TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

OSSI MARTIKAINEN

COLLABORATIVE SENSING WITH LIDAR IN AUTOMATED VEHICLES

Diplomityö

Tarkastaja: professori Karri Palovuori
Tarkastaja ja aihe hyväksytty
9. elokuuta 2017

ABSTRACT

OSSI MARTIKAINEN: Collaborative sensing with LiDAR in automated vehicles
Tampere University of Technology
Master of Science Thesis, 65 pages
January 2018
Master's Degree Programme in Electrical engineering
Major: Embedded systems
Examiner: Professor Karri Palovuori

Keywords: collaborative sensing, cooperative sensing, LiDAR, automated vehicles, 5G, ITS-G5

In recent years, traditional car manufacturers as well as other technology companies have been developing vehicles with an increasing number of automated functions. Their ultimate goal is to create an affordable, fully autonomous vehicle. One key element of autonomous vehicles is their ability to sense their surroundings. This can be done with the vehicle's own sensors but their capabilities in various scenarios can be limited. One way to tackle this problem is to exchange the sensory data between vehicles, thus improving the perception abilities of the vehicles. The method of exchanging sensory data is called collaborative sensing.

This thesis studied the different elements of collaborative sensing in automated vehicles. The work was carried out at VTT Technical Research Centre of Finland in Automated Vehicles team. In this thesis, a collaborative sensing software was implemented on VTT's two automated vehicles. The implementation utilized ranging laser scanners, LiDARs, to gather information about the vehicles' environment. Various communication methods were tested to enable the collaborative characteristics of the system.

Essential information was gathered about the LiDAR and the various communication methods. Two software test platforms were developed as well as an independent positioning module that was used in the collaborative sensing implementation. The sensor system was also tested in various weather conditions and two inventions reports were submitted regarding the use of LiDARs in adverse weather.

TIIVISTELMÄ

OSSI MARTIKAINEN: Collaborative sensing with LiDAR in automated vehicles
Tampereen teknillinen yliopisto
Diplomityö, 65 sivua
Tammikuu 2018
Sähkötekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Sulautetut järjestelmät
Tarkastaja: professori Karri Palovuori

Avainsanat: yhteistoiminnallinen havainnointi, LiDAR, automatisoidut ajoneuvot, 5G, ITS-G5

Perinteiset autonvalmistajat muiden teknologiayhtiöiden rinnalla ovat viime vuosien aikana alkaneet kehittää yhä pidemmälle automatisoituja ajoneuvoja. Jokaisen tavoitteena on lopulta tuottaa edullinen, täysin autonominen ajoneuvo. Autonomisten ajoneuvojen tärkeimpiä ominaisuuksia on niiden kyky havainnoida ympäristöään. Ympäristön havainnointi voidaan toteuttaa ajoneuvoon asennetuilla antureilla, mutta niiden suorituskyky voi tilanteesta riippuen olla rajattu. Eräs ratkaisu suorituskyvyn parantamiseen on kasvattaa datan määrää jakamalla anturidataa usean ajoneuvon välillä. Tätä toimintatapaa kutsutaan yhteistoiminnalliseksi havainnoinniksi.

Tässä työssä tutkittiin yhteistoiminnallisen havainnoinnin osa-alueita automatisoiduissa ajoneuvoissa. Työ tehtiin Teknologian Tutkimuskeskus VTT:llä Automated Vehicles tiimissä. VTT:n kahteen automatisoituun ajoneuvoon toteutettiin yhteistoiminnallisen havainnoinnin mahdollistava järjestelmä. Ympäristön havainnoinnissa hyödynnettiin ajoneuvoihin asennettuja laserskannereita, LiDAReita. Ajoneuvojen välistä kommunikaatiota testattiin useilla menetelmillä ja näiden menetelmien soveltuvuutta yhteistoiminnalliseen havainnointiin arvioitiin erilaisin kriteerein.

Työn aikana saatiin olennaista tietoa sekä LiDARin toiminnasta että erilaisista kommunikaatiomenetelmistä. Yhteistoiminnallisen havainnoinnin testaukseen kehitettiin kaksi testausalustaa. Tämän lisäksi ajoneuvoihin toteutettiin erillinen paikannusmoduli, jota käytettiin tämän työn lisäksi muissa VTT:n automatisoitujen ajoneuvojen projekteissa. Ajoneuvojen anturijärjestelmiä testattiin myös vaihtelevissa keliolosuhteissa ja näiden testien tuloksena tehtiin kaksi keksintöilmoitusta, jotka käsittelivät LiDARin hyödyntämistä haastavissa keliolosuhteissa.

PREFACE

This thesis was carried out at VTT Technical Research Centre of Finland in the Automated Vehicles team. I would like to thank the whole team for giving me their full support during the making of this thesis. Especially my supervisor Pasi Pyykönen and my colleague Ari Virtanen provided invaluable knowledge without which this thesis would not have been successful. Finally, I extend my thanks to my wife Taru, who has supported me through the entire process of this thesis.

Ossi Martikainen
Tampere, 30.1.2018

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

5G	Fifth generation of mobile communication technology
API	Application Programming Interface
CAN	Controller Area Network
CCC	Collaborative Cruise Control, also known as Cooperative Adaptive Cruise Control (CACC)
DDS	Data Distribution Service
ENU	East North Up
ETSI	European Telecommunications Standard Institute
GNSS	Global Navigation Satellite System
IDL	Interface Description Language
IMU	Inertial Measurement Unit
ITS	Intelligent Transportation System
ITS-G5	Communication technology designed for ITS
LiDAR	Light Detection And Ranging
MEC	Mobile Edge Computing
MQTT	Message Queue Transport Telemetry
OCI	Object Computing Inc.
OMG	Object Management Group
UTM	Universal Transverse Mercator
V2V	Vehicle-to-Vehicle
WAN	Wide Area Network
WGS	World Geodetic System
Qt	A cross-platform software framework

1. INTRODUCTION

Collaborative sensing is a concept in which data from multiple sensors from multiple actors is gathered and used collectively. One of its application areas is the automotive industry. Modern cars use radars to adjust the cruise control speed to match that of the vehicle in front of them. Collaborative sensing continues from where the field of view of the car ends. It enables the car to see past obstacles and through harsh weather conditions by utilizing the sensory data provided by other vehicles and intelligent infrastructure near it. This is especially useful as car manufacturers are trying to develop automated vehicles. The more the vehicle sees and understands about its surroundings, the safer its actions can be.

1.1 Needs and requirements

This thesis is tightly coupled with the VTT Technical Research Centre of Finland automated vehicle development. The ultimate goal of VTT's automated vehicle research is not to produce a completed self-driving vehicle for the market but to study the world of automated driving and its challenges. The goal of the thesis is to expand VTT's development platforms so that collaborative sensing can also be studied in automated vehicle environment.

Automated vehicles require three main components: actuators for controlling the vehicle without a human driver, sensors to perceive the vehicle's surroundings and the vehicle itself and finally a processing component to make rational decisions on how to control the vehicle. Human brain can process visual data amazingly fast. It can derive and predict essential information in a way that the most sophisticated artificial intelligence cannot. This needs to be compensated by the automated vehicle with advanced environment perception produced by intelligent processing of data from different sensors. Although sensor technology keeps evolving, it will always have its limits. Line of sight is crucial for many sensors and in various traffic scenarios the line of sight to important objects can be lost. Collaborative sensing is one way of tackling this problem.

The need for collaborative sensing is best explained with an example scenario. Lane changing is a vital function for an automated vehicle in an urban environment. Changing a lane requires a good knowledge of the free space around the vehicle. For an automated vehicle, this means that it should have a 360° field of view with its sensors. Even if the vehicle's sensors could cover its surroundings, the line of sight to the direction of the lane change could be blocked by another vehicle. If another vehicle nearby equipped with sensors could provide additional situational awareness through collaborative sensing, the

lane change could be completed much safer. Automated vehicles should be able to drive without the assistance of collaborative sensing since sometimes there is no traffic around or there are no external sensors to collaborate with. Nevertheless, it is a safety increasing technology and it should be acknowledged in the development of automated vehicles. It should be noted that the collaborative sensing technology could prove to be useful in more limited scenarios where multiple collaborative vehicles are controlled by the same actor. These scenarios could include for example an intelligent transportation fleet on public roads or automated traffic in industrial environments where the traffic is mainly controlled by a property owner.

An implementation of collaborative sensing with laser scanners was developed for this thesis project. The implementation provides a mean of testing the main research question: How well does the combination of modern sensor and communication technology meet the requirements of collaborative sensing in automated vehicles? This question has to be separated into two aspects: collaboration and environment perception.

Three key requirements for collaboration were tested in this thesis. First requirement is an accurate positioning system. All measurements have to be tied down to a single coordinate system so that every vehicle using the sensory data from external sources can understand where the measurements were made[1]. This is a requirement for both the sender of the sensory data and its receiver. The other two requirements deal with communication. The second requirement is that the transfer speeds of the communication methods have to be high enough to support the sending of sensor data from one collaborative actor to another. The third tested requirement is the low communication latency. The data needs to be sent fast enough, otherwise the environment could have changed between sending the data and receiving it. For example, a view of an intersection area could be totally useless if it is received two seconds after the measurement is made. Multiple vehicles could have come to the field of view of the sensor after the measurement was made. That would make the sent data not only useless but dangerous if assumptions about the state of the intersection are made based on it.

The overall performance of the sensor data processing in evaluated in order to find the challenges and possibilities of environment perception with LiDARs. This evaluation includes environmental considerations as well as an analysis of the object tracking and recognition software. Speed, accuracy and robustness of the system are critical. If movement of dynamic objects can be estimated well, longer latencies can be accepted assuming that the information about the movements of all the objects can be included in the sent sensor data. There are also many other requirements for collaborative sensing in automated driving such as security issues[2] but they are left outside the scope of this thesis.

1.2 Research projects

This thesis is part of two ongoing VTT projects: RobustSENSE and 5G-Safe. RobustSENSE is an EU ECSEL funded project with an aim to set the ground for automated driving in all weather conditions. 5G-Safe is a Tekes Challenge Finland project that is carried out by a consortium of 10 Finnish companies with an aim to improve different traffic related services and road safety with 5G communication technology.

1.2.1 RobustSENSE

RobustSENSE approaches the challenge of harsh weather conditions by developing a sensor platform that is able to self-monitor its operation and adapt to changes in the weather. The sensor platform is developed so that it can be utilized in all levels of vehicle operation from driver assistance to automated driving. Self-driving vehicles are being developed around the world by several companies and consortia but until the challenges of operating in all weather conditions have been overcome, fully automatic driving cannot be accomplished. By bringing together experts from digital data and transportation RobustSENSE advances the robustness of automated driving and creates new ideas for automotive industry. LiDAR software developed in this thesis project is one of the practical outcomes of the RobustSENSE project.[3]

1.2.2 5G-SAFE

5G-Safe's mission is to find new practical applications that new 5G technology enables in traffic and road safety. Data gathering services are developed to collect information about vehicles, roads, weather and other traffic related issues. This information is processed in centralized servers as well as Mobile Edge Computing (MEC) servers. The project includes several use cases whose aim is to validate the technologies possibly enabled by the 5G mobile communication technology.[4]

1.3 Structure of the thesis

Chapter 2 introduces the state of the art research in the collaborative sensing in automotive applications. It focuses on research related with automated vehicle development since the collaborative sensing implementation of this thesis is based on that.

Chapter 3 introduces the automated vehicle development and the automated vehicles used in this thesis. It also presents the different sensors used in the automated vehicles and covers the operating principle of LiDARs.

Chapter 4 explains the design of the sensing software for the LiDAR. The software includes object tracking and recognition modules that utilize the measurements of the LiDARs in VTT's automated vehicles.

Chapter 5 discusses the different communication methods used for vehicle-to-vehicle communication. These methods include DDS, MQTT and ITS-G5.

Chapter 6 explains the design of the collision warning software. This software module includes the practical collaborative sensing elements of the thesis.

Chapter 7 includes the performance assessments of all the software and hardware components developed and used in this thesis.

Chapter 8 presents the conclusions on how well the developed sensing system and communication system work as a whole.

2. STATE OF THE ART

2.1 Research areas

Research of collaborative sensing in automotive applications can be divided into two perspectives. The narrower perspective looks at direct applications that the collaborative sensing enables. Connected Cruise Control (CCC) is a good example of such an application. In CCC a fleet of vehicles exchanges speed and acceleration information so that the accelerations and decelerations of the vehicles can be optimized reducing the risk of collision, minimizing fuel consumption and improving traffic fluency. A vehicle in the fleet can measure the speed of the vehicle in front of it and broadcast that information to the vehicles behind it even if the vehicle in front doesn't have any communication capability. The vehicles behind the sensing vehicle can adjust their velocity accordingly if there are changes in the velocities of other vehicles in the fleet even though they cannot see the changes themselves. The accelerations and decelerations can also be planned ahead so that unnecessary braking can be avoided.[5]

The wider perspective deals with more abstract problems of collaborative sensing. These problems include challenges such as real-time requirements[1], restrictions in the field of view of a vehicle[6] and authentication of the data received with vehicle-to-vehicle (V2V) communication[2]. This thesis focuses on the wider perspective and provides an implementation of sharing sensory information from one vehicle to another.

Many European research projects in automated vehicles are based on the Intelligent Transportation System (ITS) standards. The subdivision of the ITS standard for collaborative sensing is the Collaborative ITS or C-ITS standards. Even though these standards have been developed since the 1980's they are still under constant refinement. The implementation of collaborative sensing developed in this thesis utilizes the same technology and information as described in the C-ITS standards, but is not fully compatible with the standards. The implementation's focus is in the practical tests and evaluations but it can easily be modified to be compatible with the standards.[7]

2.2 Collaborative sensing in automated vehicle development

Even though automated vehicle producers such as Tesla have presented advanced autonomous driving functions, the advances of collaborative sensing in automated vehicles are still limited to research projects. Separate collaborative sensing functions have been researched by simulations[5], [6], [8] and actual vehicles[1], [2], [9]. The communication protocols have also been tested and developed in many studies[10]–[12].

Thomaidis et al. developed an object tracking system[1] that merged the data from an ego-vehicle radar with a location information sent by a collaborating vehicle. They demonstrated that using a Wi-Fi connection, it is possible for two vehicles to share their information and to synchronize the transferred data so that it is valid even after latencies created by the transmission. This also proved that the positioning system of an automated vehicle can be accurate enough for the collaborative sensing purposes.

Obst et al. described a method of checking the plausibility of objects whose information has been received through V2V communication[2]. Their system analyzed the surroundings with a commercial MobilEye camera and gave an estimate of the plausibility of received V2V messages including position data of another vehicle. They demonstrated that off-the-shelf products can be used to accurately verify the validity of information received by V2V communication by using object tracking algorithms. This result indicates that the object tracking software implemented in this thesis could also be used for V2V validation.

Different traffic scenarios have been modeled in [5], [6] and [8]. The simulation results show promising results for the benefits of collaborative sensing even in scenarios where multiple surrounding vehicles are incapable of receiving or sending V2V data. The scenarios modeled include collaborative control of multiple vehicles and lane changes by sharing local maps with adjacent vehicles.

An advanced environment perception system using LiDARs was developed by Maclachlan and Mertz.[13] Their work included an object tracking software developed for a moving vehicle. Their work showed the challenges such as feature extractions in object tracking with LiDARs in automotive applications. Their system was able to create collision warnings for a moving vehicle equipped with LiDARs. They tested their software on 263 hours of recorded LiDAR measurements. The resulting software was able to correctly generate warnings but it was far from ideal. 60 % of the high risk warnings generated were false positive. Greatest reason for the false positives was an error in the velocity estimations of the tracked objects. The study was conducted with a similar LiDAR as in this thesis and it shows well how demanding the task of object tracking with LiDARs is.

This thesis is unique in a way that it incorporates fully functional automated vehicles with complete sensor setups that enable comprehensive environment perception, high-performance positioning and communication between vehicles. The only limitation is that the thesis is restricted to using only two communicating vehicles and any large-scale tests cannot be performed in practice.

3. AUTOMATED VEHICLE DEVELOPMENT

3.1 Automated vehicle platforms

Two robot cars were under development at VTT during the making of this thesis. The cars were the main development platforms on which the environment perception software was built. These vehicles were named Marilyn and Martti because it was necessary to separate the automated vehicle development from the original vehicle manufacturers, who did not have a part in the development of the automated functions.

Marilyn is based on a Citroën C4 and Martti on a Volkswagen Touareg. Both vehicles have automatic gear boxes for ease of actuator installations. The vehicles serve as research platforms for multiple VTT projects. Marilyn's focus is on projects involving automated driving in urban environments whereas Martti's development focuses on projects where automated driving is performed outside urban areas. The vehicles are presented in figure 1.

Figure 1. *Automated vehicle platforms. Marilyn is on the left and Martti on the right.*[14]

Various sensors were installed on both of the vehicles to handle mainly positioning and front side environment perception. These sensors are presented in chapter 2.3. Plans were made to increase the number of sensors to cover the rear side of the vehicles and also update the sensors to produce more robust environment perception. Main computer model for both vehicles is the Compulab IPC2 which is designed for industrial use. Both vehicles are equipped with five of these computers running Linux operating systems.

3.2 LiDAR

3.2.1 Operating principle

LiDAR is short for Light Detection And Ranging. It is a distance measuring sensor equipped with a laser transmitter and a light detecting receiver. LiDAR measures distance

by measuring the time of flight of a reflected laser pulse. LiDARs can be equipped with a rotating mirror that allows measurements in multiple directions with a single set of a transmitter and a receiver. Additional transmitters and receivers can also be installed to increase the number of scanning planes. [15]

3.2.2 Sick LD-MRS

The LiDAR used in the VTT project RobustSENSE was The Sick LD-MRS 800001 although some of the software development was conducted with almost identical IBEO LUX LiDAR. The developed tracking and object recognition software works with both LiDARs because they have identical APIs. The Sick LiDAR mounted on Marilyn is shown in figure 2.

Figure 2. *Sick LD-MRS LiDAR mounted on Marilyn's front bumper.*

The Sick LD-MRS information is based on the Operating instructions datasheet[16] from Sick AG. The LD-MRS LiDAR provides advanced range measurements simultaneously in eight layers. The eight layers are produced from two integrated four-layer scanners. The central scanning range of a single four-layer scanner is 85° but the measurements can be extended to 110°. Scanning outside the central scanning range provides measurements only in 2 planes from each scanner. The orientation of the scanning planes of the 8-layered LiDARs is asymmetrical. The angle between two planes of the LiDAR is roughly 1°, but

the value varies based on which two planes are compared and at which horizontal angle. The maximum angle difference between the highest and lowest measurement plane is 6.4°. The plane angles of a single Sick LiDAR device are visualized in figure 3.

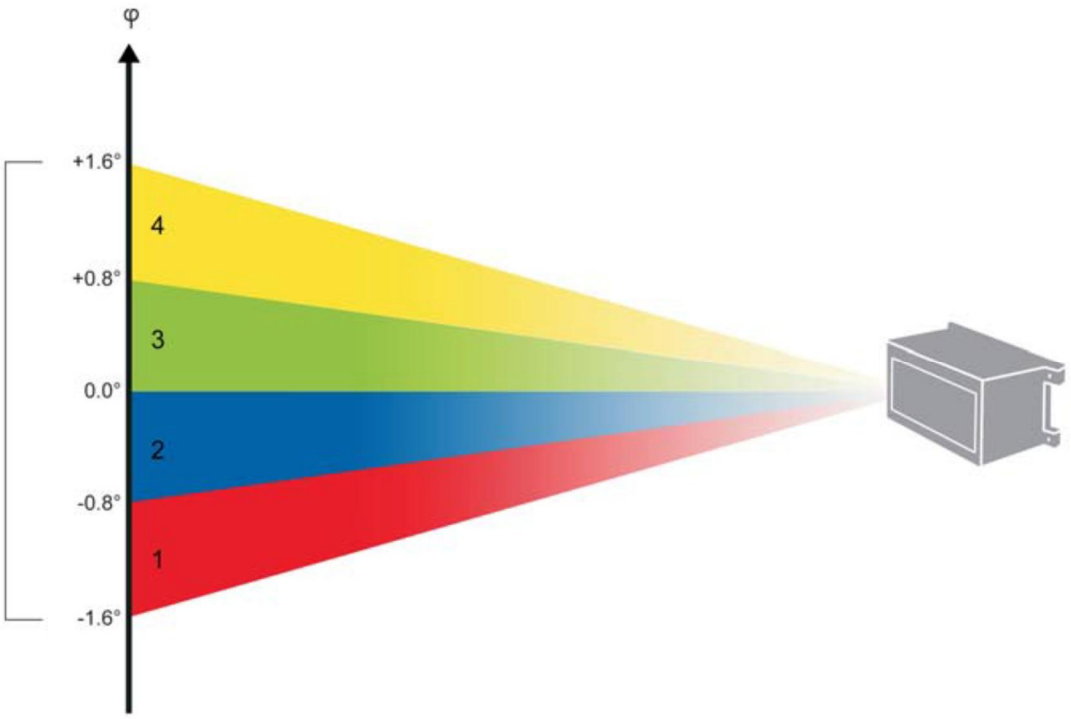

Figure 3. *Measurement plane angles of a single LiDAR device.*

The scanning frequency of the sensor can be adjusted, but it affects the resolution. The horizontal resolution range is between 0.125° and 0.5° depending on the scanning area and frequency. Central scanning area provides a higher resolution when lower scanning frequencies are used but the resolution is reduced to 0.5° in the whole field of view if the frequency is increased to the maximum value. The frequency can be adjusted between 12.5 Hz and 50 Hz. It is worth noting that a single scan provides measurements only from one device meaning that only 4 layers are measured simultaneously. The nominal operating range of the LD-MRS is from 0.5 to 300 meters but the actual maximum range is much smaller and it depends on the reflectivity of the surroundings. The actual maximum distance according to the data sheet is presented in figure 4.

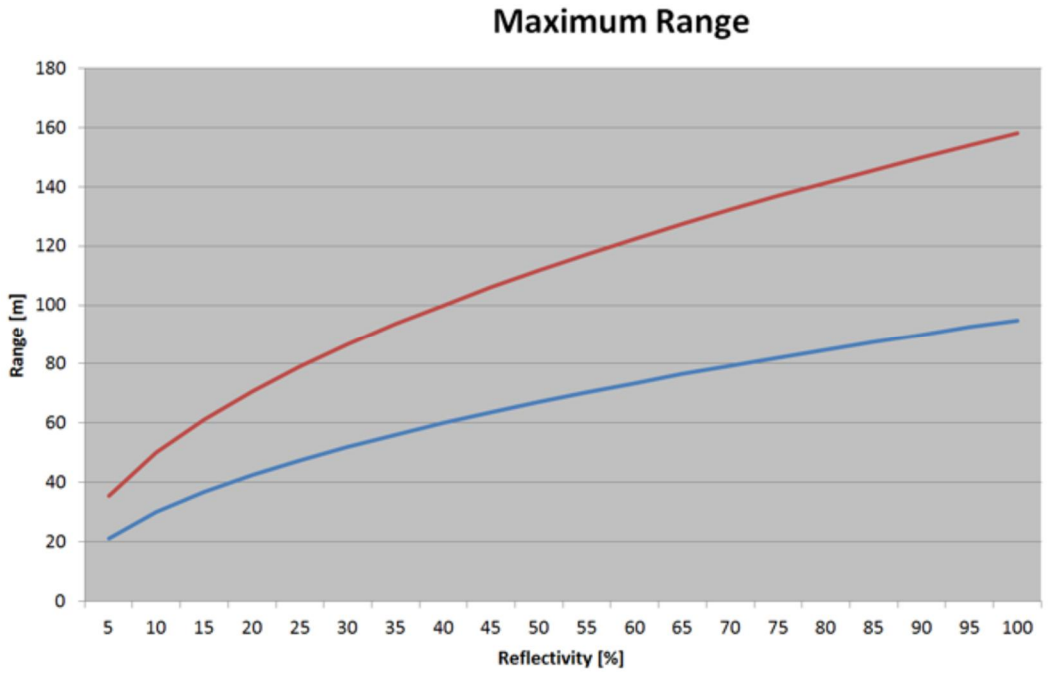

Figure 4. *Sick LD-MRS operating range dependency on object reflectivity. The upper red curve represents standard LD-MRS sensor and the lower blue curve LD-MRS HD sensor.*[16]

The range depends also on the horizontal angle of the current measurement point because of the design of measurement optics. The relation of measurement angle and maximum distance is presented in figure 5.

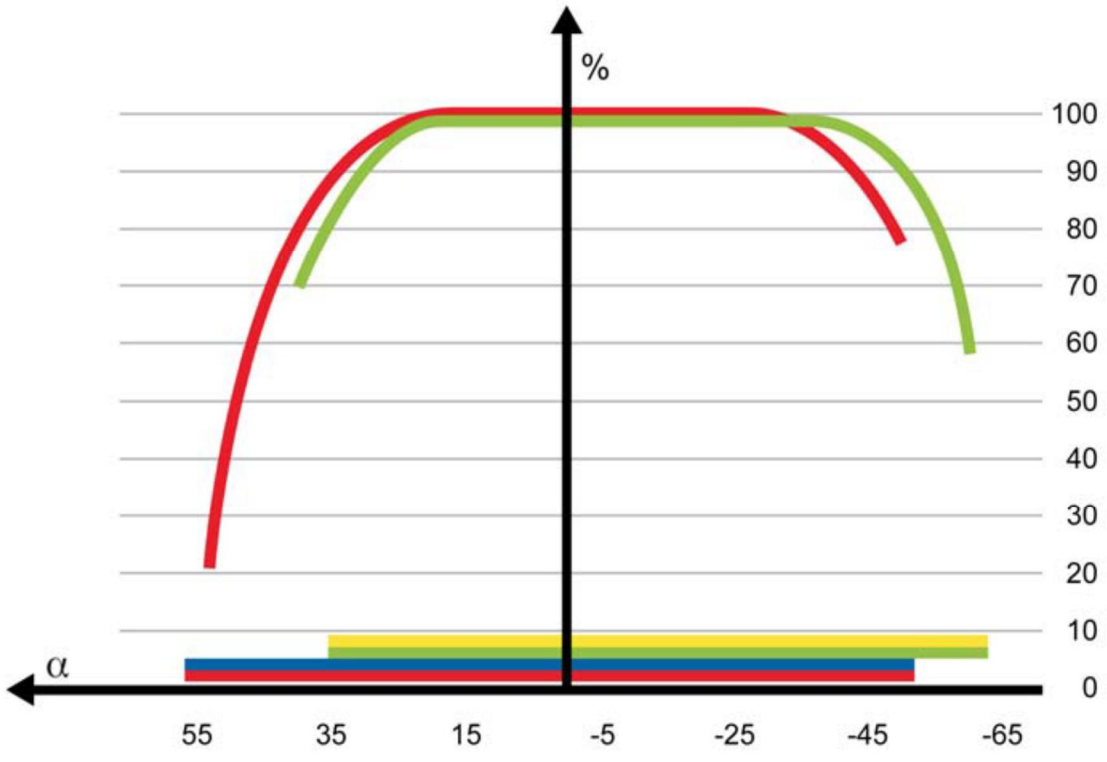

Figure 5. *Sick LD-MRS relation of view angle to operating range.*[16]

Vertical axis describes the ratio of maximum measurement distance for the angle and the absolute maximum measurement distance. Horizontal axis describes the horizontal angle of the measured point in the LiDAR's coordinates.

The sensor's operating principle is improved by measuring multiple echoes from a single laser pulse. This allows the sensor to detect objects that are located behind transparent surfaces. This also improves the sensor's performance in harsh weather conditions such as rain since the sensor also detects objects after the laser pulse has first reflected off a rain drop.

One heavy duty version Sick LD-MRS HD LiDAR was installed on robot car Martti. The sensing distance of the heavy duty version of the LiDAR is limited when compared to the regular version, but it is much less affected by weather conditions. The heavy duty LiDAR on Martti serves simultaneously as a reference sensor for the regular LiDARs and as a backup for harsh weather conditions, where the visibility to close proximity is very limited with the regular sensors.

3.3 Other sensor technologies in vehicle platforms

It is necessary to describe the other sensor technologies on the vehicle platforms because environment perception is deeply connected to sensor fusion. Many other sensors are used to support the operation of environment perception with LiDARs and ultimately the goal is to combine all sensor input to create a robust situational awareness.

A navigation module was developed alongside the main thesis project. The module combines positioning data with inertial measurements and odometry to produce valid position data between GNSS measurements and even during short periods when GNSS signal is lost or when the GNSS data is invalid. The output of the module is improved by an online inertial sensor calibration based on GNSS data. The module is the first fully functional and independent sensor fusion module for the VTT's current vehicle platforms.

The sensor setup of the vehicles was nearly identical. Both vehicles included LiDARs, thermal and stereo cameras, positioning sensors and inertial measurement units (IMUs). Data in the vehicles' CAN buses were also read but no control signals were sent to the CAN buses for safety reasons.

3.3.1 Radars

Both of the vehicles were equipped with two types of radars for measurements in shorter and longer distances. The longer range measurements are covered with a Bosch LLR2 77 GHz radar that can produce measurements from up to 200 meters. Shorter ranges are measured with a Continental SRR 20X radar. The Continental radar is used alongside the

long range radar because of its wider 150° field of view as opposed to the 16° of Bosch's radar.

3.3.2 Cameras

Both automated vehicles are equipped with a stereo camera system including infrared sensitive cameras and two infrared light sources. Marilyn is equipped with IDS HDR system that is installed right behind its wind shield. The cameras are independent and the stereo camera functionality is gained programmatically. A VisLab 3DV-E system is mounted on Martti's front bumper. The VisLab system provides a 636×476 disparity resolution. The maximum nominal operating range of the system is 88 meters but in automated vehicle scenarios the valid operating range is roughly 30 meters.[17]

Marilyn is also equipped with two FLIR thermal cameras. These cameras are used to identify warm objects and differentiate moving objects such as pedestrians and vehicles from the background.

3.3.3 GNSS

Both vehicles are equipped with two GNSS systems with separate antennas using GPS and GLONASS for positioning. The reason for using two systems is to validate the received location data. Since the antennas have fixed positions on the vehicles' roofs the distance between them is known. If the measured distance between the two is much greater than the actual distance, the measurements are known to be invalid.

3.3.4 Inertial measurements and odometry

An XSENS AHRS unit is used to measure the inertial movements of the vehicles. It provides measurements in all 6 dimensions: 3 for accelerations in its rest frame and 3 for rotations. It measures magnetic fields and works also as a compass for the vehicles.

Odometry measurements can be gathered via the CAN buses of the vehicles. Wheel velocities are the key measurements from the bus but it is also possible to read other important messages from the bus such as steering wheel and gas pedal position.

3.4 Communication systems

The vehicles are equipped with 2 communication systems for different purposes. Shorter distance communication is handled with an ITS-G5 system with small latency but also a small capacity. Larger data transfers are handled with a 4G LTE system. The 4G LTE system is also compatible with future 5G networks and it allows the testing of 5G technology still under development. Currently the 4G LTE network is used for reading GNSS correction data and aiding with software development.

All raw and processed sensor data inside the vehicles is handled with a Data Distribution Service (DDS) network. DDS is a reliable, high-performance protocol that is ideal for the real-time requirements of automated driving. The communication systems are covered more deeply in chapter 5.

3.5 Environment considerations

The research on automated vehicles at VTT is specialized in dealing with harsh weather conditions especially in Nordic weather. Rain, snow and fog bring out many challenges in automated driving. Some sensors are rendered useless by these different weather conditions but the automated vehicle should still be able to operate based on the valid data from other sensors.[18]

The challenges that weather conditions produce can be approached in three ways. The automated vehicles can be equipped with different kinds of sensors that can handle various weather conditions as a combination. The second option is to design individual sensors and processing of their data so that they can handle larger variety of weather conditions. Final option is to expand the field of view with collaborative sensing. VTT's two vehicle platforms utilize all of the approaches. They are equipped with sensors that can operate even in the harshest weather and in addition VTT is involved in projects where sensors are improved to handle specific weather challenges. For example, a LiDAR operating at longer wavelengths is under development. The longer wavelength has less absorption in water thus providing less noisy measurements in foggy and rainy weather situations. Finally, this thesis project provides the starting point for collaborative sensing between the two vehicles.

4. ENVIRONMENT PERCEPTION SOFTWARE

4.1 Software overview

The developed environment perception software consists of four main parts which are raw data processing, object tracking, object recognition and integration with inertial measurements. These parts are used to generate an entity that tracks and predicts the movements of nearby objects in the LiDAR's field of view.

Raw data processing covers all the steps that are taken to transform the raw point cloud data into trackable objects with simplified features. The steps include sorting the points based on their azimuth angle, filtering out measurements points coming from the ground, clustering the measurement points to separate objects and linearizing the clusters to create more robust and simplified tracking and recognition. Linearization in this thesis is a process where a set of measurement points is transformed into one or more linear models. The linearization process reduces the amount of handled data and creates insight on the shape of the sets of measurement points. The models created by the linearization represent the edges of the perceived objects. For example, a car seen by a LiDAR is typically shown as one or two lines, depending on the point of view. These lines can be represented as linear models. An example of a vehicle seen by a LiDAR with linearized measurement points is show in figure 6.

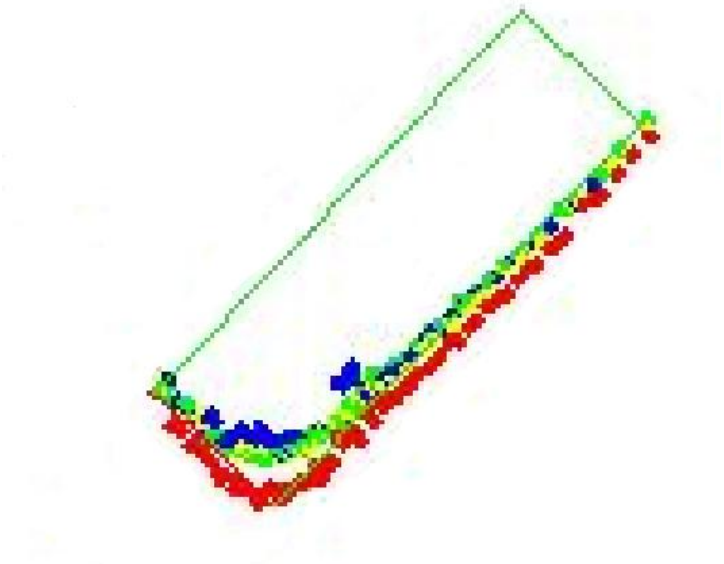

Figure 6. *Example of measurement point linearization.*

The measurement points created by the LiDAR are shown as dots and the linear models as lines. The software also projects two other models on clusters that appear to be vehicles to estimate the rest of the vehicle's edges.

The object tracking is implemented with a Kalman filter. Tracking enables predictions of the trajectories of perceived objects. Trajectory information can be utilized in e.g. predicting collisions[19].

Object recognition is implemented with a voting algorithm. The objects are divided into five classes: undefined, car, pedestrian, bike and obstacle. Each class has specific size, shape and movement characteristics. If a tracked object has the same characteristics as one of the classes the algorithm votes for that class. If a class gets a vast majority of the votes the object is set to that class. Similar voting algorithm was used by Mendes et al. for LiDAR data classification[19]. The object recognition could later on be used to estimate the possible movement of the object in the near future. For example, vehicles cannot move sideways but pedestrians can almost freely move in 2 dimensional space.

Since the software is to be used in a moving vehicle it needs to take into account the movement of the vehicle the LiDAR is attached to. Inertial measurements are produced with a separate positioning module which uses GNSS, inertial measurements from an IMU and odometry measurements from the vehicle wheels through the vehicle's CAN bus. Previous measurements from the LiDAR are transformed to the vehicle's coordinate each time a new measurement is received based on the movement of the vehicle.

In figure 7 an overview of the software operation is presented. The objects that the overview discusses are entities that are tracked aver multiple measurements with the LiDARs. The software is described in detail in chapter 4.3.

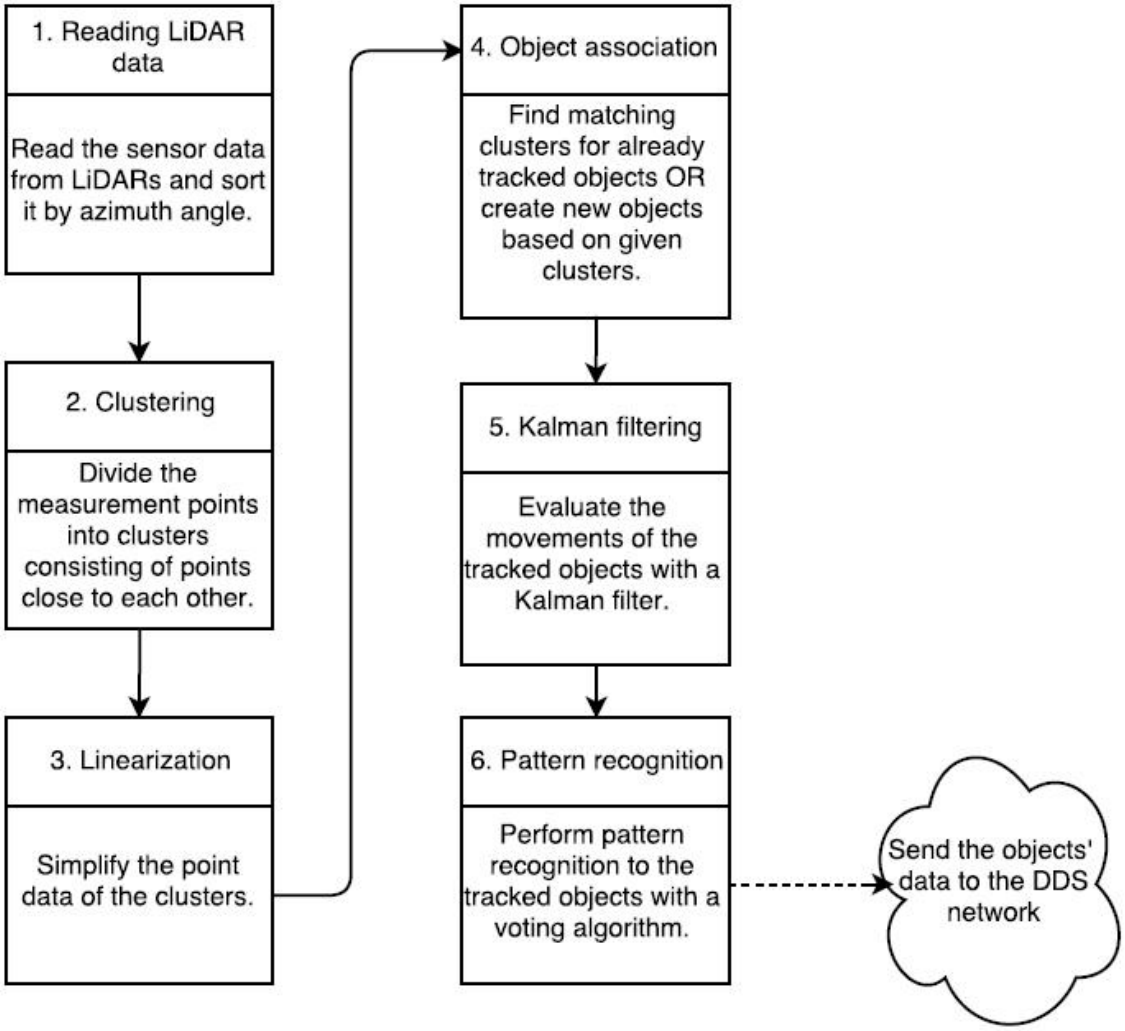

Figure 7. *Flow chart of the object tracking and recognition module.*

4.2 Theoretical background

The theoretical background of the environment perception covers four different areas: sorting algorithms, linearization, coordinate system transformations and tracking with Kalman filter. These areas are presented in the same order as they are utilized in the software. This subchapter gives a theoretical view of the algorithms used. The implementations and more detailed explanations of their characteristics are described in the chapter 4.3.

4.2.1 Sorting algorithms

Two sorting algorithms are implemented in this thesis: insertion sort and merge sort. These two algorithms are called comparison sorts. They define the order of elements in a data set by comparing a chosen attribute of an element. The performance of these sorting algorithms is typically defined by worst-case running time and average running time. The running times are expressed as the function of the number of elements in an input data

set. For example, insertion sort's worst-case and average running times are n^2, where n is the number of elements in the data set. Merge sort's worst-case and average running times are $n \cdot \log(n)$. These estimations of running times do not represent the actual time of running the algorithm with small data sets because they only take into account the most significant factors in the formulas. By increasing the number of elements in the input data set, the estimations become more accurate. They only give estimates on how the processing time increases when the number of elements in the data set is increased since absolute processing time depends on a number of different factors in the software and hardware.[20]

Insertion sort is an efficient sorting algorithm for small data sets. Its intuitive sorting method takes a single element from an unsorted data set one by one and finds a correct position for it in the sorted data set. The sorted position of the new element is found by iteratively comparing its chosen value to those of the sorted data set. In best-case scenario, the new element is placed in the first position in the sorted data set requiring only one comparison between the chosen values. In the worst-case scenario, the new element is positioned in the back of the sorted data set. This means that the comparison of values has to be done for each element in the sorted data set. It is crucial then to know if the values of the original unsorted data set are already in some sort of order. The best-case scenario, where the elements are already in order, the running time is linear.[20]

Merge sort is a more complex sorting algorithm. It is a recursive algorithm meaning that the sorting problem is first split into smaller problems until the problem of sorting becomes trivial. In merge sort, the input data set is split into smaller data sets until the remaining split data sets are sorted. For completely unsorted input data set, this means that each split data set contains only one element, thus being sorted. After the splitting, each remaining data set is merged with another data set. Since the two merged data sets are already in order, only the first elements in the two data sets need to be compared. The merge sort is an efficient sorting algorithm especially when combining two sorted data sets together and it is especially useful in this thesis when merging multiple already sorted sets of LiDAR data into one larger data set.[20]

4.2.2 Linearization

Linearization is used in the software to simplify the point clouds that the LiDAR produces. Two main linearization methods were implemented and tested. First tested linearization method utilized a combination linear regression and Random Sample Consensus (RANSAC) algorithms. The second algorithm tested was the Douglas-Peucker algorithm. Linearization aims to produce one or more linear models that fit the point cloud data as well as possible. A single linear model can be described by the following formula.

$$y = a + bx \quad (1)$$

The x and y represent coordinates in Cartesian coordinate system. The differences between the algorithms used arise from the different assumptions on the raw data. Linear regression produces the least squares fit to the available data but it assumes that the whole data set given to the algorithm belongs to the model[21]. The Douglas-Peucker algorithm also assumes that all data belongs to the model but it operates recursively based on a given threshold distance to produce an undefined number of linear models. RANSAC on the other hand allows the data set contain outliers – points that are not included in the model. It seeks for the model that contains the most inliers through iterative process.[22], [23]

Linear regression produces a single least squares fit by iterating twice over the given data set. Means of the x and y coordinates are calculated on the first iteration. On the second iteration variance of x and covariance of x and y are calculated. The values are used to calculate the value of b in the linear model with the following formula.[21]

$$b = \frac{\sum_{i=0}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=0}^{n}(x_i - \bar{x})^2} \quad (2)$$

$\bar{x}$ and $\bar{y}$ are the means of the x and y coordinates of points in the data set. $\sum_{i=0}^{n}(x_i - \bar{x})^2$ is the standard deviation of the x coordinates and $\sum_{i=0}^{n}(x_i - \bar{x})(y_i - \bar{y})$ is the covariance of x and y. The value of a in the linear model can be calculated with the following formula.[21]

$$a = \bar{y} - b\bar{x}. \quad (3)$$

The RANSAC algorithm is not the optimal solution for finding the best model but it allows finding multiple models in a single data set. The algorithm contains four elements. The first element is trial model creation. A trial model is created with a small data set. The original algorithm doesn't take a stand on how the model is produced. The second element is inlier counting. The algorithm is given a distance threshold and if a point in the data set is within that distance from the model line, it is an inlier. The third element is iteration. Model creation and inlier counting are iterated a given number of times. Increasing the number of iterations increases the possibility of finding the best possible model containing maximum amount of inliers for a linear model. The final element is model validation. If the number of inliers exceeds a given threshold, the model is accepted. This threshold can be an absolute value or a ratio of inliers to total number of points. The algorithm itself can be iterated by giving the outlier points of previous iteration to the current iteration's data set, thus allowing multiple models to be found. An example of one RANSAC trial is shown in figure 8. In the figure measurement points are shown as dots, trial model is shown as a solid line and the threshold distance is visualized by two dashed lines.[23]

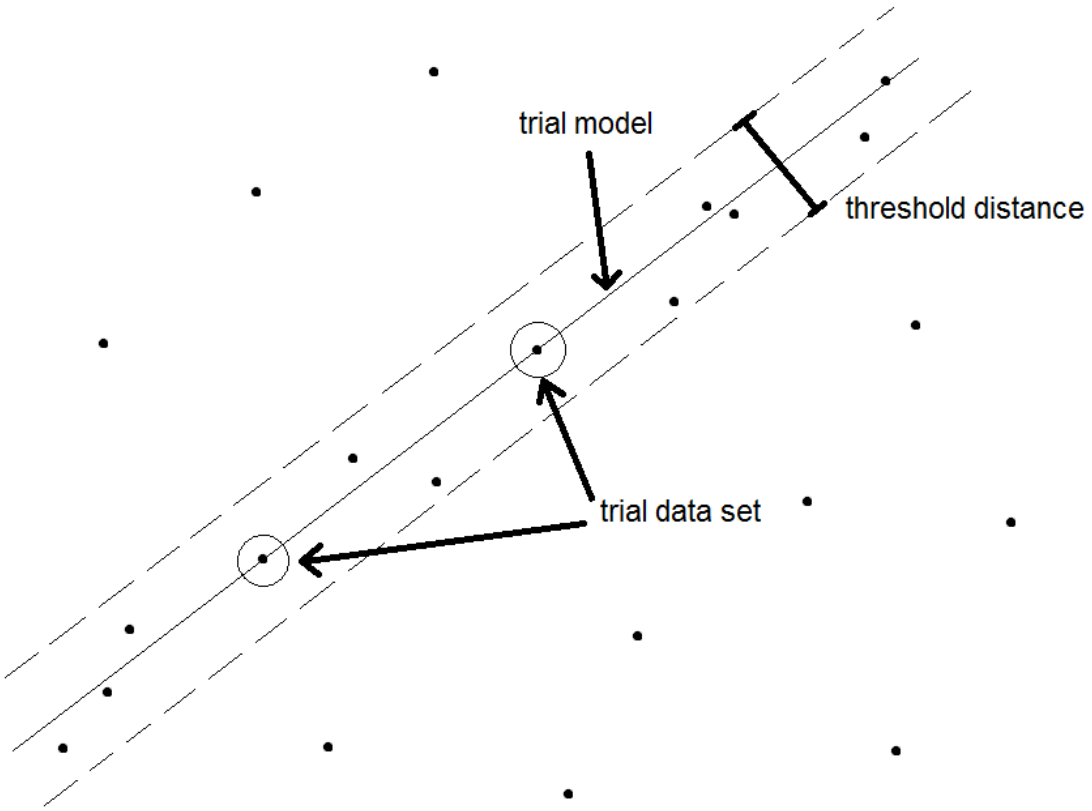

Figure 8. *Example of a RANSAC trial model.*

The figure shows a scenario where the complete data set containing all the measurement points does not show a clear line. Using linear regression would result in a random linear model, but RANSAC is able to produce a model that makes over half of the measurement points inliers. Finding a well-fitting model presented in the example figure on the other hand requires excessive iteration since there are over 250 different ways to choose a trial set of two measurement points and most of the trials result in poorly fitting models.

The Douglas-Peucker algorithm is an efficient method of reducing the number of data points in a set. In this application it is used to produce linear models of the LiDAR measurements. This algorithm assumes that all of the data points are organized by some property of the data. The LiDAR measurement points are organized by azimuth angle and thus they fit the algorithm without further processing. The algorithm works by examining a subset of the original data set on each recursive round. This data set is initially set to contain all of the points in the original data set. The recursive round begins by creating a line between the first and last points of the current data set. Then distances to each other point of the current data set are calculated. The point furthest from the line is taken under inspection. If the point is further away from the line than a given threshold value, it is set as the last point of the next recursive round's data set. If the point is closer than the threshold, the last point of the current data set is set as a corner point and the subset is

updated for the next recursive round. The first point of the next round is the new corner point and the last point is the last point of the original data set. The recursion will continue until the latest created corner point is the last point of the original data. A simple example of the algorithm is given in the figure 9.

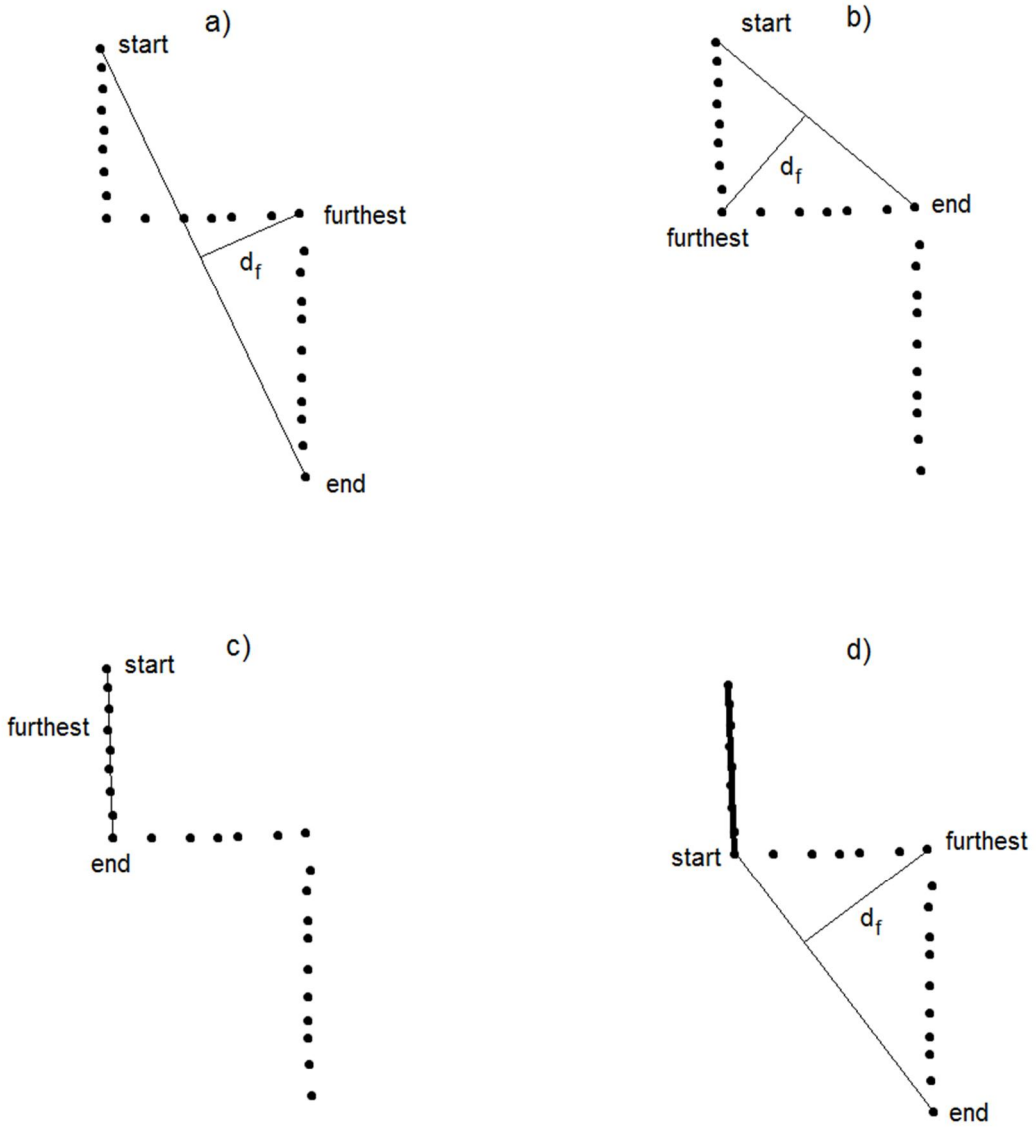

Figure 9. *Visualization of the Douglas-Peucker algorithm.*

The figure 9 represents four recursive rounds of the algorithm. The algorithm begins from the figure 9 a) where a line is drawn from the first point of the original data set to the last point of the original data set. The furthest point is sought and the distance d_f to it is calculated to exceed the threshold distance. The furthest point is thus set as the end point of the next round's data set. The next round is represented in figure 9 b). This round differs from the first round only in the way that the last point of the current data set is not the last point of the original data set. In figure 9 c) that represents the third recursive round, the threshold distance is not exceeded. This means that the created model is valid

and the data set for the next round is updated. End point of the current round's data set becomes the first point of the next round's data set and the last point of the original data set becomes the last point of the next round's data set in figure 9 d).

The comparison between the two linearization methods is covered in the chapter 7.1.3.

4.2.3 Coordinate system transforms

Coordinate system transforms are made in multiple parts of the software. The LiDAR expresses measurements in polar coordinates which are transformed into Cartesian coordinates. This transform is not necessary for the operation of the software, but it makes the calculations and functions of the software easier to comprehend. The other, necessary transforms are rotations. Two of the LiDARs in front of the robot cars are facing away from the center line of the vehicles. All measurements in the vehicle environment are represented in the vehicle's coordinate systems and thus all measurement points from these LiDARs must be rotated. The second need for rotation arises when the vehicle turns. This leads to the effect where objects appear to be moving in the LiDAR's field of view between two measurements even if they are stationary. This is corrected by performing another rotation to the previously perceived objects according to the change of heading read from the vehicle's location module. The vehicle can rotate around all three axes since a typical road is not completely horizontal. In the vehicle's coordinate system rotation around y axis is called pitch, rotation around x axis is called roll and rotation around z axis is called yaw. The coordinate system is presented in figure 10. The coordinate system is Cartesian and the positive direction of z axis is up. The y axis goes through the rear axle of the vehicle and the x axis runs through the center of the vehicle. Driving direction is to the positive direction of x axis.

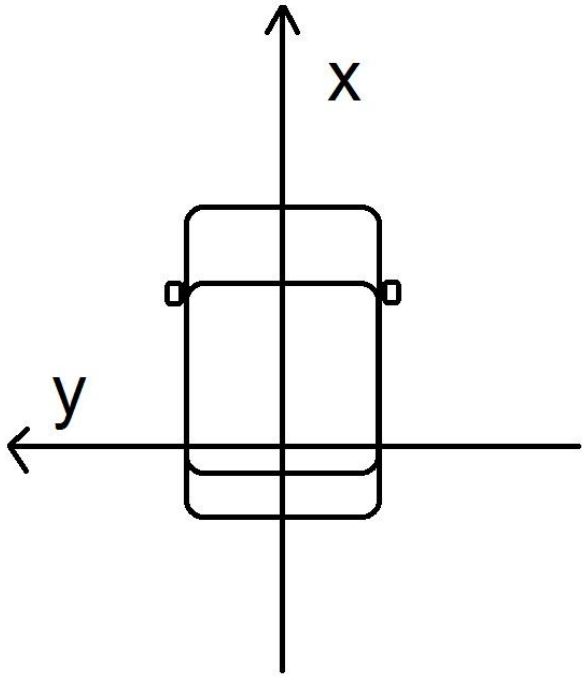

Figure 10. *Vehicle's coordinate system.*

The rotations are calculated with rotation matrices. Each rotation is calculated around a single axis. This means that three rotation matrices are needed when roll, pitch and yaw are transformed into changes in the Cartesian coordinates. The rotations in three dimensions are given with matrix R_x for rotation around the x axis, R_y for the rotation around the y axis and R_z for the rotation around z axis. These matrices are defined as follows.[24]

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{4}$$

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{5}$$

$$R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

The variables α, β and γ represent the angle of rotation around the x, y and z axes. The rotated coordinates are calculated by matrix product of the original coordinate vector and the rotation matrix. The coordinate vector $\boldsymbol{x}$ is defined as follows.[24]

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{7}$$

These rotations are performed around the origin. For filtering purposes it is also necessary to perform rotations around other fixed coordinates. These rotations are explained in more detail in chapter 4.4.1.

4.2.4 Kalman filter

The Kalman filter is an algorithm designed to provide optimal estimation of a state of the examined system. The filter reads measurements and recursively produces an estimate of the current state and a prediction of the next state of the system. The recursive nature of the algorithm allows it to be used in real-time applications. The estimates produced by the algorithm are typically more accurate than the measurements if they contain noise. One of the advantages of the algorithm is that it makes very few assumptions on the system. It only requires the system variables to have finite means and covariances. It also assumes that the noise in the system is zero-mean Gaussian noise, but it can also perform well in other cases such as the tracking of objects in this thesis.[25]

The core principles of one iteration of the algorithm are presented to express the underlying functions of the Kalman filter. One iteration of the algorithm consist of reading the measured state variables, calculating Kalman gains and covariance matrices, estimating the current state of the system, predicting the next state and re-evaluating the covariance matrices. In addition to these steps it is also necessary to determine the variances and covariances of the measurements. The covariance matrices describe the errors in the measurements and the estimations. These matrices describe what the errors of each measured and estimated state variable are and how the errors in one variable affect the errors in other variables. The estimations and the measurements have their own covariance matrices.[25]

After the measurements have been read, Kalman gains are calculated by comparing the covariance matrices of the measurements and the estimations. Kalman gain represents the confidence on the measurements and the estimations. Higher values of the Kalman gain indicate that more weight is given on the measured values and the estimations are less accurate. Lower values of the Kalman gain indicate a higher confidence in the estimations. The estimation errors are updated twice during each iteration. The first update is based on the Kalman gain and it represents how the estimations become more accurate after reading more measurements. The second update takes into account how the covariance of the state variables affects the errors. [25]

The current state of the system is also the main output of the Kalman filter. The current state is calculated as a weighted mean of the measured and estimated values. Kalman gains determine which values are given more weight. After calculating the current state an estimation of the next state is made. Estimation of the next state is based on the current state variables and their derivatives. For example, if the state variables of a system are the distance of an object and its velocity, the estimated next state is determined by taking into

account the acceleration of the object and calculating the movement of the object before next measurement. Even though the underlying mathematics are complex, the actual implementation of the Kalman filter is fairly simple and it can produce robust results.[25]

4.3 Object detection implementation

Object detection is the first step of the process in environment perception. The detection phase includes transforming the data into Cartesian coordinates, dividing the measurement points into clusters and linearizing the data from the clusters so that the tracking and recognition are easier to handle.

4.3.1 Preceding software and modifications

The developed software for the environment perception was built on two layers of API's. The first API was provided by the sensor itself and the second was an extension to the sensor API that was developed before the thesis work at VTT. The sensor provides the API through an Ethernet interface. After starting the sensor with an Ethernet command it will start sending measurement data after each measurement scan. Each Ethernet message contains raw measurement data from a single scan and a single measurement point is presented in polar coordinates.[16]

The API developed at VTT handles the reading of the raw measurement data and categorizes it into two classes: Observation and ObsPoint. Single measurement points are processed by the API and transformed from polar coordinates to Cartesian coordinates. The data of a single measurement point is stored in ObsPoint class instantiation. Points of each scan are stored to Observation class instantiation. After transformation to the Cartesian coordinate system the measurement points are rotated to match the vehicle's coordinate system which is presented in figure 10.

The API was redesigned for vehicle use because all of the sensor data is transferred through the DDS system in the vehicle. The new design included a publisher software that reads the Ethernet messages from the vehicles' LiDARs and sends them to the DDS network. The application also required a subscriber to read the data coming from the DDS network. The DDS network and other communication related software are described in more detail in chapter 5. The software was also updated to combine measurements from multiple LiDARs into a single Observation.

4.3.2 Sorting measurement points

The measurement point sorting was implemented in two phases to minimize the processing time. First organizing phase took place in the LiDAR's driver. This sorting

was necessary only for the 8-layer LiDARs with two sensor devices in them. Since the measurements of a single device are in order, the most natural selection for the sorting was merge sort algorithm[20]. Measurement points from each device were gathered into separate lists. A sorted list was created by comparing the first measurement points of each device list and then choosing the one to put on the sorted list based on its azimuth value. The point added to the sorted list was removed from the device list. The second sorting phase was implemented in the object tracking software. First tested implementation was the insertion sort. The LiDARs could be read in an order, where the measurement points are almost sorted, insertion sort was a fast and easy way to test the sorting. To improve the sorting speed, a merge sort was also implemented to be compared with the initial insertion sort implementation. The comparison results are described in chapter 7.1.1.

4.3.3 Clustering

After the points have been transformed into Cartesian coordinates they are divided into clusters. The clusters are managed by Cluster class. These clusters try to define which points are from the same object. The algorithm used for the clustering took advantage of the order in which the measurement point data is saved in the Observation class. The order was based on the azimuth angle of the measurement points. The algorithm takes a single point and calculates its distance to 100 former points. If the distance is within a threshold, the point is added to the same cluster as the point it was close to. A new cluster is created if there are no points within the threshold distance.

One of the challenges for clustering is to define the threshold distance. While the depth resolution is not distance dependent, the horizontal resolution becomes worse with distance. A simple solution for this problem was to use a distance dependent threshold which included a small constant for a more robust operation in small distances. A similar method has also been used by Thuy and Léon in their research[26].

4.3.4 Linearization

It is possible to track the clusters without further data processing but linearizing the clusters by creating simple models from a set of measurement points, the recognition process becomes much easier. The tracking can also include the direction of the cluster if the clusters are linearized even if the perceived object is motionless. The linearization process has also been used in former studies about LiDARs in automotive applications and it is proven to be a better solution than simpler bounding box[13], [27].

Two versions of the software were implemented with different linearization methods that were explained in chapter 3.2.1. The Douglas-Peucker algorithm's implementation was thoroughly explained in the theoretical background. The combination of RANSAC and simple linear regression is described next in more detail.

Application of the RANSAC algorithm is implemented as follows. All of the trial models are produced by choosing two points from the cluster and calculating a model that fits the two points. If the number of points in the cluster is small enough, all of the different two point combinations are tested. Clusters containing larger number of points are handled by randomly choosing two cluster points near each other. The randomness allows fewer number of iterations but it also creates the possibility that a fitting model that could be found is not found. This possibility of error can be lowered to almost zero with a large number of iterations. Even if the model cannot be formed, a single undefined model between two successful models can be handled by the tracker and object recognizer without major effects. After the model is formed, distances from all the cluster points to the model line are calculated. The point is considered to be inlier if the distance is within the threshold of 15 cm. The best model is chosen by the number of inliers. For the first model, at least one third of the cluster points need to be inliers. If there are none or a few outliers the model is considered to be valid by itself. If there are more outliers it is required that a second model is found. The second model is calculated again with RANSAC but it only takes the first model outlier points as input. The second model requires at least half of the points to be inliers to be considered as valid.

Linear regression is utilized after a model is found with RANSAC. By including only inlier points of RANSAC model it is possible to eliminate points that don't have a good fit from the model. Linear regression creates a model with least squares method[21] which allows a more accurate model to be created especially with small amount of model points. Consider the following situation of the figure 11. RANSAC can create a valid model that does not represent the data well since it only takes the first two points as input. Linear regression on the other hand can form a more fitting model by utilizing the whole data set.

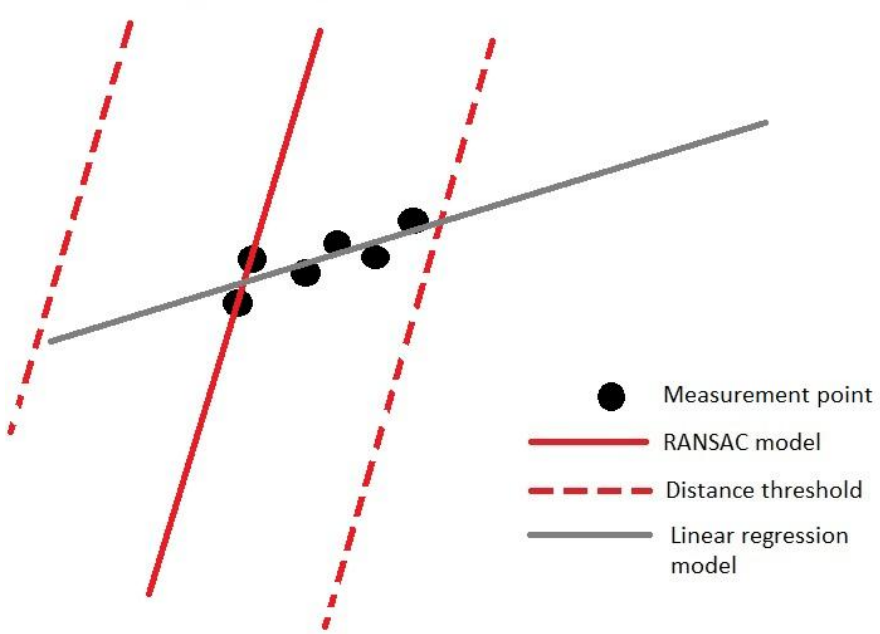

Figure 11. *Combination of RANSAC and linear regression.*

If two valid models are found they are validated once more by calculating the angle between the two lines they form. Typical trackable objects with multiple models in automotive applications are vehicles whose two model lines are perpendicular to each other. The angle between the formed lines has to be between 60° and 120° for the algorithm to accept them. The allowed angle range is large because the combination of measurement noise and randomness of linearization can create large errors in the coefficient b of the model.

After the linearization process the algorithm creates corner points for the cluster. The Douglas-Peucker algorithm produces the corner points as its output but the linear regression and RANSAC require additional calculations. First, the algorithm finds the minimum and maximum values of inlier point coordinates. Depending on the coefficient *b* it chooses either x or y coordinates. Values of b that are closer to 0 represent more perpendicular lines and it is more accurate to use minimum and maximum values of y coordinates that are also perpendicular. Values of *b* that are further from 0 are more accurate to calculate with minimum and maximum value of x coordinate. Corners are simple to calculate with minimum and maximum values for a cluster that has only one model. Cluster with 2 models is more complex since the orientation can vary and the starting point and ending point cannot be defined. The algorithm solves the problem by first calculating the intersection point of the two models. Then it calculates minimum and maximum points along both of the model lines and finally chooses the ones that are furthest from the intersection.

Other analyses are performed also on the cluster to make the recognition process more robust. A common problem for measurements with LiDARs is occlusion of objects. When

an object comes between the LiDAR and another object, the further object becomes partially or fully occluded. The size of the further object is reduced and it can even split into two separate clusters.[13]

To find partially occluded objects the software uses a method demonstrated by Maclachlan in [28]. In this method each point is marked as occluded or not occluded. A measurement point is occluded if either of its adjacent points is closer to the sensor and the closer point belongs to another cluster. After determining occlusion for each point the same is done for clusters. A cluster is occluded if its first or last point is occluded.

Another analysis for the clusters with two models is determining whether they appear as convex or concave from the LiDAR's point of view. Vehicles and other trackable objects appear always as straight lines or convex corners. Concave corners are thus easy to identify as static obstacles. To determine whether a corner is convex or concave a following method is implemented. A line is created from the first corner point to the last. If the middle corner point is further from the LiDAR than the line, the corner is concave. Otherwise it is convex.

The output of clustering and linearization in a real traffic scenario is shown in figure 12.

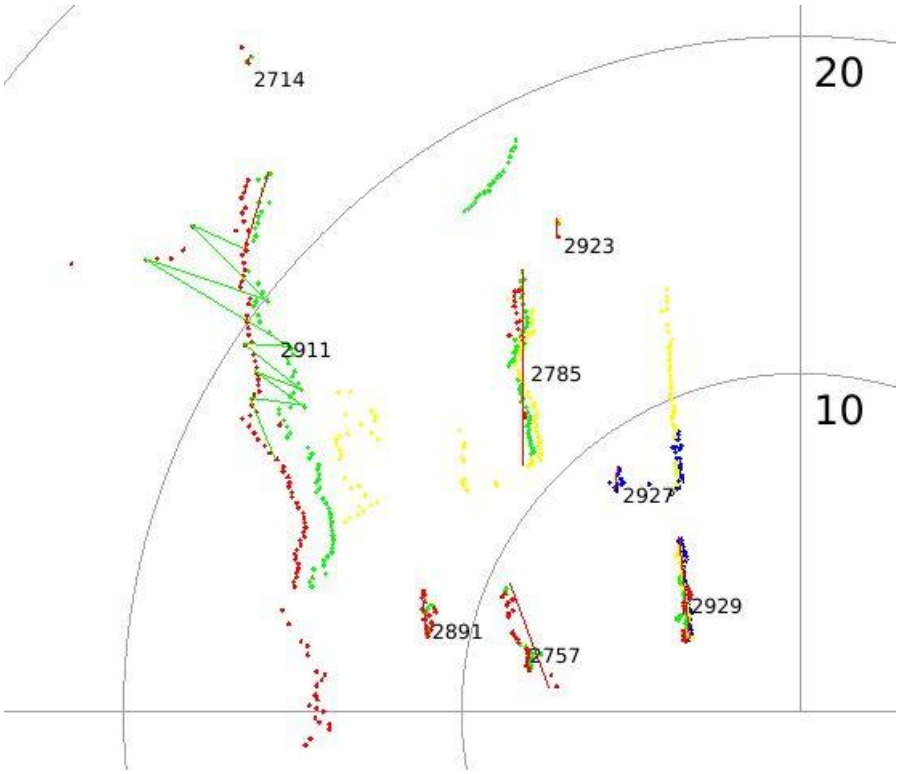

Figure 12. *Example output of clustering and linearization.*

The clusters in the figure are separated by unique identification numbers. Models created by the Douglas-Peucker algorithm are presented as lines. The figure shows a good example of the challenges in LiDAR measurement processing. Uneven ground creates

great challenges for the point filtering and results in errors. Some ground points are seen as actual objects, because they seem to be well above ground from the LiDAR's point of view. Object number 2911 is created by ground measurements. These measurements typically result in clusters that have much more corner points than actual objects and can be filtered out. On the other hand, some objects appear partially as ground points. The vehicle that is marked as object number 2927 on the right hand side of the figure is only partially interpreted as a real object as some of the measurement point are filtered away.

4.4 Tracking and recognition implementation

Tracking and pattern recognition are handled by two classes: Tracker and Object. Tracker gets fed the clusters every time they are processed and converts them into Objects. Movement of the objects is predicted with a Kalman filter that is handled inside instantiations of Objects. The tracker is run on a separate thread because the clustering and linearization requires a lot of processing time. For this reason the cache for clusters is protected with a semaphore structure.

After processing the objects with Kalman filter they are sent to a pattern recognition algorithm. The algorithm utilizes a voting principle that tries to categorize Objects into different types such as pedestrians and cars. The algorithm is more thoroughly explained in section 4.4.3.

4.4.1 Kalman filter for tracking

The object tracking is handled with Kalman filter. The filter tracks the center point of the Objects' corners. The tracking by center of corner points is not ideal since the shape of the object can change radically. This happens for example when a car's side becomes visible and one corner point is added to the Object. Introduction of new corner points and the change of the center point can create a false sense of acceleration. These situations have to be handled with special methods that are explained later.[13]

First step of object tracking from point cloud data is associating the clusters with the tracked objects. The Kalman filter needs to know what the current measured positions of the tracked objects are in order to continue the tracking. The connection between tracked objects and new clusters is made by comparing the center point location of each cluster to the predicted center points of all tracked objects. The cluster's center point is assigned as the new measured position of the object that it is closest to if the distance between the object and the cluster is small enough. If the distance between the cluster and the closest object exceeds the given threshold value, a new tracked object is created and its initial state is set according to the cluster's information.

The Kalman filter in this software has six state variables: x and y coordinates, velocities in x and y direction, angle and angular velocity. The acceleration was initially also

included in the state of the Object but it turned out to be too unstable to track. The Kalman filter is initialized when an Object is associated with a new cluster for the first time. This allows the initial state to be fairly accurate since the velocity of the object is also included in it. An iteration of the Kalman filter ends in predicting the next state. This allows a more accurate association of the Objects and clusters for the next set of measurements.

One of the key problems for the Kalman filter is that the measurements are very noisy and the linearization algorithm is not capable of reducing it enough. This was discovered after testing first versions of the filter. The object's dimensions don't typically change because of noise but the angle of the object is much more unstable. Two improvements were made to mitigate this problem. First was an algorithm that forces all found corner models to be perpendicular. The algorithm recalculates the intersection of two linear model lines by altering the linear model that creates shorter line. This produces a small improvement but it is not enough yet. The second improvement is filtering the corner point locations based on the angle of the object. The Kalman filter calculates the angle first and then rotates the corner points around the center point of the object.

4.4.2 Integration with inertial measurements

The movement of the ego-vehicle must be accounted for when processing LiDAR measurements. The movement can be split into two categories which are the rotation and the movement in the xy-plane. The pitch and roll angles of the vehicle and the LiDAR are handled by the Observation class and the angle information is transformed into changes of x, y and z coordinates of the ObsPoints. Pitch and roll affect how the sensor sees the shapes and that is why they need to be handled before the clustering. Hillsides create an exception for the rotations. On longer hills the rotation would cause all points to be considered as ground or sky since the car is tilted. The actual situation is that the LiDAR lasers point nearly parallel to the ground and it is not necessary to perform the pitch rotation. This scenario can be handled by tracking the pitch rate. If pitch rate is near zero for a longer period of time, the rotations are not necessary. The yaw angle on the other hand affects the location of the Objects but it doesn't distort their shapes. The resulting view from the current measurement after the rotations with roll and pitch describes the actual surroundings. That is why the information of the yaw angle change needs to be added to the last Objects' locations.

4.4.3 Voting algorithm for object recognition

Many studies have been done on object classification with LiDAR but they are often only superficially explained consider only specific seen objects such as pedestrians [19], [29], [30]. A simple voting algorithm was developed to classify seen objects into five different categories: cars, bikes, pedestrians, obstacles and undefined objects. Each tracked object is initialized with voting counters for each category. Votes are given for or against each

category by increasing or decreasing the category's counter. Votes are based on the object's size, shape and movement. For example, a 5 meter long object traveling at the speed of 50 km/h is much more likely to be a car than a pedestrian. The voting algorithm also takes into account the occlusion of the object. If the inspected object is partially occluded, the category counters are influenced less than if the object was fully in sight.

4.5 Playback software for recorded LiDAR measurements

Testing of the object tracking and recognition with the vehicles is time consuming and often impossible because of other development work done on the vehicles. For this reason, a playback software was developed for testing the object tracking and recognition based on recorded LiDAR measurements. The user interface of the software is shown in figure 13.

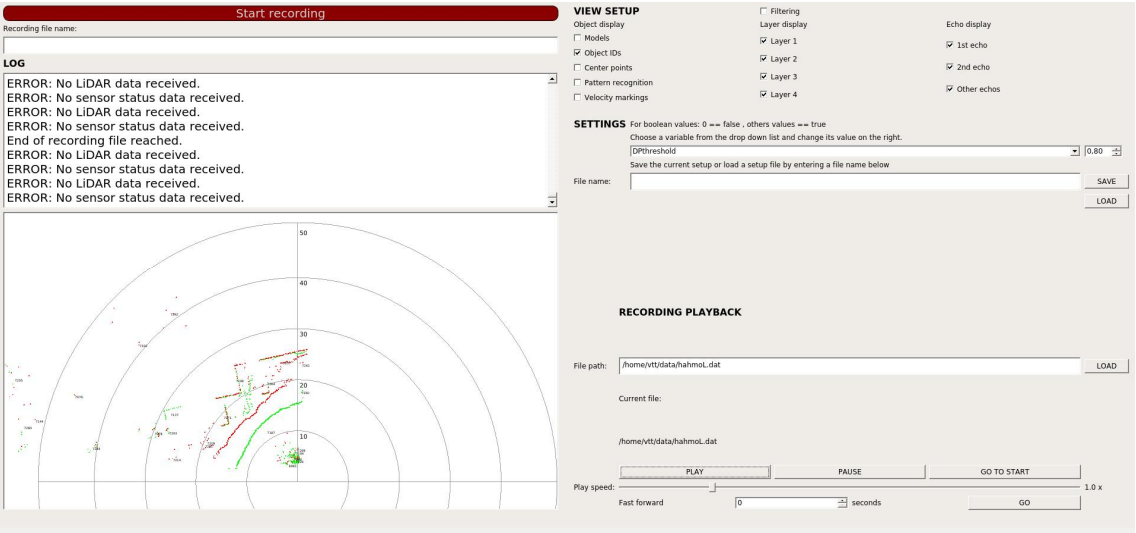

Figure 13. *Playback software for LiDAR measurements.*

The software is integrated within the object tracking software meaning that it can either listen to the LiDAR measurements received from the vehicle network or it can read the measurements from a file. Measurement points and perceived objects are shown on a 2D map of the vehicle's surroundings. The playback module emulates actual LiDAR measurements so the recorded information can be run through all the algorithms in the same way as the actual measurements. The interface also allows tuning of different algorithm variables online. This makes it easier to test various configurations. The playback speed can also be altered. This information is fed to the Kalman filter so that it can manage the tracking accurately in different playback speeds. The recording of the measurements is also handled with the same software.

Multiple display options were also added to the user interface. The user can toggle different layers of the LiDAR on and off. Also the display of multiple echoes can be

toggled. Multiple different attributes such as ID numbers and velocities of perceived objects can be shown over their location on the map.

5. VEHICLE-TO-VEHICLE COMMUNICATION

5.1 Implementation options

Communication between vehicles could be implemented with many different technologies. For this thesis, three different options were studied. The options were publish-subscribe-based DDS, Message Queue Telemetry Transport (MQTT) which is also a publish-subscribe-based messaging protocol, and finally the radio based automotive communication protocol ITS-G5. There were no prior experience at VTT on using OpenDDS over internet but since the same messaging protocol is used in VTT automated vehicles' internal networks, it was an interesting option to study for vehicle-to-vehicle communication. MQTT on the other hand had been used at VTT in automotive applications and it was a natural option for this thesis. The ITS-G5 had been also used in automotive applications and since it is designed for automotive communication, it was also a natural option.

5.2 5G technology

Even though 5G technology was only briefly tested during this thesis, it needs to be acknowledged as a key component for collaborative sensing in the future. 5G is the telecommunication standard expected to enter the consumer markets in the year 2020. The main goal for 5G technology is simple: improve the current 4G telecommunication methods to meet the increasing requirements for network capacity. The increasing requirements also apply for collaborative sensing. As vehicles become smarter and start to exchange information, the need for low-latency, high throughput network is considerable. Data sent to vehicles from other vehicles or infrastructure is extremely time-critical. In addition, the number of vehicles exchanging the data in a small area can be very high. 5G technology can possibly answer the needs of the collaborative sensing.[31]

Even though the goal for the 5G technology is simple, there are many different aspects on tackling the challenge. One of the key technologies studied at VTT is mobile edge computing (MEC). MEC is a technology where transferring and processing the sent data is moved closer to the radio access nodes to which all the clients of the network connect. The technology reduces the latencies, takes some of the load off the core network and makes a more spatially confined communication possible. These advantages can all help the development of collaborative sensing in automotive applications. For example, typical data sent to a collaborating vehicle is only useful in a small area nearby the vehicle. There is no use then to send the data to the core network if only the nearby vehicles can utilize it. The latency requirements are also easier to handle with a more compact network. MEC technology will be tested further at VTT in the future.[31]

5.3 DDS

DDS is a standard created by Object Management Group (OMG) for distributed systems. It enables reliable, high-performance machine-to-machine communication. There are many implementations of DDS and the one chosen for VTT's automated vehicles is an open-source implementation OpenDDS managed by Object Computing Inc (OCI). OpenDDS is highly configurable and allows fast and efficient communication. The messaging is controlled by assigning a topic for every sent message. A message sent by a publisher with a certain topic can only be listened by a subscriber listening to that topic. The messages are pre-defined with Interface Description Language (IDL). The IDL supports many different data types such as integers, floating point numbers and character arrays. It also enables easy serialization of data arrays and thus allows sending large amounts of data efficiently.[32]

OpenDDS was also an interesting implementation because of its own server structure called Data Centric Publish-Subscribe Information Repository (DCPSInfoRepo) that handle client discovery. The DCPSInfoRepo works as a server connecting publishers and subscribers but it doesn't deal with the data transportation. As all data is transferred point-to-point it is an ideal protocol to test the performance of communication networks since the server itself won't affect the data transfer speeds. The discovery architecture is shown in figure 14.[32]

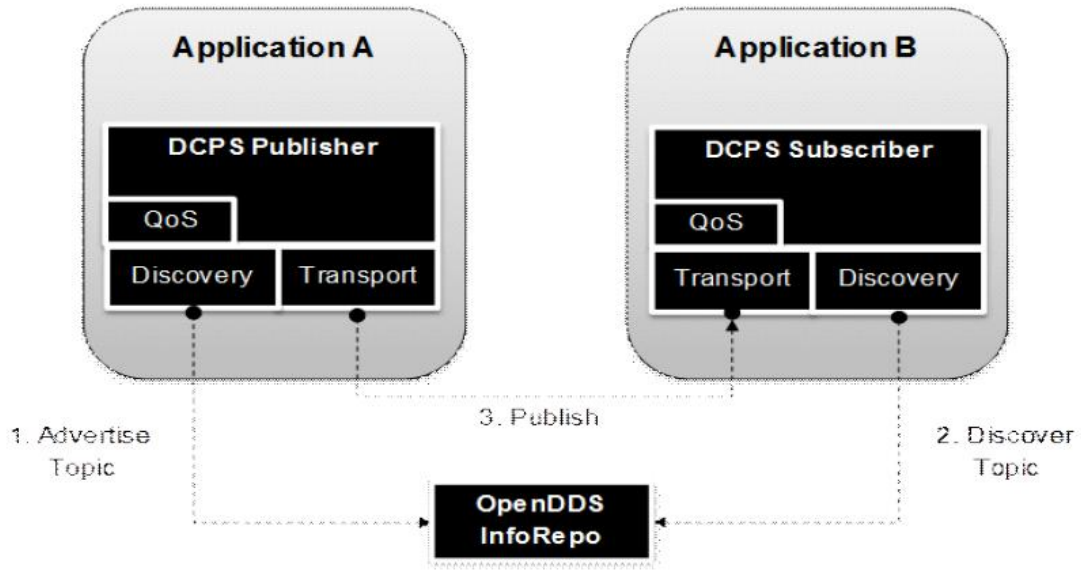

Figure 14. *DDS network structure when using OpenDDS information repository.*[32]

The communication architecture could be simply implemented by creating a public DCPSInfoRepo server available from anywhere. The vehicles could connect through the server with other clients of the server such as cars and traffic information service providers. This design raised questions about the security of the data because then clients would be listening to data coming from all other clients, even malicious ones. Since the

data sent and received through the server connections is pre-defined, this was not seen as a problem at least in the testing phase.

When OpenDDS was more deeply studied, it came clear that it is designed to be used in local networks. All examples for using OpenDDS cover only cases where all network components are inside the same network. The developer's guides for OpenDDS do not include the necessary configuration information needed for operating over the internet. By consulting OCI we were able to configure the DCPSInfoRepo so that the publishers and subscribers could connect to it. The server, publishers and subscribers indicate their IP addresses by writing them to a file and sending them when first connections are made. By default, all of the components in the OpenDDS network write their local addresses to the file and thus become unreachable for other components outside their network. The DCPSInfoRepo could be configured to force its IP address as a public IP address in the file but the same thing was not achieved with the publishers and subscribers. Time limits of this thesis forced to drop OpenDDS as an option for vehicle-to-vehicle communication but it would be a good subject to study later on.

5.4 MQTT

MQTT has a similar publisher-subscriber-method as DDS. Each message sent by a publisher contains a topic and only subscribers listening to that topic can read the messages. The main difference between MQTT and OpenDDS is that all messages go through a message broker in MQTT whereas the OpenDDS uses point-to-point messaging. The message structure is also different since MQTT only supports sending of text messages instead of pre-defined data structures. This allows more flexible communication but it also makes the communication more inefficient because all numeric data has to be converted into characters by publishers and back to numeric data by subscribers. The message payloads are more inefficient since encoded text takes up more space than raw numeric data. Conversions from numeric data to text and vice versa also require additional processing that is not necessary when using OpenDDS. Another practical difference between MQTT and OpenDDS is the topic structure. In OpenDDS all topics are unique and they contain no structures. An MQTT topic on the other hand has a hierarchy. Publishers can send messages and subscribers can listen to topics on different hierarchy levels indicated with a backslash. For example, a temperature reading of a certain room in a building could be published with a topic:

```
Building1/2ndFloor/Room221/Temperature
```

Subscribers could either listen to that specific topic or some broader topics such as all data from Building1 or all temperature measurements from the second floor of any building. The data management becomes easier with advanced topic hierarchies especially when there are more components in the network.

The chosen implementation of MQTT in this thesis was Mosquitto. It is an open-source message broker. C++ based client libraries have also been added to Mosquitto and they are also used to create the MQTT publishers and subscribers. There are many different implementations of MQTT but almost all of them are commercial products or provide only public brokers that allow limited testing of the MQTT protocol. Mosquitto as an open-source project is better suited for the automated vehicle development at VTT especially in early testing phase since it doesn't require any economical commitments. Downside of Mosquitto is its performance. It is more focused on providing easy usage than high data throughput. A benchmark test[33] of different MQTT brokers by a commercial MQTT implementation provider indicated that latencies increase in linear fashion with the amount of sent data for Mosquitto when commercial products can deal with almost constant smaller latencies. The suitability of Mosquitto for vehicle-to-vehicle communication in automated driving is covered in more detail in chapter 6.

5.5 ITS-G5

The ITS-G5 is based on ETSI standard EN 302 663[34] for vehicle-to-vehicle communication in 5.9 GHz frequency band allocated in Europe. The EN 302 663 standard is based on the 802.11p wireless communication standard. The ITS-G5 is the main technology in Intelligent Transport Systems (ITS) for vehicle-to-vehicle communications in time-critical applications. Despite the low latencies, the performance of the ITS-G5 can be limited in urban environments if there is no line of sight between the message sender and receiver. This needs to be considered when implementing critical applications in automated vehicles.[35]

By its definition the ITS-G5 is an ideal protocol for sending data between vehicles. Even though it is designed for the task, it can fail in many scenarios and thus it is important to validate also other communication methods. For comparison purposes, the ITS-G5 communication performance is tested in this thesis.

6. COLLISION WARNING SOFTWARE

The collision warning system introduces the element of collaborative sensing to VTT's automated vehicles. The software listens to the object tracking data produced by the ego-vehicle as well as other vehicles that are connected to the same communication platform. The risk of collision is analyzed by projecting the ego-vehicle's and tracked objects' trajectories and calculating if they intersect in the near future.

6.1 Collaborative sensing

The final version of the collaborative sensing in this thesis was implemented with a MQTT Mosquitto broker. Each vehicle connected to the Mosquitto broker can share the objects it has tracked and receive data from other vehicles that have their own object tracking systems. The test environment includes a Mosquitto broker that is hosted on VTT's server.

Vehicles that are connected to the Mosquitto broker need to have a common coordinate system in order to interpret each other's object tracking information. The first step for sharing the tracked object information is to perform coordinate system transformations from the ego-vehicle's coordinate system to global coordinate system. The transformations include transformations of the object's locations from the Cartesian coordinate system of the vehicle to World Geodetic System (WGS). WGS84 was chosen for this implementation as the global coordinate system because of its commonness and accuracy in all parts of the world. The second required transformation is the rotation of objects' headings from the ego-vehicle's coordinate system to universal East North Up (ENU) heading. ENU based heading was chosen because the rest of the vehicle's software also utilizes the same system.[36]

Calculating the global coordinates of each object is performed with another transformation to Universal Transverse Mercator (UTM) projection. In the UTM projection, Earth is divided into 60 zones. Locations are expressed as a combination of zone numbers and the accurate location in the zone described by easting and northing values. Each zone has a center meridian from which the easting is calculated. The northing is calculated from the equator. The advantage of the UTM projection is that it presents the position in meters rather than in angles. This allows fairly simple calculation of the objects' global positions with a single rotation of original x and y coordinates and addition of the rotated coordinates to the ego-vehicles UTM coordinates. The UTM coordinate system is not used for transmitting the objects' locations because UTM coordinates are more complex to present than WGS84 coordinates and they contain more error especially on the edges of the zones. WGS84 coordinates contain only two values,

latitude and longitude, which are expressed in degrees. UTM coordinates on the other hand require two distance values, northing and easting, as well as a zone identifier number. The accuracy of the UTM projection probably wouldn't have much effect since the use of the object tracking data is restricted to a relatively small area around the sensing vehicle. The message efficiency on the other hand is critical when limited performance communication system such as Mosquitto is used.[36]

6.2 Collision estimation

Collision estimation is based on projecting the path of each perceived dynamic object and the ego-vehicle and then examining if any of the objects paths intersect with the ego-vehicle's path. The input for the system comes from the vehicle's own object tracking as a DDS message as well as from external sources as MQTT messages. The objects' locations and headings are first converted to the vehicle's coordinate system. Then each object's trajectory is projected based on the location, heading and curvature of the object. The projection has to be processed by integrating the curvature effect on the path. The integration time used is 0.2 seconds and each object is tracked for 4 seconds into the future. The collision estimation must also take into account the possible delays between measuring the object and projecting its track. To synchronize the measurements between the different software components and measurements from external sources, a common, globally available GNSS based time stamp is used. In addition to the 4 second projection to the future, the trajectories between the measurements time and the processing time need to be calculated.

Checking of the possible collision is made by comparing the border lines of the object and the vehicle on each given time stamp from current time to 4 seconds ahead. Width and length is determined for each object. All objects are marked as rectangular boxes and the border lines are defined as linear models. Similar models were used in software described in chapter 4.2.1. If any of the ego-vehicle's border lines intersect with any border lines of an object, a collision warning is created. The software informs the estimated time stamp of the collision as well as the direction of the colliding vehicle.

A test software was developed to assist in the collision warning system development. The user interface of the software is shown in figure 15.

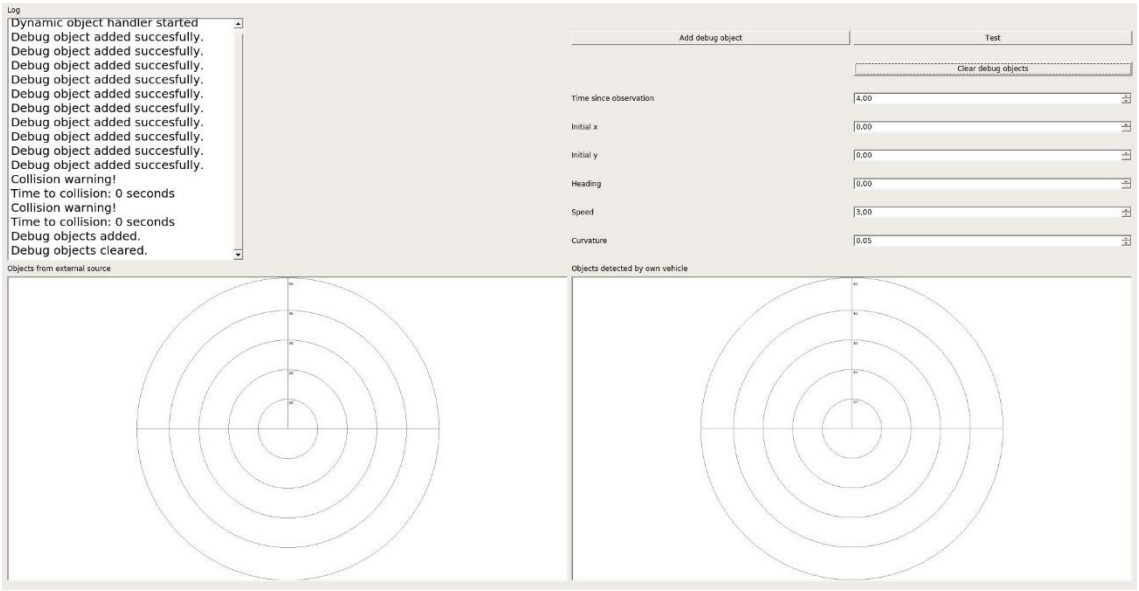

Figure 15. *Collision warning software user interface.*

The interface displays the object information received from other vehicles and the ego-vehicle on separate windows. It can also be used to create simulated objects with a simple tool. The tool was used to verify the accuracy of the collision warnings. An example of two simulated moving objects is shown in figure 16.

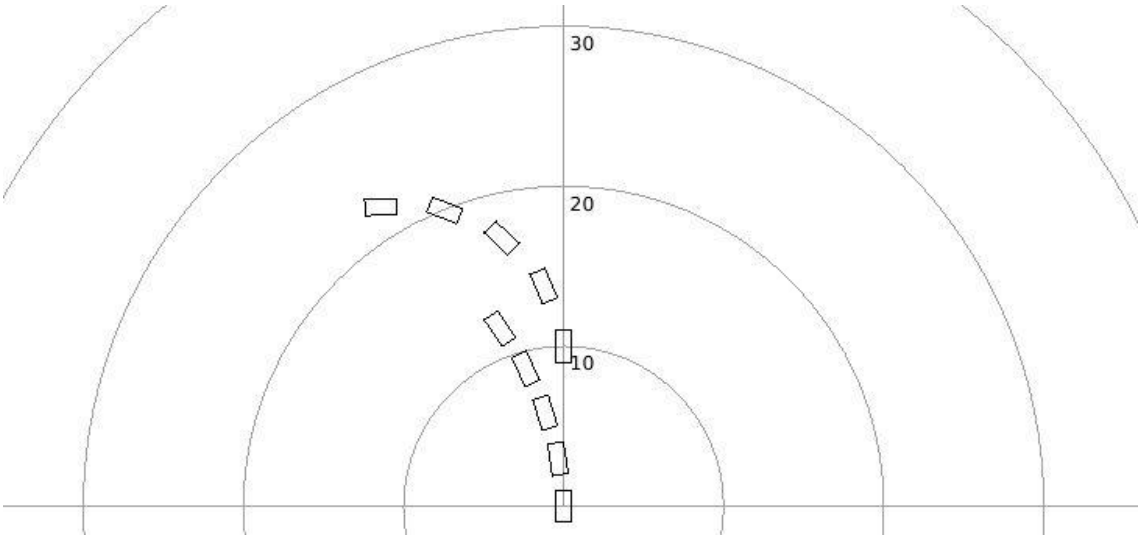

Figure 16. *Example of trajectory projection.*

The track of two simulated vehicles is projected for four seconds in the figure. The software takes into account the location, speed, heading, curvature and size of the objects. Producing all of the information for the collision warning software is a great challenge. Especially the curvature is hard to define accurately.

The collision estimation could be improved in many ways. First improvement would be a more accurate collision detection. If a smaller object collides with a larger object, it is possible that the border lines do not intersect at all but the all of the smaller object's border

lines are inside the border lines of the larger object. Second improvement could be an estimation of the projection validity. Any changes in the heading, speed or curvature of the object lead to errors in the projection. The further the trajectory is projected, the higher the chance is for these changes. Any errors in the initial state of the object also cumulate over time and the projection becomes less accurate after each integration round. The current performance of the collision estimation is presented in chapter 7.3.

6.3 Collision warnings

The collision warning itself can be used for many purposes. Even non-automated vehicles can utilize the feature by creating a warning for the driver and creating a more alert state of mind. For automated vehicles, the collision warnings can be used by reacting to the threat of collision by adjusting the vehicle controls. By knowing the estimated collision time and place and the movement of the possibly colliding object, it is possible to redesign the route for near future thus lowering the risk of the collision or making it less dangerous. The DDS network enables the easy use of this information in later development phases of the automated vehicles. The collision warnings can be produced as DDS messages and all software modules can utilize it in a suitable way.

7. IMPLEMENTATION PERFORMANCE

This chapter describes how the software and hardware components developed and used in this thesis were tested and how they performed. Each module of the software is covered individually and the hardware components associated with the software modules are included in the corresponding module review. The three different software modules of this chapter are the Object tracking and recognition module, V2V communication module and Collision warning module. All software tests were performed on the Compulab IPC2 computers on the automated vehicles Marilyn and Martti.

7.1 Object tracking and recognition module

7.1.1 Sorting

The first sorting method used a single insertion sort that read measurement points from multiple LiDARs and combined them online. This method took on average tens of milliseconds and could not keep up with the LiDAR measurements in some scenarios where the LiDARs produced more than average number of measurement points. This implementation could not be used because of the poor performance.

The first improved sorting method used a combination of merge sort and insertion sort. Merge sort was used to combine the measurement points of a single LiDAR and insertion sort was used to combine the sorted measurements of multiple LiDARs. This sorting method took on average 13 milliseconds. This was already acceptable performance for the sorting but a final improvement was tested to get near-optimal performance. The final improvement utilized merge sort algorithms both in sorting the points of a single LiDAR and combining the points from multiple LiDARs. The average processing time of this implementation was roughly 1 millisecond and was ultimately used in the software.

7.1.2 Clustering

The performance of the clustering proved to be excellent in ideal situations but very challenging in actual driving scenarios. The accuracy of the LiDARs allow a reliable separation of small objects even within a meter apart from each other. The challenges of the clustering come from drippy points and ground hits. Drippy points are one problem addressed also by MacLachlan in his article[28]. Measurement points seem to appear between objects that are next to each other but are slightly in different distances from the sensor. This can cause the objects to merge and produce invalid sense of movement.

While the dripping is a significant challenge, the main challenge of clustering is the filtering of ground hits. On an even ground, it is easy to determine which of the measurement points are from ground since the height of the measured point can be determined in relation to the LiDAR. Uneven surfaces on the other hand could produce different size clusters that appear to be even a meter above the ground level. When the ego-vehicle moves, these clusters can appear to be moving. These clusters have to be filtered out with some sort of sensor fusion because they do not differ from other types of objects. Any static map can't be used as a solution because many objects of this type are created by changing environmental factors such as piles of snow. Possible solution could be an integration with a stereo camera system. Each tracked object could be fed to the stereo cameras to be identified. This on the other increases the vital latency of creating an observation of an object. The stereo camera system could also be used to sense gradient changes in the area of the perceived obstacle. Another possible other solution could be to use LiDARs with higher number of layers. This would enable gradient-based estimation of which measurement points are from ground.

7.1.3 Linearization

The two linearization methods were tested by measuring the average processing times with different sensor and algorithm configurations and evaluating how their outputs corresponded with the original measurements points. The performance of the combination of RANSAC and linear regression produced good output results but even with fewer iterations the algorithm was far too slow and it could not process the data from multiple sensors. With a single 4-layer LiDAR a single linearization round of all clusters took between 50 and 150 milliseconds. Input from two 8-layer LiDARs were so intense on the algorithm that it could not perform the linearization between measurements. It was thus necessary to create another method for the linearization.

The Douglas-Peucker algorithm was a good solution for the linearization. It yields good output results and the processing time with two 8-layer LiDAR takes 4 milliseconds on average making it a valid choice for the linearization. Output of the linearization is shown in figure 17.

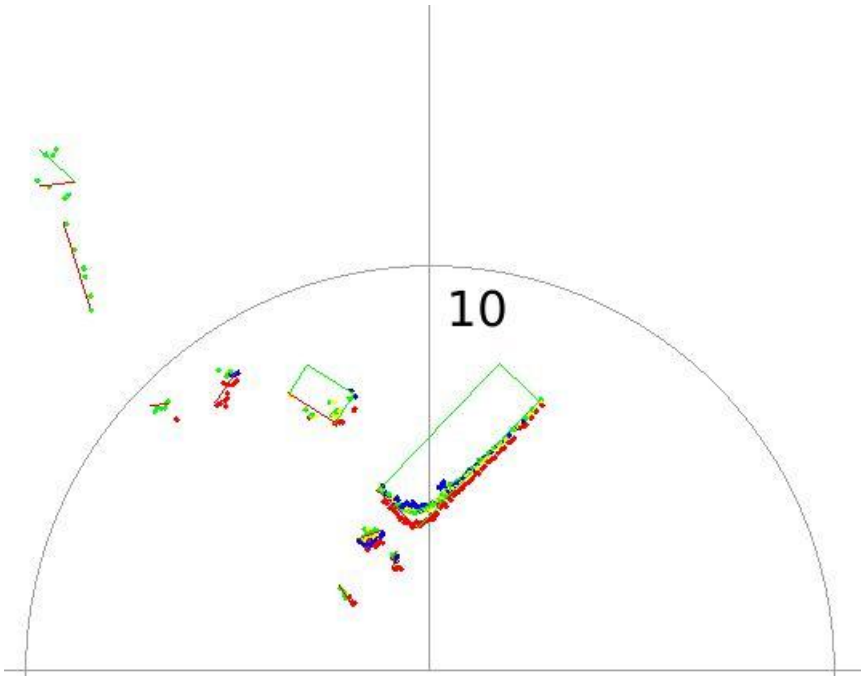

Figure 17. *Visualization of the measurement point linearization.*

LiDAR's measurements are shown as dots and the linearized models with lines between their corner points. Clusters that contain three corners and form a convex corner are processed further by projecting a fourth corner point. This is done to balance the center point of the vehicle and make the visualization of vehicles clearer.

The Douglas-Peucker algorithm has also a downside. It requires the points to be sorted by their coordinates. Sorting the points requires more processing time even though the individual LiDAR devices produce measurements that are in order. The two devices of a single 8-layered Sick LD-MRS LiDAR provide measurements that fully overlap. Another overlap happens at the center line of the vehicle's heading where the fields of view of the two LiDARs overlap. These overlaps increase the randomness of the order by coordinates and that leads to more processing time. The challenge was overcome with the two-phased merge sort algorithm and the final implementation was performing well.

7.1.4 Object tracking

The object tracking works successfully on clearly defined objects. Typically small objects such as pedestrians have stable positions in the LiDAR's field of view after the linearization because their shape doesn't change much even if they are perceived from different angles. Larger objects such as vehicles and static obstacles on the other hand created problems. The shape change caused by perceiving an object from a different angle can cause a sense of movement. Even though this effect can be diminished by increasing the weight of the predicted location in the Kalman filter, some level of movement still persists. This makes it especially hard for determining whether an object is static or not. This information is crucial for object recognition because the movements of an object

define a large portion of the object's characteristics alongside its size and shape. The static flag would also serve as a good filter for sending the object's information to the Collision warning module.

The problem of defining whether an object is static or not was addressed by adding a filtering counter. If the velocity of the object exceeded a threshold, it would increase a counter. If the counter increased on multiple consecutive measurements, the object was determined to be non-static. This served as a partial solution but did not solve the final problem. In cases where two objects merge, the merging object will receive the static flag status of the object it is merging to. A fully working solution was not found for this challenge.

7.1.5 Object recognition

The object recognition module was not successful at recognizing objects at real traffic scenarios. The greatest problem for the module was the synchronizing of the two devices of a single LiDAR. Because the measurements from the two devices come in turns, the combination of the two measurements creates a ghost image whenever the objects in the LiDAR's field of view have relative movement to the LiDAR. The synchronization problem is especially challenging with vehicles that are driving towards the ego-vehicle because their relative movement is the added speed of their velocity and the ego-vehicle's velocity. Figure 18 shows the effects of the synchronization on a perceived vehicle that is moving towards the moving ego-vehicle.

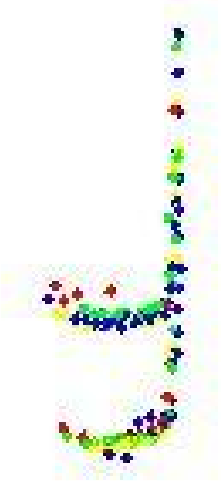

Figure 18. *Ghost image created by synchronization.*

The problem with synchronization could be compensated by three different methods. The first solution would be to increase the sampling rate of the LiDAR. This would make the time between the measurements of the two devices shorter and the errors would be smaller. The sampling rate could be increased to 50 Hz from the current 12.5 Hz so the

error would be reduced to one quarter of the original. Increasing the sampling rate would decrease the LiDAR's resolution and the same problem would persist on greater velocities so it would not be an ideal solution. In addition, other system modules using LiDARs would be affected by the lowered resolution and that could lead to more problems elsewhere in the vehicle.

The second option would be to compensate the ego-vehicle's movement for the first device measurement. This could be performed with good results but it would not solve the problem of the moving objects in the LiDAR's field of view since their movement could not be compensated.

Third option would be to use individual tracking software for each device. A vehicle using three 8-layered LiDARs would then run six individual tracking modules. This method would make the sensor fusion much harder to implement. Using the combined devices creates the advantage of having a more comprehensive view of the surrounding objects. Even if an object is on the edge of the field of view of a single LiDAR, it can be perceived fully if the adjacent LiDAR can also see it. This makes the object recognition much easier because then there are more information about the objects on the edges of the field of view. The challenging implementation and the reduced performance make this method also a poor choice but it would be the only option that would mitigate all errors created by the synchronization.

If the synchronization problem would be solved, the object recognition module could provide some results. For example, pedestrian recognition can be achieved with some accuracy if the ego-vehicle is stationary. The voting algorithm creates some false positive recognitions for pedestrians if small stationary objects are seen but a combination of thermal camera image and LiDAR could provide recognition with reasonable accuracy. In conclusion, the LiDARs that produce point clouds that are close to 2 dimensional, such as Sick LD-MRS, are not the ideal sensors for object recognition. When even a human eye has trouble recognizing the objects from the LiDARs point cloud, it is very difficult to teach a machine to produce accurate recognition.

7.2 V2X communication

7.2.1 MQTT Mosquitto

The MQTT Mosquitto implementation was tested with different kinds of network configurations to measure the performance of each component in the MQTT setup. The goal of the measurements was to find the limitations the MQTT implementation poses and define how well it scales for larger scenarios.

The performance tests focused on testing the Mosquitto broker. The performance was tested by sending messages containing the same amount of information as would be

needed to send the data of one dynamic object. The data was ASCII coded and contained the information described in table 1.

Table 1. Number of characters in an ASCII coded dynamic object data.

Information	**Typical number of characters**
Latitude	9 – 10
Longitude	9 – 10
Object width	3 – 4
Object length	3 – 4
Heading	5
Curvature	5 – 6
Speed	4 – 5
ID	1 – 8

When the values of the ASCII coded object information are separated with commas and different objects are separated with new lines, the number of characters needed to transfer data of a single dynamic objects is approximately 50 to 60 characters. The number of object lines in a single message was also varied to test how the broker throughput is affected by the size of individual messages.

The broker was tested with a varying number of publishers ranging from 1 to 200. The publishers were initialized in a single QT terminal software. The test were conducted in VTT's local network so that the network connection would not be a limiting factor in the tests. Mean, minimum and maximum transmission times were measured for each configuration. Each publisher was made to send a message every 20 milliseconds which is close to the object tracking software's output frequency.

The performance of the Mosquitto broker was very limited in the scope of exchanging object data in large-scale scenarios. Even in local network the broker could not provide reliable communication if 50 publishers and subscribers were connected to it. The latencies started to grow uncontrollably when a critical amount of data was sent through the broker. Figure 19 shows the time it takes for 10 objects' information to be sent in a network configuration of 100 publishers and 1 subscriber.

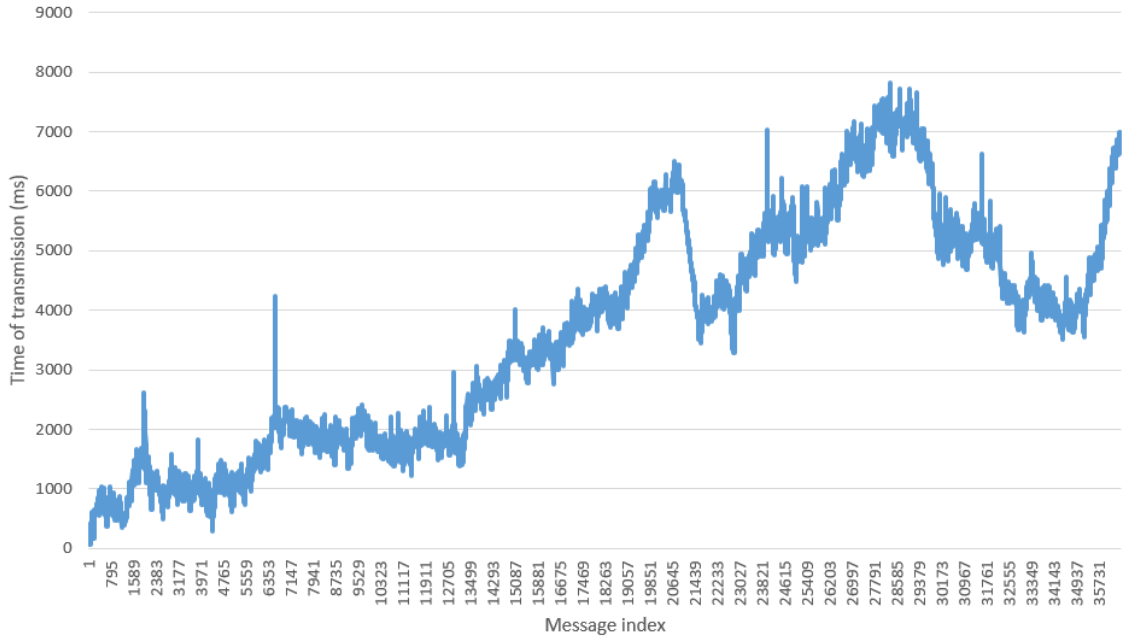

Figure 19. *Transmission time of sending ten objects' information from 100 publishers to one subscriber.*

Latencies in this scenario are too long for the broker to be useful in real-time applications of this scale. The latencies for different configurations are shown in tables 2, 3 and 4.

Table 2. Mosquitto broker performance with 1 subscriber and 1 sent object in every transmission.

Publishers	**Mean transmission time (ms)**	**Minimum transmission time (ms)**	**Maximum transmission time (ms)**
1	21,16	13	215
10	63,22	13	1692
100	413,01	19	2773
200	383,47	33	2367

Table 3. Mosquitto broker performance with 1 subscriber and 10 sent object in every transmission.

Publishers	**Mean transmission time (ms)**	**Minimum transmission time (ms)**	**Maximum transmission time (ms)**
1	22,46	13	157
10	37,78	13	1106
20	152,46	30	3599
40	383,47	33	2367
70	2884,34	18	151576
100	3563,89	61	7815

Table 4. Mosquitto broker performance with equal amount of subscribers and publishers sending and receiving 1 object in every transmission.

Publishers and subscribers	**Mean transmission time (ms)**	**Minimum transmission time (ms)**	**Maximum transmission time (ms)**
10	160,11	14	3217
20	222,49	186	1749
50	25451,00	197	51916

The measurement result tables show that the Mosquitto broker can manage some smaller scenarios and it can well be used in testing the V2X-communication with a small number of clients connected to it. Larger number of clients start to overload the broker and increase latencies. The scenario of 10 publishers sending data to one subscriber is presented as a graph in figure 20.

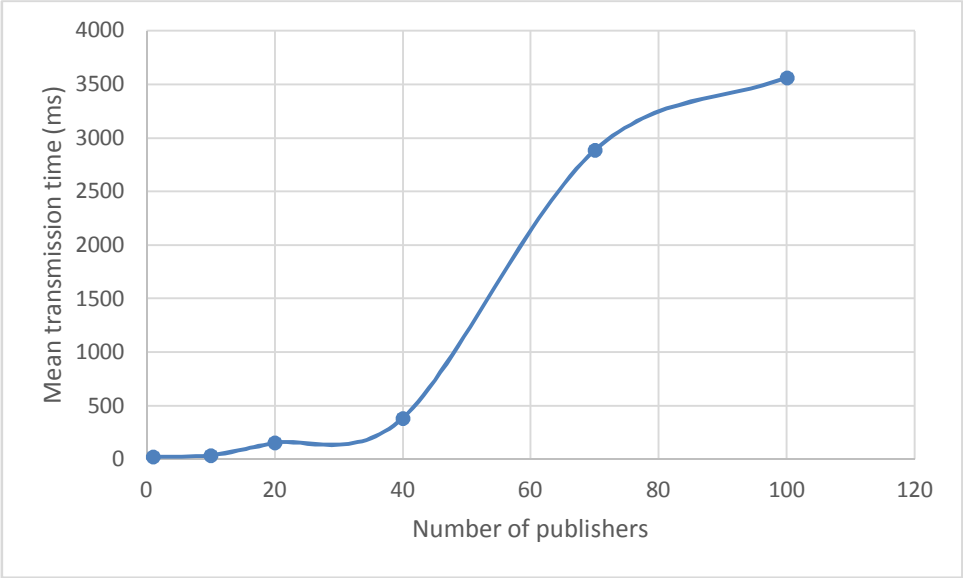

Figure 20. *Average transmission time relation to number of publishers.*

The figure shows that increasing the number of publishers even up to 70 leads to latencies of several seconds. If the number of subscribers is increased, even 50 publishers and subscribers is too much for the broker. Optimization of the sent data, longer intervals between observations and hardware improvements on the broker PC would possibly allow larger scale tests but for commercial products, a higher performance communication method would be necessary.

In addition to the controlled tests in VTT's local network, the Mosquitto was tested on multiple V2X demonstrations. The Mosquitto configuration was not working robustly enough in the demonstrations. It occasionally produced tenfold latencies for individual topics and typically disconnected the publishers of that topic. A reconnection to the broker took tens of seconds after the forced disconnection by the Mosquitto broker. The demonstrations contained only a few publishers and a single subscriber for each topic. Increasing the throughput resulted in more frequent disconnection. This would implicate that the larger data throughput also affected the Mosquitto broker by making its connections more unstable.

Other MQTT implementations were inspected for further testing of the V2V communication. Open-source projects such as VerneMQ and EMQ were found to be promising solutions because of their scalability.

7.2.2 ITS-G5

The ITS-G5 modules were not the core components of this thesis but their performance was tested for a simple comparison with the MQTT implementation. The performance of the vehicles' ITS-G5 modules were tested by sending ad hoc messages ITS-G5 messages that contained almost identical information as the test messages for the MQTT

implementation. The transmission frequency could controllably be set to 1 kHz. Each message contained information about a single perceived object. The transmission latencies were measured by taking the time stamp information from the GNSS modules and comparing them at the receiving end. The GNSS time stamps were received only every 0.5 seconds so the time stamps in between the GNSS based time stamps were created using Qt's QElapsedTimer class. The timer increased the time stamp accuracy in theory to 1 millisecond. The latencies were monitored for 60 seconds and the results are shown in figure 21.

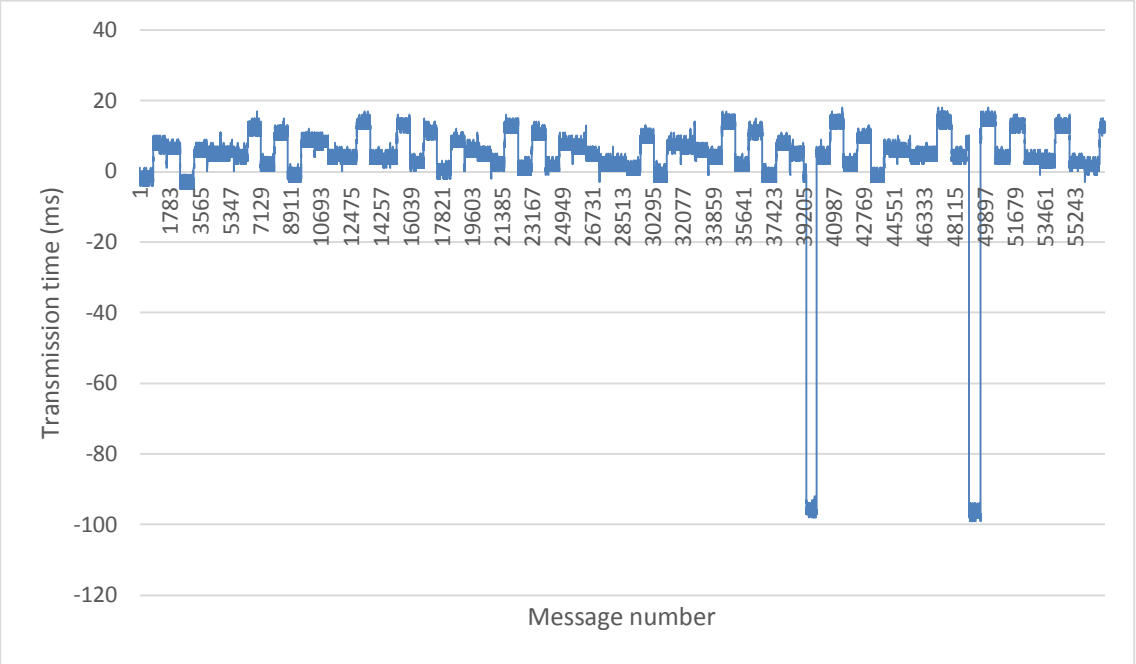

Figure 21. *Transmission time of a single object's information at 1 kHz transmission rate using ITS-G5.*

The latencies were extremely small and the time stamp inaccuracy actually became a problem for the measurements. Many measurements actually showed negative transmission time. This is probably caused by latencies in the receiving vehicle's internal network. At maximum controlled transmission rate, the latencies of the transmission were extremely short. ITS-G5 could be a valid option for collaborative sensing if the packet losses due to obstructed line of sight were acceptable.

7.3 Collision warning module

7.3.1 Positioning accuracy

Positioning is a critical part for the software because the movements of the ego-vehicle affect the analysis of dynamic objects from the ego-vehicle as well as from external sources. The critical components of the position information are the heading, speed and coordinates. Coordinates affect only the processing of dynamic objects from external

sources because all data processing inside the ego-vehicle is made in the vehicle's own coordinate system. Heading and speed on the other hand affect the ego-vehicles own measurements.

Creating an accurate heading was the greatest challenge of positioning. Even an error of a few degrees can lead to errors of several meters if the perceived object is further away. For example, an error of 1° in heading leads to an error of 0.87 meters when the object is 50 meters away. An error of 5° creates an error of 4.36 meters for objects 50 meters away. It also affects the projected trajectories of the dynamic objects.

The IMU units on both vehicles had many performance issues. The raw heading value contained a non-constant offset and turning the vehicle caused non-linear errors on the heading based on which direction the vehicle turned. The navigation sensor fusion module was used to analyze the performance of the IMU. The module reads GNSS measurements and projects the path of the vehicle between them using the IMU and the odometry data from the vehicle's CAN bus. The measurements showed a steady 7° offset when turning right but when the vehicle was turned right the offset varied between 11° and 21°. Calibrating the IMU seemed to reduce the non-linear offset but its effects would not last long. This level of offset is too large for the object tracking software since it can lead to errors of over 10 meters in the perception of objects and also create false approximations of movements of the objects when turning. A much more accurate sensor would be necessary for the needs of the object tracking software.

7.3.2 Warning accuracy

First warning accuracy tests revealed many challenges. The first tests were conducted outdoors when it was snowing and the weather created major challenges to the software. Even a light snow created a lot of noise near the sensors and created constant false positive warnings. The performance harsh weather conditions is covered in chapter 7.4. Other false positives were also a challenge. Vaguely perceived objects created the sense of movement to the tracking software and that lead to false alarms.

The solution for these challenges was intelligent filtering of the objects. Many demands were made for the objects to be accepted to the collision warning module. Size, movement, weather conditions and time of tracking were checked before the object data was sent to the collision warning module to be analyzed. Even with the filtering the tracking was not performing well enough for the collision warning module to produce accurate warnings.

7.4 Harsh weather conditions

Rain, snow, fog and dirt affect the performances of the LiDARs used in this thesis. All these weather conditions create noise in the measurements that needs to be considered

when automated driving related software is developed. During the thesis, it was possible to measure the effects of rain and snow on the LiDARs in real harsh weather conditions. Based on these measurements, a weather monitoring system was implemented on the vehicles with an attempt to filter out noise created by harsh weather conditions.

The effects of harsh weather conditions seemed to be restricted to within 10 meters of the LiDARs. Depending on the harshness, a varying amount of noise observations occurred near the sensors. The heavy duty version of the Sick's LiDAR showed much less noise even in heavy snow. Snow measurements were conducted with the robot car Martti in which the LiDARs' outputs could be simultaneously recorded. Figures 22 and 23 show point clouds created by the two sensors. The regular version of the LiDAR is facing to the left in 60° angle to the driving direction and the heavy duty version is facing straight ahead. Figure 24 shows the parking lot environment where the measurements were made. At the time of the measurements, the LiDAR angles were poorly calibrated resulting in a visualization of too small angle of the regular version of the Sick LiDAR facing left. This only affected the visualization and had no other effects on the measurements.

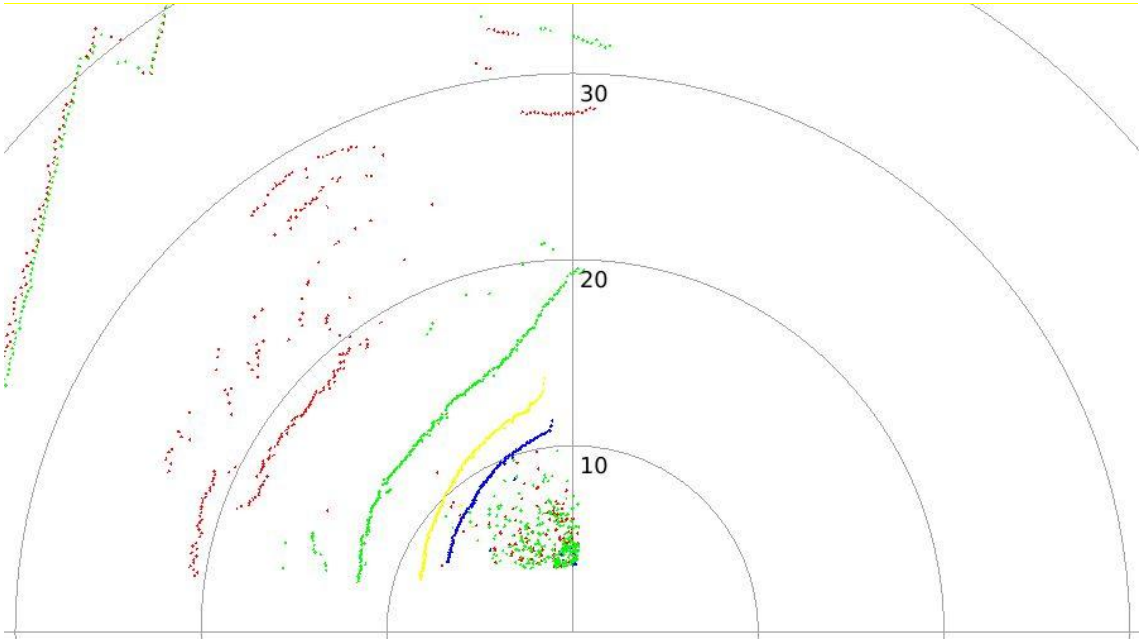

Figure 22. *Regular Sick LD-MRS measurements in snowy conditions.*

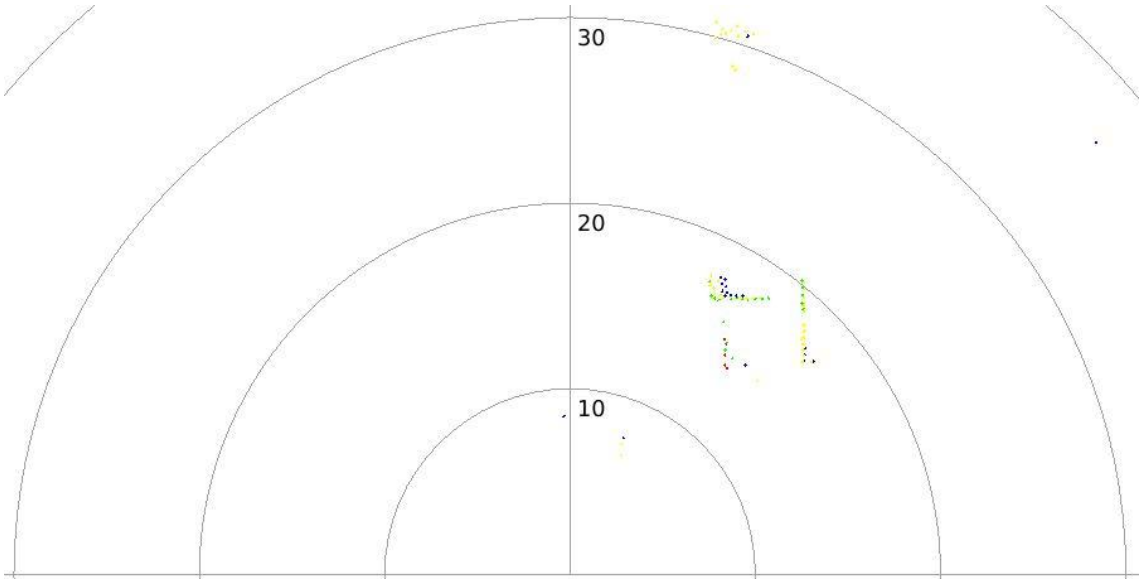

Figure 23. *Sick LD-MRS HD measurements in snowy conditions.*

Figure 24. *Measurement environment in snow tests.*

The regular Sick LiDAR is showing ground hits from roughly 8 to 20 meters which appear as uneven lines. The noise created by the snow is seen as an intense cloud of measurement points within the 10 meter circle. The heavy duty version on the other hand shows only a few noise points near the 10 meter circle. Both of the sensors are mounted on the front bumper of the vehicle half a meter from each other so they are exposed to identical

weather conditions. The amount of snow was also recorded from the Finnish Meteorological Institute websites[37] that reported 1.1 millimeter of water per hour as the amount of snow.

The next step was to analyze the characteristics of the noise created by the weather to enable noise filtering. Each weather scenario was measured for at least one minute with a regular Sick LiDAR. Results of the analysis are shown in table 5.

Table 5. LiDAR measurement attributes in different weather conditions.

	Dry weather	**Rain**	**Snow**
Mean signal strength	165,06	139,96	196,71
Mean number of echoes	1,13	1,15	1,33
Maximum pulse width	432	372	656
Relative number of measurement points within 1 meter	0	0,0018	0,0275
Relative number of measurement points within 0,5 meters	0	0,00037698	0,0069
Relative number of measurement points within 0,2 meters	0	0,00018849	0,00020539
Closest second echo (m)	4,95	2,38	0,45
Closest third echo (m)	5,1	36,24	1,82

The signal strength of the noise measurements was indistinguishable from actual measurement points. The number of echoes increased in snowy conditions but in rain, the value dropped. Maximum pulse width was also inconclusive. The only value distinguishing the weather conditions from each other was the number of measurement points near the sensors. This value on the other hand is also inconclusive because there might be an actual object near the LiDAR. This challenge can be overcome to some extent

in VTT's automated vehicles by reading all the regular LiDAR measurements and checking if there are measurement points near each sensor. This could be applied especially to Marilyn, where the LiDARs are mounted on the front as well as on the rear of the vehicle. Measurement points near one or two LiDARs could indicate a real object but points near all three sensors would mean that the performance of the sensors is limited due to weather. This approach enables auxiliary use of LiDAR's in harsh weather conditions but it cannot be used exclusively since the close proximity of the vehicle becomes invisible for the LiDARs when filtering is applied.

Since the straight-forward approach of analyzing the different average values provided merely partial solutions, more complex analyses were made. This lead to new discoveries about the Sick's LiDARs characteristics. The premise of the analysis was that noisy measurement points created by weather only appear within 10 meters of the sensor. This characteristic of the weather noise was used to isolate the noise points and analyze them further. The first test were made in an open field where only noise points appeared near the LiDARs. The tests were made when it was snowing. The signal strength in relation to the distance of the corresponding measurement points from the sensor was measured and the results are shown in figure 25. The signal strength value is the detected measurement point's echo pulse width in centimeters.

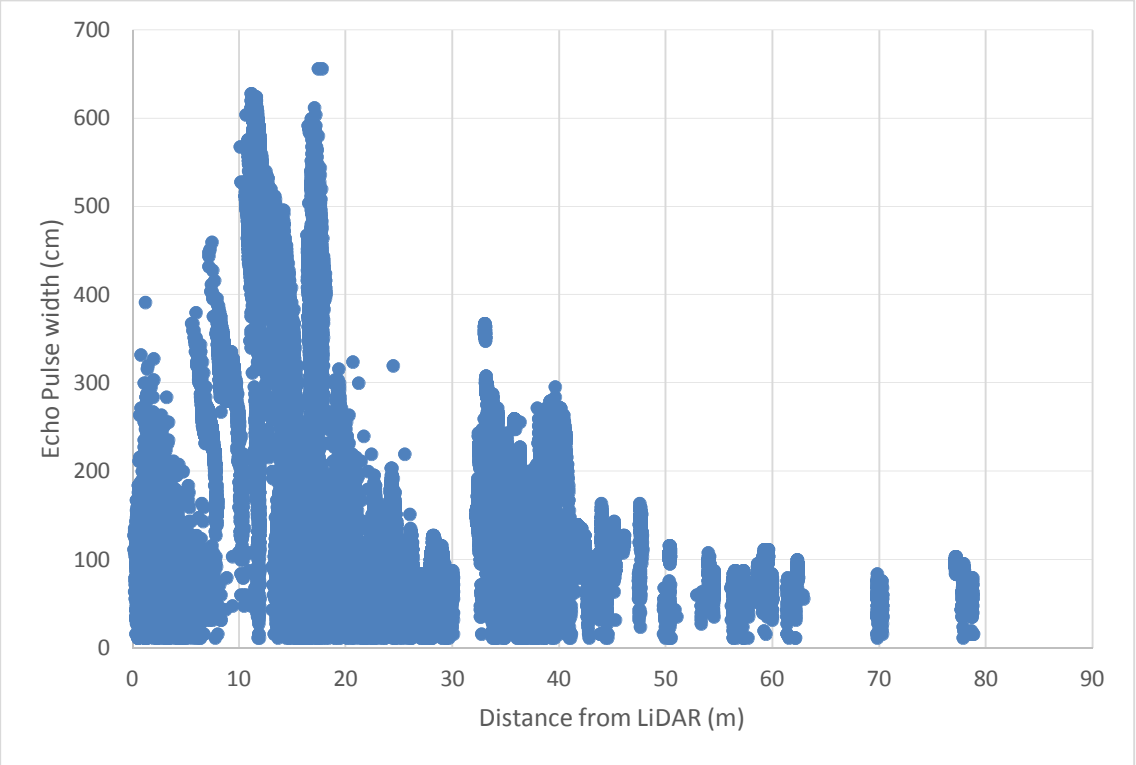

Figure 25. *Echo pulse widths of Sick LD-MRS measurement points in snowy conditions.*

The measurement points within roughly 7 meters were examined to be noise based on the LiDAR's visualization. The signal strength peaks at roughly 7-16 meters were observed to be from ground hits. This measurement showed exactly the same results as the mean

signal strength measurements. The signal strength of a noise point created by weather doesn't differ from any real measurement point based on this measurement.

Since the environment is almost fully observable even in heavy snow, it means that the noise measurement points created by weather are mainly "transparent". That means the same laser pulse that creates the noise, creates also other points behind the noise point. Both points have individual signal strengths so it was possible to measure the ratio of a noise measurement point and the actual measurement point received behind it. A point was defined to be a noise measurement point if the observation came within 10 meters and was transparent. The average ratio of 30 consecutive measurements was recorded in different weather conditions and also before a fully windowed wall. The windowed wall was tested because it also can create transparent points within 10 meters of the sensors. If no transparent measurement points were found within 10 meters, the ratio was set to 0 for a single scan. The results are shown in figure 26.

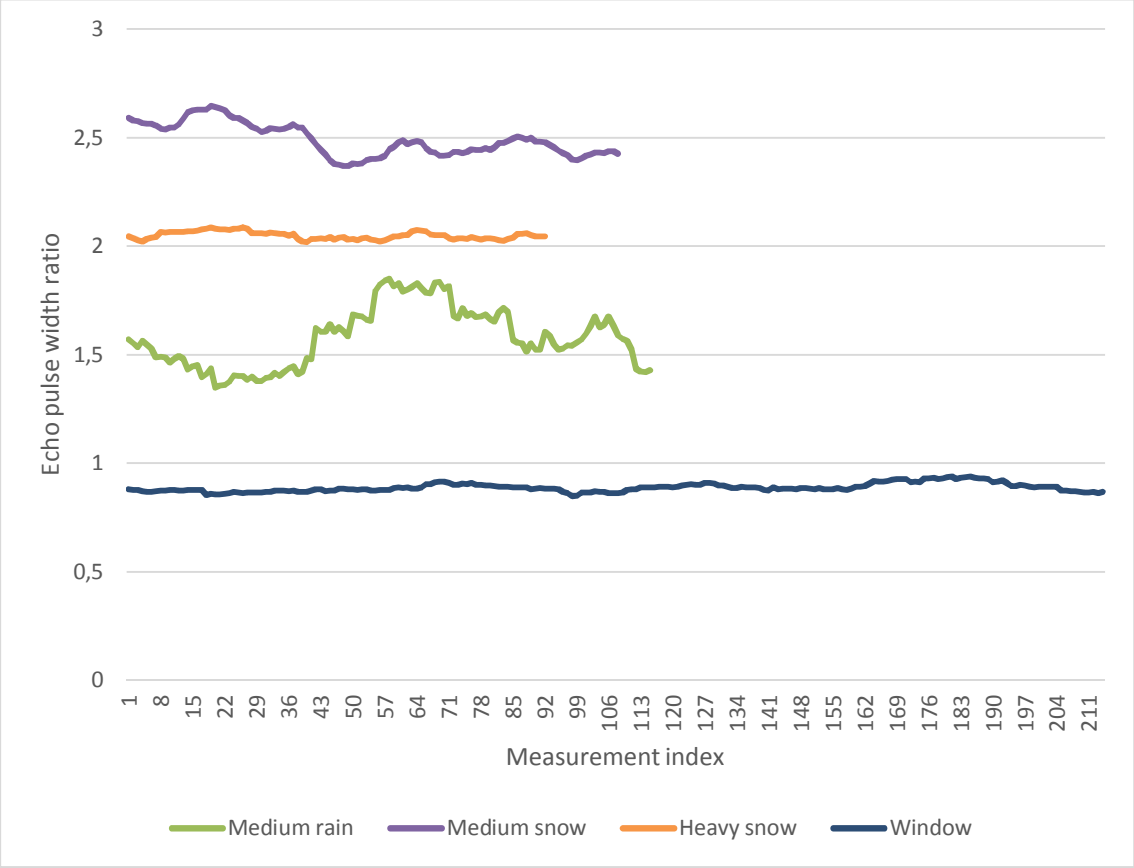

Figure 26. *Ratio between first and second measurement points' echo pulse widths in different scenarios.*

Control measurements were also made on dry weather but since there were no transparent objects, the ratio stayed at zero. These measurements showed indications that transparent points created by weather noise could be distinguished from real transparent objects such as windows.

The ratio tests were also conducted on LiDAR measurements taken in an urban environment. The temperature during the measurements stayed unfortunately close to 0 °C and there was snow on the ground so it was not possible to create a good control measurement to test false positive weather recognition. Fortunately, the measurements showed that slush splashing on the LiDARs also creates a similar effect on the pulse width ratios as snow and rain. A filtered pulse width ratio from the urban measurements is shown in figure 27.

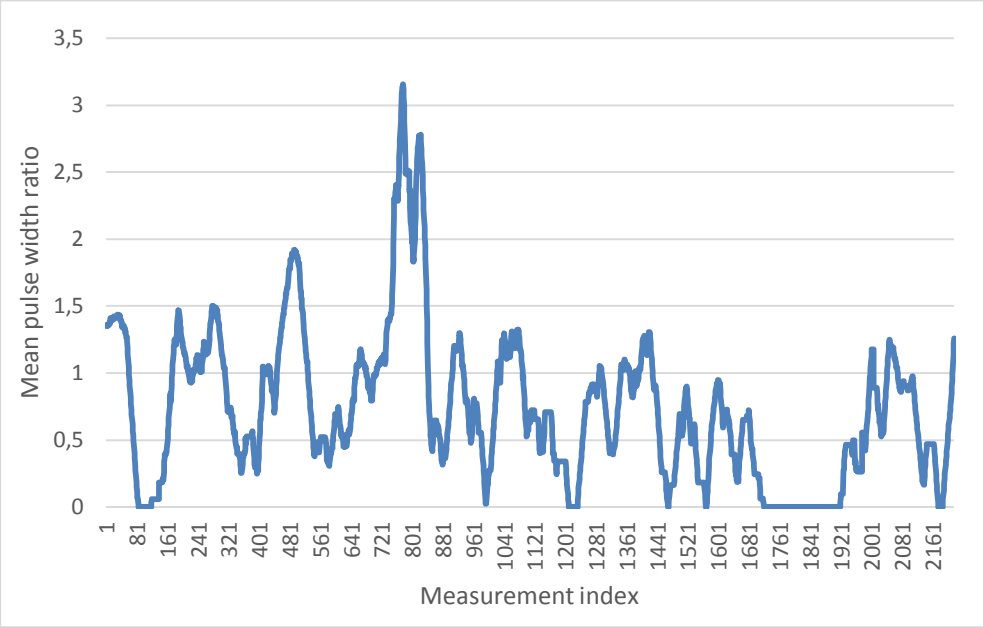

Figure 27. *Mean LiDAR pulse width ratio on an urban area drive.*

The roughly 5 minute drive in the Hervanta center consisted of multiple cases where slush got thrown in the LiDAR's field of view. Each of these cases can be seen as a spike in the signal pulse width ratio. This measurement indicates that creating an accurate evaluation about the weather solely with LiDARs requires more filtering. It also shows that the signal pulse width ratio can be used to detect other cases than rain or snow that require measurement point filtering. This is critical information especially in places like Finland, where the roads can be covered with slush for several months each year.

More measurements have to be made to create an algorithm that accurately detects the weather type with LiDARs. The first measurements showed promise in its development. The same methods will be tested with various weather scenarios and environments in the future.

The next step in this thesis was to find out whether the pulse width ratio could be used to filter out the noise created by weather. The first version of the filter removed all transparent measurement points within 10 meters of the LiDARs. The threshold distance for weather effects was later on discovered to be 6.5 meters. The first version of the filter reduced the amount of noise to roughly a third. The unfiltered test scenario where a

pedestrian is standing in snow is show in figure 28 and the output of the first version of the filter is shown on figure 29.

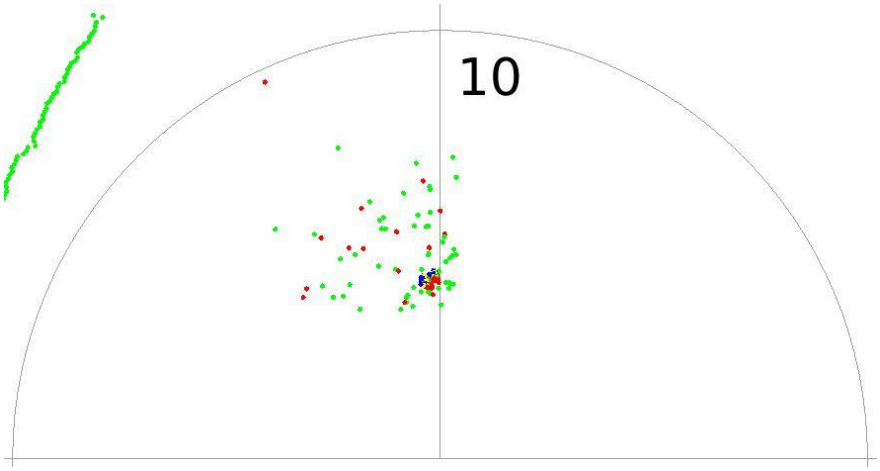

Figure 28. *LiDAR view of a pedestrian standing in snow.*

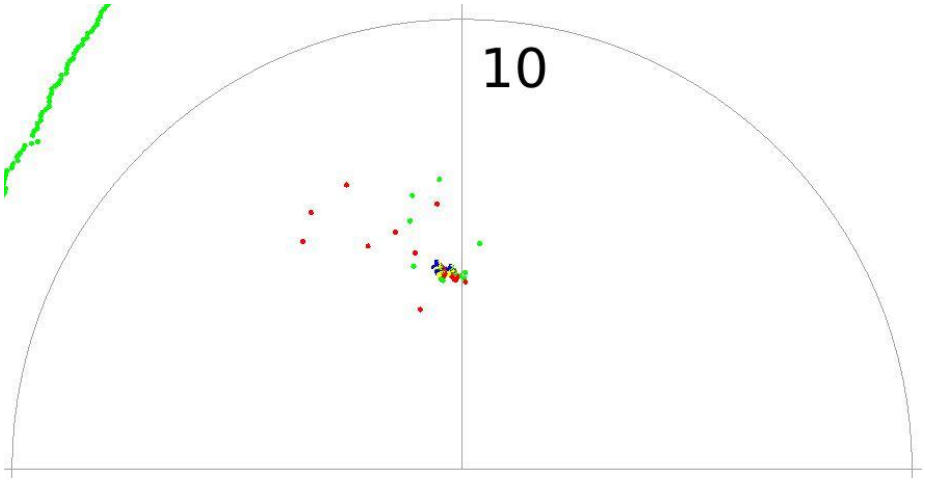

Figure 29. *Output of the first version of the weather filter.*

The next step was to filter also the measurement points within the threshold distance that came from the same laser pulse but weren't transparent. This reduced the amount of noise but did not considerably affect the perceived pedestrian. The output of the second version of the filter is shown in figure 30.

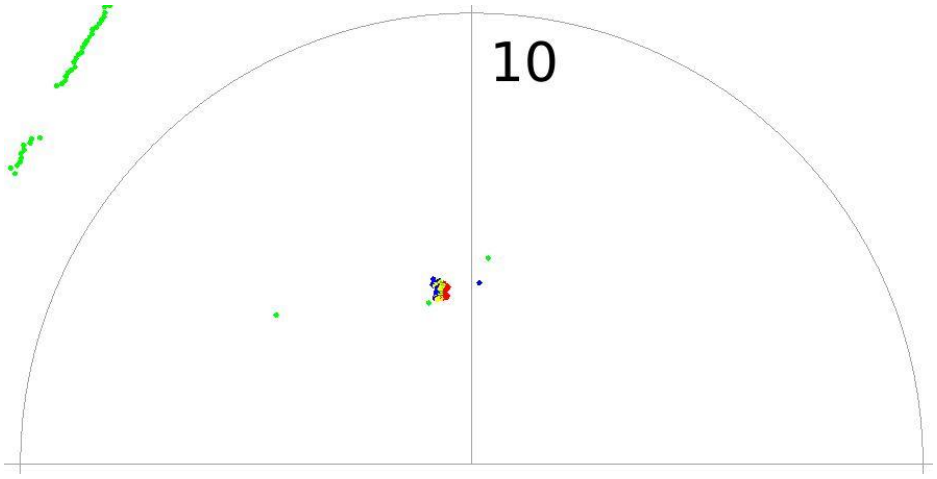

Figure 30. *Output of the second version of the weather filter.*

The second version of the filter performed well but some noise was still left in the measurements. The signal pulse widths of the filtered measurements were inspected and the rest of the noise points appeared to have relatively small pulse widths. After testing various pulse width filters, a value was found that left most of the pedestrian's measurement points intact but removed all of the weather created noise. Final filtering result is shown in figure 31.

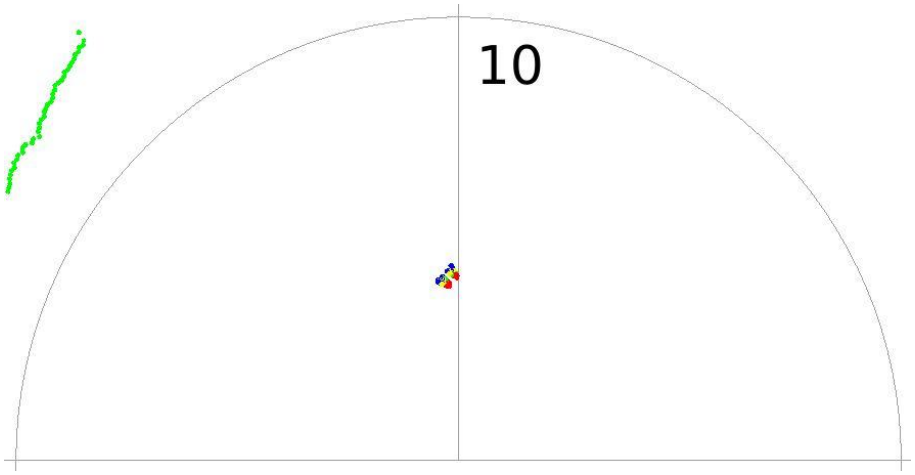

Figure 31. *Final version of the weather filter.*

The developed weather filtering should ideally be used only in cases where weather is creating noise in the measurements because it removes the lowest pulse width points. It should not affect the perceiving of real objects but there can be situations where critical information about the surroundings is lost. The combination of the signal pulse width ratio based weather detection and the filtering could prove to be valuable assets in development of automated vehicles. As VTT's automated vehicle development is focused on operating in harsh weather conditions, this filtering method is an important step forward. Two invention reports were filed at VTT regarding the estimation of the weather state and the filtering of noisy measurements.

These algorithm for weather analysis and noise filtering answer the needs of all automated vehicle developers. Much research has been published on the different kinds of utilization of LiDARs. These include such as clustering[38], negative obstacle detection[39] and object tracking[13]. All of these systems are rendered useless by harsh weather conditions if sufficient filtering is not used. The proposed algorithm provide a good start for the development of smart weather filtering with LiDARs.

8. CONCLUSIONS

The software developed for this thesis served as a good platform to seek the challenges of environment perception in automated vehicles. By the time of finishing this thesis, the collision avoidance software was not able to create accurate collision warnings in all situations. Despite the lack of fully functional software, the work in this thesis was successful in the way of developing the automated vehicle systems. Especially the work on automated vehicle's internal and external communication was useful in developing more advanced system modules. Vital information about the operating principles of the Sick's LiDARs was also gathered for this thesis and was later applied to other LiDAR-based perception modules. The information also led to filing of two invention reports at VTT regarding weather analysis and noise filtering.

This thesis also provided valuable information about systems that are not fitting for automated vehicles or possess limitations in their dedicated fields. For example, DDS system was exceptionally well performing when used in a local network but communication through a WAN would have required excessive configurations and special software and hardware solutions such as static IP-addresses which would not be available for large-scale commercial applications.

As a fully functional system was not finished, it is not possible to determine whether it would have been possible to create working collaborative sensing implementation with the modern sensor and communication technology. Challenges faced during the thesis implicate that the sensor setup used in VTT's automated vehicles is insufficient to enable robust object tracking with LiDARs in all scenarios. The communication on the other hand proved to be possible with a good choice of protocol. If the automated vehicle is concerned, data transfer capacity is sufficient enough for collaborative sensing. The challenge lies in the server end. Large-scale collaborative sensing requires the capability to manage hundreds or even thousands of connections and offer transfer rates with low latencies. The hierarchical structure of future's 5G networks are a plausible solution for the challenges of the collaborative sensing. As the data processing and transferring is handled more locally, the requirements for performance drastically decrease as opposed to centralized data processing and transferring.

Since the communication systems are capable of enabling collaborative sensing in the near future, the relevant question becomes what kind of other useful information can be exchanged between intelligent vehicles if the LiDAR's object tracking data is too unreliable. First commercial applications are probably going to deal with static observations of the environment. Information about an abnormal road condition or changed traffic arrangement could be easily exchanged with nearby vehicles since they are not as time-critical as an object tracking service. As the data processing and the sensor

technology of LiDARs advances, it will be a good sensor to rely on even in harshest weathers.

REFERENCES

[1] G. Thomaidis, K. Vassilis, P. Lytrivis, M. Tsogas, G. Karaseitanidis, and A. Amditis, "Target tracking and fusion in vehicular networks," in *IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 1080–1085.

[2] M. Obst, L. Hobert, and P. Reisdorf, "Multi-sensor data fusion for checking plausibility of V2V communications by vision-based multiple-object tracking," in *Vehicular Network Conference (VNC)*, 2014, no. January, pp. 143–150.

[3] "RobustSENSE web page." [Online]. Available: https://www.robustsense.eu/. [Accessed: 28-Sep-2017].

[4] "5G-SAFE web page." [Online]. Available: http://5gsafe.fmi.fi/index. [Accessed: 28-Sep-2017].

[5] M. Wang, W. Daamen, S. P. Hoogendoorn, and B. van Arem, "Rolling horizon control framework for driver assistance systems. Part II: Cooperative sensing and cooperative control," *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 290–311, 2014.

[6] M. Röckl, T. Strang, and M. Kranz, "V2V communications in automotive multi-sensor multi-target tracking," in *2008 IEEE 68th Vehicular Technology Conference (VTC)*, 2008, pp. 1–5.

[7] A. Festag, "Cooperative intelligent transport systems standards in Europe," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 166–172, 2014.

[8] G. Ozbilgin, U. Ozguner, O. Altintas, H. Kremo, and J. Maroli, "Evaluating the requirements of communicating vehicles in collaborative automated driving," in *Intelligent Vehicles Symposium (IVS)*, 2016, vol. August, pp. 1066–1071.

[9] M. Vasic, D. Mansolino, and A. Martinoli, "A system implementation and evaluation of a cooperative fusion and tracking algorithm based on a Gaussian Mixture PHD filter," in *Intelligent Robots and Systems (IROS)*, 2016, vol. November, pp. 4172–4179.

[10] Zheng Song, Yazhi Liu, Ran Ma, Xiangyang Gong, and Wendong Wang, "Short paper: Multi-task-oriented dynamic participant selection for collaborative vehicle sensing," in *2013 IEEE Vehicular Networking Conference*, 2013, pp. 214–217.

[11] L. Hobert, A. Festag, I. Llatser, L. Altomare, F. Visintainer, and A. Kovacs, "Enhancements of V2X communication in support of cooperative autonomous driving," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 64–70, Dec. 2015.

[12] I. Llatser, S. Kuhlmorgen, A. Festag, and G. Fettweis, "Greedy algorithms for information dissemination within groups of autonomous vehicles," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1322–1327.

[13] R. A. MacLachlan and C. Mertz, “Tracking of moving objects from a moving vehicle using a scanning laser rangefinder,” in *IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2006, pp. 301–306.

[14] “VTT | VTT’s autonomous cars take to public roads and start communicating with each other,” *VTT Web page*, 2017. [Online]. Available: http://www.vttresearch.com/media/news/vtts-autonomous-cars-take-to-public-roads-and-start-communicating-with-each-other. [Accessed: 26-Sep-2017].

[15] G. Brooker, *Introduction to sensors for ranging and imaging*. Institution of Engineering and Technology, 2009.

[16] Sick AG, “Operating instructions, LD-MRS.” 2009.

[17] VisLab, “3DV-E system.” [Online]. Available: http://vislab.it/products/3dv-e-system/. [Accessed: 19-Dec-2017].

[18] M. Kutila, P. Pyykönen, W. Ritter, O. Sawade, and B. Schäufele, “Automotive LIDAR sensor development scenarios for harsh weather conditions,” in *International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 265–270.

[19] A. Mendes, L. C. Bento, and U. Nunes, “Multi-target detection and tracking with a laserscanner,” in *IEEE Intelligent Vehicles Symposium (IV)*, 2004, pp. 796–801.

[20] T. H. Cormen, *Introduction to algorithms*. MIT Press, 2009.

[21] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 5th ed. New Jersey: John Wiley & Sons, 2012.

[22] D. H. Douglas and T. K. Peucker, “Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature,” John Wiley and Sons, 2011, pp. 15–28.

[23] M. A. Fischler and R. C. Bolles, “Random sample consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” *Communications of the ACM*, vol. 24, no. 6, 1981.

[24] I. N. Bronshtein, K. A. Semendyayev, G. Musiol, and H. Mühlig, *Handbook of mathematics*, 6th ed. Springer Berlin Heidelberg, 2015.

[25] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation*. Wiley, 2001.

[26] M. Thuy and F. P. León, “Non-linear, shape independent object tracking based on 2D lidar data,” in *IEEE Intelligent Vehicles Symposium (IV)*, 2009, pp. 532–537.

[27] C. Mertz *et al.*, “Moving object detection with laser scanners,” *Journal of Field Robotics*, vol. 30, no. 1, 2013.

[28] R. Maclachlan, “Tracking Moving Objects From a Moving Vehicle Using a Laser Scanner.” Carnegie Mellon University, 2005.

[29] K. C. Fuerstenberg, K. C. J. Dietmayer, and V. Willhoeft, “Pedestrian recognition in urban traffic using a vehicle based multilayer laserscanner,” in *IEEE Intelligent Vehicles Symposium (IV)*, 2003, vol. 1, pp. 31–35.

[30] D. Stroller, K. Furstenberg, and K. Dietmayer, “Vehicle and object models for robust tracking in traffic scenes using laser range images,” in *IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2002, pp. 118–123.

[31] B. Badic, C. Drewes, I. Karls, and M. Mueck, *Rolling out 5G: Use cases, applications, and technology solutions*. Apress Media LLC, 2016.

[32] “OpenDDS developer’s guide, OpenDDS Version 3.11.” Object Computing Inc., 2017.

[33] ScalAgent, “Benchmark of MQTT servers, version 1.1,” 2011.

[34] ETSI, “EN 302 663, Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band.” 2013.

[35] M. Jutila, J. Scholliers, M. Valta, and K. Kujanpää, “ITS-G5 performance improvement and evaluation for vulnerable road user safety services,” *IET Intelligent Transport Systems*, vol. 11, no. 3, pp. 126–133, 2017.

[36] Z. Lu, S. Qiao, and Y. Qu, *Geodesy : introduction to geodetic datum and geodetic systems*. Springer-Verlag Berlin Heidelberg, 2014.

[37] “Ilmatieteen laitos.” [Online]. Available: http://ilmatieteenlaitos.fi/. [Accessed: 23-Dec-2017].

[38] D. O. Rubio, A. Lenskiy, and J.-H. Ryu, “Connected components for a fast and robust 2D lidar data segmentation,” in *Asia Modelling Symposium 2013: 7th Asia International Conference on Mathematical Modelling and Computer Simulation (AMS)*, 2013, pp. 160–165.

[39] J. Larson and M. Trivedi, “Lidar based off-road negative obstacle detection and analysis,” in *4th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011, pp. 192–197.