its core, the method of separating background readings from moving object readings in the Cartesian coordinate system uses the same principle as the method in the spherical coordinate system. We first apply spherical-XYZ coordinate transformation for all laser readings and generate the point clouds in 3D space. Then, we analyze the location of a point in 3D space to determine whether it belongs to the background or to a moving object. The following steps are followed:

1. Select a polygon area of the intersection to determine the data points that will be analyzed (points inside the polygon).
2. Use a MATLAB® algorithm to fit the initial intersection plane to the data.
3. Calculate the distance between each point in the polygon area and the fitted plane.
4. Use the points that are in the vicinity of the plane (e.g., within 0.5 meter) to redo the plane fitting.
5. Points that are between 0.4 meter and 5 meters above the intersection plane are moving objects.



Red points belong to the road and blue points are assumed to be moving objects.

Figure 5.7 Example output of the same frame of Figure 5.5 with XYZ analysis

Choosing a polygon for the area of interest

The attempt to identify moving objects should focus on the road intersection area (or pedestrian walk area). One option is to choose the vertices of a polygon in a 2D projection of the data points to identify the area of interest (see the top view in Figure 5.8). Those vertices should well describe the intersection area of interest.

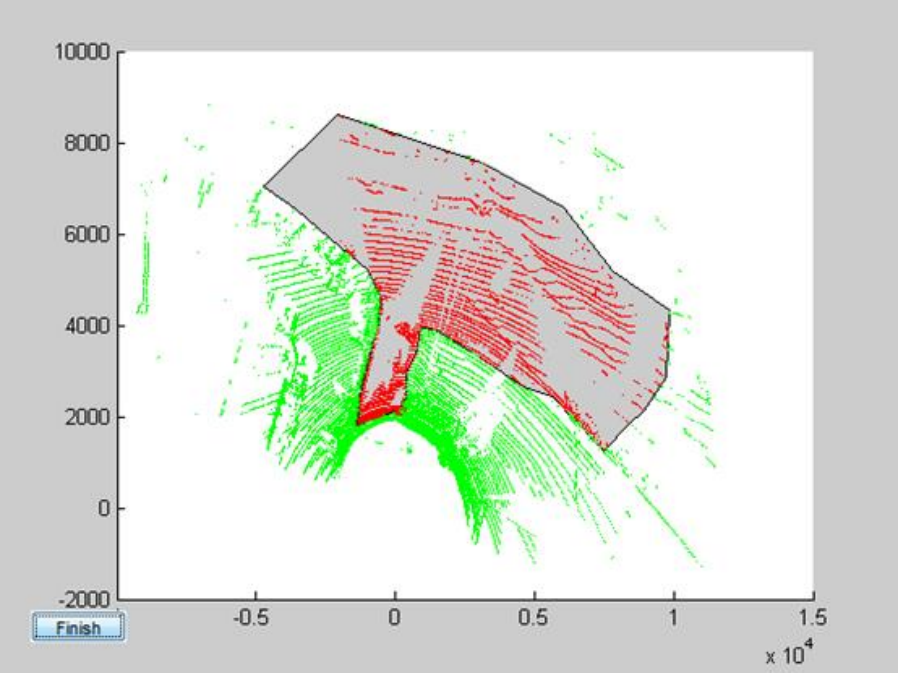


Figure 5.8 Choosing a polygon of interest during XYZ analysis

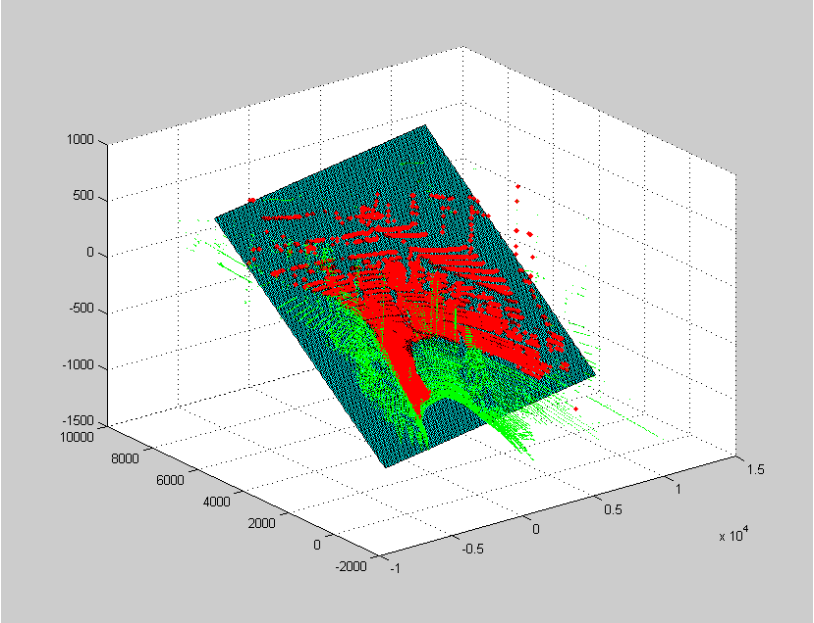
Fitting the Intersection plane

In most cases, an intersection may be assumed as a plane that can be estimated with a linear regression model focused on the best estimation of the *z* coordinates:

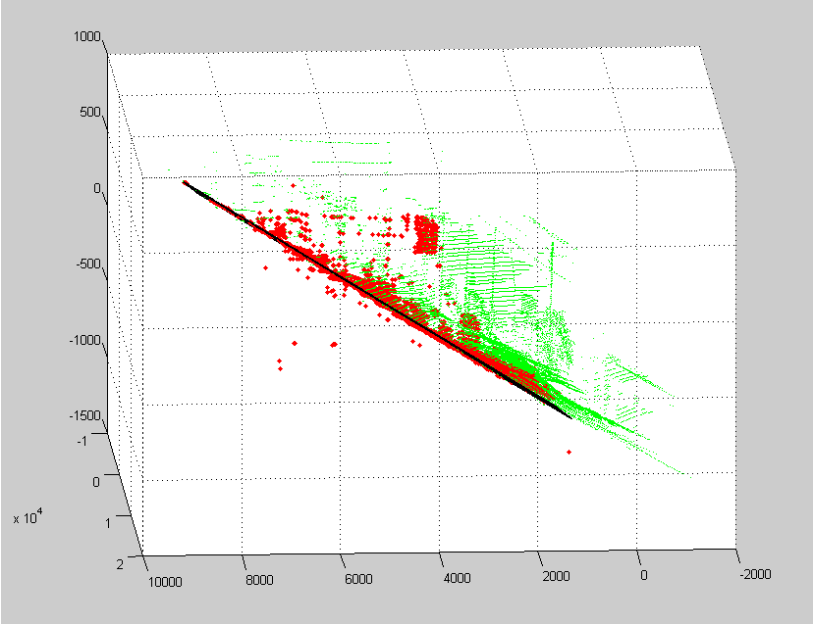
$$z_i = \beta_0 + \beta_1 \cdot x_i + \beta_2 \cdot y_i + \varepsilon_i \qquad \textbf{Equation 5.2}$$

where: $\beta_0, \beta_1, \beta_2$ are the plane parameters and ε_i is the error term.

The data points inside the intersection polygon are used to fit the intersection plane. Figure 5.9 presents the points classified as background points. The color of the points indicates whether they are below the intersection area (red) or not (green). Figure 5.10 depicts the estimated intersection plane.

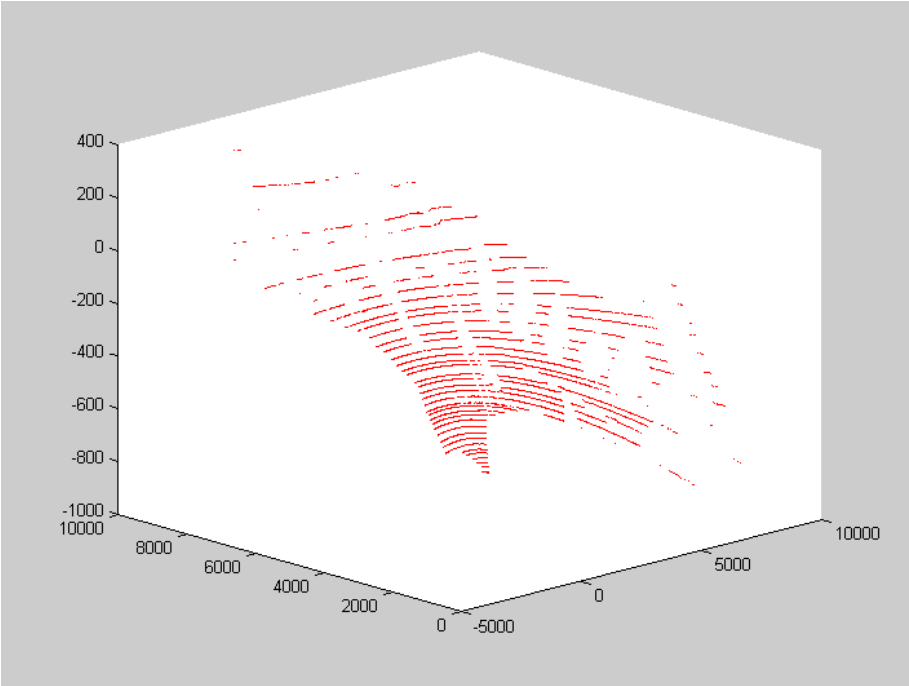


(A) **Isometric view**

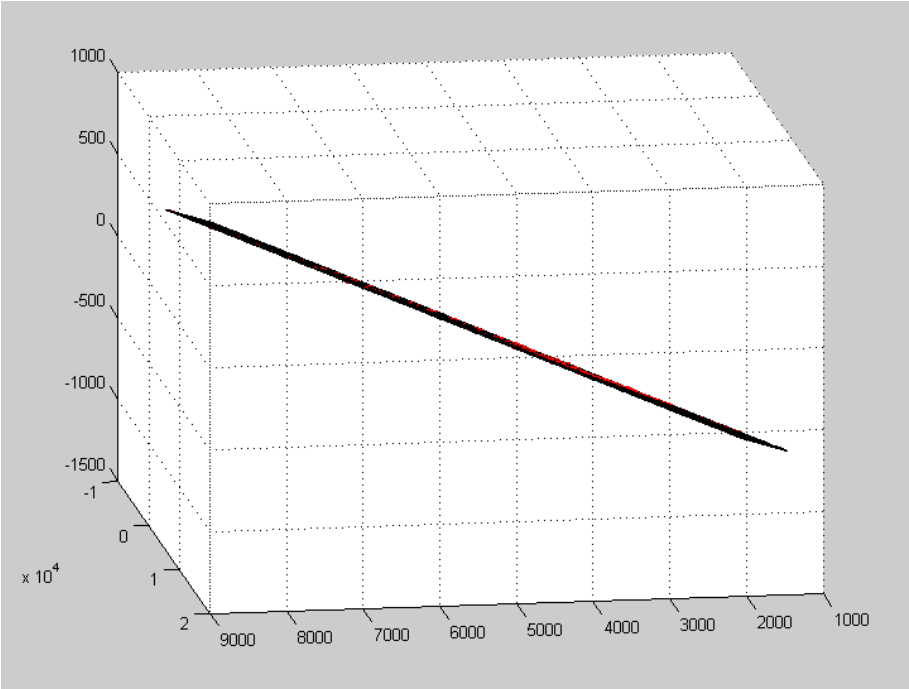


(B) Side view

Figure 5.9 Raw data that constitutes a frame (red points belong to the intersection area, green points do not)



Isometric view



Side view

Figure 5.10 Results of fitting the interaction points to the plane in a single frame

5.6 Conclusions

The analysis of the distance measurements in the same "nominal" direction (laser, horizontal angle) in the spherical coordinate system indicated the measurement variability was larger than expected for a fixed background and the error specifications by Velodyne. A laser beam drift (limited to one degree) is plausible explanation. A beam could hit different objects in different frames, but it may not be the only explanation. Other possible explanations are the motion of the sensor caused by the wind, ground vibration caused by heavy trucks, or vibration of the rotating sensor. This problem can be addressed by adding an adjustment for the sensor movement. Two approaches are taken into account: 1) adjustment with data from additional motions sensors attached to the LiDAR base and 2) adjustment with a self-calibration procedure based on the detection of non-moving objects.

Separation of the background from moving objects based on the distance measurements is a promising new technique not possible in the past with video data. It helps reduce the data by eliminating the background points from the identification and tracking moving objects.

Separation of the background from moving objects in the Cartesian coordinate system based on the z coordinate value related to the intersection plane (surface) is also a promising technique that is further specialized to the specific class of objects measured in the considered application.

A combination of the two techniques can be expected to produce even better results. First, the raw readings are converted to the Cartesian coordinate system and adjusted for the motion of the LiDAR sensor. Then, the distribution of the z coordinates in multiple frames is analyzed to separate the background points from other points. Only the background points within the intersection polygon are used to fit the intersection surface model to the background data. Furthermore, multiple planes could be assumed in a piece-wise modeling (or triangulation) of the intersection surface to more precisely identify the local z thresholds for moving objects.

A general problem with the LiDAR sensor is that some laser beams are not returned back because the light bounces away from the sensor on the reflective surfaces of vehicles or on wet or icy pavement areas. This problem results in the loss of already rather sparse data, and this particular drawback of the laser sensor is difficult to remedy. The only solution is to use the laser data as efficiently as possible and look for supplementary sources of data.

6 Stabilization with Motion Sensor Data

In this report, the term *frame* refers to the data captured by the LiDAR in one complete revolution. The LiDAR has its own origin inside its body with respect to which all the data are measured. Therefore, when the LiDAR is not fixed, the origin from which the distance is measured also moves. To improve the accuracy of measurements, they must be adjusted for the sensor motion, which can be accomplished by establishing a global frame of reference and mapping the data from all the other frames to that global frame.

The main sources of disturbances that cause the LiDAR to move are rotation of the LiDAR that cause the mast to sway, vibrations from the collection vehicle's engine and other mechanical components, movement of humans in and out of the vehicle, and wind movement that causes the mast to sway.

In order to get cleaner data that are based on a single established frame of reference, we propose a methodology by which we can automatically apply the necessary corrections to each frame and thereby improve the accuracy of the estimated positions and velocities of vehicles.

6.1 IMU – Inertial Measurement Unit

In order to determine the change in orientation and position of the origin of the LiDAR, a sensor is needed that can track motion in the six degrees of freedom. An IMU, which is an electronic device containing a gyroscope, an accelerometer, and/or magnetometer, is used for this purpose. The IMU can report the velocity and orientation of the object to which it is attached. The accelerometer measures the accelerations being experienced by the LiDar in all three directions and the gyroscope provides angular velocity information. The IMU used for the experiment is Microstrain 3dm-gx2. The specifications of the sensor are available on the vendor's website (Microstrain, 2014). To derive the position and angular orientation from the linear accelerations and angular velocities provided by the IMU, a Kalman filter is used.

6.2 Kalman Filter

The Kalman filter is an algorithm that uses a series of measurements observed over time that contain noise (random variations) and other inaccuracies. It produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone. In other

words, the Kalman filter operates recursively on streams of noisy input data to produce a statistically optimal estimate of the underlying system state.

The algorithm works in a two-step process. In the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties. Once the outcome of the next measurement is observed, these estimates are updated using a weighted average, with more weight being given to estimates with higher certainty. Because of the algorithm's recursive nature, it can run in real time using only the present input measurements and the previously calculated state and its uncertainty matrix; no additional past information is required.

Kalman's original paper derived the filter using orthogonal projection theory to show that the covariance is minimized, and this result does not require any assumption (e.g., that the errors are Gaussian) (Kalman, R. E., 1960).

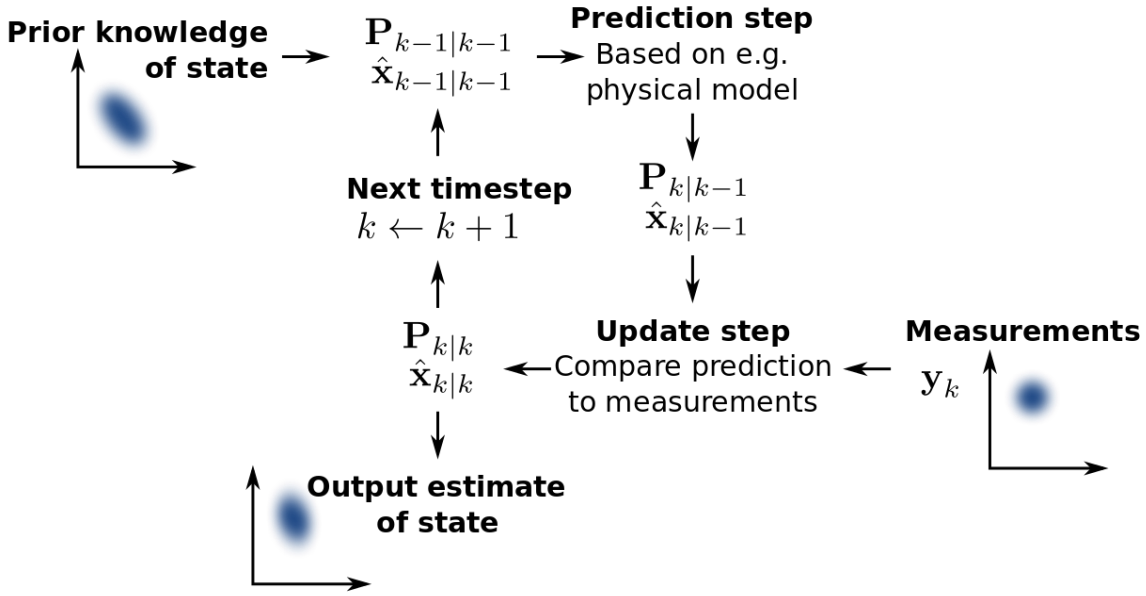


Figure 6.1 Kalman Filter overview

6.3 Data collection and applying the Kalman filter

The Kalman Filter loop is split into two parts, one for the accelerations/displacements system and the other for the angular rates/angles. We demonstrate the filter for accelerations on three directions with the assumption of a dominant frequency of 10Hz (i.e., that there is an oscillation at a frequency of 10 Hz. This oscillation in our experiments was induced by the rotation of the

LiDAR. at that frequency. However, if wind-induced vibrations at lower frequencies exist; a different frequency can be used. Indeed, it is possible to extend the Kalman filter to track the dominant frequency.

$$\begin{aligned} p_X &= -a_X / \omega_x^2 \\ p_y &= -a_y / \omega_y^2 \\ p_z &= -a_z / \omega_z^2 \end{aligned} \qquad \textbf{Equation 6.1}$$

The above equation is used for oscillation to represent the characteristics of the system in the Kalman filter. Here ω_x is the frequency of oscillation about the x axis.

The state vector for the Kalman filter loop is

$$X_k = \begin{pmatrix} a_x & a_y & a_z \end{pmatrix}^T \qquad \textbf{Equation 6.2}$$

The following equations are used in the Kalman filter loop

Difference Equation $$X_{k+1} = \Phi X_k + Q_k \qquad \textbf{Equation 6.3}$$

Measurement Equation $$Z_k = H_k X_k + R_k \qquad \textbf{Equation 6.4}$$

Where

$$\Phi_k = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, H_K = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \textbf{Equation 6.5}$$

The magnitude in each direction is obtained by calculating the standard deviation value of the entire signal. The noise attributes are determined by the initial estimation of the entire system`s shaking magnitude, which is revealed in the matrix of Q.

$$E[\nu_k \nu_k^T] = Q_k = \begin{pmatrix} \sigma_x & 0 & 0 \\ 0 & \sigma_y & 0 \\ 0 & 0 & \sigma_z \end{pmatrix} \qquad \textbf{Equation 6.6}$$

Where σ_x, σ_y and σ_z are the standard deviation in the rate of change of acceleration along the coordinate axes x, y and z, respectively, and ν_k is noise.

The measurement noise magnitude is obtained by examining the natural attribute of the inertial sensor, which could be done by either looking at the specifications or by re-calibrating the noise level. The attributes are revealed in the diagonal segments of the R matrix. If w_k is the measurement noise, then the R matrix is given by

$$E[w_k w_k^T] = R_k = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.0001 \end{pmatrix} \qquad \textbf{Equation 6.7}$$

The properties of the P matrix require an initial estimation as a starting point, which can be done by a rough estimate and is adjustable, although the system output does not rely heavily on the initial estimate as long as the initial estimate is reasonable. Having filtered the acceleration signals with the KF loop, we directly estimate the displacement with the equation of oscillation (as shown in Equation 6.1).

The same strategy is applied to the KF loop of the angular rates/angles, with the major difference being the system transition matrix. With the new properties of the P, Q, R matrices and the measurement data string, the output is capable of directly illustrating the angles in three directions.

Let the state vector X_k contain the angle for which the Kalman loop is written and the angular rate. The three angles that we need to determine are alpha (α), beta (β), and gamma (γ). For the sake of explanation, let the angle be generally denoted using theta (θ).

$$X_k = \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} \qquad \textbf{Equation 6.8}$$

The state transition model that is applied to the previous state x_{k-1} to obtain x_k.

$$\Phi = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \qquad \textbf{Equation 6.9}$$

In this case, the sampling rate of the IMU is 0.01, which is used in the actual calculations. Therefore, the state transition difference equation is

$$X_{k+1} = \Phi X_k + w_k \qquad \textbf{Equation 6.10}$$

Where, w_k represents the process noise, which is Gaussian distributed with a zero mean and with covariance Q at time k, Q_k represents the process noise covariance matrix.

Similarly, if the measurement (i.e., the observation of the true state is denoted by the vector Z_k, then the value it takes is given by the following equation

$$Z_{k+1} = H_k X_k + v_k \qquad \textbf{Equation 6.11}$$

Where H_k refers to the measurement model that maps the true state space into the observed space. In this case, since we are directly measuring the angular rates, the structure of H_k is

$$H_k = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \qquad \textbf{Equation 6.12}$$

Where v_k refers to the measurement noise and is assumed to be a zero mean Gaussian noise with covariance R at time k. The structure of Q_k and R_k are given as shown

$$Q_k = \begin{pmatrix} 0.1 & 0 \\ 0 & \sigma_\theta \end{pmatrix} \qquad \textbf{Equation 6.13}$$

$$R_k = \begin{pmatrix} 0.01 & 0 \\ 0 & 0.1 \end{pmatrix} \qquad \textbf{Equation 6.14}$$

These are essential constants which must be within a reasonable range in order for the Kalman filter to function.

Kalman Filter Loop

Now that the initial setup has been explained, the next step is to implement the Kalman loop. The loop has two stages, the prediction stage and the updating stage.

In the prediction stage, the Kalman loop will try to predict the current state and the error covariance matrix at time k. First, the current state is estimated based on all the previous states and the gyro measurement, which is given by the following difference equation

$$X_{k+1|k} = \Phi X_k \qquad \textbf{Equation 6.15}$$

Then, weestimate the a priori error covariance matrix P_{k+1} based on the previous error covariance matrix P_k , which is defined as:

$$P_{k+1} = \Phi P_k \Phi^T + Q_k \qquad \textbf{Equation 6.16}$$

This matrix is used to estimate our level of trust for the current values of the estimated state.

The update stage is next. First, the difference between the measurement Z_k and the a priori state $X_{k+1|k}$, is computed. This is also called innovation

$$y_{k+1} = Z_k - HX_{k+1|k} \qquad \textbf{Equation 6.17}$$

The matrix H is then used to map the a priori state into the observed space, and the innovation covariance is calculated

$$S_{k+1} = HP_{k+1}H^T + R \qquad \textbf{Equation 6.18}$$

This innovation tries to predict how much we should trust the measurement based on the a priori error covariance matrix P_{k+1} and the measurement covariance matrix R. The observation model H is used to map P_{k+1} in the observed space. The greater the value of the measurement noise, the greater the value of S, and the less we should trust the measurement.

The next step is to calculate the Kalman gain, which is used to indicate the level of trust we have for the innovation and is defined as:

$$K_{k+1} = P_{k+1}H^T S_{k+1}^{-1} \qquad \textbf{Equation 6.19}$$

It can be seen that if our trust in the innovation is low, the innovation covariance S will be high; and if we trust the estimate of the state, then the error covariance matrix P will be small, and the Kalman gain will therefore be small and vice versa.

Now we can update the a posteriori estimate of the current state,

$$X_{k+1|k+1} = X_{k+1|k} + K_{k+1}y_{k+1} \qquad \textbf{Equation 6.20}$$

Thus, we retrieve the position (x, y, z) and the orientation (α, β, γ) of the LiDAR from the IMU. These values are then used to determine the homogenous transformation matrix that will be discussed in Section 7.9.

6.4 Inspection of the results with IMU-based adjustments

The above method was applied to sample data collected. The IMU unit used was Microstrain 3dm-gx2. Results of the experiment are presented in Table 6.1. The values Rx, Ry, Rz represent the rotations around the x, y, z Cartesian axes and Tx, Ty, Tz represent the translations about the x, y, z Cartesian axes, respectively.

Table 6.1 IMU method

	Rz (degrees)	**Ry (degrees)**	**Rx (degrees)**	**Tx (cm)**	**Ty (cm)**	**Tz (cm)**
Mean	0.0011	0.0022	0.0008	0.0006	0.0643	0.0052
High	0.00371	0.00484	0.00304	0.2797	0.5083	0.2927
Low	0.00001	0.00001	0.00000	0.1329	0.3813	0.1190

From Table 6.1, we can see that the maximum translation that the LiDAR undergoes is 5 mm in the y axis. This agrees with what was observed visually when the data were being recorded. There was little to no wind, therefore, the reasons for the motion of LiDAR are predominantly the rotation of the LiDAR about its own axis and to a minor extent disturbances caused by human movement inside the van. The mean translation being close to zero indicates that the motion is an oscillation, which validates our assumption made when designing the Kalman filter.

7 Stabilization through Self-adjustment

Adding the IMUs makes the hardware more complex and also makes it more expensive. An IMU with a low in-run bias and temperature coefficient can cost upwards of $1,000 (Analog Devices, 2014). An alternative method was considered that attempts adjusting the readings with the measurements of the planes that pertain to the background. This self-adjustment (or self-calibration) method does not use any extra hardware, but it will increase the computational load.

7.1 Representation of an object in LiDAR data

Any object that is recorded by a LiDAR is represented as a cloud of points (point cloud) in 3D space. Each object is represented by only a certain number of points, and that number depends on the distance of the object from the LiDAR and the size of the face of the object facing the LiDAR. This means that the same object can be represented by a varied number of points in each frame. Therefore, the object cannot be defined based on the number of points in the point cloud data.

Also, the points that represent the same object in different points in time (across frames) are not the same physical points. In other words, the LiDAR lasers do not hit the same physical points on the object in every rotation. If a mid-point of the data representing the physical object is determined, that mid-point appears to drift/oscillate even if the object is perfectly stationary in real life.

The reason for the drift might be two-fold. First, it might be due to the sensor itself moving. Second, it may happen because a different set of points are being scanned every single rotation. If we can eliminate the effect of different points being scanned every time from a stationary object, then the "drift" of the object in the LiDAR data would only be due to the motion of the sensor.

7.2 Tracking Planes

As stated above, the usage of the mid-point of the data does not cancel the effect of drift. Hence a different averaging mechanism was needed that could nullify the effect of the different points being scanned each time.

If we can fit a plane to the face of the object that is being represented in the data, then the equation of the plane would not change at all, provided the LiDAR was scanning points from the same plane. Thus, by tracking constant planes (faces of stationary objects), the effect of different points being scanned each rotation can be nullified. Then, the apparent motion of that object only would be due to the movement of the LiDAR.

Therefore, we chose tracking the entire face of the object that is visible to the LiDAR by considering the face as a plane. A schematic of the current approach adopted to solve the LiDAR self-calibration is shown in Figure 7.1

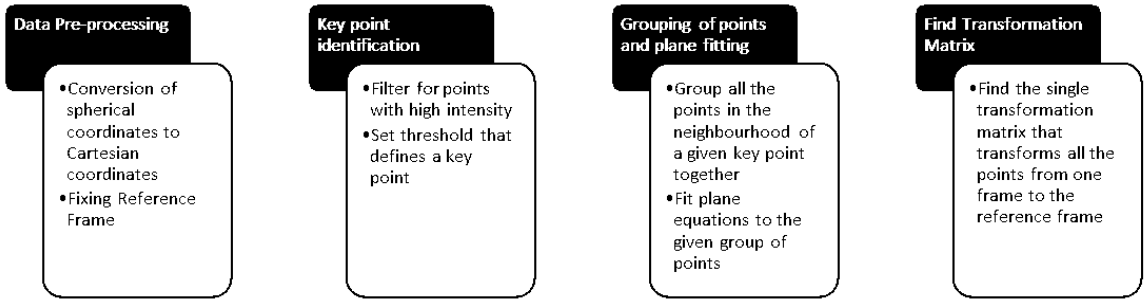


Figure 7.1: Schematic of the methodology employed for LiDAR self-calibration

7.3 Fixing Reference Frame

We first needed to fix the reference frame to which we were mapping all the data from each frame to which it is being mapped. The reference frame was chosen from the data, which avoided calculating a transformation matrix for that frame, and the reference frame was well defined as well. Moreover, the additional requirement to validate the transformation matrix to an arbitrarily fixed imaginary reference frame was avoided.

The sensor returns the data over the Ethernet using Universal Datagram Protocol (UDP). The software provided by the manufacturer saves the recorded data as a .pcap file. The structure of

the data that is present in this file is explained in the Velodyne user manual (2014) and in Section 5.2.

Since the sensor returns data in spherical coordinates; we had to convert it into XYZ Cartesian coordinates, for it was easier to analyze the velocity in Cartesian coordinates than in spherical coordinates. The sensor model used for the conversion from spherical to Cartesian coordinates is explained in (Glennie and Lichti, 2010) and also in Section 5.1.

After observing the data from the LiDAR, it was found that the first frame that was obtained was incomplete. The second frame, on the other hand, was always complete. This happened because , the data from the current packet sent by the LiDAR was the first to be recorded when the record button was pressed. This may not correspond to the zero angle of the LiDAR and therefore the frame was incomplete.

After the first frame was completed, the second frame captured had the complete 360 degree information. Thus, this was the first full frame, and we therefore chose this as our reference. We plan to map all future frames to this reference. In other words, we will be transforming data from all other frames to this frame. This will also be referred as *frame* 0.

7.4 Markers

The solution to the LiDAR self-calibration problem is in determining the transformation matrix for each frame (*frame x*) which transforms it (*frame x*) to the reference (*frame* 0). There are six parameters that needed to be identified in order to find the transformation matrix: the translations (t_x, t_y, t_z) and rotations (α, β, γ) around the X,Y, and Z axes.

As stated above, this transformation is possible by tracking stationary objects that are fixed in space across the frames. The apparent motion which then appears from frame to frame will be due only to the motion of the LiDAR. Therefore, the primary objective is to identify stationary objects and track them in LiDAR's data.

As a first step, in order to aid achieving a faster solution, markers (retro-reflective traffic signs) that were capable of reflecting light without much loss in intensity were used. These markers were arbitrarily placed around the field at an angle to the incident rays from the LiDAR. The size of the markers used for this experiment was 2 ft x 2 ft.

Since we have firsthand knowledge that these markers are fixed at a particular location, any motion of this marker that appeared across the frames in the data provided by the LiDAR would be due to the motion of the LiDAR itself. By tracking these markers, we determined the transformation matrix for each frame which transforms that frame to the chosen reference frame. In our test case, we manually placed markers; but in the future, we hope to develop a robust method whereby "markers" are identified in an autonomous fashion.

7.5 Marker Identification in LiDAR Data

As stated above, we first tried to identify static objects across frames. For a typical traffic junction, these objects were fixed structures like poles, walls, retro-reflective traffic signs, etc. These objects typically do not change their position in the physical world with respect to time. Also, objects such as traffic signs, because of their retro-reflectivity, and snow reflect light back with minimal loss of intensity. As stated above, for our test case, we placed retro-reflective traffic signs manually to act as markers.

Constraints

We took advantage of the fact that the manually-placed markers are retro-reflective and searched for a group of points having high intensity returns (*intensity constraint*). Also, there was an additional constraint that the points must be close to each other (i.e., the maximum allowed distance between two points in the group would be less than the largest diagonal of the marker (*spatial constraint*)). This value was set in the variable *threshold neighbor*. Using these two constraints, we identified the markers. The procedure is explained in the remainder of this subsection.

Filtering based on intensity

We first filtered each frame such that only the points with high intensity were chosen for analysis. This was accomplished by iterating over each point in each frame and checking whether its intensity of return value was greater than a certain threshold (*threshold intensity*). These values were at fixed locations in the data. Please refer to Section 5.2 for the structure of the data.

In our example, we set this value, *threshold intensity*, which defines the minimum intensity a point is expected to have to be considered a point with a high intensity return, as 200. The point

with maximum return intensity had a value of 255, so we therefore imposed the *intensity constraint*.

A program that scans through each frame (each frame is a separate file with the extension .asc), filtered out the points that had an intensity return greater than that of *threshold intensity* and stored these points in a new file for each frame.

Figure 7.2a shows a particular frame viewed using the software provided by the vendor, DSR. After applying the filter stated in this section, the resultant frame is seen in Figure 7.2b.

Identifying the Markers

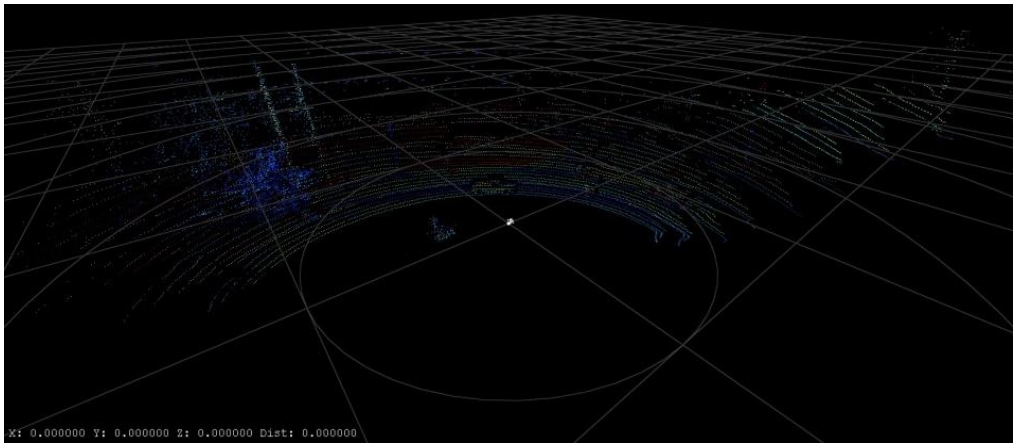
For the *spatial constraint*, we needed a method by which we could cluster a group of points that would represent the marker in its complete form as it appeared to the LiDAR with minimal noise. In order to achieve this, we first needed to identify where the marker was in the data from the LiDAR. Since we knew that the markers likely would have the highest concentration of high intensity points, we tried to determine how many neighbors each high intensity point had.

The neighborhood of a point is a sphere whose center is the point itself and the radius is given by the *threshold neighbor*. For a given point, say *x*, any other point, say *y*, is in its neighborhood, then *y* is said to be a neighbor of *x* and vice versa. For each point in the frame, the number of neighbors it has is calculated. The variable *max neighbor* contains a value that indicates the maximum number of neighbors any point has in a particular frame. A *key point* is one that has a percentage of *max neighbor* as the number of neighbors. We presume that each *key point* is a part of a marker.

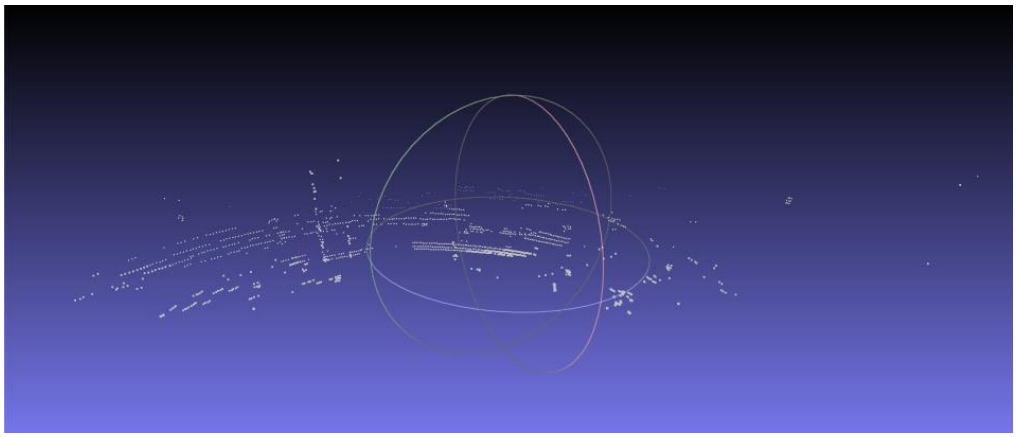
7.6 Grouping Points

Each *key point* and its neighbors together define a physical object. Hence, these points are grouped in such a way that each group of points represents an object. There may be more than one *key point key point* identified for the same physical object, and there may be one or more *key points* in the neighborhood of another *key point*. It is therefore necessary that the points are grouped together in such a way that each group uniquely represents a physical object.

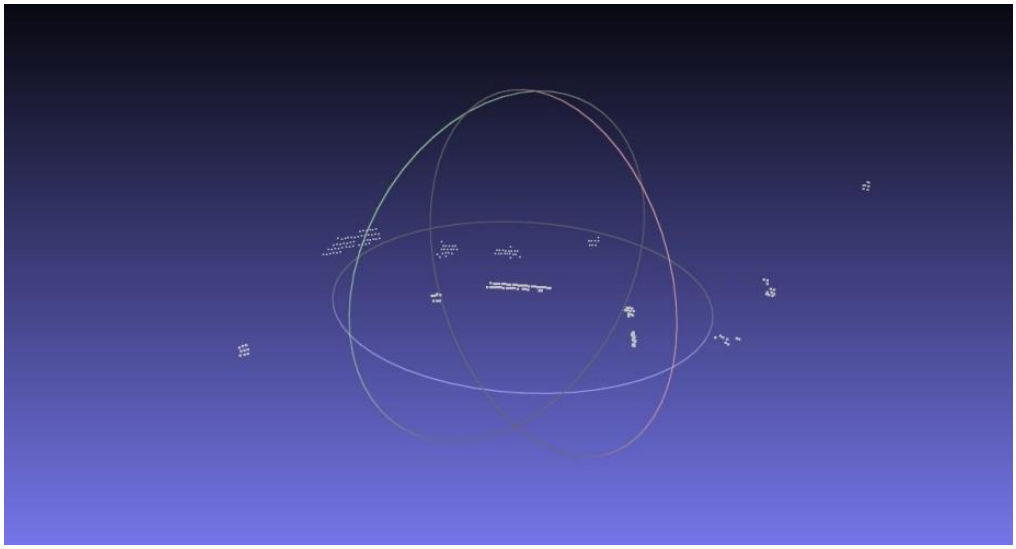
The above grouping was achieved by iterating over each *key point* and its neighbors. If two *key points* have a common neighbor, then they belong to the same physical object and is the same when there is another *key point* in the neighborhood of the *key point* in question.



(a) Raw data from sensor rendered using the vendor's software

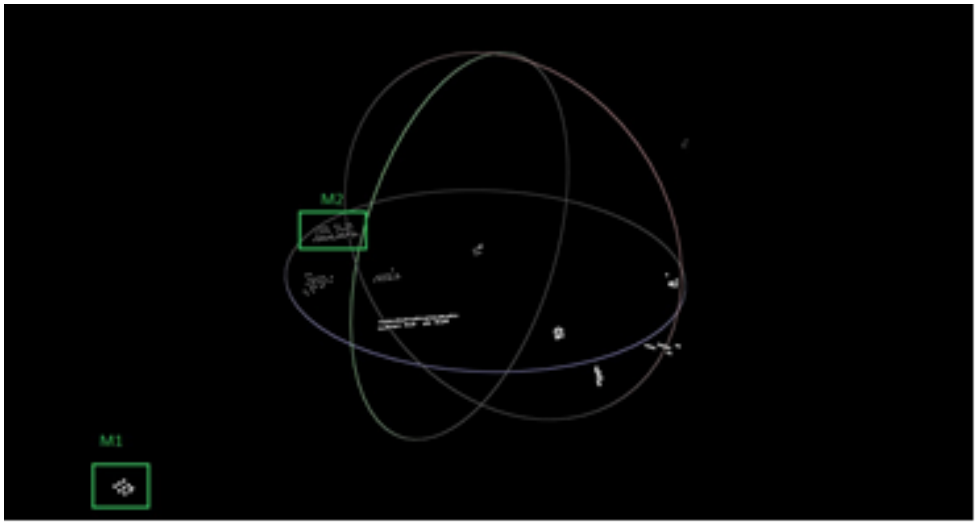


(b) Frame showing only points with high intensity, rendered using Meshlab

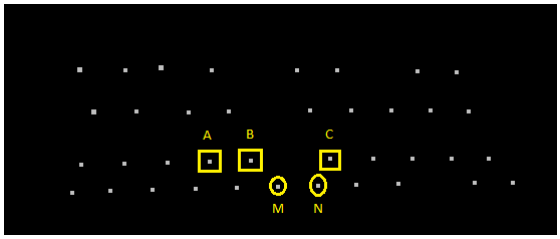


(c) Frame showing logical planes grouped together

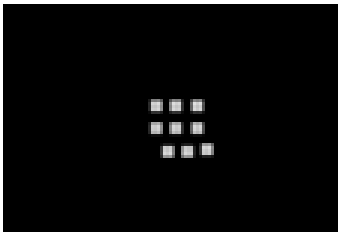
Figure 7.2 Results from the Sample Data



(a) Markers identified in reference frame



(b) Marker M2



(c) Marker M1

Figure 7.3 Illustration of grouping of points to represent markers

Figure 7.3 illustrates the points discussed previously. The close up view of marker M2 is shown in Figure 7.3b. The highlighted points A, B, and C represent the *key points*, and M, N represent the common neighbors to *key points key_points* B, C. It is apparent that *key point* A had another *key point* B as its neighbor. Therefore, it is logical to merge the neighbors of both *key points* to represent a single object. Since the *key points* have common neighbors M and N, the two *key points* and their neighbors are also grouped together. The marker shown in Figure 7.3c is a planar marker M1, with a relatively lower number of points. In reality, this corresponds to a manually-placed marker.

Each group of points thus identified was considered to represent an object. These objects were assumed to be planar because LiDAR can return only the face of the object that is visible to it. Even cylindrical objects, such as poles and pipes, are assumed to be planar because the radiuses of the curvature of these objects are very small compared to the range of the LiDAR. The grouped points represent a logical planar object.

7.7 Fitting a plane equation to an identified group of points

An equation for the plane that best fits the group of points can be found using least squares. If (x_n, y_n, z_n) represents the three coordinates of point *n* with respect to the origin of that frame, then parameters *a, b, c, d,* which define a plane by the equation $ax+by+cz=0$, is obtained by solving the following equation using least squares

$$\begin{pmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_k & y_k & z_k & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad \textbf{Equation 7.1}$$

If *A* represents the matrix of points, *x* the vector having the plane parameters, and *J* the errors, then the equation to minimize is

$$J = \| Ax \|^2 \qquad \textbf{Equation 7.2}$$

Using singular value decomposition (Lawson and Hanson, 1974)

$$A = U\Sigma V^T \qquad \textbf{Equation 7.3}$$

Therefore,

$$J = \| \Sigma V^T x \|^2 \qquad \textbf{Equation 7.4}$$

To minimize *J*

$$\Sigma V^T x = 0 \qquad \textbf{Equation 7.5}$$

The covariance of *x* is given by

$$V(x) = \sigma^2 V \Sigma^{-2} V^T = 0 \qquad \textbf{Equation 7.6}$$

Using the above procedure, equation of each plane is obtained.

Table 7.1 shows data regarding various planes that were identified in the reference frame based on the above description. It also shows the results of fitting a plane equation for the identified markers.

Table 7.1 Plane fitting applied to data from a frame

Plane	Mid-Point of group of points [x, y, z] cm			Distance from origin cm	Plane Parameters [a, b, c, d]				No. of Points
1	-357.93	3086.66	-1218.89	3337.85	0.2819	0.4864	-0.827	-2409	9
2	1349.17	5533.93	-1564.47	5906.97	-0.0898	0.9787	-0.1847	-5584	15
3	1358.32	2993.66	-1186.06	3494.83	0.2796	0.3918	-0.8765	-2592	8
4	1630.48	832.7	-663.77	1947.42	0.8029	-0.5798	0.1387	-734.2	8
5	1685.81	5587.88	-1518.85	6031.02	-0.1455	0.9592	-0.2424	-5483	44
6	2091.99	2666.92	-1136.01	3574.83	-0.0127	0.1045	0.9944	877.9	40
7	2248.7	4217.49	-1307.64	4955.18	-0.2732	-0.7806	0.5621	4642	19
8	2338.56	1433.05	-935.08	2897.74	-0.4427	-0.6129	0.6545	2526	11
9	2492.76	604.15	-848.41	2701.6	0.0013	-0.1622	-0.9868	-742.4	8
10	2628.58	3504.95	-1176.82	4536.41	0.5660	0.8183	0.1005	-4237	19
11	3459.01	2938.14	-1027.16	4653.22	0.1453	0.9445	-0.2947	-3580	8
12	3634.1	878.75	-845.54	3833.25	0.9533	0.1651	-0.253	-3823	11
13	9498.23	2237.45	-998.71	9809.18	-0.9654	-0.2373	0.1086	9808	12

7.8 Identifying the same plane across frames

The procedure described in the previous sections, results in "markers" being identified in each plane independently. Thus, before we compute a transformation matrix from given frame to the reference, we should first match the planes found in the given frame with the corresponding planes in the reference frame. A procedure to achieve this is explained in this section.

For every plane P_o in frame *o*, there may or may not be a plane identified in the reference frame, *ref*. We first chose one plane in the frame that had a lower number of planes than the other planes. Then, for the chosen plane, we first looked at the parameter *d* of the plane equation. We filtered out all the planes whose parameter d varies beyond an acceptable threshold. Then, the

distance of the midpoint of frame in question and the filtered frames were compared, and the planes that were far off were neglected, while the nearest plane is assumed to be a likely candidate. Now for the nearest frame, the difference in the individual coordinates of its midpoint and the plane in question was examined. If there is a large variation here, then that plane is unlikely to be the same physical plane as that represented by the plane in question. Thus, the most suitable candidate is identified.

7.9 Plane Transformations

Once the corresponding matching planes were identified, the next step was to find the homogeneous transformation matrix. The general homogenous transformation matrix is

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} C_\beta C_\gamma & C_\gamma S_\alpha S_\beta - C_\alpha S_\gamma & C_\alpha C_\gamma S_\beta + S_\alpha S_\gamma \\ C_\beta S_\gamma & C_\alpha C_\gamma + S_\alpha S_\beta S_\gamma & C_\alpha S_\beta S_\gamma - C_\gamma S_\alpha \\ -S_\beta & C_\beta S_\alpha & C_\alpha C_\beta \end{bmatrix} \qquad \textbf{Equation 7.7}$$

$$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Where R is the 3×3 matrix that represents the rotation around the Z Y X axis, respectively, and t is the 3×1 vector representing the translation along XYZ directions, α, β, γ, and t_x, t_y, t_z are rotations and translations about x, y, z axes, respectively. Assuming the angles of rotation are small, the S_θ and C_θ can be approximated to θ and 1, respectively, and the product of the two angles is very small, R becomes

$$R = \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix} \qquad \textbf{Equation 7.8}$$

If p_x^o represents the coordinates of point x of frame o and p_x^{ref} represents the same point in the reference frame, then the transformation is expressed as

$$P_x^{ref} = P_x^o \times T_o^{ref} \qquad \textbf{Equation 7.9}$$

Where, T_o^{ref} represents the homogeneous transformation matrix from frame 0 to frame *ref*.

If there are m planes identified in the reference frame and the frame in question, then the transformation matrix between the two is obtained by solving the equation,

$$\begin{bmatrix} P_{a1}^o & P_{b1}^o & P_{c1}^o & P_{d1}^o \\ P_{a2}^o & P_{b2}^o & P_{c2}^o & P_{d2}^o \\ \vdots & \vdots & \vdots & \vdots \\ P_{am}^o & P_{bm}^o & P_{cm}^o & P_{dm}^o \end{bmatrix} \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}_{ref}^{o} = \begin{bmatrix} P_{a1}^{ref} & P_{b1}^{ref} & P_{c1}^{ref} & P_{d1}^{ref} \\ P_{a2}^{ref} & P_{b2}^{ref} & P_{c2}^{ref} & P_{d2}^{ref} \\ \vdots & \vdots & \vdots & \vdots \\ P_{am}^{ref} & P_{bm}^{ref} & P_{cm}^{ref} & P_{dm}^{ref} \end{bmatrix} \quad \textbf{Equation 7.10}$$

Where, P_{am}^o represents parameter a of plane m in frame o.

Taking advantage of the symmetry of the R matrix (as shown in Equation 7.8), Equation 7.10 can be rewritten as

$$\begin{bmatrix} P_{b1}^o & P_{c1}^o & 0 & 0 & 0 & 0 & P_{a1}^o \\ P_{-a1}^o & 0 & P_{c1}^o & 0 & 0 & 0 & P_{b1}^o \\ 0 & P_{-a1}^o & P_{-b1}^o & 0 & 0 & 0 & P_{c1}^o \\ 0 & 0 & 0 & P_{a1}^o & P_{b1}^o & P_{c1}^o & P_{d1}^o \\ P_{b2}^o & P_{c2}^o & 0 & 0 & 0 & 0 & P_{a2}^o \\ P_{-a2}^o & 0 & P_{c2}^o & 0 & 0 & 0 & P_{b2}^o \\ 0 & P_{-a2}^o & P_{-b2}^o & 0 & 0 & 0 & P_{c2}^o \\ 0 & 0 & 0 & P_{a2}^o & P_{b2}^o & P_{c2}^o & P_{d2}^o \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \times \begin{bmatrix} r_{21} \\ r_{31} \\ r_{32} \\ t_1 \\ t_1 \\ t_1 \\ 1 \end{bmatrix}_{o}^{ref} = \begin{bmatrix} P_{a1}^{ref} \\ P_{b1}^{ref} \\ P_{c1}^{ref} \\ P_{d1}^{ref} \\ P_{a2}^{ref} \\ P_{b2}^{ref} \\ P_{c2}^{ref} \\ P_{d2}^{ref} \\ \vdots \end{bmatrix} \quad \textbf{Equation 7.11}$$

If Equation 7.11 is treated as $Ax = b$, then the solution to that equation for least squares is

$$\hat{x}_w = (A^T A)^{-1} A^T b \quad \textbf{Equation 7.12}$$

Table 7.2 shows the results of applying the proposed method for finding the transformation matrix for a frame, referred to as "frame 3," which will transform it to the same coordinates as the initial frame. The residual column shows the discrepancy between the calculated value ($A \times TF$) and the actual observed value (B). Table 7.3 shows the transformation matrix that was determined by applying the method described in this section to "frame 3" described in Table 7.2, which was used in calculating the residuals column in Table 7.2

Table 7.2 Identifying and matching the same physical plane across two frames

Plane parameters from frame 3 {A} [a, b, c, d]				**Corresponding plane parameters from ref frame {B} [a, b, c, d]**				**Residuals { $A \times TF - B$ }**			
-0.0121	0.1033	0.9946	879.46	-0.0135	0.1107	0.9938	862.66	0.0045	-0.0049	0.0006	11.93
-0.4016	-0.6878	0.6047	4580.31	-0.4230	-0.7299	0.5369	4726.50	0.0069	0.0522	0.0697	-117.90
0.4194	0.7168	-0.5571	-2523.55	0.4552	0.6505	-0.6080	-2559.73	-0.0206	0.0559	0.0489	6.73
-0.0179	0.1505	0.9885	793.84	0.0038	0.1589	0.9873	733.69	-0.0176	-0.0058	0.0008	52.92
0.6445	0.7585	-0.0963	-4459.61	0.6347	0.7635	-0.1193	-4479.29	0.0262	-0.0192	0.0207	-8.72
-0.0846	-0.9959	0.0312	3244.11	-0.0828	-0.9962	0.0268	3232.35	-0.0234	0.0022	0.0068	58.94
0.5164	0.8377	-0.1776	-6519.35	0.5390	0.8250	-0.1702	-6523.40	-0.0044	0.0011	-0.0097	-29.88
0.9720	0.2010	-0.1214	-9804.72	0.9806	0.1524	-0.1233	-9778.14	-0.0043	0.0271	0.0005	-23.71

Table 7.3 Transformation matrix for the frame shown in Table 7.2

Transformation Matrix [TF] [4x4 matrix]			
1.0000	-0.0218	-0.0009	13.0094
0.0218	1.0000	-0.0022	-48.4608
0.0009	0.0022	1.0000	0.2958
0.0000	0.0000	0.0000	1.0000

7.10 Evaluation of the Self Calibration method

IMU method vs Marker method

We conducted an experiment to compare the results from the two methods for calibrating LiDAR motion, namely the IMU method (explained in Section 6) and the method involving markers (explained in this section). The results from both are shown in Table 6.1and Table 7.4.

Table 7.4 Marker method

	Rz (degrees)	**Ry (degrees)**	**Rx (degrees)**	**Tx (cm)**	**Ty (cm)**	**Tz (cm)**
Mean	0.0250	0.0166	0.0104	19.9021	28.4689	17.9612
High	0.0837	0.0509	0.0437	64.0041	109.6369	69.5103
Low	0.0003	0.0001	0.0001	0.1028	0.5511	0.1775

It must be noted that the results are for the same instance when data were collected. It can be seen that while the IMU shows that the translations were on the order of a few millimeters, the marker method indicates that there are translations (from the reference frame to the frame in question) of more than 50 cm in certain cases. From visual inspection, it is evident that the mast containing the LiDAR did not sway a great deal. Hence, it is necessary to investigate the reason for such huge numbers being projected by the marker method. The following subsection attempts to provide an explanation.

7.11 Residual Analysis of plane fitting

Once the markers are identified based on the method suggested in Section 7.5 and Section 7.6, we tried to fit a plane equation to these markers as described in Section 7.7. The residuals obtained from fitting the planes were analyzed to determine the effect of the shape of the object. It was anticipated that, if the object was planar, then the residuals were likely to be smaller when compared to non-planar objects.

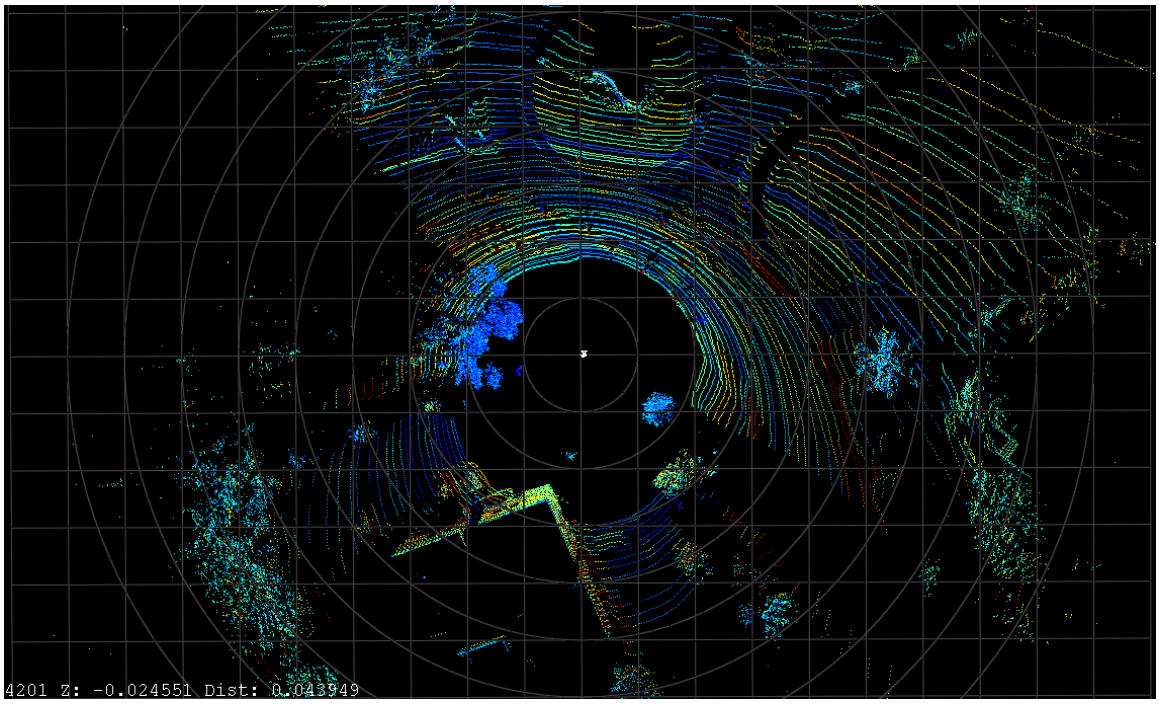


Figure 7.4 A sample frame from the experiment that also used IMU

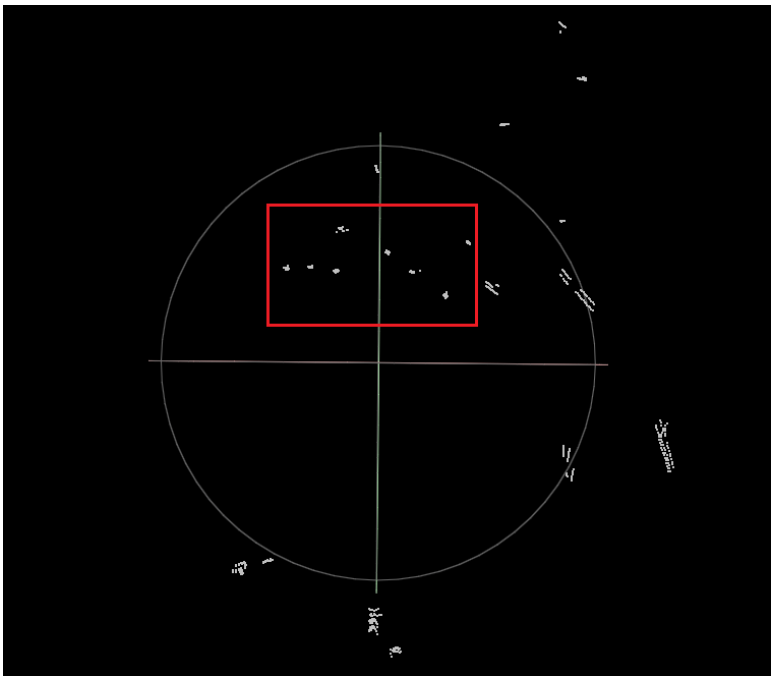


Figure 7.5 For the frame shown in Figure 7.4 with the markers identified

To give an example, consider the following. An example frame is provided in Figure 7.4 and Figure 7.5. The groups of points inside the red box are retro-reflective traffic signs that were manually placed to act as markers. There were 25 markers identified by the algorithm in the frame shown in Figure 7.4. The reference frame for this data was identified as having 28 markers.

Table 7.5 Planes in sample frame from experiment

Plane	Mid-Point of group of points [x, y, z]			Plane Parameters [a, b, c, d]				No. of Point	Sum of Residue
1	-5655.46	2853.83	-158.25	-0.3178	0.2784	-0.9064	-2735.03	12	7.15
2	-4794.94	3155.83	-56.69	-0.8420	0.3213	-0.4333	-5076.12	14	16.93
3	-4113.10	1981.02	-156.43	-0.9492	-0.1177	0.2919	-3625.36	26	19.57
4	-3446.05	-11.92	-317.33	-0.3312	0.9374	0.1075	-1095.99	13	12.04
5	-2662.39	2951.82	-370.55	-0.9526	0.1280	0.2762	-2811.59	17	18.97
6	**-2567.55**	**-578.55**	**-683.41**	**0.7649**	**0.0695**	**0.6404**	**2441.84**	**11**	**18.62**
7	**-2352.31**	**1502.45**	**-642.35**	**-0.8550**	**0.3103**	**-0.4155**	**-2744.39**	**11**	**22.32**
8	**-2193.05**	**176.54**	**-708.48**	**-0.7807**	**0.4813**	**-0.3985**	**-2079.46**	**12**	**11.93**
9	**-1952.37**	**-1089.43**	**-764.31**	**0.9797**	**0.0707**	**0.1877**	**2133.23**	**15**	**17.71**
10	**-1922.47**	**-1486.58**	**-778.14**	**0.7201**	**0.2335**	**0.6534**	**2239.97**	**10**	**5.40**
11	**-1888.84**	**-668.23**	**-759.21**	**0.6297**	**0.3213**	**0.7073**	**1941.08**	**14**	**27.23**
12	**-1882.58**	**583.02**	**-748.04**	**-0.9244**	**0.2938**	**-0.2434**	**-2093.54**	**14**	**7.02**
13	-1800.41	3145.04	-755.88	-0.1755	0.1242	-0.9766	-1444.65	17	10.26
14	-1638.71	1909.59	-819.01	0.1519	-0.0141	0.9883	1085.17	22	2.01
15	**-1495.42**	**1136.14**	**-750.69**	**0.5715**	**-0.8181**	**-0.0638**	**1736.34**	**27**	**76.74**
16	-1466.44	3463.04	-788.80	-0.1947	0.0918	-0.9766	-1373.65	25	5.83
17	1009.24	4908.67	-1031.83	-0.2615	0.6810	-0.6840	-3784.61	54	113.47
18	1148.70	3276.26	-1163.62	0.1032	0.0101	0.9946	1005.69	17	8.58
19	2951.01	-1776.12	-933.43	0.9444	0.3061	-0.1203	-2355.45	10	11.35
20	3124.36	-2272.91	-1274.59	-0.9225	0.1979	0.3314	3754.56	13	89.29
21	3163.28	-2284.34	-1032.57	0.9331	-0.2301	-0.2763	-3762.80	16	100.54
22	4045.13	-5.75	-1558.26	-0.4118	0.1683	0.8956	3062.47	12	72.79
23	4123.24	-4.21	-1351.59	0.8625	-0.4830	-0.1511	-3762.57	16	102.89
24	4161.38	-9.04	-1120.85	-0.8031	0.5533	0.2209	3594.91	16	97.54
25	4724.96	373.50	-1564.23	0.7703	0.2939	-0.5659	-4634.79	17	89.53

In Table 7.5, the planes highlighted in red are the manually placed markers. The remainder consists of those planes determined by the algorithm to be markers. In other words, the remaining identified markers may or may not be planar in nature. As one can see, the planar

retro-reflective traffic signs, (manually placed) have lesser residuals when compared with some other planes identified by the algorithm (e.g., plane 21 or 23).

Table 7.6 shows the planes identified in the reference frame, the number of times they were identified across frames, and the standard deviation of the fitted plane parameters for the same physical plane across frames.

Table 7.6: Variation of plane parameters across frames

Plane number	**Variance of parameters across frames [a, b, c] – degrees^2 and [d] – cm^2**				**No of frames in which the plane was identified (Total:2999)**
1	0.001766	0.000646	0.000779	2686.32	64
2	0.00013	0.00132	9.69E-05	6647.42	595
3	0.000308	0.000689	0.001524	10008.49	2881
4	7.18E-05	0.000426	0.000773	2712.54	2913
5	0.000271	4.54E-05	0.001396	3594.30	2906
6	0.000253	0.001332	0.002004	8450.03	2606
7	0.002058	0.000887	0.001745	7693.48	2007
8	0.002388	0.008266	0.007812	3063.71	1886
9	0.003469	0.0118	0.006265	7663.53	1315
10	9.90E-05	0.002556	0.001841	1499.13	2593
11	0.001237	0.006102	0.000373	3611.21	2669
12	0.000896	0.015751	0.00684	4183.58	2999
13	0.00187	0.00411	0.001243	1452.34	2999
14	2.91E-05	0.000409	9.04E-06	3776.65	1862
15	1.38E-06	1.76E-06	3.67E-08	15.55	2988
16	0.00308	0.002243	0.000636	1312.11	2999
17	2.40E-05	1.83E-05	1.41E-06	283.49	2573
18	8.33E-05	0.000214	0.000345	1918.36	2967
19	6.36E-05	0.000114	5.74E-07	1780.12	2769
20	0.053176	0.022897	0.084642	7184.63	42
21	0.001005	0.000136	0.004657	1535.47	759
22	4.98E-05	0.000184	0.001739	2137.11	2475
23	0.002517	0.002091	0.008869	7055.40	885
24	0.001175	0.005323	0.006295	7704.76	2487
25	0.000236	0.001854	5.98E-05	2873.98	1027
26	0.000707	0.001623	0.005254	8758.22	2190
27	0.000552	0.000532	0.003339	4796.49	683
28	0.000932	0.007219	0.005306	4540.53	2633

The number of frames in the sample data was 3,000; and since the first frame was considered to be the reference frame, there were 2,999 frames remaining for which a transformation matrix needed to be determined. As seen in Table 7.6, only two planes are consistently identified across

all the frames. On inspection, these planes were not the manually placed retro-reflective traffic signs, which meant that a varying number of markers were identified in each frame. Except for marker-plane 15, all other frames had a standard deviation of more than 10 cm. Therefore, even if the planes were stationary in the real world, due to other sources of errors in the way the LiDAR collects data, we see that the planes apparently moved. This table represents the same data used to generate Table 7.4 and Table 6.1. While the direct measurement of motion using IMU shows negligible change in the position and orientation of the LiDAR, the marker method showed very high changes. One of the reasons for that result can be understood from Table 7.6 and shows that because of the errors that are present in the way data were collected by the LiDAR, even stationary planes appeared to move.

7.12 Special cases for high residuals

There are a few special cases as to why even retro-reflective traffic signs placed manually still produced large residuals. For example, plane number 15 in Table 7.5 has a large residual sum, which is as high as 76 because of a combination of the below factors.

Additional high intensity points

In some cases, there may have been high intensity points that belonged to the background but were close to a marker (which also had high intensity points). This means that the above proposed algorithm considered those points to be a part of the marker even if the points physically were not, which distorted the plane fitting procedure. For the time being, we are trying to fit a plane to a great number of points that are part of the same physical planar object along with a few outliers (those belonging to the background or a very close by different physical object). These outliers are the reason that the residuals were skewed

Errors in the distance information provided by the LiDAR

The other error that was observed in the data from the experiment was that there were errors in the distance information obtained from the LiDAR itself. When the captured data was visualized using the vendor's software (Digital sensor recorder – provided by Velodyne), the set of points that represented the planar retro-reflective traffic sign (marker) did not appear to be part of a flat plane.

There were clearly errors. The visualization shows that some points were nearer and some farther than intended because the beam of light from which the distance information was obtained may have undergone multiple reflections before it returned to the source. Since the senor calculates the distance be measuring time, these multiple reflections mean that delays exist before a beam returns. The sensor is incapable of differentiating this, which means that the distance information is larger than the actual distance. Thus, defective distances crept in and caused a spike in the sum of the residual value.

8 Closure

The concept of a portable roadside LiDAR-based TScan unit for long-period data acquisition, real-time processing, and storage to facilitate accurate microscopic traffic measurements useful to traffic and safety engineers and researchers is proposed in this research report. This first phase of this research, presented in this report, focused on finalizing the development of an experimental unit, investigating the LiDAR sensor's capabilities and data properties, and developing prototype data management and preprocessing modules. The outcomes of this phase include a ready-to-use experimental unit, a thorough understanding of the properties of the LiDAR data collected with the HDL-64E sensor, a set of prototype MATLAB codes for data management and pre-processing, and a concept of objects identification, classification, and tracking.

In the course of the project, the existing Purdue MTL was modified by adding a LIDAR sensor, upgrading the surveillance video cameras, increasing the storage capabilities, and modernizing the supervising computer and Ethernet-based network for communication and data transfer. The hardware was tested by checking the communication between hardware components, performing initial data acquisition, and inspecting the collected data using the vendor software.

Then, the vendor software for data management was replaced with open-domain Wireshark to increase the data management capabilities. The most important capabilities of the developed experimental unit are data reduction through collecting data only in a user-defined FOV and storing the data in packets rather than in a single file. The latter reduced the risk of losing data while increasing the efficiency of data management. The following subsections summarize the most important outcomes regarding the data properties and the pre-processing functions. The concept of objects identification, classification, and tracking is provided. The future research needed to support further development of TScan is briefly discussed.

8.1 Basic Data-related Findings

The LiDAR sensor emits and receives light impulses at a selected constant frequency regardless of the rotating sensor head position. The encoder of the LiDAR has an angular resolution of 0.09°, which means that it has 4,000 possible horizontal angle values. A fixed spot in the sensor's FOV hit in a frame (all readings during one sensor rotation) is not hit in the following

ten or more frames due to "frame drift." A relatively long time is needed to collect a sufficient number of readings in the same horizontal direction for a point-based statistical analysis. Even then, there is no guarantee that the readings correspond to the same point in the real world due to possible motion by the sensor. A better approach is to identify and estimate the positions of planar and 3D objects and treat the estimation error of these objects.

The LiDAR sensor accepts the light reflected by a surface if its energy level exceeds a certain intensity threshold. A considerable number of vehicles have highly reflective bodies that, depending on the beam angle, reflect most of the energy away from the sensor. Wet or icy pavement hit by a light beam at a low angle has the same result. The received back energy is too low and the reading is blank. The same phenomenon may also lead to an overestimated distance reading if the light beam is initially reflected away from the sensor and then reflected back to the sensor by another reflective surface (a light path may be as follows: sensor-pavement-vehicle-sensor).

LiDAR sensor motion is possible and may render the distance and location estimation unreliable. Adjusting the readings for the sensor's motion is necessary.

The distribution of the distance readings in a certain direction in the spherical coordinates, even before adjusting the reading for the sensor's motion, were found to be useful in separating the background readings from the moving objects. The same concept applied to the distribution of z values in the Cartesian coordinates brought similarly encouraging results.

8.2 Adjusting for LiDAR Sensor Movement

Measuring the motion of the LiDAR sensor with the IMU provides an independent source of information useful for adjusting the LiDAR readings. Using more than one IMU would be beneficial for the following reasons:

- Any random error that may occur in one sensor can be alleviated by using data from other sensors. Thus, redundancy increases the reliability of the system.
- Since an IMU cannot be placed at the origin of the reference with respect to which the LiDAR reports the data, multiple IMUs will help in triangulating and producing a more accurate transformation matrix.

8.3 LiDAR Self-calibration

The foremost problem with the proposed method is that it relies upon detection of fixed planar objects (called markers). If sufficient markers are not identified in each frame, then the transformation matrix becomes inaccurate. Since the definition of markers in the above method is based on a high-intensity return, there may be cases where the number of markers is insufficient. One promising solution is for the user to place additional retro-reflective objects in the FOV, which may be found cumbersome by some users.

Second, the regions that are identified as markers may not necessarily be planar objects, which implies that, based on what regions of the object is identified, different plane equations will be derived. If the variation in plane parameters is large, then it will distort the transformation matrix.

Third, while trying to match the same physical plane across frames, there is a possibility that a wrong match may occur (i.e., two planes that correspond to different objects in the real world may be identified as the same plane). This occurs only if two planes are fairly parallel to each other and are within a couple of meters from each other. The chances of this happening are low.

The above three issues can be avoided by letting the user identify parts of the background that are sufficiently large, flat, and unobstructed by moving vehicles. At this point, the recommended method of dealing with the LIDAR sensor's motion utilizes IMUs.

8.4 Concept of Objects Identification, Classification, and Tracking

The outcomes of the first phase of the TScan project allow proposing the following sequence of steps leading to successful identification and estimation of the moving objects. These steps are divided into two phases:

1. TScan Calibration (data collected during several minutes up to 30 minutes)
    1.1. Converting the LiDAR spherical readings to XYZ readings,
    1.2. Adjusting for the LiDAR sensor's motion,
    1.3. Inputting the intersection area boundaries and major geometry features (user),
    1.4. Estimating the conditions to separate the background from moving objects within the intersection boundaries based on the z coordinates distribution in the x-y lattice.

2. Data Acquisition and Processing
    2.1. Converting the LiDAR spherical readings to XYZ readings,
    2.2. Adjusting for the LiDAR sensor's motion,
    2.3. Applying the separation conditions within the intersection boundaries to extract the moving objects data.
    2.4. Identifying and classifying objects by performing 3D clustering analysis bound with the objects' physical properties (feasible positions, size range).
    2.5. Tracking the identified objects across frames with the Kalman filter bound with the objects' dynamic properties (speed and acceleration).

8.5 Future Research Needs

Several areas of research have been identified to further advance the TScan research.

Object Identification

Once the LiDAR readings are adjusted for the sensor's motion and all the readings are referred to the same reference frame, the next step is to identify moving objects. We propose that motor vehicles are the initial focus. Pedestrians and two-wheelers should be considered next after the algorithm for identification of vehicles is developed.

The improvements in separating the background from moving objects in the reported research project allow focusing future research on moving objects. Figure 8.1 and Figure 8.2 show the same frame before and after the background was removed. One can see that the cars are the only predominant clusters of points in the intersection area. The remaining points are noise; and the primary source of noise outside of the intersection area in the presented example was trees. The points corresponding to cars seem to be easy to cluster in the 3D space. Vehicles close to one another may be more difficult to handle. Future work will be focused on resolving this issue by utilizing the known size range of the objects and their probable position in the intersection area with known arrangements of traffic lanes.

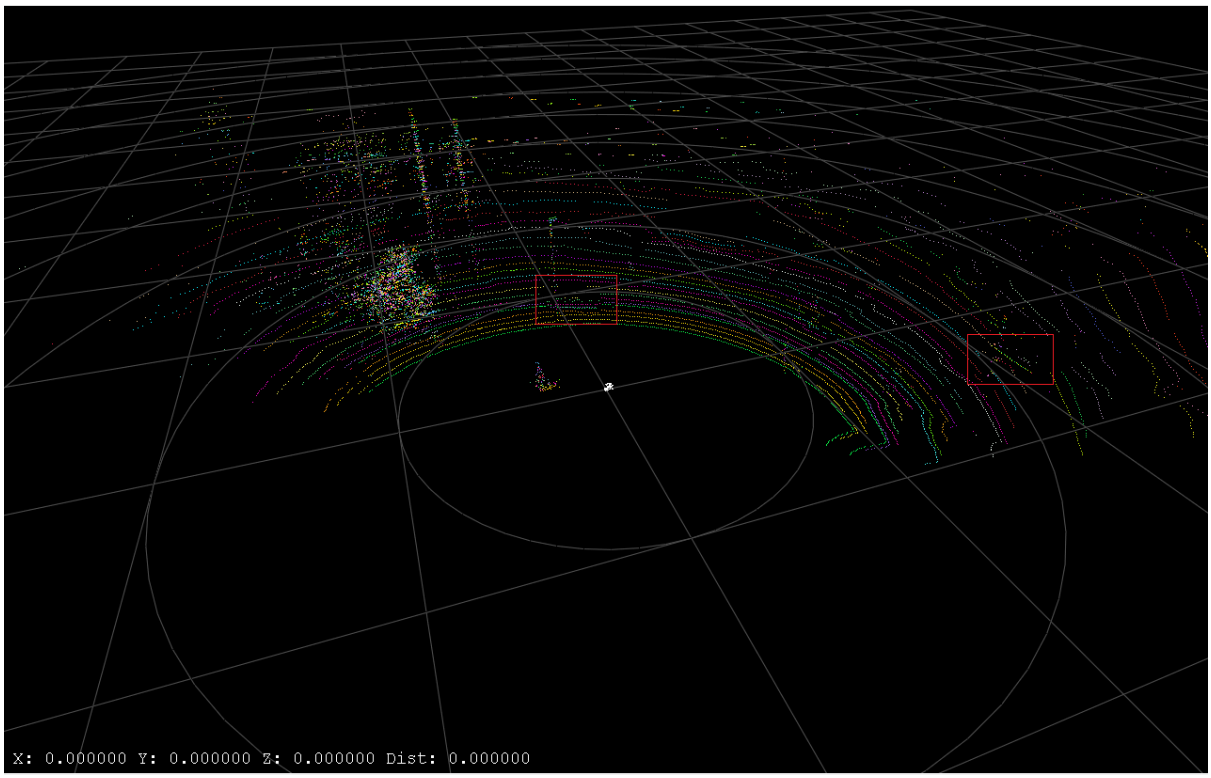


Figure 8.1: A sample frame showing two cars

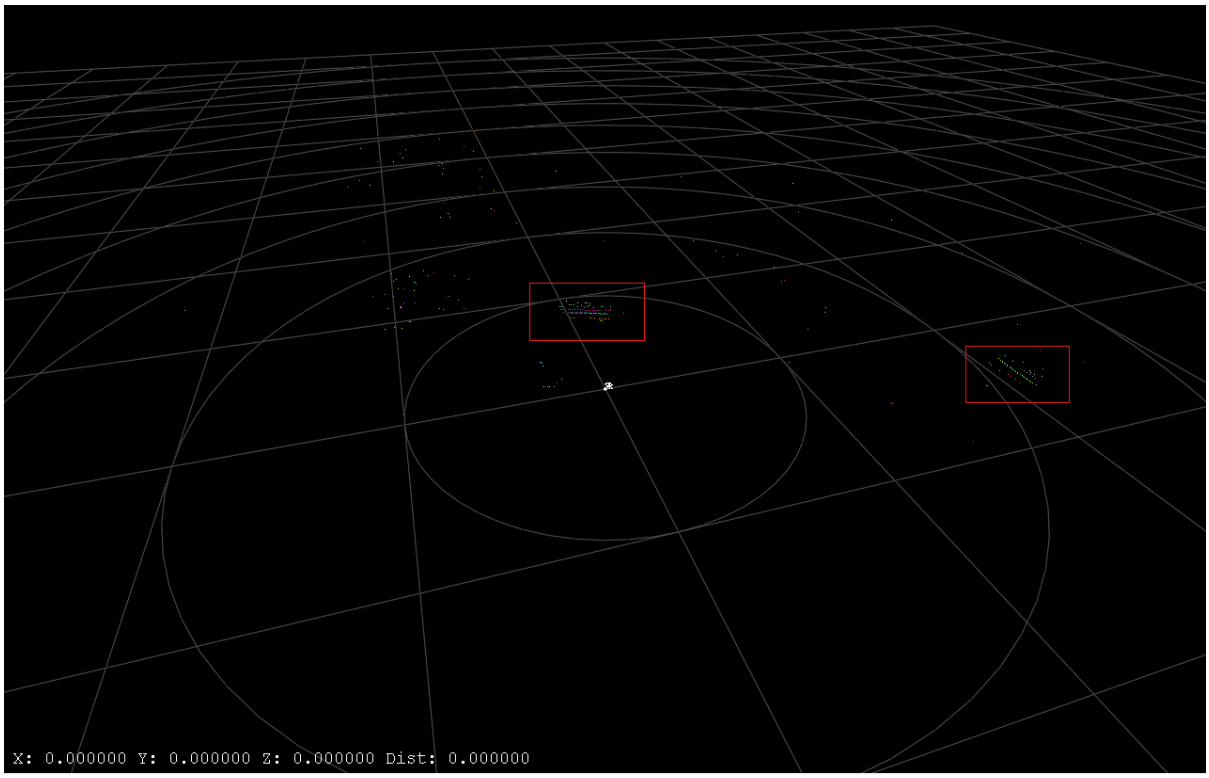


Figure 8.2: Same frame as in Figure 8.1 after background is removed

Object Tracking

Once moving objects are identified, they have to be tracked to estimate their trajectories. Known dynamic properties of vehicles (prior distribution of their speeds and acceleration further improved with the data collected in the calibration phase) will be utilized in a Kalman filter.

There are challenges that need to be better understood. The density of the TScan readings reduces with the distance from the sensor. The growing sensitivity of the points clustering and tracking to the decreasing number of points negatively affects the quality of the results. The relationship between the object distance to the sensor and the detection and tracking errors must be investigated. This would help improve tracking and establish the range within which a LiDAR with a given set of specifications is able to perform the tracking job successfully.

Real-time Processing

The prototype processing modules are developed in MATLAB®. The data processing will eventually be performed in the field during data acquisition. To speed up the processing and to investigate the computational efficiency of the ready-to-implement software, the MATLAB programs must be migrated to a much more efficient environment such as C++ which allows low-level operations on bits to avoid time-consuming overheads.

An initial test was carried on a sample data collected for 13.2 seconds including 132 frames and approximately 30000 points per frame. The MATLAB® program took more than two minutes for the data conversion and splitting the large .pcap files containing the data stream into frames. The C++ version completed the same job in approximately two seconds. Thus, the C++ variant of the program was fast enough to provide additional 10 seconds for other operations of objects identification and tracking.

Properly designed tests are needed to evaluate the feasibility of real-time processing when all the necessary modules are developed and implemented in the C++ language.

9 Bibliography

Analog Devices (2014). “MEMS Inertial Measurement Units” [Online]. Available: http://www.analog.com/en/mems-sensors/mems-inertial-measurement-units/products/index.html [Accessed 15 August 2014]

Cheung, H. (2007). Spinning LASER maker is the real winner of the urban challenge, *TG Daily*, 7 November 2007.

Dozza, M and N. González (2013). Recognizing safety critical events: Can automatic video processing improve naturalistic data analyses? *Accident Analysis and Prevention*, Vol. 60, pp. 298– 304.

Glennie, C. and Lichti, D. (2010). "Static calibration and analysis of the velodyne hdl-64e s2 for high accuracy mobile scanning," *Remote Sensing. 2,* No. 6: 1610-1624.

Jonasson, J. and Rootz, H. (2014). Internal validation of near-crashes in naturalistic driving studies: A continuous and multivariate approach. *Accident Analysis and Prevention*, Vol. 62, pp. 102p. nt.

Kalman, R. (1960). "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, no. 82 (1), p. 35–45.

Lawson, C. and Hanson, R. (1974). *Solving Least Squares Problems*, Englewood Cliffs, NJ: Prentice Hall.

Microstrain (2014). "Microstrain #dm-gx2," Microstrain, [Online]. Available: http://www.microstrain.com/inertial/3DM-GX2. [Accessed 22 July 2014].

NOAA (2013). "LIDAR - Light Detection and Ranging - is a remote sensing method used to examine the surface of the Earth," [Online]. Available: http://www.webcitation.org/6H82i1Gfx. [Accessed 04 June 2013].

Strutz, T. , (2010). *Data Fitting and Uncertainty* (A practical introduction to least squares and beyond), Viewing+Teubner, p. Chapter 3, New York, NY: Springer.

Tarko, A. (2012). Use of crash surrogates and exceedance statistics to estimate road safety. *Accident Analysis and Prevention*. Volume 45, pp. 230-240.

Tarko, A., Davis, G, Saunier, N., Sayed, T., Washington, S. (2009). Surrogate measures of safety, white paper, unpublished.

Velodyne (2014). "High Definition Laser - HDL 64E - Datasheet," [Online]. Available: http://velodynelidar.com/lidar/products/brochure/HDL-64E%20S2%20datasheet_2010_lowres.pdf. [Accessed 27 June 2014].

Velodyne Manual (2014). "High Definition LiDAR - HDL 64E User Manual," [Online]. Available: http://velodynelidar.com/lidar/products/manual/63-HDL64ES2h%20HDL-64E%20S2%20CD%20HDL-64E%20S2%20Users%20Manual.pdf. [Accessed 27 June 2014].

Wu, K. and Jovanis, P. (2013). Defining and screening crash surrogate events using naturalistic driving data. *Accident Analysis and Prevention*, Vol. 61, pp. 10– 22.