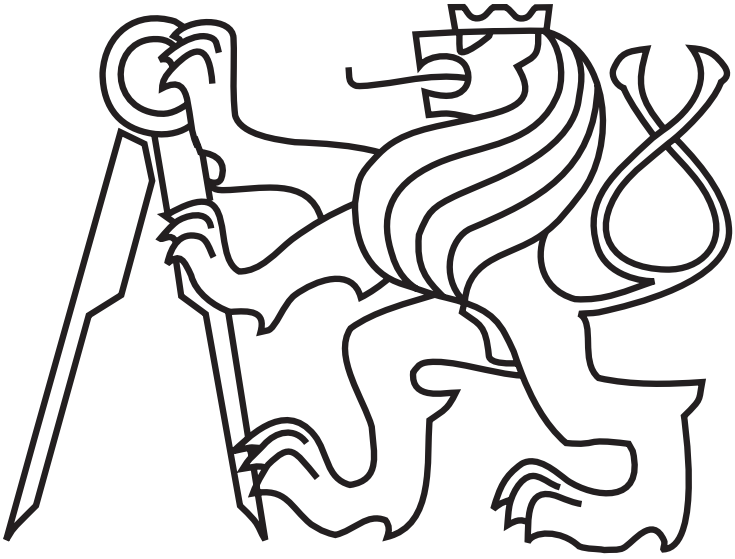
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

DIPLOMA THESIS



Filip Majer

Pedestrian detection in adverse weather conditions for autonomous mobile robots

Department of Control Engineering

Thesis supervisor: **doc. Ing. Tomáš Krajník, PhD**

August, 2020

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Majer Filip** Personal ID number: **425049**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Pedestrian detection in adverse weather conditions for autonomous mobile robots

Master's thesis title in Czech:

Detekce chodců v nepříznivém počasí pro autonomní mobilní roboty

Guidelines:

1) Learn about methods for autonomous navigation and implement a suitable system capable of traversing a given path in adverse conditions using off-the-shelf sensors.
2) Learn about sensors and methods for obstacle detection and their performance in adverse weather conditions.
3) Choose a suitable combination of sensors and methods capable to detect people in adverse weather.
4) Implement a system capable of reliable pedestrian detection in adverse weather conditions.
5) Integrate this system into the autonomous navigation method running on a real robot.
6) Design a set of experiments demonstrating capability of the navigation system to traverse a given path while avoiding collisions with pedestrians in its path in adverse weather conditions.
7) Perform the experiments using a real mobile robot.

Bibliography / sources:

[1] Krajník, Tomáš, et al. "Simple yet stable bearing□only navigation." Journal of Field Robotics 27.5 (2010): 511-533.
[2] Krajník, Tomáš, et al. "Navigation without localisation: reliable teach and repeat based on the convergence theorem." 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.
[3] Yan, Zhi, Tom Duckett, and Nicola Bellotto. "Online learning for human classification in 3d lidar-based tracking." 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017.
[4] Kam, Moshe, Xiaoxun Zhu, and Paul Kalata. "Sensor fusion for mobile robot navigation." Proceedings of the IEEE 85.1 (1997): 108-119.
[5] Barfoot, Timothy D., et al. "Into darkness: Visual navigation based on a lidar-intensity-image pipeline." Robotics Research (2016): 487-504.

Name and workplace of master's thesis supervisor:

doc. Ing. Tomáš Krajník, Ph.D., Artificial Intelligence Center, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **14.01.2020** Deadline for master's thesis submission: **14.08.2020**

Assignment valid until:
by the end of winter semester 2021/2022

doc. Ing. Tomáš Krajník, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Declaration of authorship

I hereby declare that I wrote the presented thesis on my own and that I cited all the used information sources in compliance with the Methodical instructions about the ethical principles for writing an academic thesis.

In Prague, date ....

signature

Acknowledgement

I would like to thank doc. Ing. Tomáš Krajník, Ph.D. for his supervision on this thesis, his excellent leadership throughout my master studies and gained skills in the field of robotics. Members of the Chronorobotics laboratory for their support. Dr Zhi Yan for being my ERASMUS supervisor and passing his skills to me.

My parents, for their endless support, thanks to it, I was able to study and write this thesis, my grandmother, for staying with me during my studies. My friends for their discussions and support when needed.

I would also like to thank the Shameless company for allowing me to have time to rest and think outside the university, moreover for teaching me the mechanical skills. The Pilot pub and the best bartender in Prague Pavel for having a place to go in the brightest and darkest times throughout my studies.

The work was funded by CSF project 17-27006Y STRoLL, CZ MSMT projects FR-8J18FR018.

Abstrakt

Tato práce se zabývá vytvořením spolehlivého systému detekce chodců pro autonomní vozidla, který je schopen fungovat v nepříznivých povětrnostních podmínkách. Systém využívá kombinaci lidaru a radaru, které jsou použity společně za pomocí sensorické fúze pro lepší přesnost detekcí. Systém používá nejen sensorickou fúzy, ale i učení jeden druhého. Systém pro detekci chodců je zkombinován s navigačním systémem, který dokáže autonomně projíždět dříve naučenou trasu. Schopnosti systému, které jsou ukázány v této práci jsou detekce chodců v mlze, schopnost systému se spolehlivě autonomně navigovat za pomocí reálného robotu i v případech, kdy do cesty vstoupí chodec, a nakonec je demonstrována schopnost detekce objektů na vozidlu nasazeném v poloprovozu automobilky.

Abstract

This thesis focuses on creating a reliable pedestrian detection system for autonomous vehicles, which is capable of operating in adverse weather conditions. The system uses a combination of radar-lidar sensors, which using sensor fusion, the system obtains higher precision of the detections. The uses not only sensor fusion, but also for learning one the other. The pedestrian detection pipeline is combined with a navigation system capable of autonomously traversing a previously taught path. The experiments of the system that this work shows, is the detection of a pedestrian under the influence of fog, the capability of the system to reliably autonomously navigate a real robot even when pedestrians are walking in its way, and the capability of object detection on a car-like machine, deployed at a car manufacturing plant.

Contents

List of Figures

1 Introduction

One hundred years ago Karel Čapek published his science fiction play R.U.R (Rossum's Universal Robots). This play was the first where the word robot was used. The word comes from the Czech word "robota" which means forced labour. After that, the robots received the famous three laws of robotics which appeared in the Isaac Asimov book "I, Robot" in the 1950s. Ever since, the topic of creating a universal artificial servant has received broad interest from the general public.

The recent advances in the field of robotics and artificial intelligence are pushing some of the science fiction authors' ideas towards reality and helping to make the robots become part of everyday life. One of the advances is the computational power of onboard robot computers, which allows processing a large amount of sensor data, which improves the robotic perception and situational awareness. The understanding of the environment in which the robots operate comes mainly from onboard cameras, lidars or radars. The autonomous vehicles typically use a combination of these and then with the principles of sensor fusion, they achieve a better understanding of the surrounding environment and other participants of their environment. Another advantage is the ever-decreasing cost of the hardware, which allows broader scientific interest even for not so well funded robotic laboratories or anyone at can home build a simple robot with obstacle detection or line follow algorithms. Lastly, we can observe a lot of the robotic contests which attract media attention, e.g. MBZIRC or DARPA Subterranean Challenge, which has helped with the advancement of knowledge of robotics and artificial intelligence in human society.

The growing number of robots entering human-populated areas comes with safety concerns. This applies also to the topic of autonomous cars, which are deployed in some cities around the world. These cities are typically in climates where it is mainly sunny and consistently sunny throughout the year. In these weather conditions, the cameras provide dense data of the cars, and its range is basically unlimited. The visual information obtained from the camera depends mainly on the weather and illumination conditions. For the autonomous car to operate fully autonomously and in the long-term domain, they need to address the problem of driving in adverse weather conditions, e.g. rain, fog, dust, or snow or they need to address a problem of changing environment, which can be varying illumination (day/night) or seasonal changes (winter/summer). The adverse conditions are still one of the most problematic aspects when it comes to fully autonomous driving.

The main concern of the general public is their safety when it comes to autonomous driving. However, for the safety of pedestrians and other traffic participants the authors of [1] suggest that the accident-per-mile of autonomous cars is lower than the human-driven cars, but every accident always has significant attention in the news. In particular, the crash by Uber cars [2], which happened during the night and Tesla car crashes [3], which happened in foggy weather conditions implies that for autonomous car these conditions are a challenging task.

In this thesis, I would like to investigate the possibility of a pedestrian detection system

being deployed on a real robot, which represents a possible autonomous vehicle driving through an human populated environment while meeting pedestrians on its way. I want to introduce an untraditional combination of sensors being used for pedestrian detection while focusing on the ability to sense and detect pedestrians in adverse weather conditions while driving autonomously.

This thesis is segmented in a way where firstly, I will describe current solutions to robotic navigation and pedestrian detection with a focus on adverse weather conditions and changing environments. Then I will describe the proposed system which combines a navigation system and a pedestrian detection system. Lastly, I will show a series of experiments, where their purpose is to investigate possible usage of the system on a real robot.

2 Autonomous navigation

One of the essential problems related to autonomous driving is the problem of autonomous navigation. When navigating from place to place the mobile robots are dealing with a variety of related problems. These problems typically are - where is the robot known as localization and how to get to the destination, which is known as path planning. The robot localizes itself and plans a path in an environment which can be represented by the map. The creation of maps is a robotic problem called mapping. When both of the autonomous localization and mapping is done at the same time, we refer to this problem as simultaneous localization and mapping (SLAM). For solving these problems the robots use onboard sensors, these can be cameras, lidars, radars. These sensors give the robot representations of the environment in which they operate. One simple classification of the environments is outdoor or indoor. The indoor environments can be buildings or storages, which typically tend not to change over time. The changes in these environments happen because of the people activity, such as opening doors or their movement. The outdoor environments are typical for their significant change in the appearance over time, which is a result of adverse conditions like a daily change, from day to night, or seasonal changes ranging from winter to summer.

This section summarizes different approaches to the problems as mentioned above related to autonomous driving. For this thesis, I will focus on problems of localization, mapping, navigation.

2.1 Mapping

The problem of mapping is creating a spatial model of the robot's environment. The maps can be created by driving autonomously or by teleoperated driving, where a robotic operator drives the robot through the environment manually. To obtain information about the surrounding environment, the robots use sensors, e.g. cameras, lidars, radars, GPS, IMU. However the sensors have some sort of error in their measurements, we call it the *measurement error*. Another error, which the robots encounter is the error of odometry, e.g. the error of motion sensors, that give us the position of a robot from the start [4]. These errors, in addition to the change of the environment, results in the model of a map, which is strongly affected by noise and errors that need to be addressed. There are two basic types of maps: Metric and Topological [5].

2.1.1 Metric map

The first map type is metric, that is used as a representation of geometric primitives in either 2D or 3D space. One of the used representation is called the occupancy grid, which segments the surrounding environment of the robot into evenly spaced-grids. Each

cell of the grid has its coordinates *x,y, (z)* in the coordinate system and has information on whether it is occupied (or the probability of being occupied), not occupied or unknown. The map is created by using the onboard sensors; typically by depth cameras or by range measuring sensors with the combination of the robot motion [6]. The advantage of using grid-maps is their easy creation and representation. Another advantage is the useability of the map across different robots. On the other hand, it requires a more precise position of a robot to be known, which can result in feeding the occupancy grid with a lot of noise.

The problem of occupancy grid is that it contains a large number of not used unoccupied cells, which only uses a lot of memory; thus, the geometric maps were introduced [7, 8]. The geometric maps extract geometric features about the obstacles. Their problem is the difficulty of the extraction. Another possibility is the octomap [9], which divides the space into differently sized octants. Each cell can be divided and based on the importance of the place. The division is higher to obtain a more precise map. The reasoning for these is saving space, compared to the continuous metric maps. The Figure 1 shows examples of metric maps - the occupancy grid and the octomap.

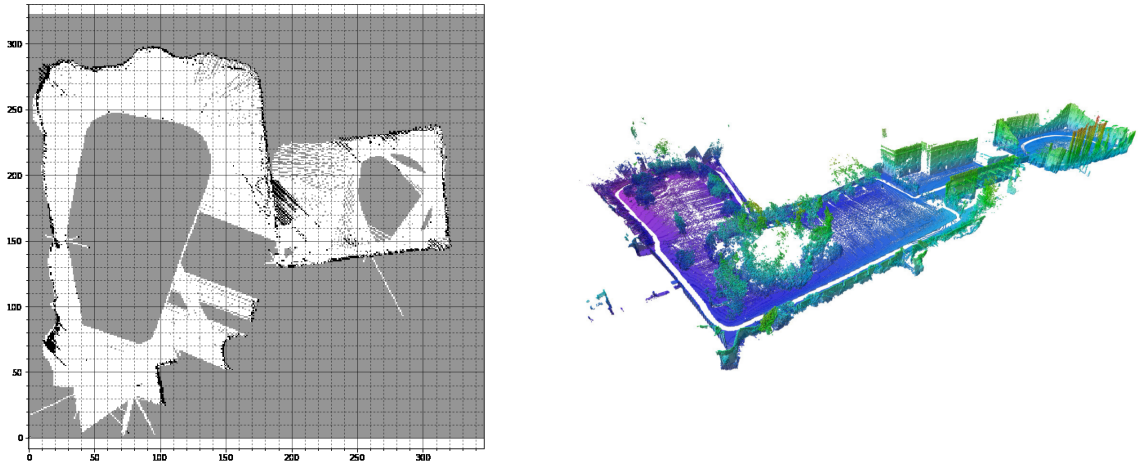


Figure 1: (left) Occupancy grid [10], (right) Octomap

2.1.2 Topological map

The representation of the topological map is a graph in which the nodes are distinct places or landmarks, and the arcs represent the connection (path) between these places. Unlike the metric maps, the position of a robot is represented relative to the specific place. The problem of the topological map is that it is sometimes hard to distinguish the places that appear similar especially if the robot can visit the specific place from multiple paths [11]. Their advantage is they do not need such a precise localization of the robot and the path-planning in them is much simpler. Also, the topological map is more human-readable and understandable in a more natural way, which can be two rooms joined by a corridor; for example this is depicted in Fig. 2.

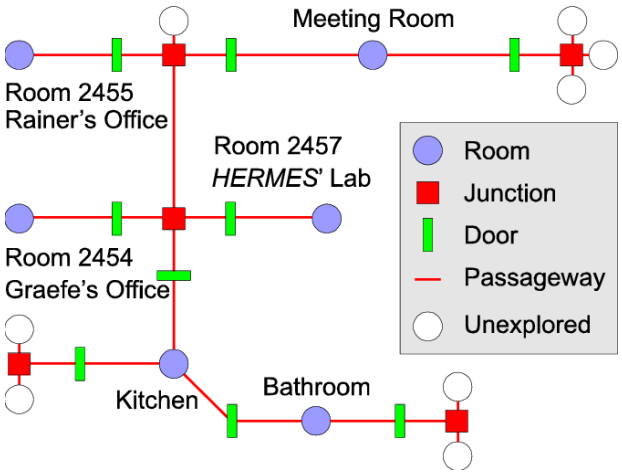


Figure 2: Topological map, where rooms, corridors, doors are represented as a graph [12]

2.1.3 Hybrid map

Hybrid maps combine both of the ideas behind the two mapping processes mentioned above - metric and topological. Typically the metric map is on a lower-level map which represents a particular place where needed, like a more complex structure of a room. Above the detailed structure of a place, there is a higher level of a topological map which joins them. The advantage is that it takes the best of both approaches, e.g. simple planning for the robot between rooms and then a detailed map of a room where the robot should operate as an occupancy grid [6, 13, 14].

2.2 Localization

Localization is the process of determining the current position of a robot. To determine the position of a robot we can use the odometry and onboard sensors. However, as described above, both of these are subject to error. The sensors and their advantages will be described in Sec. 3.

The localization can be either relative or absolute. The relative localization is determining the robot position relative to its starting position. The localization is done continuously. It is based on the previous estimation of the position of the robot. The estimation contains uncertainty, which needs to be handled during the autonomous drive of the robot. This problem is also called the position tracking.

Complete localization is determining the position of a robot in the world coordinate frame with or without the prior knowledge of its previous position. The absolute localization is sometimes referred to as global. For the absolute localization, beacons are typically used, where the robot calculates its position based on the position of the beacons. Beacons can be markers, GPS or active beacons, which transmit signals. The problem is referred to as the

kidnapped-robot problem. This problem is coming from the idea, where the robot operates in an environment, localizes itself, and then we simply take it to an entirely different place in the map, and the robot needs to localize itself again.

The popular approaches to deal with the localization is the Kalman filter or the Particle filter [15, 16].

2.2.1 Kalman filter

The Kalman Filter (KF) is based on the bayesian filters, used for linear systems. The filter takes into account the belief of the actual position, which at a specific time is represented by its mean and covariance. The motion model with added noise determines the next position (state). The correction of the position is updated by the sensor measurements also with the noise. The Kalman filter is optimal but has the biggest constraints when it can be used. The nonlinear extension, the Extended Kalman filter (EKF), uses a linearized version of the system. The improved version of the EKF is the Unscented Kalman filter (UKF), which approximates the probability distribution [17]. The localization using EKF is shown in Figure 3

2.2.2 Particle filter

The particle filter is a method for localization, which is used only for discrete systems. In this method, we use the particles as a hypothesis of the robot position in the map. The particles are a sampled probability distribution of the position. The particles are randomly placed points on the map that have the position of the robot x,y, its heading and weight of the particle. The algorithm works as follows. First is the prediction step, where we update all the particles by the motion model. Then it continues with the measurement step where we compute the particle posterior probability (weight) by the sensor measurement. The method of updating by the sensor measurements is called *ray-tracing*, where we for each particle compute the probability that the sensor measurement happened in that specific point. We then regenerate particles with new weights, and the algorithm continues. The position of a robot can be either the center of the biggest cluster of particles or we can take the particle with the highest weigth [15]. The problem of particle filters is that the position estimation is not precise, and it takes a long time to converge. However, for some of the robotic tasks, a perfect localization is not needed, and the particle filter is good enough. The problem of particle filters is the particle depletion which is when the number of particles cannot properly sample the probability distribution due to the size of the map. The localization using particle filter is depicted in Fig. 3.

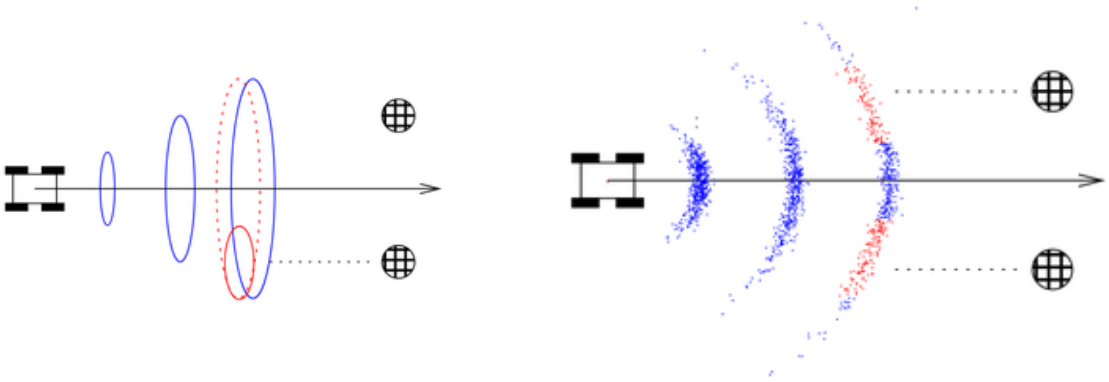


Figure 3: Localization using (left) Extended Kalman Filter , (right) particle filter [18]

2.3 Navigation

The problem of navigation is when the robot is getting from place A to place B following a path. The problem is related to localization, mapping and motion planning. On the way, the robot is capable of avoiding or assessing an unexpected encounter with an object. We can separate the navigation problem into three types: map-based, map-building based, map-less [19].

2.3.1 Navigation without metric maps

For the map-less navigation, the robots learn about the traversable structures in the environment. These can be paths in the park or roads on a highway, which are used by the robot to determine the next motion [20]. The navigation can be also done using the topological maps.

Another approach is to use the so-called *teach-and-repeat* methods, where the human operator teaches the robot the desired path. After the *teaching* phase, the robot simply *repeats* the commands from the operator and drives the desired path. The problem of this method that the stored information about the path is also under the influence of the measurement error. One of the possible solutions is to use the visual teach-and-repeat, where during the teaching phase, the robot stores visual information from the equipped camera about the environment in a way which extracts keypoints from the image. During the *repeat* phase the robot traverses the taught path and replays the stored keypoints and matches them with the currently visible ones [14, 21].

2.3.2 Navigation with metric map building

The navigation with building the maps is referred to as the Simultaneous Localization and Mapping (SLAM) [22, 23]. The SLAM problem is related to the fact that for precise mapping, we need to have precise localization in the environment; however, for precise

localization, we need good maps. The measurement from the sensors, which help the robot to perform SLAM are noisy and therefore, for the approach we use probabilistic methods to lower the noise in the sensor data. More about SLAM methods can be seen in [15]. Sometimes we want the robot to navigate around the whole area, while continuously mapping without having a predefined ending point. This is a problem called exploration. The exploration is using points of interest, which can be frontiers [24] or entropy based [25] and navigate itself using these.

2.3.3 Navigation with metric maps

The navigation in this approach is using maps which are known a priori. The map is done typically by teleoperated drive, where the operator drives through the environment or one can use the known maps for the Global Positioning System (GPS). The self-driving cars use in particular where the car localizes inside the map and then performs navigation. The usage of GPS is limited to outdoor roads. The advantage of navigating in a known map is that it focuses only on localization in this map to successfully navigate itself. The maps have a much better quality and can be manually edited and updated. The manual editation can be, for example, removing points that appear as an obstacle but are noise. An example of navigation with map known a priori is work of [26] where the hand-drawn maps are used for navigation or use CAD drawing for navigating in the environment [27].

2.4 Navigation in adverse conditions

Another division of navigation methods is if they are used in a static or dynamic environment. The dynamic environment, unlike the static and structured environment, has to deal with the changes in the environment, e.g. illumination changes, seasonal changes, weather changes, people movement. These changes inject into the system another type of noise, which has to be to handled when creating maps in which the robot navigates. The goal of the self-driving car if they want to operate in the long-term is to address this problem. For the temporary changes like people movement or a car passing the trend is to remove those by using either some sort of filtering, which can filter all the moving objects. The authors of [28] have done this by removing objects with moving shades. Dealing with the illumination changes one can use navigation by streetlight [29] or where the robot has its illumination own device [21]. One of the approaches to illumination or seasonal changes is to exploit the fact that they are periodic and thus we can predict those changes and use them for navigation [30, 31]. Most of the current approaches use the combination of sensors in a sensor fusion fashion [32, 33], which uses the fact that the lidar is unaffected by the illumination change. The illumination changes can be improved if the robot is using its own illumination device [21, 34].

The problem with the map systems is that the outdoor environment gradually changes over time, which makes the mapped environment unreliable to the point at which the robot

does not recognize any of the stored places. For this thesis, the focus is on outdoor robots that operate in the changing environments in the long-term scenarios. The focus here is on the teach-and-repeat navigation systems, which are more robust to the environmental changes where especially one can adapt to the changes [35, 31].

3 Sensors in autonomous driving

Sensors give the autonomous vehicles the ability to sense the surrounding environment and its own states. They give information about what is happening around and inside the robot. The robots typically have multiple sensors, where one can combine the information obtained through *sensor fusion* [36]. The sensors can have limitations to their range and resolution, which comes from the physical laws they are based on. The limitations can be, for example, a camera that cannot see through the walls, but the radars can see through them, or the reflectivity of the material, which the sensor is detecting.

This section describes the sensors which are used for the proposed pedestrian detection system in Sec. 5, and the navigation system in Sec. 4. I will focus on their principles and the effects of environmental changes and the effects of the adverse weather conditions. Moreover, I will describe other widely used sensors for autonomous driving, which are mentioned in the previous Section 2.

3.1 Interoceptive sensors

The sensors that give robot information about the inner states are interoceptive. The first sensor is the wheel encoder which outputs the rotation of a motor of a robot. The wheel encoders are used for the odometry. Odometry is a method which uses information about the motion of a robot by to determine its position. To obtain the position of the robot, we need to integrate the obtained velocity, which results in a linear error. The linear error is accumulating with the further distance travelled. This means that the odometry is not suitable for long-term autonomy. The error of odometry is also subject to a surface slippage or by the wheel material and the tire inflation. Best usage is for indoor robots where it robot operates for a shorter period of time.

Another interoceptive sensor is the Inertial Measurement Unit (IMU). This sensor typically combines an acceleration sensor, gyroscope, magnetometer or pressure sensors. The acceleration sensor outputs the acceleration across three axes. The gyroscope measures orientation in space. In mobile robotics, they are typically used together with GPS to correct their error. The IMU can also be used for collision detection, and if a car hits something, it will be easily detectable by the IMU. An example of the IMU sensor is shown in Figure 4.

Lastly, the Global Positioning System (GPS) is a sensor which is used for outdoor robotics. The GPS takes information from satellites around the Earth and uses this to determine its position in the global frame. However, it does provide the robot with a global position, which can be off by a couple meters, which makes it an unreliable sensor for precise localization. The problematic part can be in the city where the GPS is affected by surrounding buildings. A GPS sensor, which can be equipped to the autonomous vehicle, can be seen in Figure 4.

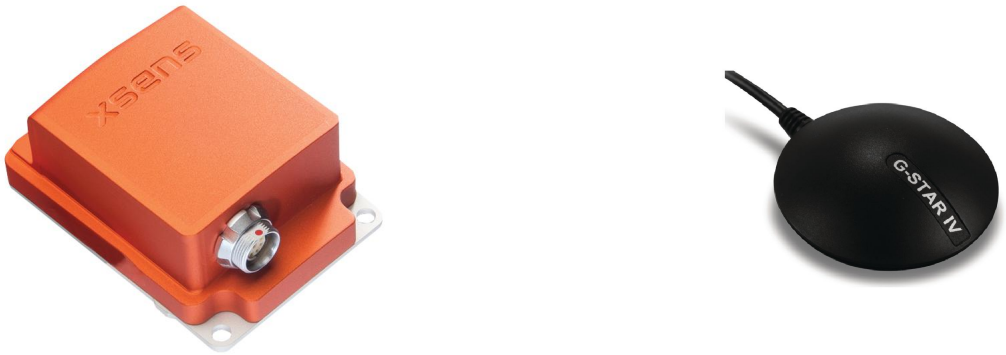


Figure 4: Example of sensors - (left) IMU XSens MTI-30-2A8G4-ND (right) GPS Module BU-353S4

3.2 Camera

Cameras became a favourable sensor to use for mobile robotics and its well-developed field of study. Their advantage is that simple monocular cameras are not expensive and they provide the robot with dense data. They have low spatial dimensions, which means they are suitable for a variety of robots and they are lightweight, so even autonomous vehicles or UAVs that have small payloads can carry them.

Cameras are composed of lenses and a shutter through which the outside light rays fall onto the light-sensitive digital sensor, which converts the light into data. The time of the opened shutter is known as exposure, which correspond to the amount of light that is exposed to the sensor. The effect of exposure on the ability of the autonomous navigation is described in [35].

Cameras used in mobile robotics vary from monocular cameras, stereo cameras from which the object position in 3D can be computed, RGB-D cameras, which have their own source of light that serves for obtaining depth images, and event-based cameras, which react to the change of brightness instead of capturing images. Another type of camera is the infra-red (IR) camera which uses the infrared light (heat) as a source. However, these are very expensive and have a low resolution. The higher their resolution is, the more expensive they are. Some of the cameras use near-infrared light, which is used for illuminating the space,e.g. surveillance cameras. Last type is the multispectral camera which is widely used in agriculture [37]. The widely used cameras for mobile robots can be seen in Figure 5.

3.2.1 Effects of adverse conditions on cameras

The camera depends on the surrounding light conditions, which makes them hard to use during the night or even throughout the day where the shades and intensity cause the environment to change dramatically even for a shorter period of time which can be a

Figure 5: Different types of cameras - (a) TARA Stereo vision camera, (b) Intel realsense D345 [38]

matter of hours or even minutes. The intensity of the sun can be problematic when the sun directly shines into the camera, which causes over-illumination, meaning that the camera image is too bright. Another problem of cameras is that they need a medium through which the light can travel. This medium has to be transparent and not translucent for the cameras. Also, the transparency of the medium should not change dramatically in shorter periods which can be caused by hot air on the surface of the road or by heavy rain. In other words, if we take dense fog where the light is dispersed and diffused, the camera will not capture any data of sufficient quality.

3.3 Lidars

The light detecting and ranging sensors (lidar) is one of the widely emerging sensors which are used to obtain in either 2D or 3D space distance measurements. They can provide the robot with up to 360° field of view, where most of the commonly used lidars have a lower field of view. Based on their resolution, the robot receives for each of the angles the distance to the object. The distance the lidar measures can be up to 300 m for the best lidars. Unlike the stereo cameras, which can correctly measure distance defined by the width of both lenses, their range is much lower than the lidar. The 3D lidars have multiple layers placed one top of the other, which takes the scan around the sensor but with various heights. Their most significant disadvantage is the price, which is the most expensive part of an autonomous vehicle when using a multilayer 3D lidar. One of a widely used brand of lidars together with data collected by this sensor can be seen in Fig. 6.

The sensor is active, which means it emits near-infrared light (laser beams) and then the sensor measures the time of flight of the specific beam and converts it to the distance to the object once reflected. An essential part of the reflection is that the material is reflective to the laser beams. Otherwise, the sensor does not recognize the object, which results in losing the data. However, the object cannot be absolutely reflective, which causes to lidar to measure the distance to the material plus the distance to the objects it reflected the

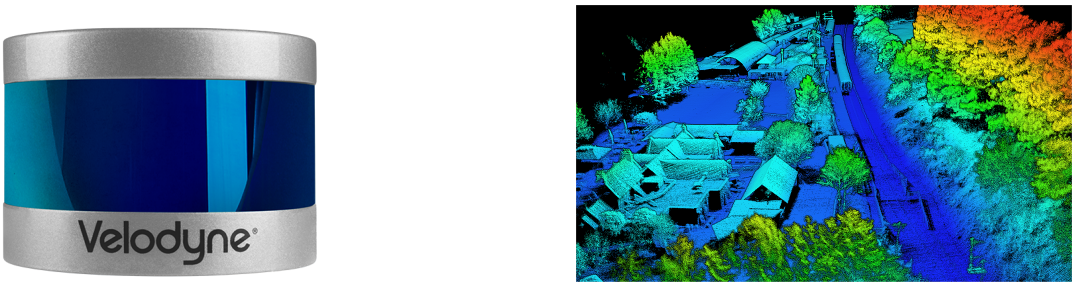


Figure 6: Example of lidar sensor the Velodyne Puck (a) and an example of data collected (b) [39]

beam to. The problematic medium is for example glass which for most of the cases needs to be modified so the infrared light can go through.

In mobile robotics, they are widely used mainly for localization, mapping, object detection, or for safety reasons. They are practical because unlike cameras, where the information about the environment is obtained from the direction the camera is facing, it receives information all around the robot, the batch size of the data can be much larger also the data is different. Contrary to cameras, they are much harder to train for the traditional neural network due to the sparsity of points [40].

3.3.1 Effects of the adverse conditions on lidars

The advantage of lidar is that they are less affected by the changing light conditions, which means that the data received from the sensor is the same during the day or during the night. They are affected by the outside heat or density of the rain. The operational ranges are typically provided by the company creating the sensors. The disadvantage of them are that they are strongly affected by the weather, e.g. the fog disperses the beam, and the data are lost [41]. With the lidar being usually used as a safety sensor, the possible error in data caused by these conditions is critical, where either it can hit a pedestrian, or it can spontaneously stop even when there is not an object in front.

3.4 Radar

Radio Detection and Ranging (Radar) sensors being used for the autonomous vehicle is not a new topic, authors of [42, 43] are showing the radar being used as a safety sensor for

obstacle detection. As the abbreviation suggests, the principle of the radar is similar to the lidar. Where for the radar instead of light beams the radio wavelength signal is emmited via the antenna, meaning that a different part of the electromagnetic spectrum is used. Once the radio signal reflects from a material, the waves are reflected and then returned to the receiving antenna. The returned signal strength in influenced by the conductivity of the material. There are two technologies that simplified using the radars for autonomous driving: The Synthetic-Aperture (SAR) which is no moving parts on the radar and Ultra WideBand (UWB) that uses wide spectrum.

The sensors can vary with the frequency they are using. The autonomous cars, the most common radars have a frequency around 77 GHz, which corresponds to the milimeter wavelength. They can provide information such as distance and velocity of the detected objects. The radars can be either 2D or 3D. The field of view varies based on the company and can be up to 360°. Sample images of radar used for an autonomous cars can be seen in Fig 7. The data can be either points in 2D space or sparse pointclouds in the 3D space.

Figure 7: Example of radar sensor - SICK RMS320

3.4.1 Effects of adverse conditions on radars

The most significant advantage of these sensors is that they are a lot less affected by weather changes. They are not affected by snow, rain, dust, extreme weather conditions or different lighting conditions, resulting in their suitability for day and night. Unlike the cameras, they can have water drops or dirt on them and still function. A comparison of the effects of adverse weather conditions can be seen in Fig 8 where a person is hard to detect by the lidar, where the typical U-shape for the legs in the 2D lidar data disappears entirely in the simulated foggy conditions.

3.5 Sensor fusion

Sensor fusion is a method of combining data from multiple sensors, which results in having better sensing abilities of the whole system. The advantage of a sensor fusion system

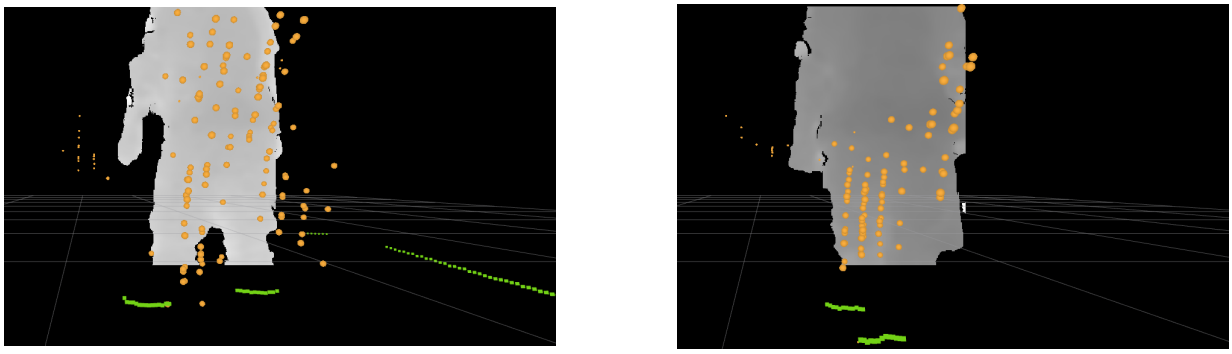


Figure 8: The effect of adverse weather conditions on different sensors in a sample scene, where a person and a wall can be observed. The Figure (left) shows normal conditions, where the orange dots represent the radar pointcloud, green dots represent a 2D lidar scan. The Figure (right) shows the sensor data when under the effect of smoke. Note the dissappearance of the wall.

is a robust and reliable system, where in the case that one sensor fails entirely or is damaged, the other can still perform the task. Their combination can be used to have a better view around the robot when the field of view of the sensors is limited. Another advantage is the increase of confidence in the detections, which can help in having much more precise detections of surrounding objects [36].

There are many different ways to divide sensor fusion methods. Based on the level of the fusion, which can be low-level meaning combining their data to produce new data; intermediate-level, which combines features of the data like positions or edges and high-level fusion which combines decisions of the algorithms. Another division is based on the sensor configuration, where the first one is the complementary configuration. This configuration is to use the same sensors to have a better view on the overall scene, like multiple cameras creating a 360° field of view. The second configuration is competitive, where each sensor delivers measurements of the same object, and the final decision is based on the voting scheme. The last sensor configuration is the cooperative configuration, where two sensors produce information otherwise not obtainable by a single sensor. An example of a cooperative configuration can be a stereo camera, which combines two monocular cameras together with the knowledge about the width of both of the lenses.

Sensor fusion allows the robot to have multiple options when it comes to adverse weather conditions. Some sensors are useful to use in different weather conditions. Due to the higher precision, the lidars are preferable for object detection during day and night, and cameras outperform the lidar during the day, however both of them struggle with adverse weather conditions where it is better to use radar.

4 Navigation system

The chosen navigation system in this thesis is called the Bearing-only Navigation (BearNav). The bearing-only stands for using bearing-only sensors, which are those that give the robot a relative orientation of the objects it sees, e.g. cameras. The BearNav method comes from the work [14] and its following work [21].

This method is based on the *teach-and-repeat* system, where the robot does not need to localize itself but can reliably navigate itself in the traversed environment. The *teach-and-repeat* methods were chosen because it is being reported that they are more robust at dealing with the adverse conditions [44]. The robustness makes the method suitable for use in long-term scenarios. To achieve higher reliability authors of [45, 46] suggest using trainable image descriptors or using adaptive techniques [35, 31].

Another factor of this method is its simple implementation for various autonomous vehicles [47], it is computationally efficient, and it does not require expensive equipment. The proposed system uses only a monocular camera, which does not require calibration and odometry for its functioning. It is also suitable for various outdoor or indoor environments [21]. All of the mentioned advantages makes it a suitable navigation system for outdoor robots operating in different scenarios under various conditions.

The *teach-and-repeat* methods works in two steps, firstly in the teaching phase the robot is taught a path to traverse by the human operator, it is analogous to the industrial robotics where the operator moves the robotic arm to perform welding and then in the repeat phase the robot follows the taught action. The repeated process is not the same because of the noise in the sensors; however, in industrial robotics, the errors are minimal because it is an entirely controlled environment, free from external processes. The used method in this work deals with the noise in a way which is described [14, 21, 18].

4.1 Teaching phase

In the *teach* phase of the navigation system, the operator drives the robot in the environment. The commands are done by teleoperated driving by using some controller where the robot traverses a path like in Fig. 9 on its way, the robot stores images at regular intervals. From those images, it extracts the image features. The images were taken on a specific location, which is denoted by the red dots in Fig. 9, which is marked by the travelled distance from the start. The odometry measures the distance from the start. Thus we have an image with its extracted features and the known distance at which this image was taken. These images at a particular distance are referred to as local maps. The blue line represents the taught path, which the robot traverses. This taught path is referred to as the *path profile*.

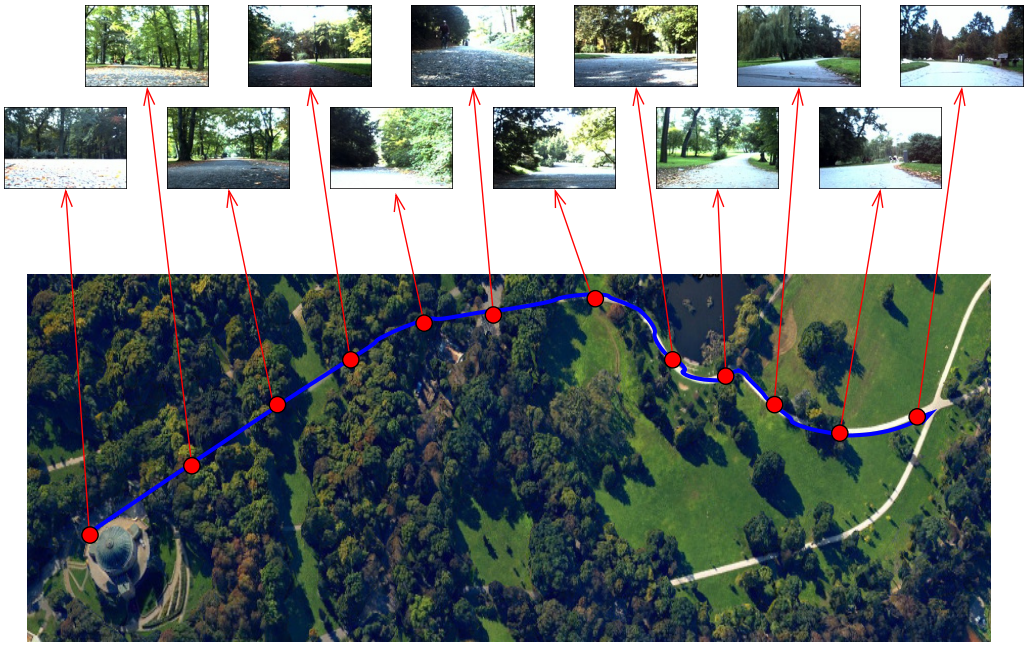


Figure 9: The taught path (blue line) and local maps (red dots) in the form of images [47]

4.1.1 Local maps

The robot on its way stores images in the form of local maps. The local maps serve as a place which the robot needs to recognize every time it travels. The information obtained from those images is called feature extraction. The feature extraction works in two steps - detection and description. The detection part locates and identifies *keypoints*. These keypoints are detected so that they can be observed from various angles, and these typically include points with high contrast or edges. This can be done by using Hessian matrix [48] or by comparing the brightness of pixels [49]. Once the keypoints are detected, their surroundings are described by the *feature descriptor*. The description can be, for example, the vector which stores information about the brightness comparison between specific pixels around *keypoints*.

For the method, the local map is the saved keypoints with their descriptors saved at a distance from the starting point. This method combines the SURF [48] descriptors and the AGAST [49] detectors for extraction. Note that the system is reconfigurable meaning that any combination of these features can be used.

4.1.2 Path profile

The *path profile* is created through the commands of the operator. At the begging the distance travelled is set to zero, once the operator gives the robot forward or angular velocity this information is saved. At every distance where the operator changes those velocities to perform a turn, the profile stores the distance from the start at which this change happened and the value of the velocity. Thus at the end, the path profile has all the velocities (forward, angular) that were applied by the operator at specific distances from the starting position.

4.2 Repeat phase

Once the *teaching phase* of the method is done, the operator tells the robot to repeat the path, and the robot starts to navigate autonomously. At the beginning, the robot pre-loads all of the stored maps together with their distance and loads the path profile. The distance travelled by the robot is set to zero, and the map at distance zero is loaded.

The robot loads the first command from the path profile and moves forward. The loaded commands adjust the movement on its way based on the travelled distance from the starting point. The measurement of the travelled distance is subject to noise from the odometry, which is increased based on the surface. Thus the path profile itself is not sufficient for navigation, because of these reasons the robot moves slowly away from the original path.

To suppress the error, [14, 21] suggests that the correction of the heading is enough to compensate for the odometric errors. For the correction of the heading, the robot uses the stored local maps. Once the robot passes the threshold distance of the specific map it loads all the features from this map.

The current view from the camera, where the robot again performs feature extraction, is then matched against the currently loaded local map. The *feature matching* is a method to match features from the two images. The robot uses the matched features to determine its correct heading by histogram voting. For every pair matched, we obtain the difference in the horizontal axis of those points. The histogram voting takes the most frequent difference between the features and steers the robot so that the difference in the horizontal axis is zero. Using the histogram voting the robot steers to the original taught path. An example of the steering influence is in Figure 10.

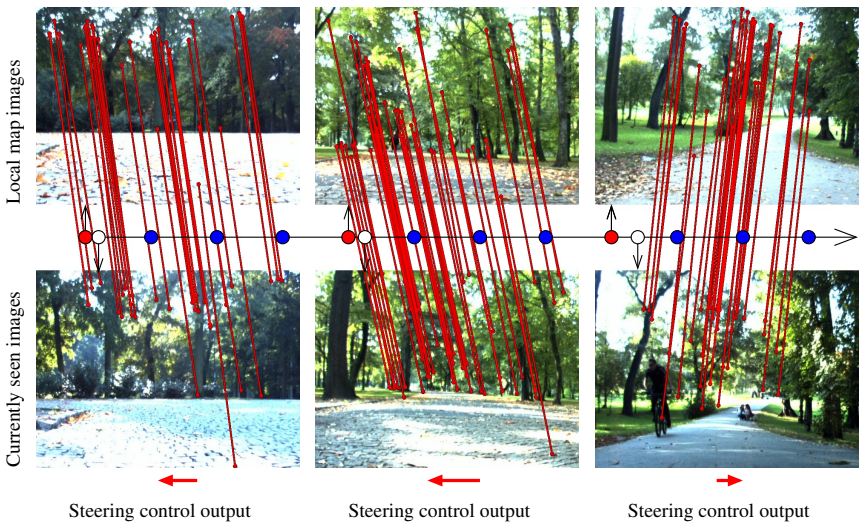


Figure 10: The Figure shows how the correction of the heading is performed, the loaded local map (white dot) and the current image view (red dot) both have their image features extracted, here we see the result of histogram voting, the most frequent horizontal displacement, which steers the robot toward zero (red arrows). The blue dots represent following local map on the traversed path. [47]

4.3 System implementation

The system is implemented in the Robotic Operating System (ROS) [50] version Melodic. The prosed system is shown in Figure 11.

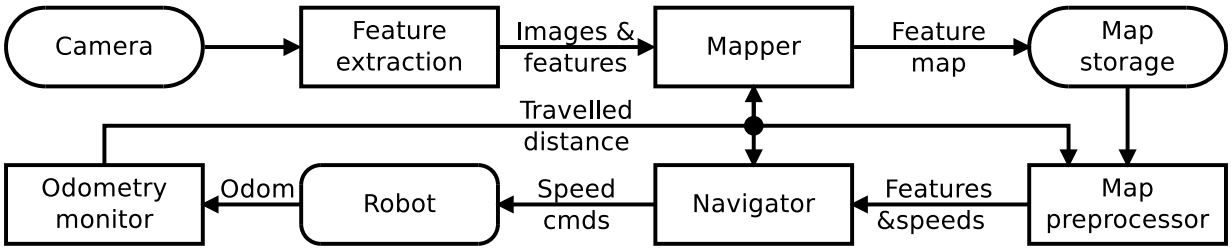


Figure 11: Navigation system in ROS [47]

The navigation system works as follows: The *odometry monitor* node obtains data from the odometry of the robot and transfers this into the travelled distance. The *feature extraction* node extracts features from the equipped camera of a robot and sends them to the *mapper* node, which saves the image and its features together with the travelled distance tag provided by *odometry monitor* node at regular intervals. The *mapper* node also stores the information about the path profile; after the teaching phase, the obtained data is stored in the robot. For the navigation, the robot loads all the images with the features

and the path profile by using the *map preprocessor*. The *map preprocessor* sends the data to the *navigator* based on the distance provided by the *odometry monitor*. The *navigator* node reads the information from the *map preprocessor* and performs feature matching and histogram voting with the current features from the camera from the *mapper* node. The robot then follows the commands from the *navigator* node.

5 People detection

There is a strong need for autonomous vehicles and especially self-driving cars to know about all the other participants of the traffic [51]. In particular to know about the most vulnerable participants such as pedestrians or cyclists.

The need for pedestrian detection is a topic which typically uses the lidars or camera or their fusion. The authors of [52] and its extension [53] proposed a pedestrian detection system while using four layers of laser scans, where they fuse the information from all of them. Zhi et al. proposed a system for human detection used in an online learning fashion [54]. Another approach is to combine depth images and lidar data [55].

However, one of the challenges for people detection is detection in adverse weather conditions. The section 3 describes the effect of the adverse weather conditions on the sensors. Most of the works, as mentioned above, do not focus on this problem. The authors of [56] are using an updated version of the popular YOLOv3 [57], used for object detection from cameras, where they used images from a thermal camera. The authors of [58, 59] are following the idea of the combination of sensors to enhance the ability of object detection.

The combination of sensors used for the same purpose, e.g. detecting the same object or used for navigation is typical for autonomous vehicles. The idea is to combine sensors which are better in some scenarios but have problems in others. The widely used cameras provide an autonomous vehicle with dense data. However, they are affected by illumination and the weather. The lidars are unaffected by the illumination, but they are affected by the weather conditions, e.g. thick fog or rain. Another factor is their representation and understandability for people, where the camera and lidar give the user straightforward data, which are intuitive. Contrary to these two sensors, the radar is less affected by both weather conditions and varying illumination, but they provide sparse and noisy data. The understandability of the radar data is a problem due to the noisiness and low amount of data coming from them. The amount of data and their understandability is critical when it comes to training and classifying objects.

The proposed system combines best of the both worlds - using already trained and shown to be suitable pedestrian detection algorithm and using it for training otherwise difficult to train sensor data from the radar. Unlike the popular methods for deep learning which require a large amount of data where those data have to be manually labelled, which is a long and tedious job to do, the proposed system goes a different way by automatically labelling the data and then performing training and classification based those labels.

This section describes the proposed system for automatic labelling, online learning and finally the sensor fusion, which comes from combining the detections from radar and lidar.

5.1 System description

The proposed system consists of four processing modules (see Fig. 12). This section goes through each of those modules and describes them.

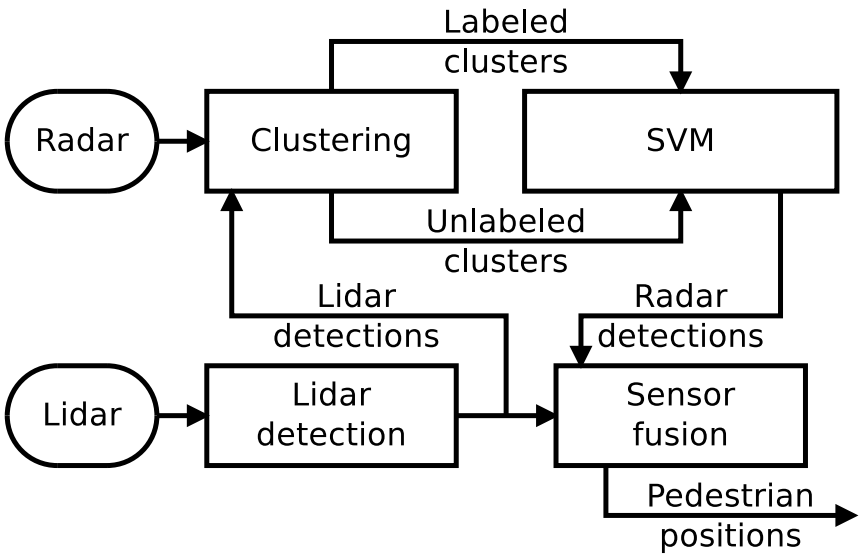


Figure 12: A diagram of the system. Lidar data in the form of pointclouds are sent for pedestrian detection. The radar sends the data in the form of pointclouds also. This pointcloud is clustered, and the clusters are labelled based on their relative position to the lidar pedestrian detections. The labelled clusters are sent to an SVM, which is trained based on the clusters, and outputs its detection to the sensor fusion module. The sensor fusion combines the position from the lidar and radar detections and estimates the final position of pedestrians.

5.1.1 Lidar detection module

The *lidar detection* module takes the input from the lidar sensor, which can be either a 2D or 3D scan. As mentioned above the proposed system uses an already developed leg detector [60] and uses a 2D lidar. This leg detector is widely used and simply accessible in ROS as a package. The leg detector uses AdaBoost as the chosen classifier. The algorithm searches for the typical U-shape of the legs in the lidar scan. For training the AdaBoost, the authors created 14 different features, which includes - number of points, linearity, circularity, radius, mean speed, mean curvature, width. The features are created from the input scan, where the scan is segmented. Each segment consists of points which represent the possible leg belonging to a pedestrian. The module output position of the pedestrian in the x,y coordinate system, moreover it also outputs the covariance matrix, reliability and the timestamp of the detection.

5.1.2 Clustering module

The *clustering* module has two inputs a radar scan and the output of the *pedestrian detection* module. The radar scan is output from the radar sensor, which is a 3D pointcloud.

The poincloud is firstly segmented into different clusters using the Euclidean clustering from the Point Cloud Library(PCL) [61]. The Euclidean clustering is a method which segments the points into clusters, where each point can be only part of one cluster. A point is put into a cluster if it lies within the distance threshold, which is variable. Then if the point is far from all the previous clusters, the algorithm creates a new cluster with only one point and continues creating clusters until all points are used. Once the module finishes the clustering, it proceeds with labelling the data. The module received an estimation of the position of the pedestrian from the *pedestrian detection* module, which gives a position in the x,y coordinates and uses these positions to search for clusters that are in the vicinity of this position, it also takes into account the reliability of the detection r. It compares the distance between the centroid of the cluster and the detected position. If the distance is lower than a certain threshold (different from the threshold used for clustering), then the particular cluster is labelled as a pedestrian if the reliability of the leg detector detection is over 0.7. The rest of the clusters are labelled as other.

Since the aim of the system and this thesis is to apply the radar detection on a real robot, which would operate for extended periods of time implies that the robot should operate over a day or possibly months. With that in mind, the clustering module uses the ideas of life-long learning, which means that the robot learns throughout the entire lifetime of its operation, the same as human counterparts [62]. The ability to continuously learn about the appearance of the pedestrians is useful for the adaptation of a robot to various situations. Also, the idea is that the robot does not rely on the immediate a priori known models but can learn continuously. Thus the module also has implemented the ideas of life-long learning. The clusters can be labelled as pedestrian only if it has received continuous detections of a pedestrian for couple of seconds, and the interval between them is lower than a specific time, 1 second is being used. When using this feature the system can be more confident about the incoming detections being true, and also if the pedestrian is not clear enough or is seen only for a fraction of time, it is simply not used as the output cluster which is used for training. Those clusters are labelled as unknown. Also if we are not receiving any detections from the *pedestrian detection*, module all the clusters are labelled as unknown.

Once the module is done with the labelling, it outputs the results. The output is composed of all the points in the respective cluster, its label (pedestrian, other, unknown), reliability, covariance matrix, and timestamp.

5.1.3 Support Vector Machine (SVM) module

The purpose of the *SVM* module is to train and classify the incoming radar clusters. The chosen classifier of an SVM is suitable for its excellent performance on small training sets and the required time for training small datasets is small. Unlike the recent popular deep learning classifiers, which require much data, and the training lasts much longer than the SVM.

Additionally, the idea is to have this done using online learning so that we can use the principles of life-long learning. Unlike the traditional offline learning, where the classifier has access to the whole training data set, and trains itself and the model it outputs does not change over time or with new data information coming. The online learning does the opposite, where at the beginning, the classifier knows only the classes to be classified, and is continuously trained with an incoming batch of data. The batch of data is split between the training rounds. For each training round the module gathers the labelled clusters from the *clustering* module for each of the known classes (pedestrian, other), which are equal. In the proposed setup, the size for each class is set to 10. Thus, every time the *SVM* module obtains ten samples of a pedestrian cluster and ten samples of the other cluster, it starts the training round. Note that for the next training round, it uses all of the previous samples plus those newly obtained. The online learning is using the new data, which helps the robot to adapt to the new data, which, due to the long-term deployment of the autonomous vehicles in the real-world environment, can be outdated.

Another used technique for the training and classification is using the transfer learning [63]. Transfer learning, in general, uses two learning tasks and two domains. The learning tasks are called the source and the target task. The idea is to use the results from the source task, and then use the output of this to improve the learning of the target learning task (the SVM). This results in the source task transfering its knowledge to the target task. For the case of this work, the source learning task is the lidar-based leg detector, which is trained offline, and its domain is the lidar data. The target task is the radar-based pedestrian classifier with its domain, the radar data. This classifier is trained from the knowledge of the leg detector. The transfer learning allows to avoid labelling the radar data, which is hard to interpret for the human eye and also is very noisy. Both of these factors make the individual labelling of the clusters for the SVM a hard task. Combining the two principles of online learning and the transfer learning brings the robot towards more robust and higher ability to operate long-termly.

The SVM itself is using the features which were originally developed for 3D lidar. This can be done because the radar and the 3D lidar output pointclouds which have similar properties. The classifier is using 7 different features, which are shown in Table 1. The 7 features have together 62 dimensions.

The features, which were introduced for the 3D lidar are, $(f_1, \ldots, f_4)$ from the work of [65], *Kidono et. al* [66] introduced the usage of f_5 and f_6. The last feature f_7 is a feature which was introduced in [64] specifically for the radar.

The SVM is using the Gaussian Radial Basis Function kernel [67]. The SVM classifier is trained to clasify two classes - pedestrian or other.

From the *clustering* module the SVM also obtains clusters which are labelled as unknown, which means that these are not being used for training but only for classifying.

Feature	Description	Dimension
f_1	Number of points included in the cluster	1
f_2	Minimum cluster distance from the sensor	1
f_3	3D covariance matrix of the cluster	6
f_4	Normalized moment of inertia tensor	6
f_5	Slice feature for the cluster	20
f_6	Reflection intensity's distribution (mean, standard dev. and normalized 1D histogram)	27
f_7	Average velocity of all points in the cluster	1

Table 1: Features for radar-based human classification [64]

5.1.4 Sensor fusion module

The *sensor fusion* module takes the outputs of both the *SVM* and the *lidar detection* modules and estimates the final position of the detected pedestrian. The need for having a module like this is related to the fact that when combining the two sensors, the final estimation from both is better than when the detections run separately.

The sensors are placed on different places on a robot, which means that firstly they need to be aligned in the same coordinate frame. Otherwise, the detections would differ by their distance. A separate calibration achieved the obtained transformation, which using the iterative closest point (ICP) and outputs of both modules for detections, the *lidar detection* and *SVM*. At first, the transformation between the sensors is set to be zero, and then it proceeds with taking the output positions of the system and from that using the ICP the transformation from one sensor frame to another is obtained.

The *sensor fusion* module uses all of the outputs provided by both of the detection modules, e.g. the pedestrian position, the reliability of the detection, reliability, covariance matrix and the timestamp of the detection. When running the detectors and observing their covariance matrices pointed to the idea of the position uncertainty being a sphere (3D radar/3D lidar) or a circle (2D lidar). The covariance matrix has dimensions of 3×3 for the radar detections and 2×2 for the 2D lidar detections. Having a spherical or circular covariance matrix can be simplified as $\mathbf{C}_{r,l} = \sigma_{r,l}^2 \mathbf{I}$, where $\mathbf{C}_{r,l}$ is the covariance matrix for radar or lidar detections, $\sigma_{r,l}^2$ is their variance and $\mathbf{I}$ is the unit matrix. With that in mind,

the variance for the respective detection is the only number to characterize the uncertainty of the position estimation.

For the fusion, two filters were implemented - *weighted* and *switching* filters. The *weighted* filter fuses the position estimations provided by both the *lidar* and *SVM* modules, and uses the weighted uncertainties of the particular position estimations. The *switching* filter uses the position estimation but instead of weighting the estimation simply outputs the position with the lower uncertainty.

For both filters, the idea is that it is expected that the pedestrian detection is not continuous, which means that for a certain amount of time the *sensor fusion* module obtains the position of the pedestrian which comes from the most recent data from the sensor, but there will be gaps in these detections. These gaps can happen thanks to the pedestrian walking out of the field-of-view of the sensors, the classification fails, or in the case of lidar, it is under the effect of adverse weather conditions. These lower or completely shuts down the ability to detect. The filters are designed so that if the detections are old, which can be simply checked by using the timestamp of the detection, the detection is given less weight. With that in mind the inverted variance (precision) $g_{r,l}$ is computed using the following equation:

$$\begin{aligned} g^r &= (\sigma_r^2\, e^{t-t_r})^{-1} \\ g^l &= (\sigma_l^2\, e^{t-t_l})^{-1} \end{aligned} \tag{1}$$

Where t corresponds to the current time, $t_{r,l}$ corresponds to the most recent timestamp of the detection and $\sigma_{r,l}^2$ is the variance of the detection. Here, the important part is the $e^{t-t_{r,l}}$, which inflates the covariances over time. Thus the older detections are penalized, making the system depend on more recent observations or older, which have lower uncertainty. The module working with the *weighted* filter has the final position estimation determined by the equation:

$$\mathbf{x} = \frac{g^r\, \mathbf{x}^r + g^l\, \mathbf{x}^l}{g^r + g^l} \tag{2}$$

Which is the position estimation provided by the *SVM* module for radar detections and the *lidar* detections multiplied by the weighted inverted variances. For the *switching* filter the final position estimation is done by the equation:

$$\begin{aligned} \mathbf{x} &= \mathbf{x}^r \;\; if \;\; g^r \geq g^l \\ \mathbf{x} &= \mathbf{x}^l \;\; if \;\; g^r < g^l \end{aligned} \tag{3}$$

The *sensor fusion* module has two filters implemented, which both provide the final position estimation of the pedestrian, when combining the radar-lidar system. The inflation helps the system to be less reactive but more robust because it is exponential, which means that the inflation is slower when the detection modules stops sending pedestrian positions but gradually is getting higher with time. It results in old data having a low impact on the final pedestrian position.

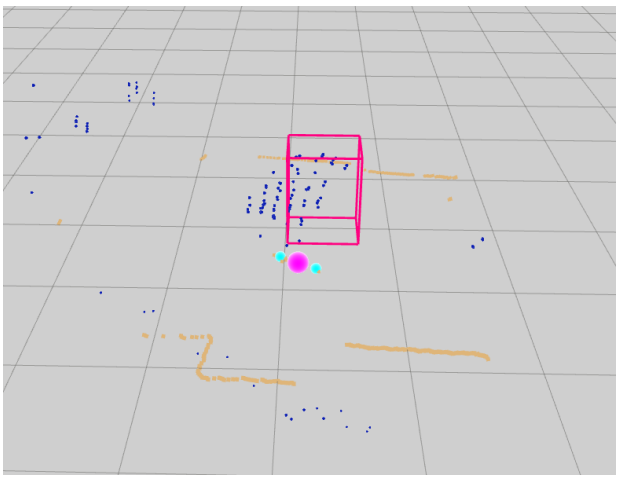


Figure 13: Sample image from RVIZ during the detections obtained from both radar and lidar data, while having a person walking in front of the sensors. The orange dots represent the incoming lidar data, where the wall can be recognized from each side. The cyan spheres are the detected legs from the leg detector, and the magenta sphere is the estimated position of the person from the lidar. Blue dots are representing the radar data. The pink bounding box represents all of the points classified as a pedestrian from the SVM. Note that the bounding box is off by one frame due to the visualization having being processed.

5.2 System implementation and summary

The system was implemented in the ROS [50] framework, which allows for simple reuse of the system. The system is implemented in the nodes and topics following the structure of the diagram 12. The system has one node for the *clustering*, which takes the incoming radar data, clusters them by using the PCL library and labels based on the relative position of the pedestrian position provided by the *lidar detection*. The *lidar detection* is a module based on the leg filter, which is not an implemented node but only a package in ROS. The *SVM* uses the LIBSVM library [68], which trains the classifier by using the labelled clusters and then classifies the unlabelled clusters. Both detectors output the pedestrian position, which is combined in the *sensor fusion*, which gives the whole framework's final position estimate from the combination of the radar-lidar sensors. The final positions are output from the two filters implemented - switching and weighted. Figure 13 show sample detections from the ROS framework, namely the RVIZ (ROS 3D Robot Visualizer) [69], which is used to display any sort of sensory data or outputs of its processing.

The described system in this section uses ideas of transfer learning to support the training of the radar, where the labels from the leg detector are used to train the SVM via the clustering. The sensor fusion used for this combination allows the system to either be more precise under normal conditions or rely more on the radar detections during the adverse weather conditions, where the radar is less affected than the lidar. The lidar

detector used is well developed and will have a lower error in position estimation. However, suppose the autonomous vehicle enters fog. In that case, the detections will either not appear at all, which makes the radar the only sensor to determine any possible detections or the lidar detection will have a much lower reliability and accuracy of detections, which also makes the radar immediately the favourable sensor from the sensor fusion pipeline.

6 Experiments

The experimental evaluation is aimed at evaluating the accuracy of pedestrian detection by using the radar-lidar combination. The second experiment is about the implementation of the pedestrian detections framework onto the real robot, while navigating using the proposed navigation system from Sec. 4. The last experiment was designed to show the ability of the radar to perform potential collision detections of other objects on a real robot. The purpose of the last experiment is to show the systems ability to detect and stop (or slow down) the robot if an unexpected obstacle appears in front of the moving vehicle.

6.1 Experiment 1 - Pedestrian detection

The purpose of this experiment is to show the ability of the system to detect a person in normal conditions and adverse conditions (fog). The setup for this experiment is depicted in the Figure 14.

6.1.1 Hardware setup

The experiment uses a RGB-D camera, a 2D lidar, and a 3D radar as the sensors.

One can see in the setup the 3D radar from Texas Instruments IWR1443BOOST mmWave, which has a frequency bandwidth from 76 to 81 GHZ, 30° field of view and an update rate of 10 Hz. Its range depends on its settings, where the range is a maximum of 50 m, but this setting only provides single points in a 2D plane. The 3D setup offers a maximum range of 10 m which is the setting, which was used for this experiment. Another parameter the radar offers to tune is the maximum velocity of the points to be detected, and clutter removal to remove the noise. The clutter removal setting removes most of the static points and extra noise. For the evaluation of the experiment, we used the setup with static removal for the system to be trained only from relevant points, also with a lower chance of being noise.

The 2D lidar is the Hokuyo URG-04LX with 240° field of view, and 0.352° angular resolution. The range of this sensor is up to 5.6 m, and its update rate is 10 Hz. The reasoning for using simple 2D lidar is its price, which is around $300USD.

For this experiment, the fog machine used is the Beamz S500 Fazer, which can provide up to 30 s continuous stream of haze. The hazer has another limitation which is the thickness of the fog, which affects both of the sensors. In Figure 14 we can observe the cardboard box, which serves as a structure for mounting the sensors and also as the plane, where the fog can float and not disappear right away and can affect both the radar and lidar the same way. The 2D lidar was placed on the bottom of the cardboard box, where one can see only the scanning plane of the sensor in Figure 14. The radar was placed at the end of the cardboard box not far from the 2D lidar.

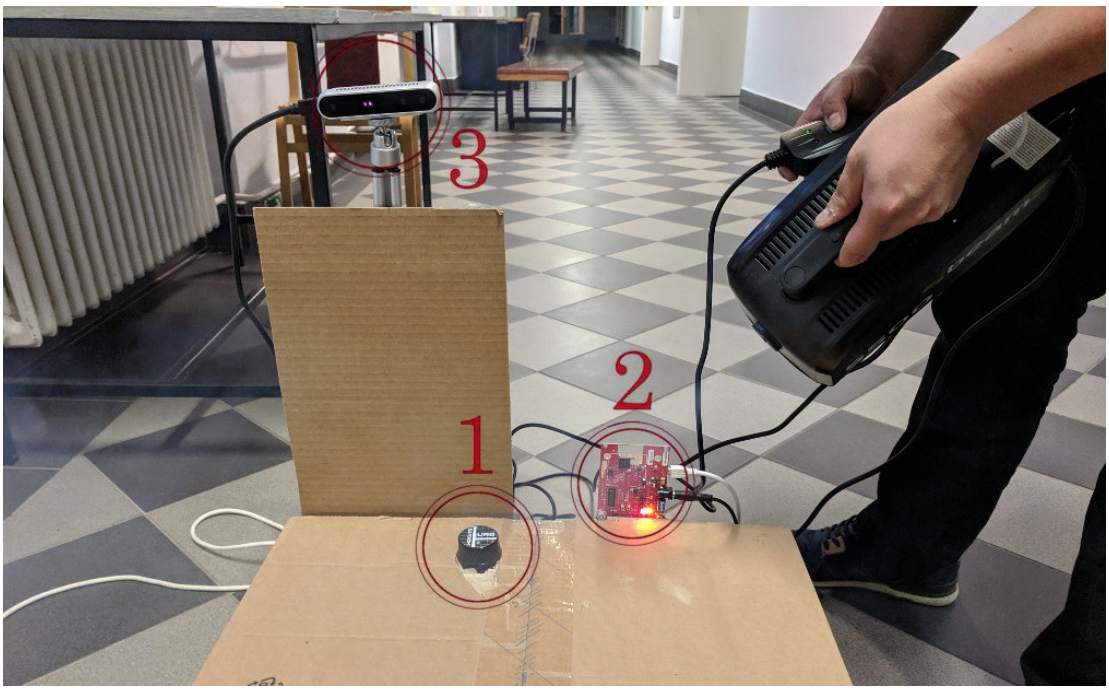


Figure 14: A photo from the experimental setup, where 1) is a Hokuyo URG-04LX 2D lidar, 2) a Texas Instruments IWR1443BOOST 3D radar, and 3) an Intel Realsense D415 RGB-D camera. The Figure also contains the haze generator Beamz S500 Fazer for generating the adverse conditions

Outside of the structure and 30 cm above the cardboard box we placed the RGB-D camera, the Intel RealSense D415 camera. Its purpose is to provide the system with ground truth data, and its placement was designed not to be affected by the fog. Even though the leg detector is well established, one metric for the experiment could be to compare the ability of radar detection with the leg detector, but in this case, the idea is to show that the sensor fusion of both detections results in more precise detections, thus an external detection system to obtain the ground truth is needed.

Every sensor was connected to the laptop, which had an Intel i5 processor, 8GB RAM and was running on Ubuntu 16.04 LTS and ROS Melodic.

6.1.2 Experimental setup

The experiment took place in a corridor of the Czech Technical University in Prague on Charles square. This experiment aims to show the ability of the pedestrian detection framework to detect people in both normal conditions and adverse conditions reliably. Thus, the setup was designed to evaluate the precision of the detections for both lidar and radar, and also their sensor fusion. The experiment is composed of the hardware setup

mentioned above and a single demonstrator. The demonstrator walks in front of all the sensors and remains there for the entire duration of the experiment. One could argue that the person could walk away and then come back, to show the ability of the framework, that if there is no person in front of the sensors, there will be no detections. However, the field of view of the radar, lidar and camera is different. This means that for some of the frames, the person is detected by the two sensors, but the third sensor will not see it. So to have a fair comparison, the person stays the entire experiment in the field of view of all the used sensors. Note that during the testing of the framework the person walked out of the scene and no detections were present.

The idea of the experiment is that the pedestrian moves around for two minutes in which the radar performs the online training. After the two minutes, we start the haze generator, putting the fog directly onto the surface of the cardboard to affect both the radar and lidar. Due to the hardware limitations of the haze generator, this part lasts for approximately 30 s.

6.1.3 Obtaining the ground truth

For obtaining the ground truth, we use the RGB-D camera mounted above the cardboard structure. From the RGB-D camera, only the depth images are used. Firstly the image was cropped in a way to remove walls on the sides, pixels with a distance too far in the scene (more than 6 m) and pixels closest to the camera (less than 1 m). This results in having a rectangular shape in the horizontal plane where the demonstrator moves. In addition to removing pixels pointing to the floor and ceiling the lowest point where setup to 0.5 m and the highest to 1.5 m. Thus, we received a bounding box where the demonstrator walks, the remaining points in the image are clustered, and the biggest cluster is considered to be the detected pedestrian. To obtain the position of the pedestrian, we simply take the centroid of this cluster for each incoming image from the RGB-D camera. The Figure 15 shows the sample image from the RGB-D camera and then the filtered one with the biggest cluster.

Figure 15: Obtaining ground truth (left) Standard depth image (right) Filtered and clustered person

6.1.4 Experimental evaluation

The experimental evaluation is composed of comparing the position of the pedestrian detection framework - its outputs from the radar, lidar and the sensor fusion from both of the sensors in the form of the two filters - switching and weighted. The metric used in the evaluation is the Multiple Object Tracking Precision (MOTP) from [70], which is used only to track a single person. Note the system is designed that for the future work it can detect multiple people. Thus the metric for multi people detection is introduced. Moreover, the paper introduced the Multiple Object Tracking Accuracy (MOTA), but this metric is based among others on the false negatives, which do not exist in the experiment, where the person always appears in the scene, thus the MOTP is used as the only metric. The MOTP metric can be writen for a single object to be detected as:

$$\epsilon(t_k) = \frac{1}{k} \sum_{i=1}^{k} ||\mathbf{x}(t_i) - \mathbf{x}_g(t_i)||. \tag{4}$$

The $\epsilon(t_k)$ is the average position error over time, t_k is time from the start of the experiment, $\mathbf{x}(t_i)$ is position estimation by the pedestrian detection framework and the $\mathbf{x}_g(t_i)$ is the ground truth position at time t_i. Thus, the equation gives the average error at time t_k, which is computed from all the previous times, t_i and the error is the Euclidean distance between the estimation and the ground truth. The reasoning for averaging the error is that occasional false positive detections will be visible in the graph and also it gives us an idea of how the online training process affects the entire evaluation and how the system converges over time.

6.1.5 Experimental results

The idea behind the experiment is a person walking in front of three sensors for some time. At the beginning of each experiment, the SVM from the framework is being trained, and once the learning time is over, the experiment continues with a person walking so we can observe how the particular detection or sensor fusion filters converge over time. The primary purpose of this experiment is to show that not only we can obtain reliable detections from the online trained radar, which is trained from the lidar, but more importantly, if we use their combination in the form of sensor fusion, we obtain a system that is capable of detections with much higher precision. The evolution of the error under normal conditions is shown in Figure 16.

Figure 16 shows four different position estimate errors radar-only, lidar-only, switching filter, weighted filter. The graph shows the learning phase of the radar, where the error rapidly grows in the first couple of seconds, once it passes the first round of training. Next visible is a drop in the accuracy of the radar-only at 20 s, which is the time where the last training round happened, which took much longer to train. Overall the radar-only error is 0.13 m. The most important part of the graph are the two filters switching and

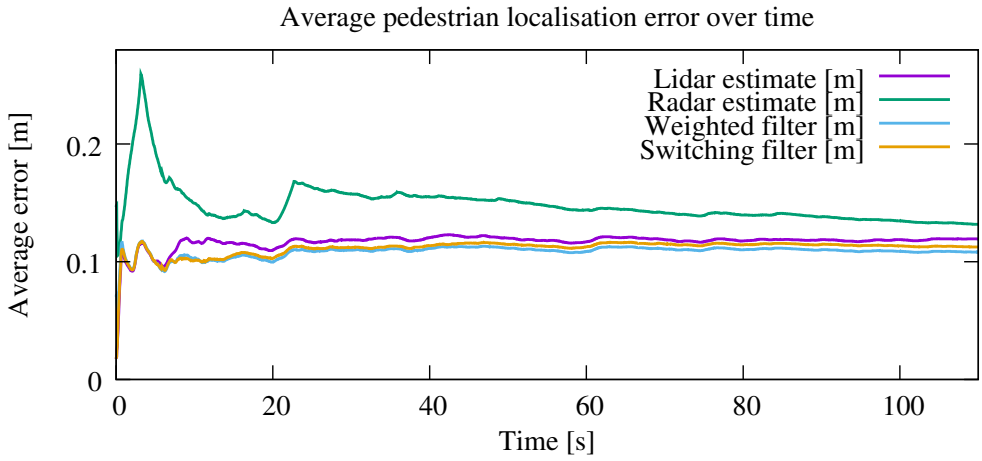


Figure 16: Evolution of the pedestrian localisation error during learning in normal conditions [64]

weighted, where the error is 0.11 m. Both of them are slightly better than the lidar-only error and better than the radar-only error, which corresponds to the hypothesis that the sensor fusion of radar-lidar will achieve higher accuracy than the respective sensor-only errors.

The experiment then followed with starting the haze generator, which was running for approximately 30 s. The evolution of the error is visible in the Figure 17.

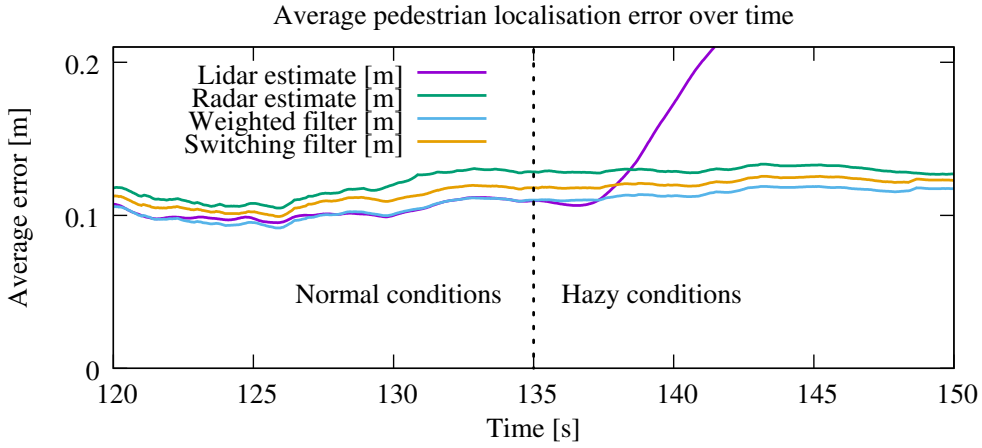


Figure 17: Evolution of the pedestrian localisation error during learning in adverse conditions of fog [64]

The second part in the adverse conditions, which in the Figure 17 is visible by the change of the lidar error. Here we can observe that the lidar error rapidly increased, resulting in an error of 20 cm. During the experiment it was visible that the entire incoming data from the

lidar disappeared, which results in the lidar-only system not beiing capable of detecting the pedestrian. Also, we can observe that sensor fusion is much better than if we are using the sensor detections separately. In the end, the filters and the radar-only detection have an error of 10-12 cm, with the weighted filter having the best results. Table 2 summarizes the values obtained in the experiment.

Conditions	*Sensor*		*Fusion*	
	Lidar	*Radar*	*Switching*	*Weighted*
Normal+Hazy	20.8	12.5	**10.6**	11.0
Normal	11.9	13.0	**10.7**	11.2

Table 2: Average pedestrian localisation error [cm] of different schemes in different conditions.

6.2 Experiment II - Real robot

The purpose of this experiment is to show the ability of the pedestrian detection framework to reliably detect a person while operating on a real robot and prevent a collision with the pedestrian while preserving the ability to navigate autonomously in the terrain. The idea is to use the navigation system from Section 4, teach the robot a path, then let the robot traverse it autonomously. After the repeat phase, the experiment continues with turning on the pedestrian detection pipeline and observing if the navigation succeeds and if it does not diverge from its previous path because of the deviation from the path profile.

This experiment runs only under normal conditions because the ability of the pedestrian detection system was described above in the previous experiment. Also, the reason is limited hardware, where ideally some sort of chamber, which could be filled with smoke is needed. Moreover, the used haze generator does not generate enough fog for this experiment. The fog was emulated by switching off the lidar after the training phase.

6.2.1 Experimental setup

For this experiment we are using the setup depicted in the Figure 18.

The Husky robot is a wheeled platform produced by Clearpath. It is capable of speeds up to 3 m/s, the chassis is capable of traversing mud, light rocks, and steep inclines, which makes it suitable for most of terrains. Lead-acid batteries power this platform. The payload of the Husky platform is up to 50 kg [34]. The robot is controlled by NUC-i7, which runs all the frameworks, and provides the communication and control between the human operator and the hardware of the robot. The NUC is running Ubuntu 18.04 LTS and ROS Melodic. For this experiment, we are using the same 3D radar and RGB-D camera as in the previous experiment.

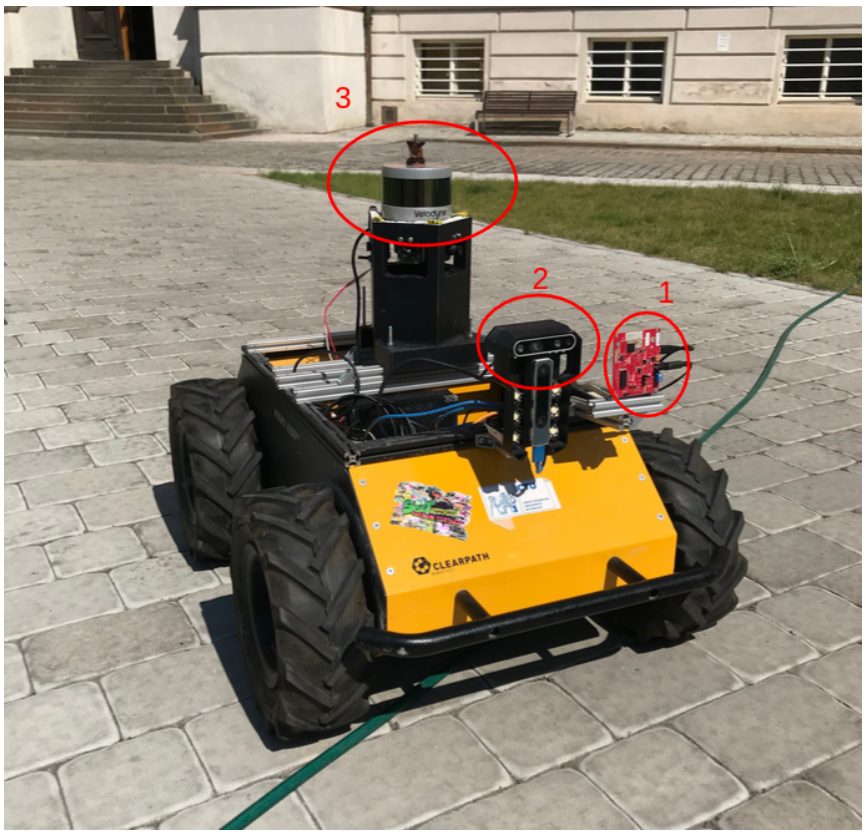


Figure 18: Husky A200 platform equipped with sensors. The red circled sensors are those being used for this experiment as follows: 1) a Texas Instruments IWR1443BOOST 3D radar, 2) an Intel Realsense D415 RGB-D camera, and 3) a Velodyne Puck VLP-16.

The robot is shared within the Center for Robotics and Autonomous Systems (CRAS), where the unique setup of the robot, its sensors and their position on the robot is needed for each experiment. I decided to leave the already equipped Velodyne Puck VLP-16 3D lidar, where the 16 stands for the amount of layer (rings) the sensor is measuring with. This lidar has 30° × 360° field of view. Its range is up to 100 m. For the same conditions as in the previous experiment, only a single ring is used in the height of the pedestrian legs. Note that other sensors equipped on the robot are not being used, for experimental consistency.

The experiment took place in the yard of the Czech Technical University in Prague. The first part of the experiment was the teaching phase of the navigation system. The taught path is depicted in Figure 19, where the start of the path is at the location of the end of the arrow.

The reason for having a circuit-shaped path is that one can quickly determine if the robot repeats the path and does not diverge from it when running multiple iterations. The initial experiments with the navigation system showed that it can traverse an arbitrarily-

Figure 19: Schoolyard of CTU with the taught path (red)

shaped path but if the end of it ends on a different place or the robot is facing other direction, the operator needs to place it back to the original one. Thus, for the simplicity of the experiment, the start of the path is the same as the end.

After the teaching phase, the robot was moved 50 cm away from its starting point and repeated the taught path several times, the amount of the iterations of the traversed path was limited due to the low capacity of the batteries. Note that for the navigation system to work the robot needs to see at least some part of the first taught image. Meaning that if we moved the robot and turn it, so none of the features could be matched the navigation system would fail. This happened because of the prolonged teaching phase, which needed to be run multiple times because there was a car parked right in front of the robot and the lack of skills of the operator to avoid obstacles.

After the teaching phase, the batteries were changed, and the experiment continued with restarting the pedestrian detection framework. For this to have the same framework with online learning, a pedestrian needed to walk in front of the lidar and radar in order to teach it. After a few training rounds of the SVM, the experiment followed again with moving the robot away from its initial position by 50 cm and starting the navigation system.

The collision detection was implemented in a way where a bounding box in front of the robot was created. This bounding rectangle was 0.5 m to 1.7 m in front of the robot, with 0.5 m to the left and 1.1 m to the right because of the placement of the radar. If a person was detected in this bounding rectangle, the robot would stop and wait for the pedestrian to leave it. Pedestrians appearing further than the 1.5 m were ignored, and anything lower

than 0.5 m meant an immediate stop, but this would mean a pedestrian jumping right in front of the robot or getting hit from side of the robot. During the autonomous traversal, the operator walked in front of the robot several times in random places on the taught path, making it detect the person in front of it and investigating its ability to detect a pedestrian and avoid collision with it. Sample image from the experiment show how the person was walking into the path of the autonomous vehicle, can be seen in Figure 20. Note that I have encountered a problem with the leg detector, where its parameters needed to be changed, which prolonged the training of the radar. Also, the detections were not perfect, meaning that a few times the pedestrian was not detected and had to escape from the front of the robot. This was happening mainly at the beginning of the experiment, where the system was not trained well enough. Thanks to the lengthy training, the batteries of the robot were enough to repeat the traversed path 6 times, which is the same amount as in the navigation-only traversals.

Figure 20: Experiment on a real robot with pedestrian crossing its path during the autonomous navigation.

6.2.2 Experimental evaluation

The experimental evaluation takes the error of the autonomous navigation without the pedestrian detection framework and then when it is turned on. The error is computed as the Euclidean distance from the starting point of the path and its ending.

The Figure 21 shows that without the pedestrian detection framework, the error was about 25 cm. However, the system did not diverge, which means that the robot was able

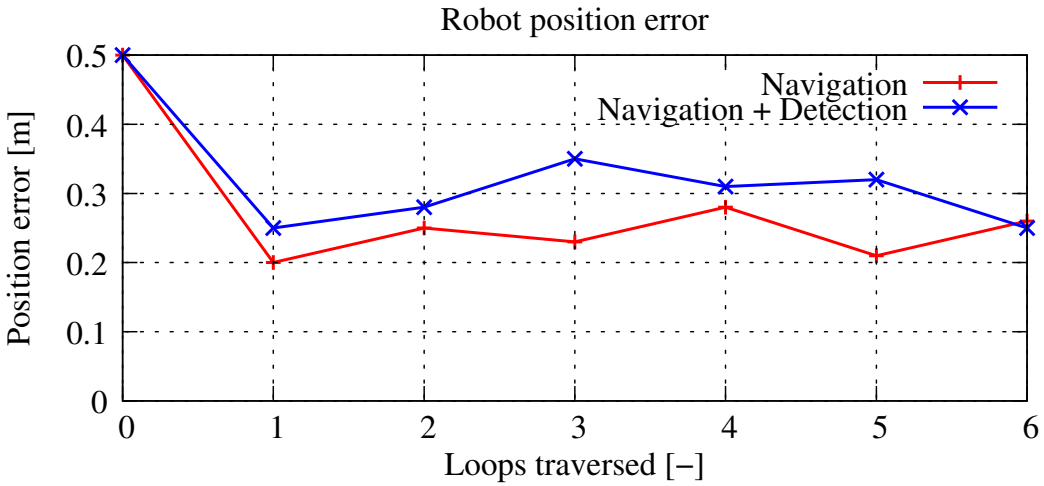


Figure 21: Evolution of the error during the autonomous traversals with the real robot with the navigation system and navigation with the pedestrian detection framework.

to traverse to the taught path repeatedly. One of the reasons for higher error in the taught path is that the ending and the beginning of the path were not the same, and was only estimated. This means that their difference appears in this error.

During the experiment with collision detection on, the person was stepping in front of the robot from different sides and different places on its traversed path. Those places were randomly selected during each traversal. Figure 21 shows that the overall error was about 30 cm, having higher error during some traversals. The difference in the error is not significant enough to diverge the whole navigation system. The whole path was about 100 m, which means that the robot traversed 600 m autonomously while encountering pedestrians on its way, successfully detected them, stopping and once the pedestrian left continued with the navigation, where the error was about 0.3 m at the end of the experiment.

6.3 Experiment III - obstacle detection

For this experiment, I decided to investigate the ability of the radar to detect any object in the direction of the heading of the robot. The idea is to show the possible usage of the radar as a sensor for collision detection in adverse conditions for any object. The pedestrian detection has a much higher priority since the pedestrians are the most vulnerable traffic participants. Detecting other objects that can be in front of the vehicle is also crucial because of the safety of the driver in autonomous cars or safety of the autonomous vehicle without any crew in the vehicle.

For this experiment, only the 3D radar (same as in previous experiments) and the robotic platform is needed. The platform used for this experiment is different, which is a heavy four tons industrial car-like machine. This platform is capable of loading and unloading

regular cars. The platform can be seen in Figure 22. The experiment also tested how the steel vehicle structure affects radar reliability.

Figure 22: Car-like platform where (left) front image with the radar in red circle (right) Platform from the back

For this experiment, the pedestrian detection was changed; the entire training radar from lidar was removed. The algorithm takes the incoming pointcloud from the radar clusters using the same Euclidean clustering as in Section 5. After the clustering, the clusters with a low amount of points were removed because the radar is very noisy. The number of points needed was set empirically to five. Thus the entire pointcloud is segmented into clusters with variable size. After the segmentation, the important information for further processing is the centroid of each cluster. If the centroid of the cluster lies somewhere within the bounding box in front of the sensor the robot reacts accordingly. The bounding box was set to be 0.5 m over each side of the vehicle, and from 1-2.5 m in the direction, which the robot is facing. If a centroid of a cluster appears in the bounding rectangle, the vehicle starts to slow down until the centroid lies within the 1 m range. Clusters closer to the vehicle than 1 m stops the car immediately.

This experiment involved placing objects in the bounding rectangle of the robot and observing if the object is detected, and if the robot slows when getting closer. After this, the navigation system was turned on. However, due to the heavy rain where the platform was still in the development phase, meaning it was not waterproof. The testing had to be indoors, where thanks to its dimension and the testing indoor area not being ready, the taught path was only a couple of meters long line in the warehouse of the robot. The robot was facing the direction of the heavy rain, and a pedestrian was standing in the rain. During the repeat phase, the robot was able to detect an object in front of it and also the pedestrian in the rain. The robot slowed down or stopped based on the distance to the object while preserving the ability to traverse the taught path.

The experiment proved that the steel structure of the vehicle is not negatively affecting the radar detection, and it detected a pedestrian which was standing outside during the heavy rain.

7 Conclusion

This work introduces an autonomous navigation system which combines a visual *teach-and-repeat* system and a *pedestrian detection* system. The pedestrian detection system is capable of the reliable detection of vulnerable traffic participants in adverse weather conditions and uses a 2D lidar and 3D radar. Apart from traditional sensor fusion scheme, we use the lidar detection results to train the radar detection pipeline. The online learning allows the robot to train the radar data processing continuously and perpetually, leading to a gradual improvement of the detection accuracy. The radar processing training takes a few minutes of the system operation after which the system can detect pedestrians in heavy fog and rain.

To achieve safe long-term operation, the pedestrian detection module is integrated into a teach-and-repeat navigation system. This navigation method requires a simple monocular camera and can be deployed for extended periods of time in adverse weather conditions.

In the set of experiments, the ability of the radar-lidar framework to detect a person in adverse weather conditions was demonstrated and proved that the *learning-fusion* has better precision even if one of the sensors fails due to environmental conditions. The following experiment combined the navigation method and the pedestrian detection on a real robot, which autonomously traversed a given path while evading collisions with pedestrians. This experiment proved that the pedestrian detection does not negatively affect the navigation accuracy, and the system can avoid collisions with pedestrians while reliably navigating. The last experiment showed the possibility to use the radar for collision avoidance on a heavy duty robotic platform, and it also showed that the system is able to detect pedestrians in heavy rain.

This thesis shows the evolution of setting up the system from a piece of cardboard, to obtain pedestrian positions, then having the pedestrian detection on an experimental robotic platform with the evolution ending with setting the system to the industrial machine.

The navigation system which I helped develop is a subject of works [71, 21, 47], and was used for experiments in [35, 31]. The pedestrian detection system using the learning-fusion principle was introduced in [64] and was selected among the ten best papers of the conference. An extended version of the paper was accepted with minor revisions in a special issued of the Robotics and Autonomous Systems Journal [40].

References

[1] Nidhi Kalra and Susan M Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016.

[2] Puneet Kohli and Anjali Chadha. Enabling pedestrian safety using computer vision techniques: A case study of the 2018 Uber Inc. self-driving car crash. In *Future of Information and Communication Conference*, pages 261–279. Springer, 2019.

[3] Josh Horwitz and Heather Timmons. There are some scary similarities between Tesla's deadly crashes linked to autopilot. *Atlantic Media, 20.9.2016*, 2016.

[4] Sebastian Thrun. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, pages 1–35, 2002.

[5] David Filliat and Jean-Arcady Meyer. Map-based navigation in mobile robots:: I. a review of localization strategies. *Cognitive Systems Research*, 4(4):243–282, 2003.

[6] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[7] M. Kulich, P. Štěpán, and L. Přeučil. Feature detection and map building using ranging sensors. In *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No.99TH8383)*, pages 201–206, 1999.

[8] Jari P Saarinen, Henrik Andreasson, Todor Stoyanov, and Achim J Lilienthal. 3d normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments. *The International Journal of Robotics Research*, 32(14):1627–1644, 2013.

[9] Kai M Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, volume 2, 2010.

[10] Jugesh Sundram, Duong Van Nguyen, Gim Song Soh, Roland Bouffanais, and Kristin Wood. Development of a miniature robot for multi-robot occupancy grid mapping. In *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 414–419. IEEE, 2018.

[11] Karel Košnar, Tomáš Krajník, and Libor Přeučil. Visual topological mapping. In *European Robotics Symposium 2008*, pages 333–342. Springer, 2008.

[12] Rainer Bischoff and Volker Graefe. Hermes—a versatile personal robotic assistant. *Proceedings of the IEEE*, 92:1759 – 1779, 12 2004.

[13] Karel Košnar, Tomáš Krajník, Vojtech Vonásek, and Libor Preucil. Lama-large maps framework. In *Proceedings of Workshop on Field Robotics, Civilian-European Robot Trial*, pages 9–16, 2009.

[14] Tomáš Krajník, Jan Faigl, Vojtěch Vonásek, Karel Košnar, Miroslav Kulich, and Libor Přeučil. Simple yet stable bearing-only navigation. *Journal of Field Robotics*, 27(5):511–533, 2010.

[15] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*, volume 1. MIT press Cambridge, 2000.

[16] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999(343-349):2–2, 1999.

[17] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[18] Tomáš Krajník. *Large-scale mobile robot navigation and map building*. PhD thesis, Czech Technical University in Prague, 2011.

[19] G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002.

[20] P. De Cristóforis, M. Nitsche, and T. Krajník. Real-time image-based autonomous robot navigation method for unstructured outdoor roads. *Journal of Real Time Image Processing*, 2013.

[21] Tomáš Krajník, Filip Majer, Lucie Halodová, and Tomáš Vintr. Navigation without localisation: reliable teach and repeat based on the convergence theorem. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1657–1664. IEEE, 2018.

[22] Steven Holmes, Georg Klein, and David W. Murray. A Square Root Unscented Kalman Filter for visual monoSLAM. In *International Conference on Robotics and Automation (ICRA)*, pages 3710–3716, 2008.

[23] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[24] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE, 1997.

[25] S. Otte, J. Kulick, M. Toussaint, and O. Brock. Entropy-based strategies for physical exploration of the environment's degrees of freedom. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 615–622, 2014.

[26] F. Boniardi, B. Behzadian, W. Burgard, and G. D. Tipaldi. Robot navigation in hand-drawn sketched maps. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6, 2015.

[27] A. Murarka and B. Kuipers. Using cad drawings for robot navigation. In *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)*, volume 2, pages 678–683 vol.2, 2001.

[28] Syaimaa Solehah Mohd Radzi, Shahrul Nizam Yaakob, Zulaikha Kadim, and Hon Hock Woon. Extraction of moving objects using frame differencing, ghost and shadow removal. In *2014 5th International Conference on Intelligent Systems, Modelling and Simulation*, pages 229–234. IEEE, 2014.

[29] Peter Nelson, Winston Churchill, Ingmar Posner, and Paul Newman. From dusk till dawn: Localisation at night using artificial light sources. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5245–5252. IEEE, 2015.

[30] Tomáš Krajník, Jaime P Fentanes, Joao M Santos, and Tom Duckett. Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments. *IEEE Transactions on Robotics*, 2017.

[31] Lucie Halodová, Eliška Dvořáková, Filip Majer, Tomáš Vintr, Oscar Martinez Mozos, Feras Dayoub, and Tomáš Krajník. Predictive and adaptive maps for long-term visual navigation in changing environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7033–7039. IEEE, 2019.

[32] Timothy D Barfoot, Colin McManus, Sean Anderson, Hang Dong, Erik Beerepoot, Chi Hay Tong, Paul Furgale, Jonathan D Gammell, and John Enright. Into darkness: Visual navigation based on a lidar-intensity-image pipeline. In *Robotics Research*, pages 487–504. Springer, 2016.

[33] Colin McManus, Paul Furgale, Braden Stenning, and Timothy D Barfoot. Visual Teach and Repeat using appearance-based lidar. In *2012 IEEE International Conference on Robotics and Automation*, pages 389–396. IEEE, 2012.

[34] Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanský, Daniel Heřt, Matěj Petrlík, Tomáš Báča, Vojěch Spurný, et al. DARPA subterranean challenge: Multi-robotic exploration of underground environments. In *International Conference on Modelling and Simulation for Autonomous Systesm*, pages 274–290. Springer, 2019.

[35] Lucie Halodová, Eliška Dvořáková, Filip Majer, Jiří Ulrich, Tomáš Vintr, Keerthy Kusumam, and Tomáš Krajník. Adaptive image processing methods for outdoor autonomous vehicles. In *International Conference on Modelling and Simulation for Autonomous Systesm*, pages 456–476. Springer, 2018.

[36] Wilfried Elmenreich. An introduction to sensor fusion. *Vienna University of Technology, Austria*, 502:1–28, 2002.

[37] Lei Deng, Zhihui Mao, Xiaojuan Li, Zhuowei Hu, Fuzhou Duan, and Yanan Yan. Uav-based multispectral remote sensing for precision agriculture: A comparison between different cameras. *ISPRS Journal of Photogrammetry and Remote Sensing*, 146:124–136, 2018.

[38] *Intel RealSense Depth Camera D435*, accessed 09.08.2020. https://www.intelrealsense.com/depth-camera-d435/.

[39] *Puck Lidar Sensor*, accessed 09.08.2020. https://velodynelidar.com/products/puck/.

[40] Filip Majer, George Brougton, Tomáš Rouček, Yassine Ruichek, Zhi Yan, and Tomáš Krajník. Learning to see through the haze: Multi-sensor learning-fusion system for vulnerable traffic participant detection in fog. *Robotics and Autonomous Systems*, 2020. to appear.

[41] M. Kutila, P. Pyykönen, H. Holzhüter, M. Colomb, and P. Duthon. Automotive lidar performance verification in fog and rain. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1695–1701, 2018.

[42] Michael Skutek, Moheb Mekhaiel, and Gerd Wanielik. A precrash system based on radar for automotive applications. In *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No. 03TH8683)*, pages 37–41. IEEE, 2003.

[43] Walter Ulke, Rolf Adomat, Karlheinz Butscher, and Wolfgang Lauer. Radar based automotive obstacle detection system. Technical report, SAE Technical Paper, 1994.

[44] Pablo De Cristóforis, Matias Nitsche, Tomáš Krajník, Taihú Pire, and Marta Mejail. Hybrid vision-based navigation for mobile robots in mixed indoor/outdoor environments. *Pattern Recognition Letters*, 53:118–128, 2015.

[45] Nan hang, Michael Warren, and Timothy Barfoot. Learning place-and-time-dependent binary descriptors for long-term visual localization. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016.

[46] Tomáš Krajník, Pablo Cristóforis, Keerthy Kusumam, Peer Neubert, and Tom Duckett. Image features for visual teach-and-repeat navigation in changing environments. *Robotics and Autonomous Systems*, 2017.

[47] Filip Majer, Lucie Halodová, Tomáš Vintr, Martin Dlouhý, Lukáš Merenda, Jaime Pulido Fentanes, David Portugal, Micael Couceiro, and Tomáš Krajník. A versatile visual navigation system for autonomous vehicles. In *International Conference on Modelling and Simulation for Autonomous Systesm*, pages 90–110. Springer, 2018.

[48] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[49] Elmar Mair, Gregory D Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *European conference on Computer vision*, 2010.

[50] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[51] I. Jegham and A. Ben Khalifa. Pedestrian detection in poor weather conditions using moving camera. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 358–362, 2017.

[52] S. Gidel, P. Checchin, C. Blanc, T. Chateau, and L. Trassoudaine. Pedestrian detection method using a multilayer laserscanner: Application in urban environment. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 173–178, 2008.

[53] S. Gidel, P. Checchin, C. Blanc, T. Chateau, and L. Trassoudaine. Pedestrian detection and tracking in an urban environment using a multilayer laser scanner. *IEEE Transactions on Intelligent Transportation Systems*, 11(3):579–588, 2010.

[54] Zhi Yan, Tom Duckett, and Nicola Bellotto. Online learning for human classification in 3D lidar-based tracking. In *In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 864–871, Vancouver, Canada, September 2017.

[55] C. Premebida, J. Carreira, J. Batista, and U. Nunes. Pedestrian detection combining rgb and dense lidar data. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4112–4117, 2014.

[56] P. Tumas, A. Nowosielski, and A. Serackis. Pedestrian detection in severe weather conditions. *IEEE Access*, 8:62775–62784, 2020.

[57] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[58] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the DARPA Grand Challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[59] Hyunggi Cho, Young-Woo Seo, BVK Vijaya Kumar, and Ragunathan Raj Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1836–1843. IEEE, 2014.

[60] Kai O. Arras, Óscar Martínez Mozos, and Wolfram Burgard. Using boosted features for the detection of people in 2D range data. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3402–3407, 2007.

[61] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, 2009.

[62] Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46, 1995.

[63] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[64] Filip Majer, Zhi Yan, George Broughton, Yassine Ruichek, and Tomáš Krajník. Learning to see through haze: Radar-based human detection for adverse weather conditions. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–7. IEEE, 2019.

[65] Luis E. Navarro-Serment, Christoph Mertz, and Martial Hebert. Pedestrian detection and tracking using three-dimensional ladar data. In *Proceedings of the 7th Conference on Field and Service Robotics (FSR)*, pages 103–112, 2009.

[66] Kiyosumi Kidono, Takeo Miyasaka, Akihiro Watanabe, Takashi Naito, and Jun Miura. Pedestrian recognition using high-definition LIDAR. In *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 405–410, 2011.

[67] S. Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

[68] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:1–27, 2011.

[69] Dave Hershberger, David Gossow, and Josh Faust. Rviz, 3D visualization tool for ROS. *URL: http://wiki. ros. org/rviz [cited 09-08-2020]*, 2019.

[70] Keni Bernardin, Alexander Elbs, and Rainer Stiefelhagen. Multiple object tracking performance metrics and evaluation in a smart room environment. In *Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*, volume 90, page 91. Citeseer, 2006.

[71] Filip Majer, Lucie Halodová, and Tomáš Krajník. A precise teach and repeat visual navigation system based on the convergence theorem. In *Student Conf. on Planning in AI and Robotics (PAIR)*, 2017.

Apppendix

CD Content

Directory name	Description
text	diploma thesis in pdf
sources	source codes
datasets	recorded rosbags for experimens

Table 3: CD Content