POLITECNICO DI TORINO

MASTER DEGREE OF COMMUNICATIONS AND COMPUTER NETWORKS ENGINEERING

MASTER THESIS IN:

DESIGN OF AN INDOOR LOCALIZATION SYSTEM BASED ON LIDAR

SUPERVISORS

ROBERTO GARELLO

ORLANDO TOVAR ORDONEZ

CANDIDATE

LORENZO ALFONSO

ACADEMIC YEAR 2019/2020

CONTENTS

ACRONYMS

ACRONYM	DESCRIPTION
GPS	Global Positioning System
SLAM	Simultaneous Localization And Mapping
UWB	Ultra Wide Band
RANSAC	RANdom SAmple Consensus
IMU	Inertial Measurement Unit
TOF	Time Of Flight
SPAD	Single Photon Avalanche Diode
AMCW	Amplitude Modulated Continuous Wave
GNSS	Global Navigation Satellite System
FOV	Field Of View
DARPA	Defense Advanced Research Projects Agency
SNR	Signal to Noise Ratio
MEMS	MicroElectroMechanical System
UAV	Unmanned Aerial Vehicle
VFH	Vector Field Histogram
RMSE	Root Mean Square Error
URI	Uniform Resource Identifier
IP	Internet Protocol
ROS	Robot Operating System
UGV	Unmanned Ground Vehicle

1 INTRODUCTION

Many different technologies like GPS and radar have been used to localize, to map environments and to detect objects over time those let vehicles to move.

In this thesis, I analyze Lidar, especially in indoor, to let robots, drones, cars, etc. to move without a driver.

One of the most popular sensor, the GPS (i.e. the most popular GNSS technology), doesn't give a good accuracy in dense urban environments because the signal availability changes over time, therefore its potential for autonomous vehicles is limited; moreover, when it's integrated with IMU, errors can be accumulated when the vehicle travels because the measured positions doesn't follow the true position.

Camera based localization has the disadvantage to be sensitive to the changes of illumination, to observation angle and it can be affected by weather too; furthermore, when it's equipped with RANSAC algorithm, it's not good to distinguish stationary objects from objects in movement.

Ultrasonic based localization requires instead a very long processing time because its extractions of feature points aren't accurate and, when it's integrated with other sensors, errors can be accumulated; in addition, it's incompatible for autonomous vehicles also due to its short range to detect obstacles.

Radar based localization has the disadvantage to have many sources of noise, those make it a sensor with low robustness; moreover, it has

some problems to distinguish target's surface properties, out of its data association.
UWB sensor works well in indoor environment, but needs a reference node and it's bad in outdoor environment because its signals have short range.

Actually there isn't a dominant sensor technology based on autonomous navigation. One of the candidates is Lidar, which guarantees smaller positioning errors respect to other localization techniques like GPS, radar, sonar, camera, etc., but it's also more expensive respect to those ones. In autonomous navigation, it's important that a car/robot/drone is able to estimate its position, all near objects and the surrounding environment in every instant, to move and follow a path towards a destination and to avoid every object/obstacle in every instant. In common words, it's very important to make a safe journey without accidents.

1.1 OBJECTIVE

In the past, people explored unknown environments risking to be injured; then they created vehicles, which were at first non autonomous and due to that reason there were accidents, especially when the driver is distracted; only in the last years people developed robots, drones and autonomous cars which are finally able to avoid obstacles not only when they follow a path, but also when they explore in an unknown environment. The aim of this thesis is to describe the Lidar, explaining the reasons for which it was created and used, the

differences respect to other technologies, why is so important and how can be used for autonomous navigation.

1.2 OVERVIEW

Chapter 2 explains the role of Lidar in autonomous navigation systems, their problems and their respective solutions through obstacle avoidance algorithms. Chapter 3 describes Lidar's technology, history, characteristics, tipologies, advantages, disadvantages, use cases (especially for autonomous navigation), models (I will not include everything), context, measured data and its errors. Chapter 4 compares different localization algorithms and map extraction techniques used in Lidar and describes SLAM. Chapter 5 describes Matlab's Robotics System Toolbox explaining its commands too.

2 OVERWIEV OF AUTONOMOUS NAVIGATION SYSTEMS

2.1 LEVELS OF AUTONOMY

In the last years and in the near future, scientists and engineers are making efforts to increase vehicles autonomy (especially for autonomous cars, considering robots and drones have just got more autonomy). The Lidar, due to its very good accuracy respect to other localization technologies, can improve autonomy of vehicles and, for this reason, scientists and engineers are developing cheaper Lidar sensors, with enough small dimensions (especially for robots and drones), a good range (longer for cars and trucks), a good FOV and a good frequency of measurements. A Lidar sensor equipped on a vehicle can potentially avoid 80% of accidents, which are caused almost exclusively (about 90%) by human mistakes.

Scientists and engineers classified vehicles automation in six different levels.
At level 0, called “no automation”, the human driver must manage every driving task in the whole movement time.
At level 1, called “assisted driving”, there are few autonomous implementation like proximity alert, cross traffic alert, blind spot

detection, those can help the human driver either to accelerate/decelerate or to steer the vehicle.

At level 2, called “partial automation”, there are some autonomous implementation like lane keeping, adaptive cruise control, traffic jam assist, front/rear/intersection collision avoidance, autonomous emergency braking, emergency steer assist, those can both steer both accelerate/decelerate the vehicle. The human driver must monitor the driving environment.

At level 3, called “conditional automation”, also monitorization becomes autonomous, which allows autonomous driving in some modes like highway. The human driver can intervene, especially in harsh environment.

At level 4, called “high automation”, the vehicle is enough autonomous that can manage every driving task in most environmental condition, in most roads and in almost every instant. Pedals and steering wheels are still needed, while human driver isn’t mandatory.

At level 5, called “full automation”, the automated driving system can manage every driving task at every environmental condition, in every road and in every instant, while pedals, steering wheels and the human driver aren’t necessary. In this level, vehicles become robotic vehicles.

2.2 OBSTACLE AVOIDANCE ALGORITHMS

2.2.1 VECTOR FIELD HISTOGRAM

The Vector Field Histogram (VFH) isn’t only an obstacle avoidance algorithm, but also a path planning method. It’s very used on robots, it can execute in real-time, it’s robust also in cluttered obstacle

environments and it’s suitable with inaccurate sensor fusion and sensor data.

The VFH utilises a 2d histogram as an environmental map (it’s also possible consider it as a 2d histogram grid) that is created and updated at every scan time of the Lidar sensor. The 2d histogram is later reduced to a 1d polar histogram that divides the active window in a number of sections at a fixed angular resolution. Each section of the polar histogram around the temporary position contains a polar obstacle density value. The most appropriate section with a small density value is therefore chosen from all histogram sections and aligned with the vehicle steering.

2.2.2 MODEL PREDICTIVE CONTROL

The aim of the Model Predictive Control is to use a system model to be controlled to optimize and predict its future behavior. In fact, the MPC is an optimal control based state feedback controller. It’s possible to obtain the feedback law through an iterative online optimization over a finite moving prediction horizon (one of its advantages is the capacity to execute real time optimization even with hard constraints).

The most important components of the MPC are the dynamic optimizer, the cost function and constraints, and the vehicle model. An optimal control problem is defined through the equations describing the vehicle dynamics and also the cost function and constraints. Later, the dynamic optimizer is used to solve the optimal control problem to obtain the optimal steering angle to avoid obstacles. The state values of the given steering sequence are predicted by the vehicle model.

The inputs of MPC are the target direction, the obstacle information and the estimate vehicle state values.

A 2d Lidar sensor is used to obtain the obstacle information in an unknown environment. The Lidar sensor (modelled without noise and delay), in every radial direction at a negligible angular resolution, returns the distance to the nearest obstacle boundary. The angular range is [0°; 180°], while the vehicle heading direction is oriented at 90°. The Lidar sensor returns the maximum detection range for directions without obstacles inside its detection range.

The vehicle states are necessary for the initialization of the vehicle model used in MPC.

The control signals for the UGV are the outputs.

3 OVERVIEW OF LIDAR TECHNOLOGY

3.1 INTRODUCTION

The original aim of Lidar (LIght Detection And Ranging, or called also Laser Imaging Detection And Ranging) was the analysis of atmosphere with searchlights to measure air density and to determine scattering intensity, height information and cloud base heights without sending instruments up.
Lidar technology would not exist without laser invention (in 1960), that is based on stimulated emission (theorized by Einstein).
There are many different tipologies of Lidar which have been developed like elastic-backscatter, differential-absorption, raman, resonance fluorescence, doppler, bathymetric, geiger mode, spinning, solid-state, mechanical scanning, optical phased array, flash, single-photon.
Lidar, after its development in 1961, it was used also for different purposes like military since Vietnam war, space exploration since Apollo missions, sea depths analysis, forests analysis, mapping, autonomous navigation (this one is the most fluorishing application of Lidar currently) since DARPA Grand Challenge in 2004, archeology since 2000.

Lidar technology have had improvements not only from laser technology (like Q-switched laser), but also from optical filters, data

acquisition filters, detectors, computer able to analyze data, map techniques and localization algorithms.

Section 3.2 explains at first the technology of Lidar and of its components, then the differences among different types of Lidar like spinning, solid-state, mechanical scanning, optical phased array, etc., and finally the performances and their roles of some Lidar models produced by different companies.
Section 3.3 describes at first the characteristics of Lidar, later its error parameters and finally its data like accuracy, range, field of view, angle of incidence, etc..
Section 3.4 observes how Lidar can be used in different use cases like robotics, automotive, drones, military, space exploration, archeology, bathymetry, mapping, forest analysis, agriculture, industrial, etc..

3.2 TECHNOLOGY

In Lidar, we use wavelengths between ultraviolet and infrared bandwidth (including also the visible one). Older Lidar models used nitrogen, ruby, CO2 and copper-vapor lasers, later high power excimer and Nd:YAG: these two recent laser types both direct Lidar emitters both pump secodary laser transmitters.
A Lidar sensor consists of a source and a destination basically. The source contains a laser and a beam expander, while the destination contains a telescope, an optical analyzer, a detector and a computer.
The beam expander has the role to reduce light divergence generated by laser before sending it in the surrounding environment; in this way,

less photons are detected in the atmosphere and background light is decreased.

Lidar sensors rarely have lenses telescopes because backscattered beams would not be detected, analyzed and their data would not be collected; out of those reasons, most of Lidar sensors have mirror telescopes.

Only a part of light returning beam is measured because the it can’t be fully reflected onto the detector at short distance. This happens out of the degree of compression of a signal dependent on geometric arrangement of the source and of the destination.

Optical analyzer has a filter (in front of optical detector) that sends a light beam in a certain wavelength and limits light outside that wavelength.

The optical detector has the role to detect received light beams and to count those photons. This capability is possible because the optical detector is usually made of avalanche photodiodes or photomultiplier tubes. The photoncounting technique is individual, sensitive, made in Geiger mode and is applied if the deflected signal isn’t enough strong or the checked region is far from the Lidar sensor.

In the computer stage, Lidar data are coded and extracted.

3.2.1 TIPOLOGIES OF LIDAR

In this subsection, different Lidar tipologies related to autonomous navigation are described. Lidar tipologies can be classified according to the technology and to the number of dimensions for which a Lidar sensor can measure distances and/or create maps.

The two most popular Lidar tipologies according to the technology are the mechanical/scanning and the solid state one, which includes flash Lidar, MEMS Lidar and OPA Lidar.
Solid state Lidar is cheaper, smaller, lighter, more reliable, has a lower FOV, better performances, consumes less power and hasn't rotating mechanical components respect to the mechanical Lidar.

SINGLE PHOTON

SPAD (Single Photon Avalanche Diode) Lidar is based on TOF method and it's suitable for SLAM, autonomous navigation and ADAS too, due to its high sensitivity and high reliability in harsh environments like fog. It has a maximum range of 70-80m and produces range data, monocular data and peak intensity data.
A SPAD Lidar sensor can have many SPADs, which detectors contain readout and quench circuits. Inside a SPAD, in the moment where a single photon triggers an avalanche event, the readout circuit records an electrical pulse.
To create a macro pixel, element of SPAD detector array, the detectors will share a set of readout and quench circuits. A histogram is created through photon counts of returning repeated light pulses from taken over objects. The photon counts bounced from objects in a single

accumulation period is much lower (also dozens) because Lidar works also in harsh environments and laser power is pushed lower.

FLASH

A Flash Lidar has a single large area laser, where its pulse lights up the surrounding environment, and a focal plane array of photodetectors, placed next to the laser source, which collects the returning light.
Flash Lidar works in a similar way to that of a digital camera and it's able to obtain data very quickly because, respect to mechanical scanning Lidar, it can catch the whole scene in a single image and due to this reason, it's more immune to vibration effects, which can distort the picture.
Unfortunately, Flash Lidar requires a very high peak laser power to enlighten the whole scene and see far enough. Retroreflectors can make Flash Lidar sensors useless (they can be blinded) because they reflect most of the laser beam and deflect very little.

OPTICAL PHASE ARRAY

OPA (Optical Phase Array) Lidar has an optical phase modulator that checks the light speed going across the lens. The control of the light speed enables the check of the shape and orientation of the wave-front from the combination of the emission from the synchronized waveguides. The middle and the bottom light beams are retarded by increasing quantity, while the top light beam isn't delayed. This event allows the deflection of the light beam, that is steered to aim toward different directions. OPAs can achieve high scanning speed (also over

100k Hz) because they are robust and insensitive to external constraints like acceleration, due to absence of mechanical moving parts.
OPA Lidar can be in a single chip and compact, but has the disadvantage of insertion loss of the laser power.

MICROELECTROMECHANICAL SYSTEM

3.3.2 LIDAR MODELS AND CONTEXT

In this subsection, considering this thesis is focused on Lidar based localization and the large number both of Lidar companies both of their respective models, only one Lidar model per company specialized on autonomous navigation will be included. In these Lidar models, the tipology, the use cases and the measured data will be explained, if possible.

The Turtlebot 3 burger it's not only a Lidar sensor, but also a robot. It operates at SLAM, navigation and manipulation; it has a range of 3,5m, an angular resolution of 1°, a precision of 1cm (if the distance is less than 0,5m) or 3,5% (if the distance is larger) and an accuracy of 1,5cm (at shorter distances) or 5% (at longer distances).

Puck is a Lidar model produced by Velodyne, of which models are spinning Lidars. Puck works in different sectors like robotics, drone/UAV, mapping, security, industrial, smart city; it has a range of

100m, an accuracy of 3cm, an angular resolution of 0,1°*2° and a FOV of 360°*30°.

Innoviz One is the latest Lidar model made by Innoviz, of which models are mechanical scanning Lidars. Its use cases are the automotive, the mapping, the UAVs, the industrial and the security; it has a range of 250m, an accuracy of 3cm, an angular resoluton of 0,1°*0,1°, a resolution of 1cm and a FOV of 115°*25°.

OS0 is the most recent Lidar model developed by Ouster, of which models are spinning Lidars. Its applications are robotics, security, drones, autonomous vehicles; it has a range of 50m, an accuracy of 5cm (for lambertian tagets) or 10cm (for retroreflectors), a resolution of 0,3cm, a precision of 1,5cm (between 1 and 10m of distance) or 3cm (if the distance is between 0,25 and 1m and between 10 and 30m) or 10cm (if the distance is larger than 30m) and a FOV of 360°*95° (larger than other Lidar models made by that company).

Rs-lidar 16 is a Lidar model produced by Robosense. It operates at autonomous driving, robotics, industrial, UAV mapping and V2R; it has a range of 150m, an accuracy of 2cm, an angular resolution of 0,1°*2° and a FOV of 360°*30°.

Leddar Pixell is the latest Lidar model made by Leddartech, which is specialized on solid-state Lidars. It works in different sectors like autonomous shuttles, robotaxis, delivery vehicles, commercial vehicles, transit buses; it has a range of 41m, an accuracy of 5cm and an angular resolution of 1,9°*2°.

S3 is the most recent Lidar model developed by Quanergy, which focuses on optical phased array Lidars. Its applications are transportation, industrial automation, security data analytics and mapping; it has a range of 150m, an accuracy of 0,5cm, a resolution of 5cm and a FOV of 120°.

Pandar64 is a Lidar model made by Hesai. Its use cases are autonomous driving, hd mapping and autonomous logistics; it has a range of 200m, an accuracy of 5cm (between 0,3 and 0,5m of distance) or 2cm (if the distance is larger than 0,5m), an angular resolution of 0,2°*0,167° and a FOV of 40°.

UST-20LX is a Lidar model produced by Hokuyo. Its applications are autonomous robots, unmanned aerial vehicle, air touch panel, interactive exhibit, people counting and analysis of human movement patterns; it has a range of 60m, an accuracy of 4cm, an angular resolution of 0,25° and a FOV of 270°.

Sense One is a Lidar model developed by Sense Photonics, of which models are flash Lidars. It operates at robotics, material handling, activity monitoring, last-mile delivery, volumetric measurement, motion tracking, 3d inspection and palletization; it has a range of 28m, an accuracy of 3,5cm, an angular resolution of 0,27°*0,27°, a resolution of 10cm and a FOV of 95°*75°.

Opal Performance Series Conical 3d is a Lidar model made by Neptec. It works in different sectors like security, marine, aerospace, transport,

construction, mining, oil and gas; it has a range of 1000m, an accuracy of 2,5cm, a precision of 2cm and a FOV of 120°.

3.3 CHARACTERISTICS

When a light beam is sent form the Lidar sensor, an almost conical space is covered; and when a part of that conical space detects an object, the light beam gets the infrared reflectivity of that object, then the light beam turns back to the Lidar sensor, thus in this way we will get the distance between the Lidar sensor and the object in that time instant.
The Lidar, compared to other localization systems, guarantees much lower error localizations (also <2cm) because of light beams, which are much quicker than other mediums.
Lidar is not only able to localize object, but also to map environments: in fact it uses point cloud maps.

One of the disadvantages of Lidar is the necessity of a very large memory because Lidar can collect thousands or even millions of points for every second out of data point clouds.

Lidar has two methods to acquire range: TOF (Time Of Flight) and phase shift. The TOF can give larger range, while the phase shift offers better accuracy.
According to TOF method, the Lidar emits a light beam to a target and range measurement is determined by time difference between the emitted and received laser beams.

According to phase shift method, Lidar range is obtained by phase difference between the emitted and received deflected signal of an AMCW.

Lidar sensors can be integrated with cameras, IMU and GNSS.

Like every instrument of measuration, the Lidar, because it measures the distance between the sensor and the object, has some error parameters and measurement errors.

3.3.1 MEASURED DATA

The first measured data is the accuracy, which is the difference between the measured position and the true position of a Lidar sensor in the environment. Lidar accuracy errors are generally in the order of centimeters, while other localization systems, out of lower accuracy, generally require larger orders of magnitude. There are two types of accuracy: lateral and longitudinal. Lateral accuracy is generally smaller than the longitudinal accuracy because robots and vehicles have a longitudinal speed much greater than the lateral speed. Considering accuracy errors can never be equal in measurements of a Lidar sensor, a RMSE is usually made in datasheets and in sets of measurements.

The second measured data is the range, that is the maximum distance a laser beam sent from a Lidar sensor can detect objects in order to avoid obstacles. The maximum range of a Lidar model can depend on its purposes and where it can be used: if it’s developed to work on

robots in indoor environments, it can be <10m; if it's built to be equipped on autonomous cars in outdoor environments, it can be of several hundreds of meters out of higher velocity of cars, otherwise the driver would not have enough time to react and to avoid obstacles; it's also possible to find Lidar models which can work both in indoor both in outdoor environments, and in these cases the range is generally between 10m and 100m except for few exceptions those have larger range. The reflectivity can influence the range of Lidar model: at lower reflectivity, the range will be reduced significantly; while at higher reflectivity, the range will be almost at the maximum of that Lidar model. However, Lidar has problems to detect objects at very short distances.

The third error parameter is the FOV, which describes the angular extension of a scene that is projected by a laser beam. Horizontal FOV (it's sometimes also 360° and it's called azimuth too) is generally bigger than vertical FOV (it's rarely larger than 90° and it's called elevation too) because robots, cars and trucks moveents are in 2 dimensions.

The angular resolution (called angle of incidence too) isn't only one of the measured data, but also one of error parameters of Lidar. Most of Lidar sensors have an angular resolution smaller than 1° and it's very hard to find a Lidar model with angular resolution larger than 2°: in fact, if the angular resolution is very large, part of wave front is rebounced earlier and the returning laser beam will trigger the threshold either earlier or too late according to detector design of the front-end. If instead the angular resolution of the Lidar model is enough small, its

pulse detects an object surface perpendicularly and the whole laser wave front is reflected at the same instant.

Another error parameter of Lidar is temporal synchronization, because Lidar sensors equipped on moving robots can produce skew if it has been left uncompensated. It's possible to compensate through a continuous motion model, which requires timestamping and synchronization of sensor data.

The third error parameter of Lidar are the boundary effects, because the elliptical wave front, when it's moving, may cross part of objects and this means that spurious returns can happen at the boundary of those objects. Due to these reasons, the average of the resulting measurement range is often calculated and a spurious measurement is generated pendent between objects in empty spaces.

Object surface properties are the forth error parameter, because the amount of reflected laser beam can vary significantly. Objects which absorb infrared light (they generally pop up dark for humans) often don't reflect enough light, obtaining measurements shorter than expectations. Mirrors, water and highly specular objects reflect most of the laser beam, obtaining measurements related to further objects. It's possible to get better measurements if an object reflects light diffusely.

Another error of parameter is ambient light, because direct sunlight and other strong ambient reduce the range of those Lidar sensors which use infrared notch filter (it's also very popular), needful to improve the SNR and to work in outdoor. A typical sensor model retains there is

free space, which can vary according to the intensity of ambient light, until a fraction of the Lidar sensor's maximum range, if the returning light beam isn't enough strong to trigger a measurement. It's impossible to determine when happens missing Lidar measurements out of strong ambient light or actual free space, without other sensors.

3.4 USE CASES

Lidar technology has been used in different sectors through its detection capability during the years. It was used at first to detect atmospheric particles and in the last years to move vehicles such as cars, robots and drones following a path.
In this section, some use cases will be described.

3.4.1 AUTONOMOUS NAVIGATION

DARPA Grand Challenges, perfomed respectively in 2004, 2005 and 2007, were the first important step about Lidar in autonomous navigation. In fact, after the first Grand Challenge, Velodyne started the development of spinning Lidars, which were used later by most of participants at the last Grand Challenge. After the spinning Lidar, also different tipologies of Lidar like mechanical scanning, flash, optical phased array, solid state, single photon have been developed. In this use case, Lidar is equipped on robots, drones and vehicles (they are becoming autonomous) like cars and trucks. All of them must localize their own position, avoid obstacles, follow a path and map surrounding environments; robots and drones can also watch over surrounding environments, carry objects. SLAM is required especially in outdoor

(out of high speed of vehicles in roads), but it's possible to find moving objects in indoor too. Autonomous navigation will continue to be the most fluorishing application of Lidar in the near future because of increasing autonomy of trucks and especially of cars, which market has a very large size.

3.4.2 SPACE EXPLORATION

In space exploration, Lidar is used mainly on space probes and space satellites to map planets and natural satellites surfaces (for example Icesat). The first uses of Lidar on this use case were in Apollo missions 11 (in 1969), 14 and 15 (these ones in 1971). Mars Global Surveyor (equipped with Mars Orbiter Laser Altimeter) space probes was sent to Mars, while Chang'e, Selene and Chandrayan spacecrafts were sent to the Moon in 2007-2008. In the future, also rovers will be equipped with Lidar, considering Spirit, Opportunity and Curiosity haven't it: in fact, rovers can use only Lidar with SLAM, which hasn't been fully developed and mature yet also because of complexity of spatial missions and of high costs not only of Lidar technology, but also both of launch both of operations.

3.4.3 SEARCH AND RESCUE

Natural disasters, like earthquakes of higher Richter magnitude, can modify environments due to the eventual destruction of buildings. For this reason, in those environments, a priori maps could no longer exist and in these cases, an USAR response is necessary, but the time to ago across rubbles, to localize survivors and to create maps is often

limited; the Lidar sensor with SLAM can help these situations. Considering those environments are not usually safe for people, USAR robots have been developed, but they are often teleoperated by humans. The next aim of this use case is to improve USAR robots autonomy.

3.4.4 MILITARY

Military use case is one of the oldest, excluding the atmosphere one. Lidar sensors are often equipped on robots and aircrafts in these situations and they are generally used to localize enemies (for example nuclear warheads or a group of soldiers hidden in a forest or in a building) and to map environments (it's possible to distinguish a tank from a photo taken from an aircraft) in order to prevent attacks. War environments are usually dangerous not only for people, but also for robots: in fact, explosions can change environments, causing the destruction of buildings. Thus, like in search and rescue use case, Lidar sensor with SLAM is recommended also because GPS can be spoofed or jammed. One of the first uses of Lidar technology in war was in Vietnam.

3.4.5 MINING AND EXTRACTION

The wish to improve safety and rising labor costs have stimulated automation in mining industry in the last years. The GPS (its signal disappears in underground) and the UWB (it must be installed in underground) aren't very useful in those environments. Lidar sensors, often equipped on autonomous robots and trucks (which move rocks)

in these use case, are a good solution because they provide localization and obstacle detection, in order to avoid accidents. This use case is correlated with “search and rescue”: in fact, also this time Lidar sensors with SLAM is recommended because in those environments can happen unexpected accidents those modify environments.

3.4.6 ARCHEOLOGY

Lidar was used in archeology since around 2000 (in fact it’s one of the most recent use cases). In archeological sites, the Lidar sensor, which is usually equipped on robots and drones, can scan the surrounding environment and create digital elevation models through maps. Documentaries are sometimes made and it’s even possible to approximate the original status of the archeological site through Lidar measurements. SLAM technology can be requested especially if a robot must navigate through tight spaces considering archeological sites are unknown environments for humans.

3.4.7 ATMOSPHERE

Atmosphere is the oldest use case and also the original aim of Lidar: in fact, in 1930s, scientists wanted to investigate atmosphere without sending instruments up, but laser technology hasn’t existed yet.
The first Lidar measurements were did with searchlights, while in the following years have been done with aircraft, probes, satellites
In this use case different tipologies of Lidar are used, such as elastic-backscatter, differential-absorption, raman, fluorescence and doppler.

Lidar technology did a big contribution to the knowledge of Earth's atmosphere in the last decades.
Lidar can analyze atmospheric compososition and can measure clouds, trace gases, aerosols, temperature, pressure, wind and humidity. It's possible to study diurnal cycle, meteorolgical phenomena like hurricane, the measurements of water-vapor and ozone fluxes. Stratospheric measurements would be impossible without Lidar. In additon, Lidar can recognize ice crystals from water droplets, investigate stratosphere after volcanic eruptions and detect desert dust, air pollution and forest-fire smoke. In mesosphere, Lidar demonstrated the presence of layers of metallic ions and atoms.

3.4.8 BATHYMETRY

Bathymetry use case is one of the oldest, excluding the atmosphere one. If Lidar sensors are equipped on flying vehicles like aircrafts, helicopter and UAVs, they are used for depth sounding, fluorosensing and to create not only topografic maps but also hydrografic charts. If Lidar sensors are equipped on water vehicles like ships, boats, submarines and water robots, they need SLAM to avoid fishes, cetaceans, corals, plancton, octopi, reef, other ships, boats, etc., because a priori maps in underwater environments don't exist.

3.4.9 FOREST

In forests, Lidar sensors weren't very used in the first years, but there have been progresses since 1990s, like SLICER in 1994 (to study forest composition) and Jigsaw program in 2001 (that uses a Geiger-

mode Lidar to see under tree canopies). In this use case, Lidar sensors were equipped only on aircrafts and helicopters, while in the last years they are equipped also on space stations, drones and robots. In this sector, Lidar is used to distinguish different types of vegetation, to measure the thickness of trunks (it's easier for robots) and tree heights (also called canopy heights), to calculate biomass and volume, to study forest composition and to estimate the position of trees; furthermore, robots, drones and helicopters need SLAM because not only they have to avoid birds and other animals hidden in the bushes, but forests can be very far from urban environments. The canopy height is calculated as the difference between those Lidar returns aren't categorized as terrain.

4 LOCALIZATION ALGORITMHS AND MAP EXTRACTION TECHNIQUES

4.1 LOCALIZATION ALGORITHMS

4.1.1 PARTICLE FILTER

The particle filter is a sequential version of Montecarlo localization algorithms and works well in nonlinear-non-gaussian systems. The posterior distribution of a stochastic process is obtained through a set of particles distributed in a way identical and independent. The particle filter has four steps called inizialization, prediction, update and resample respectively.

At the inizialization step, every generated particle has weight 1/N, considering N as the number of particles. All the following steps can be repeated at every time instant.

At the prediction step, the prior probability is predicted by the posteriori probability of the previous instant time, then the set of particle is sampled from that prior probability. The posteriori probability of a determined time instant is obtained from the prior probability and the sampled particles.

At the update step, the particle weights are updated through likehood functions and later normalized.

At the resample step, old particles are replaced by newer particles generated and the weight of each particle is updated to 1/N'.

4.1.2 MONTECARLO LOCALIZATION

The Montecarlo localization is a famous method used by robots to localize. It's a Bayesian method localization able to estimate almost any distribution of a robot pose. It has a very good execution time and it's applicable with both local both global localization problems, with linear or nonlinear systems, Gaussian or nongaussian systems, parameterized or not. Although the Montecarlo localization has many advantages, it isn't able to find the local minima if there are many particles. The Montecarlo localization has an iterative cycle, which has two steps called prediction and update.
At the prediction step, the motion model (based on a conditional probability) is resampled to create a new set of particles.
At the update step, the weights of new particle are updated according to the sensor's measurement model (based on a likehood).
After the iterative cycle, the importance factors are normalized.

4.1.3 ITERATIVE CLOSEST POINT

The Iterative Closest Point is both a localization algorithm both a map extraction technique. It has an iterative cycle, which includes five steps. It's slower than other localization algorithms because of its iterative cycle and its correspondences searches in the whole point cloud to find the best possible transformation: for these reasons, the ICP isn't very good in real-time applications. Although the Iterative Closest Point is

time consuming, its distance metric has quadratic divergence and it can guarantee good accuracy and robustness in general environments. Furthermore, it's applicable in unstructured environments since the aim of ICP is to align the current scan and the reference scan without extracting features.

At the first step, inertial sensors give initial translation and rotation.

At the second step, the current scan is transformed through current rotation and translation.

At the third step, a research of the two closest corresponding points in the reference scan is made for every point of the transformed current scan.

At the forth step, the sum of the square distance between the point of transformed current scan and the line segment containing the two closest corresponding points is minimized.

At the fifth step, there is a convergence check: if it isn't reached, the ICP will return to the second step to search new correspondences and repeat the proceedings; otherwise, the ICP will go on the process of the next new scan.

4.1.4 KALMAN FILTER

The Kalman filter gives linear (the estimate is a linear combination of the current measurements and of the previous estimates), unbiased (the mean estimation error is zero), optimal (the variance of the estimate is minimized) and recursive (the current estimate is obtained only by the current measurement and the previous estimate) estimates of a dynamical system state from its noisy measurements.

The state of a dynamical system is expressed as the sum between the input, the system noise and the product between the transition matrix and the previous state of the system.

The current measurement is expressed as the sum between the measurement noise and the product between the measurement matrix and the state of the system.

In frequency analysis, a filter is a system that choose a particular bandwidth of the frequency system. Without the filter and the noise, it's impossible to obtain the estimate of a trend.

In a non-recursive estimator, the mean value is expressed as the ratio between the sum of the mean value of measurements and the number of measurements. Instead, in a recursive estimator, the mean value is expressed as the ratio between the sum between the product between the previous mean value and the number of previous measurements, and the mean value of the new measurement, and the number of measurements. The recursive estimator has fixed computational cost and memory usage, while the non-recursive estimator has both of them linear.

A classical example of the Kalman filter is the motion of a boat on the ocean. Moreover, the Kalman filter is often used in localization systems and its noises are normally distributed.

The system model equation has the same form both in the continuous-time, both in the discrete-time (in fact the system noise and the input can be also expressed as integrals, while the transition matrix can be exponential). The mean value of the system noise is equal to zero, while the covariance of the system noise (also called covariance matrix) is expressed as the mean value of the product between the system noise and its transposed.

The estimator is expressed as the sum between the product between the extrapolated Kalman gain and the extrapolated state estimate and the product between the Kalman gain and the measurement. The Kalman gain is linear and recursive.

The estimator is also called as the updated state estimate and can be expressed also as the sum between the estimation error of the updated state estimate and the state of the system. The extrapolated state estimated is instead expressed as the sum between the estimation error of the extrapolated state estimate and the state of the system. The mean value of the estimation error of the updated state estimate is equal to zero, in order to have an unbiased estimator.

It’s possible to obtain the Kalman gain only if the estimator is optimal (the variance of the estimate is minimized, it means that the mean value of the absolute value of the estimation error of the updated state estimate is minimum).

The mean squared length of the estimation error of the updated state estimate is easier to be minimized, while the updated error covariance matrix is expressed as the mean value of the product between the estimation error of the updated state estimate and its transposed.

It’s possible to find the optimal estimator for the Kalman gain that minimizes the mean squared length of the estimation error of the updated state estimate expressed as the trace of the updated error covariance matrix.

Deriving the updated error covariance matrix, the extrapolated error covariance matrix is expressed as the mean value of the product between the estimation error of the extrapolated state estimate and its transposed, while the mean value of the product between the

estimation error of the extrapolated state estimate and the transposed of the measurement noise is equal to zero.

The updated error covariance matrix is later expressed as the sum between the product between the Kalman gain, its transposed and the mean value between the product between the measurement noise and its transposed, and the product between the extrapolated error covariance matrix, the difference between the identity matrix and the product between the Kalman gain and the measurement matrix, and the transposed of the difference between the identity matrix and the product between the Kalman gain and the measurement matrix.

The Kalman gain is expressed as the ratio between the product between the measurement matrix and the extrapolated error covariance matrix, and the sum between the mean value between the measurement noise and its transponsed, and the product between the extrapolated error covariance matrix, the measurement matrix and its transposed.

In addition, the extrapolated Kalman gain is expressed as the difference between the identity matrix and the product between the Kalman gain and the measurement matrix. Therefore the updated state estimate is expressed as the sum between the product between the Kalman gain and the current measurement, and the product between the extrapolated state estimate and the difference between the identity matrix and the product between the Kalman gain and the measurement matrix.

The state estimate extrapolation is expressed as the sum between the input and the product between the transition matrix and the previous updated state estimate. Instead the error covariance extrapolation is expressed as the sum between the mean value of the product between

the system noise and its transposed, and the product between the previous extrapolated error covariance matrix, the transition matrix and its transposed.

4.2 MAPS EXTRACTION TECHNIQUES

5 MATLAB'S ROBOTICS SYSTEM TOOLBOX

In this chapter, Matlab's Robotics System Toolbox (updated to version 2019a) will be described together with its commands relative to Lidar based localization. In fact, Matlab it's one of these programs which able to move a robot equipped with a Lidar sensor from a point "A" to a point "B" and avoiding both static both moving obstacles.

The Robotics System Toolbox gives not only hardware connectivity with many different types of robots like manipulators, humanoid, aerial and ground vehicles, but also algorithms which allow robots to avoid obstacles, to scan the surrounding environment, to localize its own position, to plan paths, to follow paths.

This toolbox gives an interface with the ROS and at least one between Matlab and Simulink. The ROS allows tests on robot simulators like Gazebo and on ROS based robots.

5.1 COMMANDS ABOUT ROS

Robot Operating System (ROS) of Robotics System Toolbox allows the user to interact with ROS and to use its operations in Simulink and Matlab. It's possible to access to ROS networks and robots, to collect data, to send and receive own messages, and to allow simulations in Gazebo.

5.1.1 COMMANDS ABOUT ROS NOT RELATED TO GAZEBO SIMULATOR

rosinit

The rosinit has the role to connect to a ROS node and its use is mandatory for most ROS-related tasks in Matlab because the communication with a ROS network needs a ROS node linked to a ROS master and the ROS functions in Matlab work on the global ROS node or on objects those depend on the global ROS node. It's usually used to start a ROS core and a global node, to start a node and link to the ROS master at specified IP address, to start a global node at given IP and node name. The command rosinit hasn't output parameters, but input parameters like the host name (or IP address), the port number, the URI for ROS master and the global node name.

rosshutdown

The rosshutdown has the role to shut down a ROS system: in fact it shuts down the global node and, if it's operating, the ROS master. It's used when all the tasks related to the ROS network are finished and it has neither input parameters nor output parameters.

rostopic

The rostopic has the role to return informations (usually as messages) about a specific ROS topic. It's usually used to get a list of ROS topics,

to get ROS topic informations and to get a ROS topic message type. In addition, the rostopic command has the ROS topic name as input parameter, while its output parameters are the list of topics from the ROS master, the ROS message for a given topic, the information about a given ROS topic and the message type for a given ROS topic.

rospublisher

The rospublisher has the role to publish messages. It's used to create a ROS publisher before the expedition of data, to create a ROS publisher and view properties, to publish data without a ROS publisher and to use a ROS publisher object. Moreover, its properties are the name of the published topic, the message type of published messages, the indicator of whether publisher is latching and the number of subscribers.

rossubscriber

The rossubscriber has the role to subscribe to messages on a topic. It's used to create a subscriber and get data from ROS, to create a subscriber that uses a callback function and to use a ROS subscriber object. Furthermore, its properties are the name of the subscribed topic, the message type of the subscribed messages, the latest message sent to the topic, the buffer size and the callback property.

rosmessage

The rosmessage has the role to create ROS messages. It's used to create an empty string message, to create and access an array of ROS messages, to preallocate a ROS message array and also to create a ROS publisher and send data. The command rosmessage has the ROS message as output parameter, while its input parameters are the message type, the ROS publisher, the ROS subscriber, the ROS service client and the ROS service server.

send

The command send has the role to publish a ROS message to the topic specified by the publisher. It's used to send ROS messages after their creation and before their reception. The command send hasn't output parameters, but input parameters like the ROS publisher and the ROS message.

receive

The command receive has the role to wait for a new ROS message. It's used to get data from ROS after the creation of the subscriber, to read a specific field from the point cloud message and to receive ROS messages after their creation and expedition. The command receive has the ROS message as output parameter, while its input parameters are the ROS subscriber and the timeout for receiving a message.

5.1.2 COMMANDS ABOUT ROS RELATED TO GAZEBO SIMULATOR

5.2 COMMANDS ABOUT GROUND VEHICLE ALGORITHMS

Ground Vehicle Algorithms of Robotics System Toolbox are specialized on mobile robotics operations. It's possible to execute SLAM, develop robot path planning, create environment maps, regulate controllers to follow a set of waypoints, estimate localization based on robot sensor data and to execute obstacle avoidance.

lidarScan

The lidarScan has the role to create object for storing 2d Lidar scan. It's used to plot Lidar scan and remove invalid points and also to match Lidar scans; furthermore, its properties are the range readings from Lidar, the angle of readings from Lidar, the cartesian coordinates of Lidar readings and the number of Lidar readings.

robotics.LidarSLAM

The robotics.LidarSLAM has the role to perform localization and mapping using Lidar scans. It's used to perform SLAM using Lidar scans; moreover, its properties are the underlying pose graph that connects scans, the resolution of occupancy grid map, the maximum range of Lidar sensor, the pose graph optimization function, the threshold for accepting loop closing, the search radius for loop closure detection, the number of attempts at finding loop closures, allow

automatic rollback of addedd loop closures, the number of loop closures accepted to trigger optimization and the minimum change in pose required to process scans.

robotics.ParticleFilter

The robotics.ParticleFilter has the role to create a state estimator based on particle filter. It's used to predict and correct particle filter and also to estimate robot position in a loop using particle filter; in addition, its properties are the number of state variables, the number of particles used in the filter, the callback function for determining the state transition between particle filter steps, the callback function calculating the likehood of sensor measurements, the indicator if state variables have a circular distribution, the policy settings that determine when to trigger resampling, the method used for particle resampling, the method used for state estimation, the array of particle values, the particle weights, the best state estimate and the corrected system covariance.

robotics.MonteCarloLocalization

The robotics.MonteCarloLocalization has the role to localize robot using range sensor data and map. It's used to estimate robot pose from range sensor data; furthermore, its properties are the initial pose of robot, the covariance of initial pose, the flag to start global localization, the minimum and maximum number of particles, the likehood field sensor model, the odometry motion model for differential drive, the minimum change in states required to trigger update, the

number of filter updates between resampling of particles and the use of lidarScan object as scan input.

robotics.VectorFieldHistogram

The robotics.VectorFieldHistogram has the role to avoid obstacles using the vector field histogram. It's used to create a vector field histogram object and visualize data; moreover, its properties are the number of angular sectors in histogram, the limits for range readings, the radius of robot, the safety distance around robot, the minimum turning radius at current speed, the cost function weight for target direction, the cost function weight for current direction, the cost function weight for previous direction, the thresholds for binary histogram computation and the use of lidarScan object as scan input.

robotics.PRM

The robotics.PRM has the role to create a probabilistic roadmap path planner. Its properties are the maximum distance between two connected nodes, the map representation and the number of nodes in the map. The command robotics.PRM has the map representation and the maximum number of nodes in roadmap as input parameters.

robotics.PurePursuit

The robotics.PurePursuit has the role to create a controller to follow a set of waypoints. Its properties are the desired linear velocity, the look-ahead distance, the maximum angular velocity and the waypoints. The

command robotics.PurePursuit has the position and orientation of robot as input parameter, while its output parameters are the linear velocity, the angular velocity and the look-ahead point on path.

buildMap

The buildMp has the role to build an occupancy grid from Lidar scans. It's used to build an occupancy map from Lidar scans and poses, and at the end of execution of SLAM using Lidar scans. The command buildMap has the occupancy grid as output parameter, while its input parameters are the Lidar scans, the poses of Lidar scans, the resolution of occupancy grid, the maximum range of the Lidar sensor, the width of occupancy grid and the height of occupancy grid.

robotics.OccupancyGrid

The robotics.OccupancyGrid has the role to create a 2d occupancy grid with probabilistic values: every cell in the grid has a value that corresponds to the probability of having an obstacle in that cell (the maximum value is equal to 1 and corresponds to certainty, while the minimum value is equal to 0 and corresponds to absence). It's used before inserting laser scans into the occupancy grid and to convert a portable graymap image to map; moreover its properties are the threshold to consider cells as obstacle free, the threshold to consider cells as occupied, the saturation limits for probability, the number of rows and columns in grid, the grid resolution, the minimum and maximum world range values of x-coordinates, the minimum and

maximum world range of y-coordinates and the [x, y] world coordinates of grid.

REFERENCES

- Jens Einsiedler, Ilja Radusch, and Katinka Wolter; *Vehicle Indoor Positioning: A Survey*; 2017 14TH WORKSHOP ON POSITIONING, NAVIGATION AND COMMUNICATION (WPNC);

- Sampo Kuutti, Saber Fallah, Konstantinos Katsaros, Mehrdad Dianati, Francis Mccullough, Alexandros Mouzakitis; *A Survey of the State-of-the-Art Localisation Techniques and Their Potentials for Autonomous Vehicle Applications*; DOI 10.1109/JIOT.2018.2812300, IEEE Internet of Things Journal, IoT-2371-2017.R2;

- Levente Tamas, Gheorghe Lazea, Mircea Popa, Istvan Szoke, Andras Majdik; *Laser Based Localization Techniques for Indoor Mobile Robots*; 2009 Advanced Technologies for Enhanced Quality of Life;

- Eiliv Hägg, Yingzhi Ning; Master's thesis in the Signal and Systems department, Chalmers; *Map Representation and LIDAR-Based Vehicle Localization*; Department of Signal and System,

CHALMERS UNIVERSITY OF TECHNOLOGY, Gothenburg, Sweden 2016;

- Robert George Reid; Thesis presented for the degree of Doctor of Philosophy; *Large-Scale Simultaneous Localization and Mapping for Teams of Mobile Robots*; School of Electrical, Electronic and Computer Engineering, The University of Western Australia, July 2016;

- Yun-Ting Wang, Chao-Chung Peng, Ankit A. Ravankar and Abhijeet Ravankar;*A Single LiDAR Based Feature Fusion Indoor Localization Algorithm*; Sensors 2018, 18, 1294; doi:10.3390/s18041294;

- YUAN XU1 , (Member, IEEE), YURIY S. SHMALIY2 , (Fellow, IEEE), YUEYANG LI1 , (Member, IEEE), XIYUAN CHEN3 , (Senior Member, IEEE), AND HANG GUO4; *Indoor INS/LiDAR-Based Robot Localization With Improved Robustness Using Cascaded FIR Filter*; *Digital Object Identifier 10.1109/ACCESS.2019.2903435;*

- Yue Xiao Yongsheng Ou* Wei Feng; *Localization of Indoor Robot based on Particle Filter*

with EKF Proposal Distribution; 2017 IEEE 8th International Conference on CIS & RAM, Ningbo, China;

- Claus Weitkamp; *Lidar: Range-Resolved Optical Remote Sensing of the Atmosphere*; Springer Science & Business, 3 June 2006;

- I. Puente ⇑, H. González-Jorge, J. Martínez-Sánchez, P. Arias; *Review of mobile mapping and surveying technologies*; journal homepage: www.elsevier.com/ locate/measurement;

- Todd Neff; https://toddneff.com/books/lidarhistory/extras/lidarhistory-timeline/;

- Sung-You Tsai, Yu-Cheng Chang, and Tzu-Hsien Sang* ; *SPAD LiDARs: Modeling and Algorithms;*

978-1-5386-4441-6/18/$31.00 ©2018 IEEE;

- Rémy Guyonneau, Sébastien Lagrange, Laurent Hardouin & Philippe Lucidarme; *Guaranteed interval analysis localization for mobile robots*; *Advanced Robotics*, 2014, Vol. 28, No. 16, 1067–1077, http://dx.doi.org/10.1080/01691864.2014.908742;

- Lei Zhao, Zhun Fan, Wenji Li, Honghui Xie, Yang Xiao; *3D Indoor Map Building with Monte Carlo Localization in 2D Map*; 2016

International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration;

- Louay Eldada; *Solid State LiDAR Sensors: The Future of Autonomous Vehicles*; http://www.imaging.org/Site/PDFS/Conferences/ElectronicImaging/EI2019/AVMKeynotes/LouayEldada_Quanergy_AVM%202019.pdf;

- https://www.egps.net/datasheets/quanergy-m8-datasheet.pdf;

- https://quanergy.com/wp-content/uploads/2017/01/CES-2017-S3-Press-Release_Final.pdf;

- G. Ajay Kumar, Ashok Kumar Patil, Rekha Patil, Seong Sill Park and Young Ho Chai; *A LiDAR and IMU Integrated Indoor Navigation System for UAVs and Its Application in Real-Time Pipeline Classification*; Sensors 2017, 17, 1268; doi:10.3390/s17061268;

- Yanbin Gao, Shifei Liu,*, Mohamed M. Atia and Aboelmagd Noureldin; *INS/GPS/LiDAR Integrated Navigation System for Urban and Indoor Environments Using Hybrid Scan Matching Algorithm*; *Sensors* 2015, *15*, 23286-23302; doi:10.3390/s150923286;

- https://autonomoustuff.com/wp-content/uploads/2019/11/velodyne-lidar-product-brochure-as-branded.pdf;

- Rajat Sagar; *Making cars safer through technology innovation*; © 2017 Texas Instruments Incorporated;

- https://www.symphotony.com/wp-content/uploads/Pandar64-64-Channel-Mechanical-LiDAR-3-1.pdf;

- https://www.hokuyo-aut.jp/search/single.php?serial=167;

- Santiago Royo1,2* and Maria Ballesta; *An overview of imaging lidar sensors for autonomous vehicles*; Appl. Sci. 2019. 9 4093, doi:10.3390/app9194093;

- Lorenzo Galleani, Patrizia Tavella, *Time and the Kalman filter*, IEEE Control System Magazine;

- Kevin Lima,*, Paul Treitza, Michael Wulderb, Benoît St-Ongec and Martin Floodd; *LiDAR remote sensing of forest structure*; *Progress in Physical Geography* 27,1 (2003) pp. 88–106;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> robot operating system (ROS);

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> ground vehicle algorithms;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → rosinit;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → rosshutdown;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → rostopic;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → rossubscriber;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → rospublisher;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → send;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → receive;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → rosmessage;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → lidarScan;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → robotics.LidarSLAM;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → robotics.ParticleFilter;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → robotics.MonteCarloLocalization;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → robotics.VectorFieldHistogram;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → robotics.PRM;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → robotics.PurePursuit;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → buildMap;

- Matlab 2019a-> getting started -> documentation home -> robotics system toolbox -> functions and other reference → robotics.OccupancyGrid;

- Seigo Ito, Shigeyoshi Hiratsuka, Mitsuhiko Ohta, Hiroyuki Matsubara, Masaru Ogawa; *SPAD DCNN: Localization with Small Imaging LIDAR and DCNN*; 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) September 24–28, 2017, Vancouver, BC, Canada;

- Seigo Ito *, Shigeyoshi Hiratsuka, Mitsuhiko Ohta, Hiroyuki Matsubara and Masaru Ogawa; *Small Imaging Depth LIDAR and DCNN-Based Localization for Automated Guided Vehicle*; Sensors 2018, 18, 177;

- Jiechao Liua , Paramsothy Jayakumarb, Jeffrey L. Steina and Tulga Ersala; *A study on model fidelity for model predictive control-based obstacle avoidance in high-speed autonomous ground vehicles*; VEHICLE SYSTEM DYNAMICS, 2016 VOL. 54, NO. 11, 1629–1650 http://dx.doi.org/10.1080/00423114.2016.1223863;

- Jiechao Liua , Paramsothy Jayakumarb, Jeffrey L. Steina and Tulga Ersala; *A MULTI-STAGE OPTIMIZATION FORMULATION FOR MPC-BASED OBSTACLE AVOIDANCE IN*

AUTONOMOUS VEHICLES USING A LIDAR SENSOR; Proceedings of the ASME 2014 Dynamic Systems and Control Conference DSCC2014 *October 22-24, 2014, San Antonio, TX, USA;*

- Jian Tang 1,2, Yuwei Chen 2,3,*, Antero Kukko 2, Harri Kaartinen 2, Anttoni Jaakkola 2,

Ehsan Khoramshahi 2,4, Teemu Hakala 2, Juha Hyyppä 2, Markus Holopainen 5

and Hannu Hyyppä 6,7*; SLAM-Aided Stem Mapping for Forest Inventory*

with Small-Footprint Mobile LiDAR; Forests 2015, 6, 4588–4606; doi:10.3390/f6124390*;*

- Motaz Khader, Samir Cherian; *An Introduction to Automotive LIDAR;* © 2018 Texas Instruments Incorporated*;*

- Jing Li1 & Hong Bao1 & Xiangmin Han1 & Feng Pan1 & Weiguo Pan1 & Feifei Zhang1 & Di Wang1; *Real-time self-driving car navigation and obstacle avoidance using mobile 3D laser scanner and GNSS*; Multimed Tools Appl (2017) 76:23017–23039 DOI 10.1007/s11042-016-4211-7;

- Lorenzo Galleani; *Kalman Filter*; Signal processing: methods and algorithms, Lecture 9, 22/11/2017;

- Lorenzo Galleani; *Kalman Filter*; Signal processing: methods and algorithms, Lecture 10, 6/12/2017;