Object detection and avoidance using LiDAR on a hydrofoil boat

Erik Söderberg

Department of Automatic Control

MSc Thesis
TFRT-6074
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

Robots in factories and vehicles on our roads are rapidly reaching the point of automation. Commercial markets require high standards of safety and reliability, which in turn demands strict requirements on software and hardware. This thesis presents an analysis of driver assistance for a hydro-foil speedboat using LiDAR technology. The LiDAR sensor measures 2D positions of surrounding objects and the control software processes the data in order to distinguish clusters and movement.

Keywords: LiDAR, Point cloud, clustering, real-time system

Acknowledgments

First of all, I would like to thank Candela Speedboat AB for providing me this master's thesis opportunity and for letting me work in their facilities. Everyone has been very helpful and supportive during these months. An extra big thank you to my supervisors, Kristian Sloth Lauszus and Anders Robertsson for their guidance and knowledge.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Problem formulation

"Is it feasible to implement driver assistance on a speed boat using a 2D scanning LiDAR sensor?"

The purpose of this master's thesis was to investigate and develop a driver assistance prototype for a hydro-foil speedboat, with LiDAR sensor technology. The goal of the software was to be able to detect and track obstacles. Other than focusing on the development of the real-time application running the driver assistance software, other important milestones was to address difficulties in the configuration of the hardware, software and identify limitations in real-time performance.

1.2 Proposed solution

The proposed solution to this thesis problem formulation was to use a 2D LiDAR sensor with 360° field of view in order to collect sufficient data of the dynamic surroundings around the boat. The data was then to be analyzed in real-time in order to provide as good estimations as possible of obstacles and their movement. By isolating tests for the performance of the LiDAR sensor, sensible conclusions could drawn about its durability and precision. By also testing the software solutions in dynamic settings key performance aspects could be addressed and analyzed.

1.3 Outline of thesis

This section is to describe the overall outline of this master's thesis.

In **chapter 2** all fundamental theories and tools, that underlie the practical work and the analysis in this report, are described.

Chapter 3 lists previous and similar scientific studies, that have been deemed relevant for this master's thesis.

In **chapter 4** work methodologies are covered in order to thoroughly detail the major steps taken during the investigation and development of the driver assistant prototype.

In **chapter 5** all relevant and quantifiable results are listed. The chapter covers the result from investigating and testing the LiDAR hardware, the first prototype in Python, the ROS obstacle detection package and the testing of the authors implementation and PCL.

Chapter 6 covers the analysis of the results and facts gathered to address flaws and imperfections in hardware and software.

In **chapter 7** future work and improvements are proposed for both hardware and software.

2 Background

Candela Speedboat AB has been developing an electric hydrofoil boat since 2014, with the goal of competing with fossil fuel speedboats. They are combining hydrofoils with advanced techniques in automation and electronics to achieve a long-range and almost silent ride. They have long been interested in moving towards driver automation or assistance and welcomed this master's thesis as a first prototype to their future investigation.

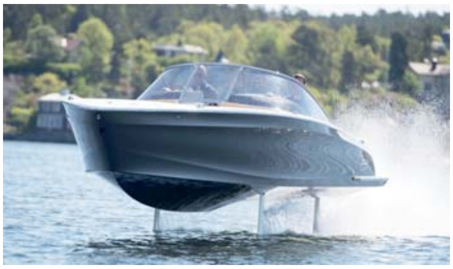


Figure 2.1: The Candela Speedboat [2]

In the initial stages of this master's thesis, a list of necessary hardware was conducted. As mentioned in section 1.2, a LiDAR sensor with a wide field of view was needed together with some stabilizing platform to compensate for sea turbulence. In order to keep costs down without loosing too much functionality, it was early decided to use a 2D rotating LiDAR. During the research phase a lot of work was put into investigating the possibility of building a stabilizing platform or gimbal. The idea was to order and assemble the hardware and to write a closed loop control algorithm that would read attitude of the sensor and quickly respond to tilt. After long consideration it was later recognized to take too much time and the idea was discarded. The idea of sensor stabilization is however vital for 2D LiDAR sensors, due to them having no vertical field of view. Hence the basic concept of such a device will be covered in this chapter as well.

2.1 Hardware configuration

2.1.1 LiDAR technology

LiDAR stands for Light Detection and Ranging. It is a technique used for determining distances to surrounding obstacles and environments. The technique utilizes the principle of time-of-flight in order to give distance measurements. By transmitting light and measuring how long it takes to bounce back, it is possible to estimate the distance to the reflecting object. As mentioned in [6] factors such as reflexive index and smoothness of surface of the reflecting object directly affect the intensity of the reflected light.

LiDAR technique is accurate and fast and has been used in a large variety of applications ranging from urban planning to autonomous vehicles and robots. The basic constructs and techniques are described below starting with the one-dimensional range finder.

2.1.1.1 Laser range finder

A laser range finder is a device that can measure a distance from itself to a *point*, aimed at by the device. They are widely used by golfers, military staff and in commercial products.

A single ray of laser light is emitted from the device. The light is bounced back to upon first contact with a solid material. The laser range finder device also has a photo diode that awaits light of the same, or similar wavelength that was emitted by the laser. By measuring the time between the laser emission and the reply, it is possible to estimate the distance to an object by using an approximation of speed of light in air. The formula for estimating the distance d with this, so called time of flight technique, can be seen in equation: 2.1. Here c is the speed of light, t is the time between the transmission and the laser response. Division with 2 is done to only acquire half of the total distance traveled by the light, namely the distance *to* the object.

$$d = \frac{c \cdot t}{2} \tag{2.1}$$

2.1.1.2 2D mechanical scanning LiDAR

2D Scanning LiDARs utilize the same technical concept as laser range finders. The difference between them is that scanning LiDAR are mounted on a rotary motor making it possible to take continuous samples as it is rotating. Scanning LiDAR sensors are also equipped with rotary encoders to be able to map a laser sample to a rotary angle. By doing this it is possible to acquire a point cloud of samples represented in the plane. Since there is still no vertical field of view to these sensors, they are not able represent their

Figure 2.2: Scanse's 2D scanning LiDAR "Sweep" [18]

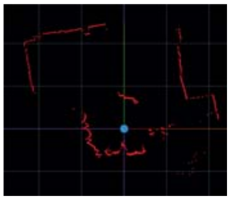


Figure 2.3: Example of point cloud output from the Sweep LiDAR

surrounding environment in 3D. The angle between two consecutive samples is called azimuth or angular resolution and can for some LiDAR sensors be tuned manually by adjusting the sampling - and rotational frequency of the device, as can be seen in equation 2.2. Furthermore, the term sample will relate to a single point in a point cloud and a sample frame will refer to a full point cloud.

$$azimuth = \frac{f_{sampling}}{f_{rotation} \cdot 360} \tag{2.2}$$

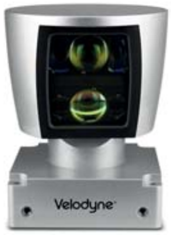


Figure 2.4: Velodyne's 3D scanning LiDAR HDL-64E [14]

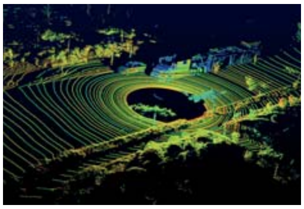


Figure 2.5: Example of point cloud output from the Velodyne 3D scanning LiDAR HDL-64E [14]

2.1.1.3 3D mechanical scanning LiDAR

3D scanning LiDARs are scanning LiDARs with vertical field of view. To obtain their vertical field of view, there are currently different approaches. Market-leading Velodyne currently uses 64 lasers in their flagship product HDL-64E with equally spaced angles between each laser emitter [14]. Other 3D LiDAR variants use mirrors to aim the laser beams, in order to gain vertical field of view [15].

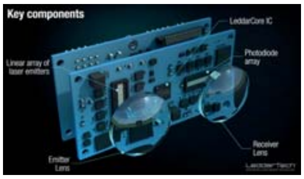


Figure 2.6: Solid state LiDAR with laser flash technique [12]

2.1.1.4 Solid state LiDAR

2017 was a big year for remote sensing, due to the emergence of solid state LiDARs. The term *solid state* is used interchangeably between various technical domains, but is most well known in digital storage. The term solid state refers to the hardware having no mechanically moving parts. Due to recent research the same technique has been utilized in LiDAR sensors. Instead of using a rotary motor to spin the sensor module, solid state LiDARs are used as stationary units. They usually have a narrower field of view, and machinery requiring 360 degree of vision usually requires multiple sensors installed. One of the market-leading companies, LeddarTech, offers two different Solid state LiDAR configurations for the automotive industry [12]. In figure 2.7 and 2.6, two solid state LiDARS, based on laser flash and hybrid flash technology, respectiveley, are depicted.

Both techniques use either some lattice or diffuser lens to break a ray of light into a scanning line of light. For flash LiDARs that line a laser light is then propagated horizontally by using an array of laser emitters. Figure 2.7 shows the hybrid flash solid state LiDAR. This LiDAR projects the laser beam onto a very small mirror oscillating on high frequencies, in order to scan the environment. Due to their compact design solid state LiDAR are much smaller than their predecessors, cheaper and more durable.

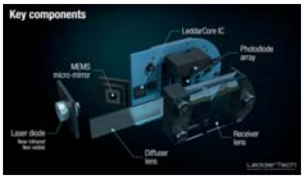


Figure 2.7: Solid state LiDAR with hybrid flash technique [12]

2.1.1.5 Scanse's Sweep LiDAR

For this master's thesis the Scanse's Sweep LiDAR was used. The sensor can be seen in figure 2.2 and is a 2D scanning LiDAR. It has two customizable parameters for rotational frequency and laser sampling frequency. Rotational frequency of the motor can be set to values between 1 - 10 Hz and the sampling rate can be set up to 1000 Hz. It collects all samples of a full revolution before sending the fully populated point cloud over a USB connection. This means that data arrives in a synchronous manner. An estimated error function is shown in figure 2.8, provided by the company Scanse.

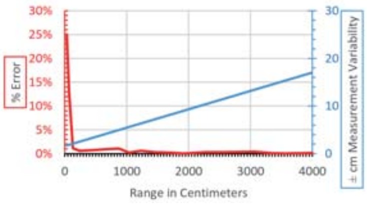


Figure 2.8: Graph of measurement error for detected obstacles up to 40m, produced by Scanse [18]

The sensor lacks an otherwise common function among scanning LiDARs, called fixed scanning. Fixed scanning means that samples do not only occur at an exact frequency but also conducts all samples at the exact same azimuth per each revolution. A scanning LiDAR with fixed scanning intervals would e.g., always have its first sample reading at 0°, whilst opposite LiDARs have a sporadic order for when samples occur [17].

The Sweep LiDAR is shipped with a predicted lifetime of 45 million rotations or approximately 2500 hours of moderate use at only 5 Hz of rotation frequency. The company warns of an even shorter lifetime if the sensor is exposed to turbulent or vibrating environments.

2.1.2 Sensor stabilization

Despite having multiple attitude sensors in the Candela speedboat and hydraulic motors to keep the boat and deck stabilized, it was early recognized that further sensor stabilization was needed. Since the Sweep sensor had no vertical field of view it was crucial to have the aiming of the laser beam correct. As can be seen in figure 2.9 and in equation (2.3), it was recognized that the sensor had to keep a horizontal orientation, with an approximate error of ±1.86° in order to not hit surface of water and to not miss boats and obstacles that were too small.

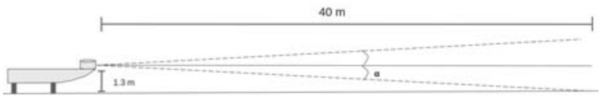


Figure 2.9: Image showing the crucial angle α for which the laser beam will hit the surface of the water

$$\alpha = tan^{-1}(\frac{1.3}{40}) \implies \alpha \approx 1.86° \tag{2.3}$$

This puts a rather strict requirement on hardware and software. However, by investing in an attitude sensor, a micro controller and a custom made construction housing two brushless motors, a stabilizing platform could be built and automated with a Kalman filter, to solve such a problem [1].

2.2 Obstacle detection

As mentioned in section 2.1.1 the output from LiDAR sensors, are point clouds. Regardless of using a 2D LiDAR or a 3D LiDAR the most common way of detecting obstacles are with point cloud clustering.

2.2.1 Clustering

The idea of point cloud clustering is to group data points together based on a predefined condition. Theoretically this condition can be any quantifiable state that a point holds, but we will for this master's thesis only focus on spatial clustering. As for detecting solid objects from point cloud data, the most obvious approach would be to measure the relative distance between points. The algorithm then determines the clusters upon a predefined minimum threshold value of this parameter. Such an approach was proposed in [3]. Here, the Euclidean clustering algorithm was effectively used to detect houses and buildings from airborne LiDAR data sets.

Given two data points p and q, in the three-dimensional space, represented in Cartesian coordinates, their Euclidean distance d is defined as:

$$d = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2 + (z_p - z_q)^2} \tag{2.4}$$

In order to speed up the clustering it is recommended to structure the input point cloud in something called a *k-d* tree. A *k-d* tree is a binary search tree used for organizing points living in an arbitrary number of dimensions. The first step of creating a *k-d* tree is by dividing the input data in two subsets. This is done by finding the median value of a specific dimension of the data set, say the x-coordinate of the points. The point, having its x value closest to this median value will represent the root of the *k-d* tree. The subset of points whose x value is smaller than the root's will appear as the left child of the root and the right-hand subset will appear as the right child of the root. After this initial step, all children of the graph are processed recursively in the same manner, until a predefined number of points are left in a subset [10].

2.3 Obstacle tracking

In order to achieve multi-target tracking of dynamic obstacles, there are mainly two important building blocks of the general tracking algorithm that need to be incorporated, namely:

- An accurate function for associating data (detected obstacles) between two consecutive sample frames
- An efficient algorithm that solves tracking problems like overlapping targets and obstacle fission and fusion.

Associating data or features between samples is usually referred to as the correspondence problem. It involves mapping data, or in this case obstacles, between samples that are statistically most likely to be the same. To find correspondences between samples or point clouds it is useful to utilize a correspondence function, that measures the similarity of states for all present obstacles. Depending on the granularity of analyzed data such a correspondence function could look very different. Such function can measure obstacle correspondences on states such as position and velocity and provide a measure of similarity. By incorporating such a function together with a Kalman filter for all detected obstacles, it is possible to estimate, predict and associate obstacles between point clouds [7][16].

3

Related work

In recent years much has happened in the fields of vehicle automation and assistance. The car company Tesla announced in 2015 that their Model S cars were to be shipped with autopilot and since then a lot has happened. Recent research suggests that there will be 10 million self-driving cars on the roads in 2020 [11] and more companies than Tesla are joining the future of technology. The urge for improved autopilots has boosted the development of sensing hardware and resource efficient software.

3.1 The ROS obstacle detector package

The ROS obstacle detector package is an open source software developed by Mateusz Przybyła from Poznań University of Technology. The package was developed for a small indoor mobile robot with two 2D scanning LiDARs utilizing fixed azimuth scanning. It uses a Kalman filter for eliminating noise and providing optimal state estimations of all surrounding obstacles. It has two algorithms running in parallel in order to detect walls (straight lines of points), and dynamic obstacles. Dynamic, or moving obstacles, are detected when the shape of a point cloud cluster deviates from that of a straight line. Once an obstacle is detected it will remain in the graphical interface as a trace until it finds an adjacent obstacle that is considered similar enough for it to have moved.

The reason why this package is considered important for this paper is due to the fact that it addresses the key features of 2D LiDAR obstacle tracking mentioned in chapter 2.3, namely data association and tracking difficulties of dynamic obstacles. Data association is solved here with a correspondence function. The correspondence function measures the similarity between one obstacle in a sample frame and compares it to all detected obstacles in the succeeding sample frame. The function does this by calculating a cost value based on the obstacles position and shape [16].

3.2 A Simple Reactive Obstacle Avoidance Algorithm and Its Application in Singapore Harbor

In the paper by Tirthankar Bandyophadyay, Lynn Sarcione and Franz S. Hover [4], a solution to detecting and avoiding stationary and dynamic boats and obstacles in the Singapore harbor, is pressented. By equipping an autonomous surface craft with a two-dimensional scanning LiDAR the authors propose a model-free approach for avoiding obstacles

This scientific paper is considered import due to the fact that it addresses some key difficulties in detecting and tracking obstacles at sea using a 2D laser scanner. It also covers common problems of usage that can occur for 2D scanning LiDARs at sea [4].

4 Methodology

This section aims to clarify the major steps taken in this master's thesis and answer what, when and how the work was conducted.

4.1 Stabilizing platform

In the initial stage, the possibility of building a gimball was considered. Inspired by an online article describing the process [1], time was put into finding relevant hardware and evaluating their specifications. After two weeks of investigation, both the time needed for building the hardware and writing the software was considered to take too much time. Instead, the idea of building a stabilizing platform was discarded to make more time for investigating the LiDAR and the software.

4.2 Prototype in Python

To first test the Sweep LiDAR and all necessary drivers, a prototype software was written in Python. The aim of this part of the master's thesis was to get acquainted with the device and also how to extract and use the output data provided by the Sweep. By sending commands to it, it was also possible to change and investigate different settings of the sensor, such as rotational frequency and sampling frequency. The final goal was to produce a graphical interface that could plot the point clouds in real-time, as data arrived from the sensor.

4.3 The obstacle detector package

In the next step of the practical work, an already existing software package was tested. This was deemed relevant to, in an early stage, address difficulties with object tracking from 2D point clouds. The software package chosen was the "The obstacle detector package" by Mateusz Przybyła [16]. The reason this package was chosen was because it proposed a solution to tracking of dynamic obstacles from 2D LiDAR data.

In order to interface the Sweep to this package, several preparatory measures had to be made. Being a ROS package [8] a ROS node and driver for the Sweep had to be configured in order to publish data to the package. Since the Sweep differed greatly in performance compared to the LiDAR used in The obstacle detector package, model parameters had to be tuned to work with the new hardware.

4.4 PCL and author's implementation

The last milestone in this master's thesis was to construct a driver-assistance prototype software based on the knowledge gathered from literature studies and testing. To not lose too much time on underlying implementation, the open source Point Cloud Library (PCL) was used [13]. PCL offers various processing tools for point clouds, and was early considered useful, mostly for its tools on graphical interfaces. The library is available in C++ and is built with the building tool CMake.

In the architectural design of this software, three parallel processes were recognized as necessary:

1. Read sensor data
2. Analyze data and run underlying algorithms
3. Update the graphical interface

To isolate these crucial processes three separate threads were introduced. As can be seen in the UML diagram in Chapter 5, all threads read from and wrote to a central monitor, holding shared resources and methods, such as point clouds and detected obstacles. Due to quite new and unstable versions of the PCL graphics module at the time, a lot of work was put into getting the visualization working. Once the input point cloud was successfully plotted, time was spent on clustering and improving the graphics.

To solve the correspondence problem in obstacle tracking the following correspondence function C was used:

$$C(\alpha,\beta) = \sqrt{(\alpha_x - \beta_x)^2 + (\alpha_y - \beta_y)^2 + w \cdot (\alpha_{points} - \beta_{points})^2} \tag{4.1}$$

The function takes two obstacles, α and β and compares two attributes between them; position and number of constituting points. The variable w in equation (4.1) is a customizable weight to how much the difference in number of points should affect the correspondence value of the function. By calculating the similarity between obstacles from the previous point cloud and the obstacles from the current point cloud, it was possible to obtain a measure of which old obstacles most possibly corresponded to new ones. All correspondence values were put in a matrix to easily compare all correspondence values, before deciding which obstacles transformed into which.

When the correspondence function was working, time was put into implementing the Kalman filter for obstacles.

4.5 Experiments

To verify and measure all implemented functionality, numerous experiments were set up throughout the thesis work, all of which will be covered in this section.

4.5.1 Indoor testing

After the realization that it would not be possible to use a stabilizing platform, it was immediately recognized that testing on the boat would not be possible. Instead, the majority of all software testing was done indoors in the Candela office space. By looking at how well and fast obstacles were detected, it was possible to tune parameters for the algorithms and evaluate the performance. By analyzing moving obstacles, such as walking people, it was possible to measure the speed of the obstacle tracking.

4.5.2 Outdoor testing with a car

By setting up a test environment at a big and empty parking lot, the obstacle detector package was thoroughly tested, by circulating the Sweep sensor with a car. The experiments included testing of sensor range, obstacle detection at different ranges and also the software performance at different speeds of the car.

4.5.3 Hardware testing

To test the LiDAR sensor's real performance a hardware test was conducted. The main idea was to establish the precision in *distance* measuring as well as the ability to track the azimuth to the centroid of a static obstacle. The goal was to abstract the sensor variance in order to compare precision to the proposed error function provided by Scanse 2.8. As can be seen in figure 4.1, the test obstacle was composed of one 1.5 meter wide and 1.5 meter high fence dressed in tarpaulin. The obstacle was then moved to a soccer field in order to have a wide and flat area with no interfering environments. Due to difficulties aiming the device at the obstacle at too large distances, the successful test was conducted at ten meters away from the obstacle, with an approximate azimuth of 20° to the obstacle.

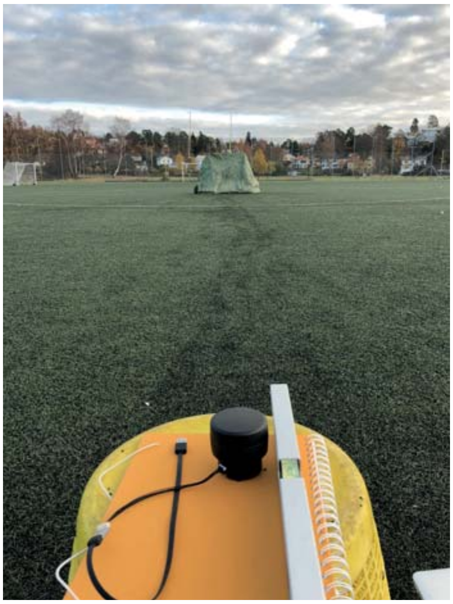


Figure 4.1: The test environment for the hardware test

5 Results

5.1 Hardware performance

Since the hardware test was only meant to measure how much the collected data deviate from the real stationary position of the obstacle, the mean of all azimuth values are not the interesting parameter but rather the standard deviation of the azimuth values.

Throughout all testing and development, minor wobbling of the rotating module could be observed for the Sweep LiDAR.

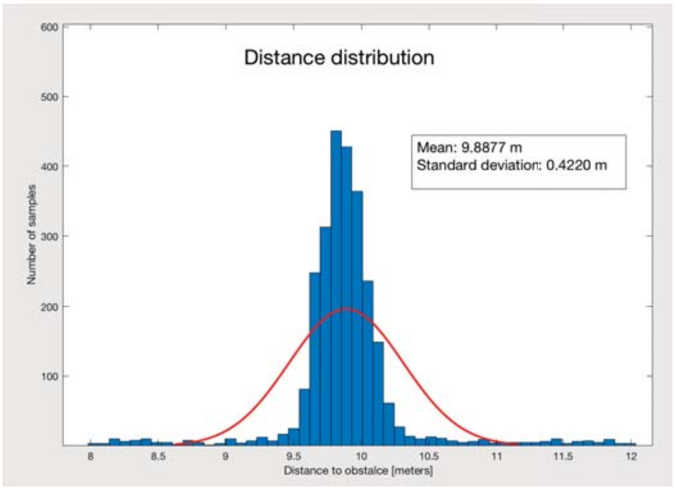


Figure 5.1: Distance distribution to a static obstacle at a range of approximately 10 meters. Result from experiment mentioned in section 4.5.3

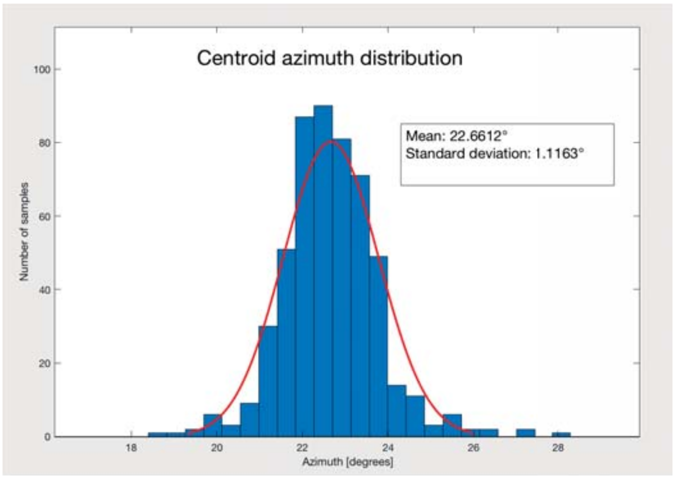


Figure 5.2: Result of the experiment to verify the precision and tracking of the centeroid of a static obstacle. Result from experiment mentioned in section 4.5.3

5.2 Prototype in Python

The first graphical prototype was written in Python using the library PyQt-Graph [5] and can be seen in figure 5.3. It successfully plotted all data from the LiDAR in real time, with no visible latency. Throughout the testing of this prototype points appeared to move around their actual positions. Walls would not appear entirely straight despite being so and the errors appeared to increase as the distance between the Sweep and the static objects were increased.

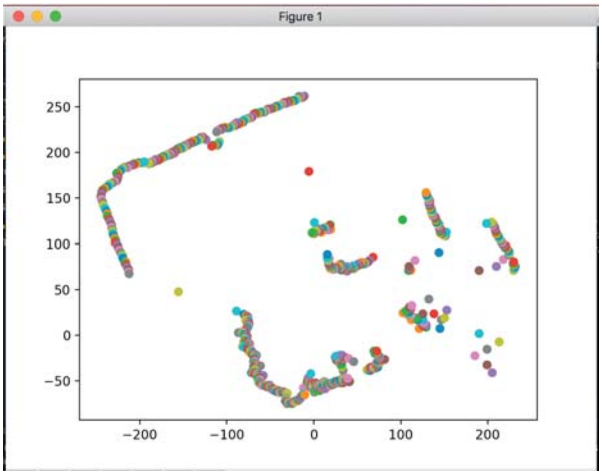
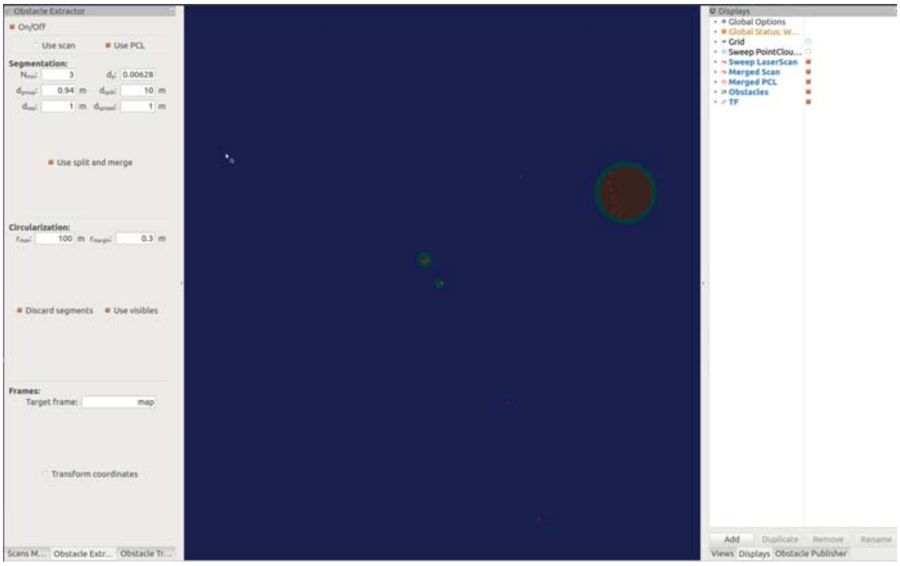


Figure 5.3: Visualization prototype written in Python for the Sweep

5.3 The obstacle detector package

The obstacle detector package was tested in both static and dynamic environments. The dynamic environment took place on an open field with a circulating car. All data was recorded and analyzed afterwards. Figure 5.4 shows the car as the top right big circle and figure 5.5 shows the car as two detected circular obstacles. Despite the Sweep's stated maximum range of 40 meters, no data points were received from the car if driven 15 meters or further away from the sensor. The maximum speed for which the package could detect and follow the car as an object within the range of 15 meters, was approximately 9 km/h. Figure 5.6 shows the result of a less strict parameter tuning. Here the Sweep is located in the forest with no moving obstacles in its surrounding and still struggles hard in determining obstacles. Despite having the Sweep sending point clouds in a synchronous manner, the application would mostly lag between samples and not have the same update frequency in the GUI as the rotational frequency of the sensor.



Obstacle Extractor

[✓ On/Off]

Use scan [✓ Use PCL]

Segmentation:

N_{min}: 3 d_p: 0.00628

d_{group}: 0.94 m d_{split}: 10 m

d_{sep}: 1 m d_{spread}: 1 m

[✓ Use split and merge]

Circularization:

r_{max}: 100 m r_{margin}: 0.3 m

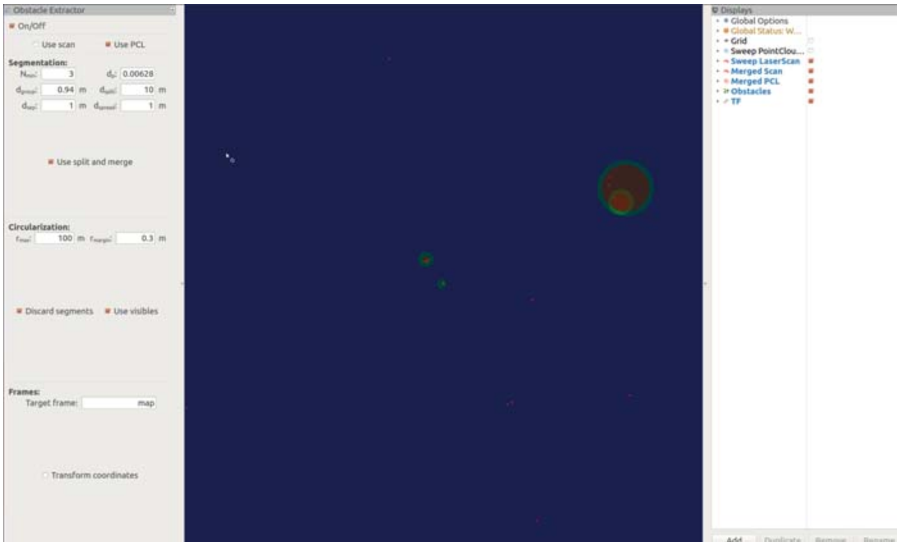
[✓ Discard segments] [✓ Use visibles]

Frames:

Target frame: map

Transform coordinates

Figure 5.4: The obstacle detector package detecting a car as one cluster



Obstacle Extractor

🟥 On/Off

○ Use scan 🟥 Use PCL

Segmentation:

N_{min}:	3	d_p:	0.00628
d_{group}:	0.94 m	d_{split}:	10 m
d_{sep}:	1 m	d_{spread}:	1 m

🟥 Use split and merge

Circularization:

r_{max}:	100 m	r_{margin}:	0.3 m

🟥 Discard segments 🟥 Use visibles

Frames:

Target frame: map

○ Transform coordinates

Figure 5.5: The obstacle detector package detecting a car as two overlapping clusters

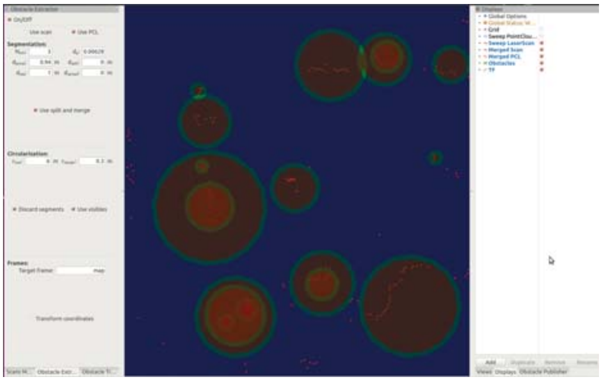


Figure 5.6: The obstacle detector package struggling with identifying trees in a forest

5.4 PCL and author's implementation

After the disappointing sensor performance in outdoor environments all following tests for this implementation was done indoors with both stationary and moving obstacles. The test environment was the Candela office space and the performance of the application was measured in how well walking people could be detected and tracked. The centroid of all detected obstacles in the GUI are marked with a white circle. All red points belong to a cluster are hence obstacles.

Most of the time all obstacles were detected fast and tracked flawlessly in the application. No obstacles were detected that should not have been detected. There was however a threshold of performance around 18 obstacles at a time, for which the GUI started to update slower than the rate of which data arrived. Figures 5.8 and 5.9 show around 10 detected obstacles for which the application would run with no visible lag.

Due to some problematic functions in the PCL library there were difficulties in updating detected obstacles and clearing old ones. Red points would at times linger between point clouds, as a result of an unsuccessful clearing between point clouds. This phenomenon can be seen in figure 5.9 around the left-most detected obstacle. There, a number of red points linger from the previous point cloud without belonging to any obstacle in the current point cloud. All tests were conducted with a rotational frequency of 4 Hz and a sampling frequency of 1000 Hz.

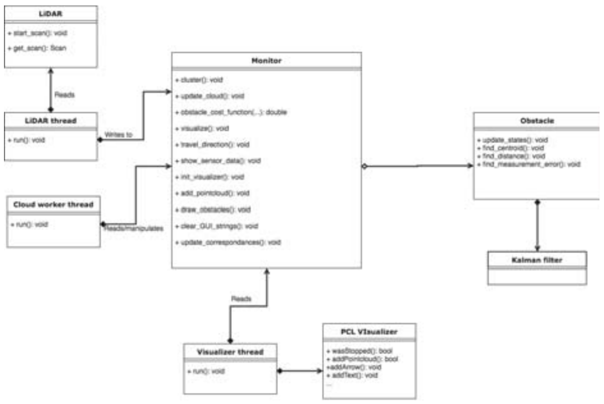


Figure 5.7: UML diagram of proposed software solution

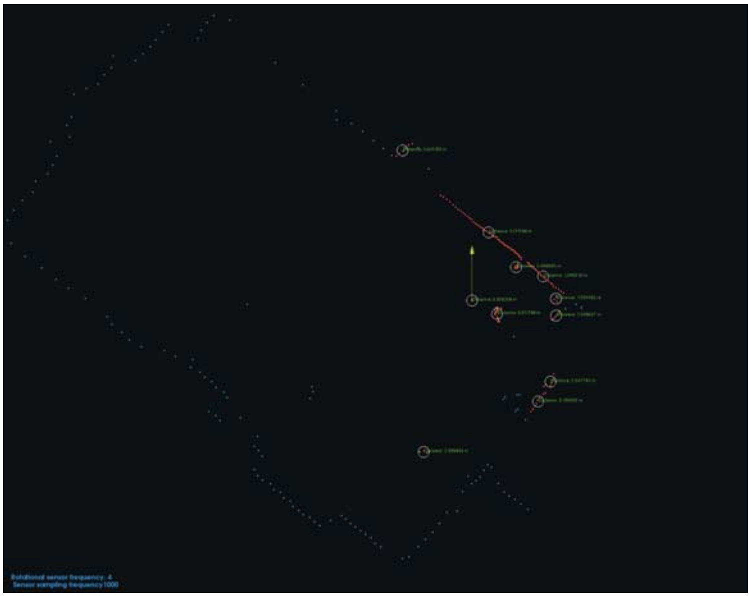

Rotational sensor frequency: 4
Sensor sampling frequency:1000

Figure 5.8: A screenshot of the author's implementation, showing a room scan. The yellow arrow marks the direction of the Sweep

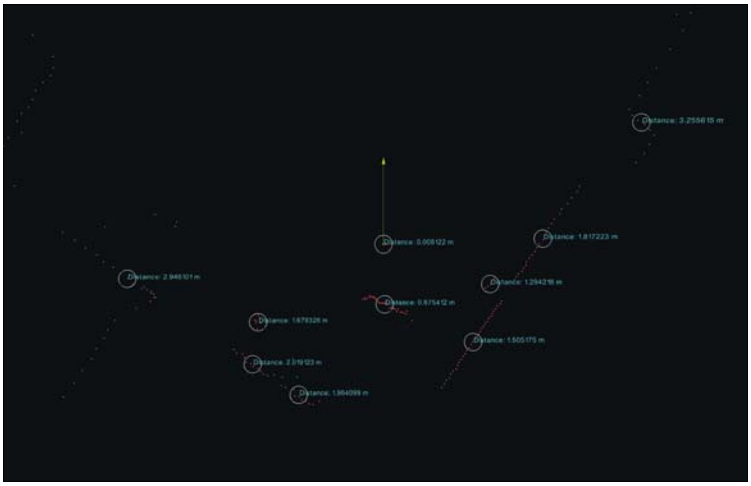


Figure 5.9: A screenshot of the author's implementation, showing a close up of a room scan

6

Analysis

6.1 LiDAR sensor

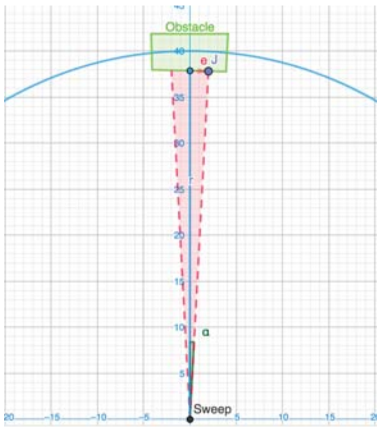


Figure 6.1: Aerial image showing a detected obstacle

$$e = r \cdot tan(\alpha) \tag{6.1}$$

In an optimal scenario, obstacles that are stationary, should also appear to be stationary by the LiDAR sensor. In a more technical approach this would mean that the measured distance and estimated azimuth to the centroid of a stationary obstacle should be the same between all consecutive point clouds. When working with real time systems that utilize sensors, the property of precision and resolution is important due to the fact that it can leave less to approximations and estimations. The tests conducted for this LiDAR sensor were done in order to obtain the degree of uncertainty of all measurements. As can be seen in figure 5.2 the hardware test showed an interesting attribute of the sensor. Despite experiencing some visual fluctuations of points during the initial Python prototype phase, this hardware test proved this suspicion of the sensor inaccuracy. A standard deviation value of $1.1°$ for the azimuth to the centroid of an obstacle is a significant value. Looking at equation (6.1) and figure 6.1, it is possible to convert this error and conclude that obstacles can on average appear to fluctuate 0.78 m perpendicular to the laser beam from its actual position, at the maximum range of 40 meters. In general, inserting the azimuth error value of $1.1°$ in equation (6.1) gives a linear equation between the distance to an obstacle and its approximate spatial error perpendicular to the laser beam.

Moving on to the sensors precision in distance measurement, it is harder to conclude the performance for the sensor's entire range. Figure 5.1 shows a standard deviation of 0.4 meters at the obstacle distance of around 10 meters. Since there were no clear indications of distance errors of such proportion when testing the author's implementation at the Candela office, there is reason to believe that the result from distance measurement test conducted in section 5 is faulty. Article [9] discusses how airborne LiDAR technology can be used over forests to measure the height of trees and vegetation by counting how many laser pulse replies are received from a single laser pulse transmission. When a reflecting object is not entirely solid a part of the light might pass through which is the case for leaves and bushes. This is a probable outcome of the tarpaulin fence; the laser light passed through the first layer to reflect a second time upon hitting the surface of the second tarpaulin layer, located approximately 0.3 meters behind the first layer. Not only does this give an insight to the inaccuracy of the distance test but also one of the drawbacks of object detection for this specific application. In cases of an undesired aiming of the Sweep, all laser samples for a boat might only hit the windows. Either that could result in no obstacle detected at all or a faulty distance measurement.

When analyzing the range of the Sweep LiDAR there are multiple things to consider. With no vertical field of view, the reliability of the aiming of the laser becomes very important. As mentioned section 5.1 the Sweep appeared to wobble slightly throughout all testing. Once again, looking at equation (2.3) small changes in the aiming angle can greatly influence where laser beams hit, depending on the distance to the target. With no specific apparatus for aiming the Sweep and with no stabilizing platform, small mechanical imperfections as this could have influenced the detection of the testing obstacle and the detection of the car when investigating the obstacle detector package, possibly missing the targets by aiming too high. Adding to this the great variety of heights for different boats, it becomes clear that a stabilizing platform for the Sweep would probably not be sufficient for this application. It would be hard deciding a good inclination, that compensates for the mechanical imperfections of the Sweep.

As mentioned in section 2.1.1 rotating LiDARs tend to break more often than solid state LiDARs due to moving parts. Having kept that in mind, along with the device's estimated lifetime it is possible to conclude scanning LiDARs have a great weakness when used in mobile configurations. Despite always testing the Sweep in a stationary position, it broke and needed to be repaired twice during the thesis project. This also gives an indication that sensor breakdown would be likely to occur on the Candela boat even with or without a potential stabilizing platform.

6.2 The ROS obstacle detector package

The obstacle detector package was originally written for an indoor robot with two scanning LiDARs utilizing fixed azimuth scanning. The software would not track an obstacle if its point cloud shape was close enough to a straight line. This was early recognized as a problem due to two reasons. Firstly, computational power would be consumed to look for walls that are non existing at sea. Secondly, obstacles that were supposed to be tracked risked, at certain angles, to have their point cloud shapes close enough to straight lines, to get lost in the tracking. As mentioned in section 5.3, the tracking of dynamic obstacles would sometimes appear inconsistent. They would disappear and only have a trace of its last detection lingering some two seconds in the GUI before also disappearing. The reasons for this inconsistent tracking with the obstacle detector package are many. First of all, the author of said work uses a 2D scanning LiDAR of much higher angular resolution and yet higher rotational frequency than the Sweep. The parameters for his models were hence tuned for a sensor of much higher performance. Despite adjustment of these model parameters, no desirable result could be obtained and inconsistent tracking would still occur. Lastly, it is possible that the mentioned car in the tracking experiments of section 4.5.2 would, for some angles, have its point cloud shape close enough to a straight line for it to be considered a wall. It would then fall in and out of the tracking algorithm and could thus explain why some obstacles have overlapping circles, or traces of circles from where it has traveled, seen in figures 5.5 and 5.6.

6.3 PCL and authors implementation

The open source library PCL offered many beneficial functionalities for the final implementation of this master's thesis. Multiple flaws were however detected during the development and testing, which will be discussed in this section. The most important function it provided was the graphical interface. Building a graphical interface from scratch is very time consuming especially for real-time applications. With the results presented in this thesis report, it is possible to conclude that the library is more optimized for static point cloud processing rather than for real-time processing. As discussed in section 5.4 there were no resource efficient functions for working with point clouds or the geometric figures plotted in the window. There were many problems detected when trying to update the view between point cloud samples, such as clearing detected obstacles and entering new ones. To solve the problem, many new functions had to be implemented and customized, such as `"draw_obstacles()"` and `"clear_GUI_strings()"` seen in figure 5.7. These functions had to be iterative and redundant in order to work with the library, slowing the overall performance of the application. This is most likely the reason why the application would run slower as more obstacles were detected, forcing the resource inefficient functions to process more data. If it would have been possible to work with a more streamlined library, built specifically for real-time processing and presentation of point clouds and geometrical figures, the performance of the the application could have been improved greatly.

7 Conclusion

In conclusion, this master's thesis addresses multiple key factors of object detection and avoidance using LiDAR technology intended for a hydrofoil speedboat.

The initial steps included a lot of investigative work regarding sensor models and required sensors stabilization. After deciding that the 2D LiDAR Sweep was the most feasible sensor for this thesis work, together with the fact that building a stabilizing gyro for the sensor would not be possible, efforts were put into investigating the hardware precision and developing the software.

By using and analyzing three different software approaches, it was possible to address key features and find weaknesses in all approaches. The main difficulty found was to visualize data and obstacles in real-time. The best approach was using the C++ compatible library PCL, for which there was some support for visualizing point clouds in real-time. It was possible to plot point clouds, clusters and shapes. However, using PCL for this purpose proved to be not optimally suited and resource inefficient, resulting in crashes and poor performance.

The performance of the Sweep LiDAR also proved to be less than promised and not well suited for this application. By analyzing the sensor specifications and its tested precision in range and horizontal perception to obstacles, it was possible to conclude that resolutions and precision were not enough for this application. To summarize the result of this master's thesis, and to answer the problem formulation question, it is not feasible to implement driver assistance for a hydrofoil speedboat, with mentioned techniques and hardware.

7.1 Future work and improvements

7.1.1 Hardware improvements

As discussed in section 6.1 rotating LiDARs entail many difficulties that are solved with solid state LiDARs. Not only would such sensors last longer and be more resilient for this specific demanding configuration, but also improve object tracking, power consumption, device cost and reliability. Having a sensor device that is much less likely to break in a critical situation means much in the sense of reliability and safety. Furthermore, having a LiDAR that produces 3D point cloud data rather than data in 2D, implies more details and information about each detected object. As discussed in section 4.4 the difficulty in mapping the same object between two consecutive sample frames, is to have a good enough correspondence function. By knowing more about an object's 3D shape, it is possible to utilize a more rigorous cost function, making tracking more reliable by minimizing the risk of mixing objects together.

7.1.2 Software improvements

The major software flaw in the proposed solution with PCL is the effectiveness of the graphical drawing tools. PCL's visualization packages appear to be intended for offline plotting of point clouds rather than real-time plotting. According to the documentation the visualization class can not be used by multiple threads in an application, and warns about possible crashes. Further more there are no efficient functions to redraw shapes or update texts in the GUI. Before each new point cloud can be shown in the visualizer, all previous geometric shapes and texts need to be erased sequentially and new ones need to be drawn, forcing excess computational power. In addition to this, all showable objects, such as texts, shapes and point clouds share the same memory space of IDs, meaning there is no uniform way of only deleting texts or only deleting point clouds. This is most probably the reason why the GUI started to run slower once more objects were detected.

Bibliography

[1] The making of a d.i.y brushless gimball with Arduino. `http://www.instructables.com/id/DIY-Brushless-Gimbal-with-Arduino/`, 2015. Accessed: 05/09/17.

[2] Candela Speedboat AB. Achieving the impossible. `http://www.candelaspeedboat.com/`. Accessed: 02/01/18.

[3] Ramiya Anandakumar M., Nidamanuri Rama Rao, and Krishnan Ramakrishan. Segmentation based building detection approach from lidar point cloud. *Egyptian Journal of Remote Sensing and Space Sciences, Vol 20, Iss 1, Pp 71-77 (2017)*, (1):71, 2017.

[4] Lynn; Hover Franz S. Bandyophadyay, Tirthankar; Sarcione. A simple reactive obstacle avoidance algorithm and its application in singapore harbor. *Field and Service Robotics. Ed. Andrew Howard, Karl Iagnemma, Alonzo Kelly. Vol. 62. Berlin, Heidelberg: Springer Berlin Heidelberg*, page 455–465, 2010.

[5] Luke Campagnola. Pyqtgraph. `http://www.pyqtgraph.org/`. Accessed: 01/09/17.

[6] Stacey Carrier. Considerations for diffuse reflection spectroscopy. *American Laboratory*, 48(7):1 – 4, 2016.

[7] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. *2015 IEEE International Conference on Computer Vision (ICCV), Computer Vision (ICCV), 2015 IEEE International Conference on, Computer Vision, IEEE International Conference on*, page 3029, 2015.

[8] Open Source Robotics Foundation. Ros. http://wiki.ros.org/. Accessed: 08/03/18.

[9] Celia García-Feced, Douglas Tempel, and Maggi Kelly. Lidar as a tool to characterize wildlife habitat: California spotted owl nesting habitat as an example. 108:436–443, 12 2011.

[10] Fabian Gieseke, Cosmin Eugen Oancea, Ashish Mahabal, Year = 2015, Christian Igel, and Tom Heskes. Bigger buffer k-d trees on multi-many-core systems. http://ludwig.lub.lu.se/login?url=http://search.ebscohost.com.ludwig.lub.lu.se/login.aspx?direct=true&db=edsarx&AN=edsarx.1512.02831&site=eds-live&scope=site. Accessed: 16/01/18.

[11] Business Insider. 10 million self-driving cars will be on the road by 2020. http://www.businessinsider.com/report-10-million-self-driving-cars-will-be-on-the-road-by-2020-2015-5-6?r=US&IR=T&IR=T, Online accessed 23-01-2018. Accessed: 16/01/18.

[12] LeddarTech. Automotive lidar solutions. https://leddartech.com/automotive/, 2018. Accessed: 25/02/18.

[13] Point CLoud Library. Pcl. http://pointclouds.org/. Accessed: 01/01/18.

[14] Velodyne LiDAR. Hdl-64e. http://velodynelidar.com/hdl-64e.html. Accessed: 16/01/18.

[15] Panasonic. Panasonic develops 3d lidar sensor enabling 3d detection of distances with wide angle of view. http://news.panasonic.com/global/press/data/2017/09/en170911-2/en170911-2.html. Accessed: 16/01/18.

[16] Mateusz Przybyła. *Online Detection and Tracking of 2D Geometric Obstacles from LRF Data*. PhD thesis, Poznań University of Technology, 2017.

[17] Scanse. Theory of operation. `https://support.scanse.io/hc/en-us/articles/115006333327-Theory-of-Operation`, Online, accessed 09-02-2018.

[18] Scanse. User's manual and technical specifications. `https://s3.amazonaws.com/scanse/Sweep_user_manual.pdf`, Online, accessed 29-01-2018.

Lund University **Department of Automatic Control** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER'S THESIS *Date of issue* February 2019 *Document Number* TFRT-6074
Author(s) Erik Söderberg	*Supervisor* Kristian Sloth Lauszus, Candela Speed Boat AB, Sweden Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden Karl-Erik Årzén, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

Object detection and avoidance using LiDAR on a hydrofoil boat

Abstract

Robots in factories and vehicles on our roads are rapidly reaching the point of automation. Commercial markets require high standards of safety and reliability, which in turn demands strict requirements on software and hardware. This thesis presents an analysis of driver assistance for a hydro-foil speedboat using LiDAR technology. The LiDAR sensor measures 2D positions of surrounding objects and the control software processes the data in order to distinguish clusters and movement.

Keywords
LiDAR, Point cloud, clustering, real-time system

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-42	*Recipient's notes*
Security classification		

http://www.control.lth.se/publications/