Multi Sensor Ramp Detection and Localization for Autonomous Valet Parking

Master thesis

Felix Saalfrank
368199

May 12, 2022

Supervisor:
Lars Schürmann
Person 2
Gutachter:
Prof. Dr. Gühmann
Prof. Dr. Orglmeister

Technische Universität Berlin
Faculty IV - Electrical Engineering and Computer Science
Department of Energy and Automation Technology
Chair of Electronic Measurement and Diagnostic Technology

TU Berlin | Einsteinufer 17 | 10587 Berlin

Fakultät IV Elektrotechnik und Informatik
Institut für Energie und Automatisierungstechnik

Fachgebiet für Elektronische Mess- und Diagnosetechnik

Prof. Dr.-Ing. Clemens Gühmann

Sekretariat EN 13 Raum EN 538
Einsteinufer 17
10587 Berlin

Telefon +49 (0)30 314-29393
Telefax +49 (0)30 314-22120
clemens.guehmann @ tu-berlin.de

Bearbeiterin
Ewa Heinze

Telefon +49 (0)30 314-22280
Telefax +49 (0)30 314-22120
ewa.heinze@tu-berlin.de

Berlin, 3. Januar 2022

Masterarbeit für Herrn Felix Saalfrank, Matrikel-Nr. 368199

Multi-Sensor Ramp Detection and Localization for Autonomous Valet Parking

Problem: Autonomous Valet Parking (AVP) will make parking easier in the future, by allowing the driver to exit the car in a drop off zone in front of a parking garage, and the car will find a parking spot on its own. When the driver calls the car again, it will also autonomously find its way to the driver. For this to work, a map of the parking garage and precise localization of the car is necessary. A challenging part is the necessary change of levels during the procedure because the ramps in parking garages are usually very narrow and require precise localization and control of the car. Therefore, information about whether or not the car is driving onto a ramp is necessary. This allows the controller of the car to adjust for the changing road conditions, e.g. increasing or decreasing the motor output power when driving up or down respectively. Also, because the maps used for the localization of the car are usually stored separately for each parking level, the loading of the new map should be initiated while the car is on a ramp.

Task: The goal of this thesis is to implement an algorithm for a car, which can detect ramps. Besides the detection, ramp properties such as the inclination angle or length should be measured. To implement this, various sensor setups will be used and compared. An Inertial Measurement Unit (IMU) will be the main sensor and will be responsible for the exact measurement of the ramp's properties, in conjunction with a wheel odometer. Additionally, a Light Detection and Ranging (LiDAR) sensor will be used to allow for the detection of the ramp, before entering it. The data of the LiDAR could also be fused with the IMU data to prevent false detections by the IMU. A camera will be tested as well for the early detection of a ramp and compared to the LiDAR. Test drives in one specific parking garage and test car will be performed. A camera will be used to validate if the detection was at the right time and the estimated ramp properties will be compared to manual measurements.

Research steps:
- Research of current methods to determine road grade angle using IMU, LiDAR or camera
- Comparison and selection of the most appropriate method for each sensor
- Implementation of a ramp detection algorithm
    - using an IMU
    - using a LiDAR sensor
    - using a camera
- Testing and optimizing of the methods
- Comparison and evaluation of the different methods used

● Documentation and presentation of the results and thesis

Organization
This thesis will be written in collaboration between TU-Berlin and Expleo Germany GmbH. Felix Saalfrank's supervisor is Lars Schürmann. All necessary documents and resources will be provided by Expleo Germany GmbH. The presentation of the researched literature, the analytical work and the experiments will be carried out according to the rules of best scientific practice. The results will be presented publicly in the seminar of the Electronic Measurement and Diagnostic Technology chair.

Mit freundlichen Grüßen

Prof. Dr.-Ing. Clemens Gühmann

Kurzfassung

100-200 Wörter Kurzfassung (deutsch)

Abstract

100-200 word abstract (english)

Declaration

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Contents

List of Figures

List of Tables

List of Acronyms

AHRS Attitude Heading Reference System

AP Average Precision

AVP Automated Valet Parking

CNN Convolutional Neural Network

COCO Common Objects in Context

FOV Field of View

HPF High-Pass Filter

IMU Inertial Measurement Unit

IoU Intersection of Union

LiDAR Light Detection and Ranging

LPF Low-Pass Filter

mAP Mean Average Precision

MEMS Microelectromechanical Systems

R-CNN Region-based Convolutional Neural Network

RADAR Radio Detection and Ranging

RANSAC Random Sample Consensus

RMSE Root Mean Square Error

ROI Region of interest

ROS Robot Operating System

SLAM Simultaneous Localization and Mapping

List of Symbols

$\mathbf{a}$ Linear acceleration

θ Pitch angle of the car

$\mathbf{g}$ Gravitatinal acceleration

$\mathbf{j}$ Rotation axis

$\mathbf{n}$ Normal vector

${}^{\mathcal{A}}_{\mathcal{B}}\mathbf{q}$ Quaternion to transform a vector from frame $\mathcal{A}$ to $\mathcal{B}$

$\otimes$ Quaternion multiplication

${}^{\mathcal{A}}_{\mathcal{B}}\mathbf{M}$ Rotation matrix to transform a vector from frame $\mathcal{A}$ to $\mathcal{B}$

R^2 Coefficent of determination

γ Road grade / Ramp angle

α Rotation angle

$\mathbf{v}$ Vector

${}_{\mathcal{A}}\mathbf{v}$ Vector in coordinate frame $\mathcal{A}$

$\|\mathbf{v}\|$ Norm of vector

$\hat{\mathbf{v}}$ Unit vector

$\boldsymbol{\omega}$ Angular velocity

Todo list

Chapter 1

Introduction

Write introduction

1.1 Motivation

Just copied this from the expose, needs improvement
maybe some more sections about the specific topics or about goals

Autonomous driving ...
Parking is one of the most challenging driving tasks and the cause of almost half of the car accidents [1]. Current cars are already able to fully automated park on their own in parallel or perpendicular parking spaces. But due to the very limited space in cities, parking garages are often used in central areas [2]. Automated Valet Parking (AVP) allows for a fully automated parking experience. The car is left in a drop-off zone and finds a parking spot on its own. Afterwards the driver can give a command and the car leaves the parking spot again and picks up the driver. AVP saves time, the hassle of remembering the parking level and spot and furthermore allows to use the available space more efficiently and also minimizes the risk of collisions. For this to work an exact mapping of the environment and localization of the car in the garage is necessary.
This can be done either by Simultaneous Localization and Mapping (SLAM) or the area can be mapped beforehand (e.g. using Light Detection and Ranging (LiDAR) sensors) in which case only a localization of the car is necessary. The mapping can be done in 2D or 3D. 2D maps only show information of the current level. Hence if the car is driven up or down a ramp, the new map of the corresponding floor has to be loaded. Because the localization usually only works in a 2D-plane, a change of levels would not be detected. To solve this problem, a ramp detection has to be implemented.

- Semantic map
- Multi storey
- Control of car before / on ramp

https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/

1.2 Goals

The goal of this thesis is the development of a ramp detection system. The system should be able to detect the presence of a ramp and the position of the ramp. Furthermore, ramp

properties such as the slope and the length of the ramp should be estimated. Different sensors will be used and compared to each other.

1.3 Outline

This thesis is structured in the following way. In chapter 2 the theoretical foundations needed for the methods used in this thesis are described. A brief mathematical overview is followed by a description and explanation of the working principle of the different sensors used in this thesis. Lastly, a short introduction into the field of computer vision and especially in neural networks is given. Other existing methods and approaches to solve the problem posed in this thesis are reviewed in chapter 3. The different implemented methods are described in chapter 4. Different approaches to estimate the road grade using an Inertial Measurement Unit (IMU) are presented in section 4.3, one method using a LiDAR in section 4.4 and finally using a camera in section 4.5. The exact type of sensors used for the recording of the data and the environment in which they were taken, is described in chapter 5. Chapter 6 contains the results of the different methods. A summary of the thesis and potential future work is given in chapter 7.

Chapter 2

Background

In this chapter the fundamentals to understand the methods used in this thesis are explained. The first section gives some mathematical background for different ways to represent orientations. In section 2.2 the functionality of the different sensors used in this thesis are explained. A brief overview of the field of computer vision is given in section 2.3. Lastly, in section 2.4 the Robot Operating System (ROS) framework, which allows for the communication to the different sensors, is introduced.

2.1 Mathematical Background

2.1.1 Mathematical Representations of 3D Orientations

Different sensors will be used in this thesis and each sensor will make its measurements in its own coordinate frame. In order to obtain meaningful results, all frames must be aligned to one common frame. The necessary 3D rotations can be expressed in several ways, the ones used in this thesis will be briefly described.

2.1.1.1 Rotation Matrix

A rotation matrix ${}^{\mathcal{B}}_{\mathcal{A}}\mathbf{M} \in \mathbb{R}^{3\times3}$ transforms an arbitrary vector from the coordinate system $\mathcal{B}$ to the coordinate system $\mathcal{A}$. Rotation matrices have the following properties

$$\mathbf{M}\mathbf{M}^\intercal = \mathbf{M}^\intercal\mathbf{M} = I_3, \quad \det\mathbf{M} = 1. \tag{2.1}$$

The rows of the rotation matrix are the axes of the new coordinate system. A rotation matrix corresponds to two different rotations, but uniquely describes an orientation. Rotation matrices can be concatenated, but this must be done in reverse order, e.g.

$${}_{\mathcal{C}}\mathbf{p} = {}^{\mathcal{A}}_{\mathcal{C}}\mathbf{M}\,{}_{\mathcal{A}}\mathbf{p} = {}^{\mathcal{B}}_{\mathcal{C}}\mathbf{M}\,{}^{\mathcal{A}}_{\mathcal{B}}\mathbf{M}\,{}_{\mathcal{A}}\mathbf{p} \tag{2.2}$$

to transform the point ${}_{\mathcal{A}}\mathbf{p}$ over the intermediate frame $\mathcal{B}$ to the coordinate system $\mathcal{C}$.

2.1.1.2 Euler Angles

Euler angles are the closest to intuition but mathematically the worst way to represent orientations. Every orientation can be produced by a concatenation of three rotations around each of the coordinate axes. Because the resulting orientation depends on the order of which the rotations were performed, there are different conventions. Furthermore, the conventions can be divided into intrinsic and extrinsic. Intrinsic rotation means that the coordinate system moves with the moving object, whereas with extrinsic rotations the original coordinate system remains static.

The most common convention is the zyx (intrinsic), which first rotates an angle ψ around the z-axis, followed by a rotation of θ around the y-axis and finally an angle ϕ around the x-axis, see fig. 2.1. The angles are called yaw (or heading), pitch and roll respectively. Roll

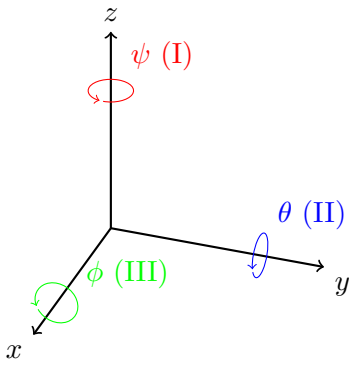


Figure 2.1: Definition of the zyx Euler convention. The rotation starts with a rotation ψ around the z-axis and ends with a rotation ϕ around the x-axis.

and pitch angle together are also often referred to as inclination.
The disadvantages of Euler angles are the many conventions and the possibility of singularity. Unlike rotation matrices, Euler angles do not uniquely describe a rotation. Singularity, which is also referred to as gimbal lock when occurring in mechanical gyroscopes, occurs when the pitch angle is set to $\theta = \frac{\pi}{2}$. The other two angles are then undefined, because both their rotations axes are parallel to each other [3].

2.1.1.3 Quaternions

A quaternion is defined as

$$\mathbf{q} = q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k} \tag{2.3}$$

with a real part q_w and three imaginary parts $\mathbf{i}, \mathbf{j}$ and $\mathbf{k}$ satisfying

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1. \tag{2.4}$$

The quaternion can also be written in vector form by dividing the quaternion in a real or scalar part q_w and an imaginary or vector part $q_v = \begin{pmatrix} q_x & q_y & q_z \end{pmatrix}^\intercal$

$$\mathbf{q} = \begin{pmatrix} \mathbf{q}_v \\ q_w \end{pmatrix} = \begin{pmatrix} q_x & q_y & q_z & q_w \end{pmatrix}^\intercal. \tag{2.5}$$

Note that in some literature the order of the real and imaginary part are swapped. If the quaternion is a unit vector, which means it satisfies

$$|\mathbf{q}| = \sqrt{\mathbf{q}^\intercal\mathbf{q}} = \sqrt{|\mathbf{q}_v^2 + q_w^2|} = 1 \tag{2.6}$$

then it can be used to represent rotations. The rotation is then described with

$$\mathbf{q} = \begin{pmatrix} \hat{\mathbf{j}} \cdot \sin\left(\frac{\alpha}{2}\right) \\ \cos\left(\frac{\alpha}{2}\right) \end{pmatrix} \tag{2.7}$$

with α being the rotation angle and $\hat{\mathbf{j}}$ being the normalized rotation axis. Each rotation is uniquely described by one unit quaternion, but each orientation can be described by two

unit quaternions with opposite signs. A rotation to transform a point ${}_{\mathcal{A}}\mathbf{p}$ from frame $\mathcal{A}$ to $\mathcal{B}$ is defined as

$$ {}_{\mathcal{B}}\mathbf{p} = {}^{\mathcal{A}}_{\mathcal{B}}\mathbf{q} \otimes {}_{\mathcal{A}}\overline{\mathbf{p}} \otimes {}^{\mathcal{A}}_{\mathcal{B}}\mathbf{q}^*, \tag{2.8} $$

where

$$ {}_{\mathcal{A}}\overline{\mathbf{p}} = \begin{pmatrix} 0 & {}_{\mathcal{A}}\mathbf{p}^\intercal \end{pmatrix}^\intercal, \tag{2.9} $$

and ${}^{\mathcal{A}}_{\mathcal{B}}\mathbf{q}^* = {}^{\mathcal{B}}_{\mathcal{A}}\mathbf{q}$ is the conjugate defined as

$$ \mathbf{q}^* = \begin{pmatrix} -\mathbf{q}_v \\ q_w \end{pmatrix}. \tag{2.10} $$

The symbol $\otimes$ denotes the quaternion multiplication, which is given by

$$ \mathbf{p} \otimes \mathbf{q} = \begin{pmatrix} p_w q_w - p_v \cdot q_v \\ p_w q_v + q_w p_v + p_v \times q_v \end{pmatrix}. \tag{2.11} $$

A quaternion can be converted to a rotation matrix with

$$ \mathbf{M} = \begin{pmatrix} 1 - 2q_y^2 - 2q_z^2 & 2(q_x q_y - q_z q_w) & 2(q_x q_z + q_y q_w) \\ 2(q_x q_y + q_z q_w) & 1 - 2q_x^2 - 2q_z^2 & 2(q_y q_z - q_x q_w) \\ 2(q_x q_z - q_y q_w) & 2(q_y q_z + q_x q_w) & 1 - 2q_x^2 - 2q_y^2 \end{pmatrix}. \tag{2.12} $$

More information about quaternions can be found in [4, 3].

2.1.2 Vector Projection

Vector projection is used later in this thesis to align different coordinate frames, this section describes the theory behind it. In general, the rotation to align a vector $\mathbf{v}_1$ with a vector $\mathbf{v}_2$ can be expressed using a quaternion. At first both vectors must be normalized, resulting in the two unit vectors $\hat{\mathbf{v}}_1$ and $\hat{\mathbf{v}}_2$. The rotation axis is perpendicular to both vectors and can thus be calculated using the cross product, which is then divided by the norm, to get a unit vector

$$ \mathbf{j} = \frac{\hat{\mathbf{v}}_1 \times \hat{\mathbf{v}}_2}{\|\hat{\mathbf{v}}_1 \times \hat{\mathbf{v}}_2\|}. \tag{2.13} $$

The angle between the two vectors can be calculated by

$$ \cos(\alpha) = \frac{\hat{\mathbf{v}}_1 \cdot \hat{\mathbf{v}}_2}{\|\hat{\mathbf{v}}_1\| \cdot \|\hat{\mathbf{v}}_2\|} = \hat{\mathbf{v}}_1 \cdot \hat{\mathbf{v}}_2 \implies \alpha = \arccos(\hat{\mathbf{v}}_1 \cdot \hat{\mathbf{v}}_2). \tag{2.14} $$

The denominator is equal to 1 (because the norm of a unit vector is 1) and thus omitted. Using eq. (2.7) the rotation can then be expressed as a quaternion.

2.2 Sensors

Sensors are necessary to acquire information about the environment. In this section the functionality and advantages and disadvantages of each sensor used in this thesis will be described.

2.2.1 Inertial Measurement Unit (IMU)

An Inertial Measurement Unit (IMU) is used to track the orientation and position of an object. Common uses are in the aerospace or automotive industry, often in combination with other sensors, to give information about the pose (position and orientation) of a vehicle. More recently with the invention of Microelectromechanical Systems (MEMS) and specifically MEMS-IMUs which allow for a very small form factor at a low cost, IMUs are also used in consumer electronics such as smartphones or fitness tracker. An IMU usually consists of the three following sensors.
The acceleration is measured using an accelerometer and can be used to determine the velocity and the covered distance by integrating the acceleration with respect to time once respectively twice. The gyroscope gives information about the change of orientation. The third part is the magnetometer, which is able to measure the earth's magnetic field and is used to correct the measurements of the gyroscope. It allows for the determination of the absolute heading, whereas the gyroscope can only measure relative change. But because it is very sensitive to other magnetic objects, it is often omitted. IMUs can be typically divided into the two following categories.
In the first type, the stable platform systems, the inertial sensors are mounted in such way, that they are always aligned with the reference frame. This is achieved using gimbals, which allow movement along all three axes. The gyroscopes on the platform measure the rotation and send them to torque motors, which rotate the gimbals to keep the platform in alignment with the reference frame. A typical setup of a stable platform system can be seen in fig. 2.2. The advantage of a stable platform systems is that the calculation of

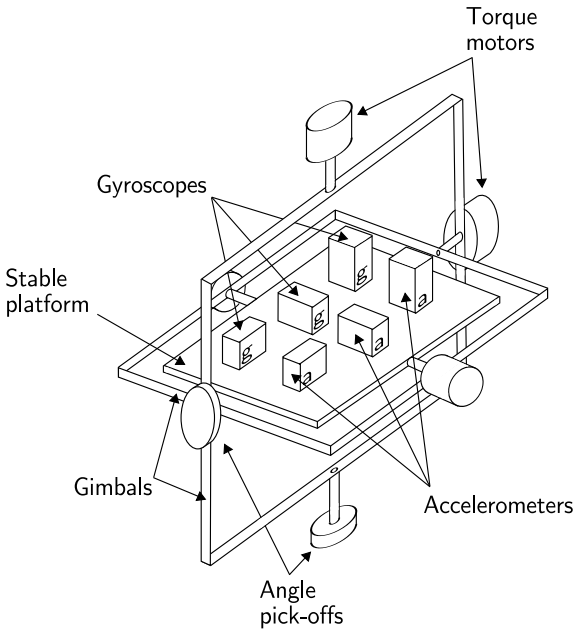


Figure 2.2: Setup of a stable platform IMU [5]. The platform always stays in alignment with the reference frame due to the gimbals.

orientation and position is straightforward. The angles of the gimbals can be measured to receive the orientation and position, the accelerometer measurements have to be corrected for gravity and be integrated two times. No coordinate transformation is necessary. The disadvantages are that the mechanical structure of the setup is complex, needs regular maintenance, requires a lot of space and has high costs.
The second type are strap down systems, which are mostly used today. As the name suggests all the parts are fixed onto the device and are thus not anymore always aligned

with the reference frame. Advantages are that due to the lack of gimbals and motors a significantly smaller build is possible while also being cheaper to mass produce. A disadvantage is that the calculation of the orientation and position is more complex, the rate gyroscopes have to be integrated to get the orientation and can then be used to transform the accelerometer signals into the reference frame. But with the decrease of computational cost this disadvantages continues to diminish.
There are many types of gyroscopes and accelerometers such as mechanical, optical or solid state, but only the functionality of MEMS will be described, because those will also be used in the experiment. Information about the working principle of other systems and also much more information about IMUs in general can be found in [5].
MEMS consist of electrical and/or mechanical components in the size of 100 nm to 1 mm, allowing for a very small form factor. Other characteristics of MEMS are that they can easily be mass-produced allowing for low cost and usually also need less power than traditional systems, because everything is integrated on the chip [6]. Almost all consumer grade electronics uses MEMS-IMUs nowadays, but they also find more and more use in many industry segments, as their accuracy continues to improve [7].

2.2.1.1 MEMS Accelerometer

The accelerometer is used to measure the acceleration. Additional to the dynamic acceleration there is the static and constant gravity acceleration on earth, which is measured by the IMU in upward direction. This allows for the determination of one axis of the IMU, even if it is not moving. Often times only the dynamic accelerations are of interest and to get them the acceleration data during stand still must be measured and subtracted.
The microstructure of a MEMS accelerometer is shown in fig. 2.3. A mass (gray) is

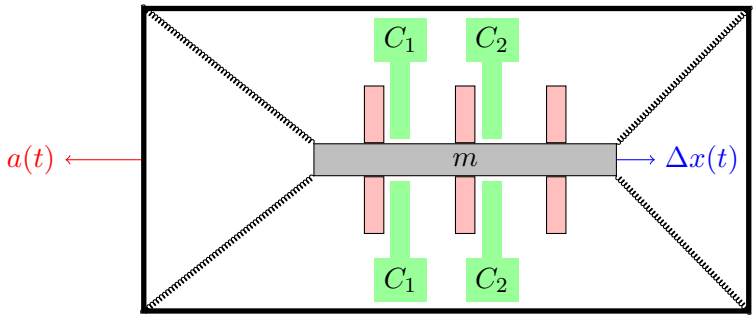


Figure 2.3: The microstructure of a MEMS accelerometer. The proof mass is suspended and moves delayed, in relation to the outer case and fixed capacitors plates, due to its inertia. The acceleration can be calculated from the change in capacity.

suspended by springs along one axis and if an acceleration along this axis occurs, the mass moves in the opposite direction due to Newton's second law. The mass has little fingers (pink) perpendicular to the axis along which the movement occurs, which affect the capacity between the fixed plates (green). The change of capacity and thus voltage can be measured, from which the acceleration can be calculated. To be able to measure the acceleration along all three axis the same setup is used three times, perpendicular to each other.
The accelerations $\hat{\mathbf{a}}$ measured with a MEMS accelerometer are not free of noise and can be modeled according to [8] in a simplified way as

$$\hat{\mathbf{a}} = \mathbf{a} + \mathbf{b}_\mathrm{a} + \mathbf{n}_\mathrm{a}. \tag{2.15}$$

The measurement differs from the true acceleration $\mathbf{a}$ due to random noise $\mathbf{n}_\mathrm{a}$ and a slowly time-varying bias $\mathbf{b}_\mathrm{a}$. More about the specific error terms can be read in section 2.2.1.4.

2.2.1.2 MEMS Gyroscopes

A gyroscope measures the angular velocity. The setup of a MEMS gyroscope is similar to that of a MEMS accelerometer. A proof mass is suspended on a frame and responds to an input force. MEMS gyroscopes make use of the Coriolis effect, which states that a rotating object with the angular velocity $\boldsymbol{\omega}$ of mass m and velocity $\mathbf{v}$ experiences a force

$$\mathbf{F}_{\mathrm{C}} = -2m(\boldsymbol{\omega} \times \mathbf{v}). \tag{2.16}$$

To measure the effect, a mass is vibrating along one axis, which in turn is also suspended. If the mass is oscillating along one axis and a rotation is applied, a second oscillation on the axis perpendicular to the rotation axis can be observed. E.g. if the mass oscillates along the x-axis and a rotation around the z-axis is applied, a vibration along the y-axis can be observed. By measuring the amplitude and phase of the secondary oscillation the absolute value and direction of the angular velocity can be calculated. While MEMS gyroscopes do not achieve the same accuracy as optical gyroscopes they offer many advantages such as smaller physical properties (weight and size), lower power consumption and startup time as well as a significantly lower cost. MEMS gyroscopes have replaced other gyroscope types in most areas, but in areas where the highest precision possible is necessary, typically in military industry, optical gyroscopes are still used today [7].
Similar to the accelerometer, the measurements $\hat{\boldsymbol{\omega}}$ of the MEMS gyroscope are influenced by errors, which can be modeled according to [8] as

$$\hat{\boldsymbol{\omega}} = \boldsymbol{\omega} + \mathbf{b}_{\mathrm{g}} + \mathbf{n}_{\mathrm{g}}. \tag{2.17}$$

The measurement differs from the true angular velocity $\boldsymbol{\omega}$ due to the slowly time-varying bias vector $\mathbf{b}_{\mathrm{g}}$ and the random noise vector $\mathbf{n}_{\mathrm{g}}$, which has a mean of zero.

2.2.1.3 (MEMS) Magnetometer

A magnetometer measures the local magnetic field. Most sensors work using the Hall effect. A current is set to flow through a conductive plate. Without the presence of a magnetic field the electrons flow in a straight line, but if a magnetic field is introduced the electrons do get deflected to one side. The voltage between the two sides can then be measured, from which the strength and direction of the magnetic field can be determined.
Without any magnetic disturbances, the magnetometer measures a constant local magnetic field vector. The vector points to magnetic North and can thus be used to determine the heading. Because the magnitude of the earth's magnetic field is very low (25 µT to 65 µT), the magnetometer readings can easily be influenced by other objects [9].
The distortions can be divided into two categories: hard or soft iron. Hard iron distortions are created by objects which actively produce a magnetic field, causing a permanent bias. Soft iron disturbances are due to deflections or altercations of an existing magnetic field. Both types of disturbances can be removed with a proper calibration if their position and orientation, relative to the sensor, stays the same [10]. Every time the sensor is placed in a (magnetically) new environment, a recalibration is necessary.

2.2.1.4 Typical MEMS Errors

The errors can be divided into two categories: systematic and stochastic errors [11].
Systematic errors or also known as calibration errors are constant over time and can be eliminated by calibration. Typical examples are bias (offset), scaling or axis misalignment. Integrating a constant bias once or twice leads to a drift (error grows linearly with respect to time) or a second-order drift (error grows quadratically) respectively. Hence, the elimination of the bias is necessary to get reliable estimations of the orientation, velocity

or position, which are calculated by integrating the angular velocity or the accelerometer measurements.
Stochastic errors change at every measurement and can be modeled using a statistical approach. The turn-on bias is different every time the IMU is powered up, but can be eliminated after a rest period. Errors due to temperature fluctuations influencing the measurements are also common, but because most IMUs are equipped with a temperature sensor, the introduced error can be eliminated. Harder to correct is the introduced error due to thermo-mechanical noise, which is measured as white noise. The integration of white noise leads to a random walk. Angle errors introduced by random walk are usually the hardest to correct and are the reason, why the measurements of the gyroscope can not be trusted over a long period of time [5].

2.2.2 Light Detection and Ranging (LiDAR)

Light Detection and Ranging (LiDAR) is a method to measure distance to objects. Similar to other systems such as Sound Navigation and Ranging (SONAR) or Radio Detection and Ranging (RADAR), LiDAR uses the Time-of-Flight (ToF) principle. A short laser pulse with the velocity of light c is sent into the environment and the reflected light is analyzed. The duration Δt it took from sending to receiving can then be used to calculate the distance s between the LiDAR and the object that the light hit with

$$s = c\frac{\Delta t}{2}. \tag{2.18}$$

The change of intensity and wavelength of the returning light are measured as well and can provide information about the reflectivity of the object (intensity) or the chemical composition of the air (wavelength). Common uses of LiDAR are the analysis of earth's atmosphere, 3D mapping of environments or in the field of autonomous driving for object detection, tracking and Simultaneous Localization and Mapping (SLAM). Basically all applications which use RADAR can also be used with a LiDAR instead, allowing for a greater accuracy.
There are different LiDAR types, but their working principles are similar. A transmitter generates a signal and sends it into the environment using a scanning system and a transmission optic. As transmitter a laser with a wavelength of 850 nm to 950 nm (near-infrared) is typically used. The scanning system allows the laser to explore a large area

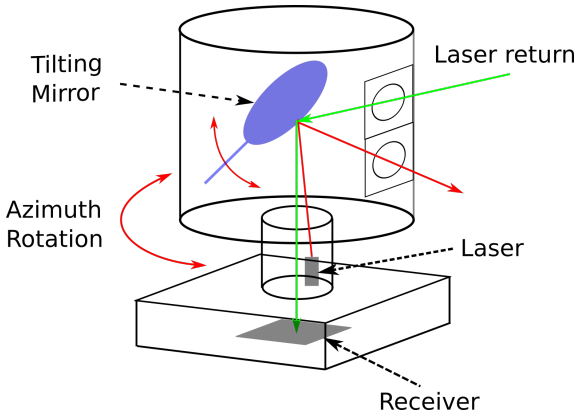


Figure 2.4: Setup of a mechanical spinning LiDAR [12]. The laser is reflected off a tilting mirror and is received through a transmission optic by the receiver. A rotation of the base independently of the mirror allows for a 360° horizontal Field of View (FOV).

instead of only a single point by steering the light at different azimuths and vertical angles

and can be divided in mechanical spinning or solid state systems. Mechanical spinning systems is the oldest technology and is still mainly used today. A mirror which can be rotated around an axis is used, allowing for a greater vertical FOV. Moreover, the whole LiDAR base on which the laser is mounted can be rotated independently from the mirror, allowing for a 360° horizontal FOV. To get a sufficient resolution the LiDAR has to spin at a high speed, but some LiDARs also use additionally a vertical array of lasers instead of only one, to further increase the density of the generated point cloud. The working principle of a LiDAR using the mechanical spinning method is shown in fig. 2.4. While mechanical spinning systems are very precise and offer a great FOV, they are bulky, need a lot of power and are expensive [13].
Solid state systems and especially MEMS LiDARs try to overcome those problems. MEMS-LiDAR are quasi-static, the only part that moves is the on the chip embedded mirror, but because of the small size (1 mm to 7 mm diameter), very little power has to be used to move it. The mirror can be rotated on up to two axes, but because the base cannot be rotated as with mechanical systems, a horizontal view of 360° is not possible. Though by using multiple lasers with different incident angles the FOV can be increased. Advantages of MEMS-LiDARs compared to mechanical systems are the smaller form factor and lower cost [14].
After transmitting the laser signal the reflected light passes through the receiving optic and is received by photodetectors. A processing unit then generates a 3D point cloud from all the received measurements.

2.2.3 Wheel Speed Sensor

The wheel speed sensors measure the speed of each wheel and allow for the calculation of the car velocity. The measurements are also used by many driver assistance systems such as Electronic Stabilization Program (ESP) or Anti-Lock Braking System (ABS) to be able to detect wheel slip. Different techniques exist to measure the speed with the most common ones being magnetoelectric and Hall type wheel speed sensors.
The magnetoelectric sensor is composed of a sensor head and a ring gear. The head is mounted stationary on the car frame while the ring gear is mounted on the wheel hub or axle and rotates with the wheel. The sensor head is composed of a permanent magnet core and a coil. When the wheel and thus the ring gear turns the teeth and gaps of the

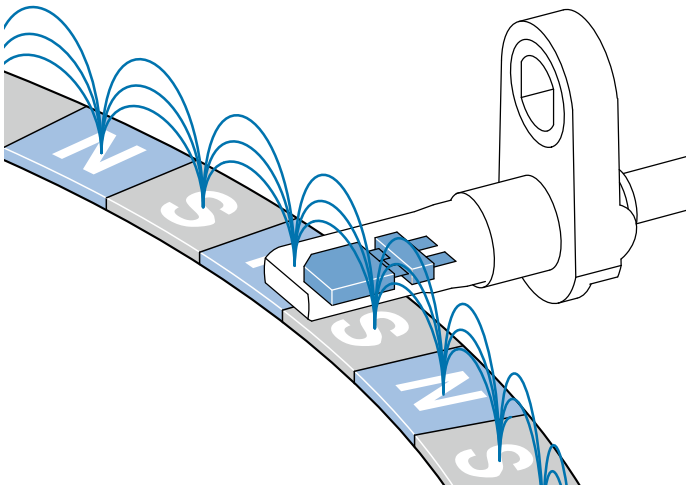


Figure 2.5: Setup of an active wheel speed sensor [15]. The sensor head measures the alternating magnetic field. The amplitude of the induced voltage can then be used to determine the wheel speed.

wheel pass by the sensor head and change the magnetic field which induces an alternating voltage in the coil. The amplitude and frequency of the induced voltage increases with increasing wheel speed. Advantages of the technique are the low cost, robustness and good performance even in the presence of mud etc. A disadvantage is the frequency dependency, very low speeds can not be measured due to the induced voltage being too small, while at very high speeds the changes can not be picked anymore up by the head [16].
Nowadays, almost exclusively the hall wheel speed sensor is used. The functionality is similar to that of the magnetoelectric sensor, but instead of a ring gear, a ring, on which alternating north and south magnets are placed, is used. The hall element in the sensor head measures the alternating magnetic field. A signal amplifier and processing unit is integrated in the sensor head and thus allows for a greater detection rate and range [15]. The setup is shown in fig. 2.5. Due to the embedded processing unit the hall wheel speed sensor is also often called an active sensor, while the magnetoelectric sensor is referred to as passive sensor.

2.3 Computer Vision

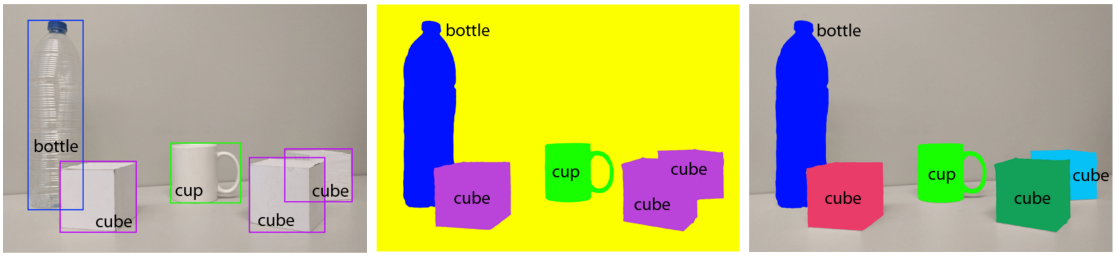
This section gives an overview of computer vision techniques and their applications. In this thesis a trained neural network is used to detect objects in images. The basics of object detection and image segmentation are explained. Furthermore, the framework of the neural network used for the training and its predecessors are discussed. Most of the following information is from [17], which gives a great overview over the field of computer vision.

2.3.1 Object Detection

Object detection is a common task in the field of computer vision and is used to detect, localize and classify objects in images or videos. In contrast to image classification, which only assigns a label to an image, objection detection allows for the detection of multiple objects by drawing a bounding box around each individual object. Object detection can be divided into machine learning-based methods and deep learning-based methods. In the traditional machine learning-based methods, features from the image must be chosen manually. This process is very time-consuming and tedious, that is why nowadays almost exclusively deep learning-based approaches based on Convolutional Neural Network (CNN) are used. Here, the features are extracted automatically.

2.3.2 Image Segmentation

Image segmentation partitions an image into multiple segments or objects. It can be divided into the three categories semantic segmentation, instance segmentation or panoptic segmentation [18].
In semantic segmentation all pixels corresponding to the same class are clustered together and assigned a label. Multiple different classes can be detected, but no difference is made between multiple instances of the same class. Every pixel is associated with a class.
Instance segmentation goes one step further by also differentiating between different individual objects of the same class. It combines objection detection, objection localization and object classification. However, it does not classify every pixel anymore, things that are hard to enumerate such as walls and roads are ignored.
The panoptic segmentation combines both methods. A visualization of the different mentioned methods is shown in fig. 2.6.



(a) Object detection (b) Semantic segmentation (c) Instance segmentation

Figure 2.6: The evolution of object recognition [19].

2.3.3 Neural Networks

An Artificial Neural Network (ANN) is a way for a computer to detect patterns and is inspired by the biological neural networks found in animal brains. It is based on a collection of neurons which are connected with each other. In the input layer each neuron receives a signal, processes it and then sends a signal to its connected neurons. Each neuron has an associated weight and threshold (also known as bias), which determines the activation of the neuron. If the threshold has not been reached, the neuron does not fire. The output of each neuron is calculated by a non-linear function.

2.3.3.1 Convolutional Neural Network (CNN)

A CNN is a class of ANN and is usually used in the field of computer vision. It consists of three different type of layers, convolutional, pooling and the fully connected layers. The convolutional layers help to abstract the input image as a feature map using a filter or kernel. Using overlapping blocks of pixels, each region of the image is scanned. Each block is then assigned a value based on the filter. The pooling layers reduce the complexity by pooling all pixels of a feature map together. Redundant information is removed in this step, which reduces the calculation time, storage requirement and also reduces the risk of overfitting. The last layer is the fully connected layer. In the previous layers neurons of the same layer were independent of each other, whereas in this final layer all neurons are connected to each other. In this layer the actual classification is done. While a CNN provides great improvement, in more complex situations with multiple objects in an image the CNN architecture is not ideal.

2.3.3.2 Region-based Convolutional Neural Network (R-CNN)

That is why in 2014 a deep convolutional network called Region-based Convolutional Neural Network (R-CNN) has been introduced [20], which can detect up to 80 different types of objects in images. R-CNN is a type of machine learning model designed for computer vision tasks, specifically for object detection. It sets bounding boxes across the object regions and then applies convolutional networks independently on all the Region of interests (ROIs). The working principle is shown in fig. 2.7.
In the first step the image is divided into approximately 2000 region proposals using the selective search algorithm. The selective search algorithm groups similar regions based on their color, texture, size and shape compatibility. After extracting a feature vector from each region proposal a pretrained Support-Vector Machine (SVM) algorithm is applied to each region and classifies the proposal either to the background or to one of the object classes. Disadvantages of R-CNN are that the training of the network requires a lot of storage, and it is not possible to run R-CNN in real time. There exist multiple improvement methods with Fast R-CNN [21] and Faster R-CNN [22]. Fast R-CNN greatly improves the

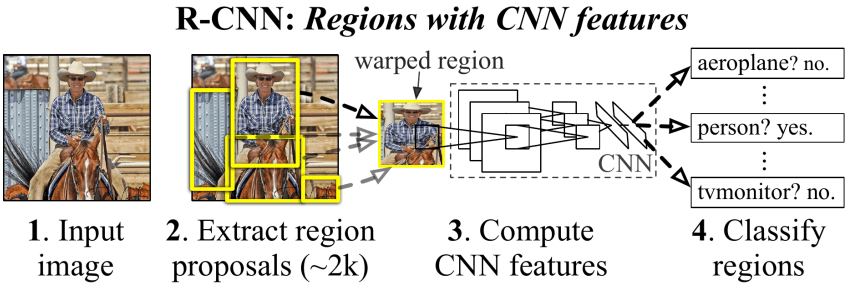


Figure 2.7: The R-CNN framework [20]. The image is divided into subregions and each subregion is classified as one of the object classes or the background independently.

performance by using a new ROI pooling layer, which allows sharing computations across all proposals instead of doing it for each proposal individually. Furthermore, it reduces the necessary storage for training the network while also providing better results. Faster R-CNN improves the performance even further. It uses a Region Proposal Network (RPN) instead of the selective search algorithm to generate region proposals. Advantages of RPN over the selective search algorithm are that the network can be specifically trained and customized for the detection task. And by using the same convolutional layers for the RPN and the Fast R-CNN detection network, the region proposal does not take any extra time. Mask R-CNN [23] is an extension of Faster R-CNN and is a form of instance segmentation, which combines objection detection and semantic segmentation. It adds a third branch to the Faster R-CNN, by also returning an object mask in addition to the bounding box and class label. The process can be seen in fig. 2.8. In the first stage ROIs are selected using a RPN, just as with the Faster R-CNN. The network has only to be run once on the image. But in the second stage, a binary mask is predicted for each ROI in parallel to predicting the class and bounding box.

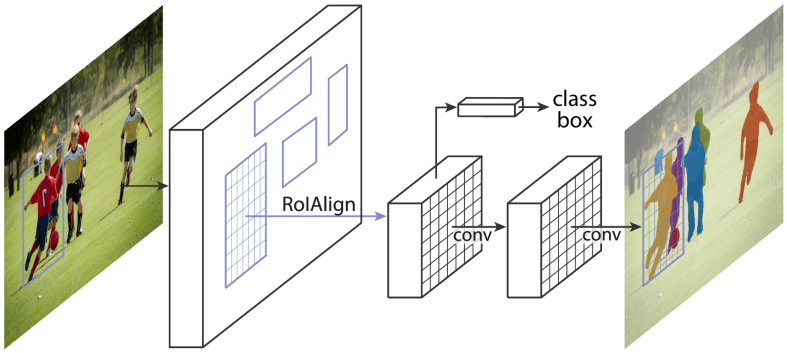


Figure 2.8: The Mask R-CNN framework [23]. It also generates a segmentation mask in addition to the bounding box and class label.

2.4 Robot Operating System (ROS)

Robot Operating System (ROS) is a framework that allows the communication between sensors and actuators of a robot and is typically run on Linux. It is a meta-operating system and provides services such as hardware abstraction and low-level device control. Different languages such as C++, Python or Lisp are supported. The fundamental concepts are nodes, messages, topics and services [24]. A node is a process that performs computation and should be responsible for only one task. A package can contain multiple nodes, which can interact with each other. The communication between nodes is done using messages.

There are different type of messages, but they all consist of standard types such as integer, float or bool, but can also contain other messages. The messages are published on a specific topic. The topic can then be subscribed by other nodes, to retrieve the messages. Services allow for a synchronized communication, instead of the broadcast type topics.
A typical communication setup between two nodes can be seen in fig. 2.9. Node 1 publishes data, e.g. the position of a robot, on a topic. Node 2 listens to the published data and can then process it. All the communication is running through the ROS master.

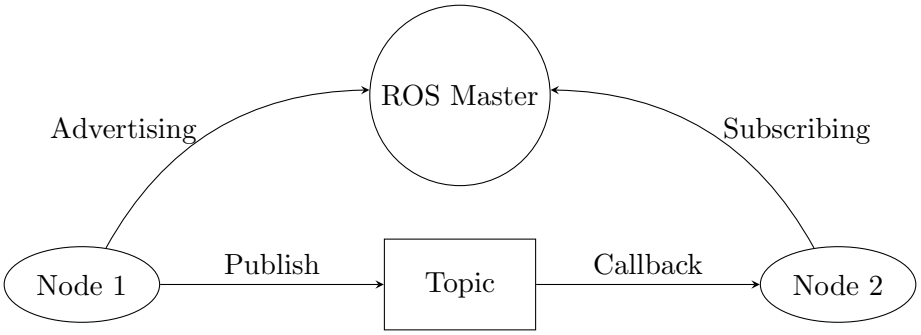


Figure 2.9: ROS communication between two nodes.

Chapter 3

State of the Art

In this chapter an overview of different existing methods, on which a ramp detection algorithm can be build on, will be described. Since a ramp is the connection between two different levels, it must have a measurable slope compared to the ground. Different methods to estimate the road grade are described in the first section. In the second section different approaches to detect objects or shapes in a 3D point cloud or a 2D image will be described.

3.1 Road Grade Measurement

In [25] different methods to estimate the road grade angle are discussed. There exist methods without Inertial Sensors relying on a model describing the longitudinal movement of the vehicle and the topology of the road. Both models are fused using a Kalman filter to improve the accuracy of the estimation [26]. A Kalman filter is also used in [27], where vehicle sensor data and Global Positioning System (GPS) data are fused. Besides the road grade, the vehicle mass is often also unknown and estimated as well, using common sensors of heavy-duty vehicles [27, 28]. More methods such as recursive least squares, extended Kalman filtering and a dynamic grade observer are discussed in [29] Another method using GPS data and IMUs to calculate the vertical and horizontal velocity change respectively and thereby the road grade is proposed in [30]. [31] omits the IMU and relies on a GPS sensor and a barometer.
GPS satellites broadcast information about their position and exact time to a GPS receiver, which than can calculate its position using triangulation [32]. While an accuracy of up to 1 m can be achieved when outside, the performance significantly drops when used indoors. The radio waves sent from the satellites are scattered, attenuated or blocked completely by walls and other obstacles, resulting in a very weak or even a complete loss of the signal [33]. Most of the previously described methods are not suitable for the road grade estimation in indoor requirement, due to the reliance on GPS. Furthermore, many internal measurements such as the engine torque, brake system usage, selected gear etc. can not easily be accessed and thus might not be available.
A method which does not rely on GPS, but only on accelerometers and wheel odometers instead is described in [34] and [35]. The vehicle acceleration, calculated by deriving the wheel speed measurements in respect to time, is subtracted from the accelerometer signal in longitudinal direction. The remaining part is then the gravitational acceleration, which is zero if driving on flat ground, but non-zero if driving on an elevated road, and can be used to calculate the road grade angle. A similar approach is used in [36]. [37] adds a gyroscope to the accelerometer and fuses their estimations using a quaternion unscented Kalman filter. The gyroscope measurements get integrated over time to receive the pitch angle. The angle from the angular velocity is accurate in short-term, but is suspect to

drifting over time. The drift can be corrected by using the accelerometer signal, which is accurate in the long-term, but unlike the gyroscope not accurate in the short-term. [38] uses all components of an IMU (meaning also the magnetometer) and fuses them using a complementary filter. The estimated quaternions using the accelerometer and angular velocity measurements respectively are fused, and the magnetometer data is used to improve the quaternion estimation from the accelerometer, but only if there are no magnetic disturbances. [39] also use a complementary, but fuse the estimated angle from the accelerometer and gyroscope instead of the quaternions.
Due to the available sensor stack, not all mentioned methods can be tested. Only an IMU and wheel speed sensors are available for the experiments. Different IMU based methods will be tested and compared to each other. The most promising results were achieved using the gravity filter, a complementary filter or a Kalman filter. While a Kalman filter achieves very good results, it is generally complex and the precise knowledge of process and measurement noise is necessary [40]. Hence, only the gravity method and complementary filter will be tested and compared to each other, as well as to the estimation when using only the accelerometer or gyroscope data.

3.2 Object Detection and Plane Segmentation

While the in the previous section mentioned methods allow for the estimation of the current road grade angle, they can not be used to detect changes in the road grade ahead of the vehicle. For this purpose, RADAR, LiDAR or camera sensors, to name a few, must be used instead. Since no RADAR sensors were available for the experiments conducted in this thesis, the focus will be on the LiDAR and camera instead.
As described in section 2.2.2, the LiDAR generates a 3D point cloud of the environment. Structures in point clouds can be either identified by using an object detection approach, which allows for the identification of complex objects (e.g. cars or people), or a segmentation approach, in which points are grouped into homogeneous regions.
Many methods use manually crafted features to identify the objects in the point cloud. In some approaches the point cloud is projected into a perspective view, on which then similar feature extraction methods, as used for 2D images, are applied [41, 42]. Other approaches use handcrafted features on a rasterized point cloud to extract the objects [43, 44]. There are also more sophisticated approaches, based on deep learning [45, 46, 47].
Segmentation on the other hand is typically used to classify the point cloud into different geometric shapes, e.g. the ground or a wall in the case of autonomous driving. Several techniques to detect planes in a point cloud exist. As described in [48] the methods to segment 3D point clouds can be divided into five categories: edge based methods, region based methods, attributes based methods, model based methods and graph based methods. Edge based methods segment objects by their shape, by identifying points with a rapid change in intensity [49]. While they allow for fast segmentation, their accuracy is very limited due to their sensitivity to noise and uneven populated point clouds. The region growing algorithms start from certain seed points and add neighboring points to the region if they share a similar model. Different patches can then be merged together, if they are consistent with each other. Disadvantages are that in the initial implementations [50, 51], the performance was highly dependent on a good selection of the starting points. This is why [52] introduced a new type of region based method, which is based on a top-down approach, where all points are grouped into one region, which is then successively divided into smaller regions. The attributes based methods are more robust and cluster points with similar attributes. An advantage is that the attributes can be chosen according to the problem, but this also means that the quality of the segmentation is highly dependent on a good selection of the attributes. The fourth type of segmentation method is the model

based method, which will also be used in this thesis. It uses geometric primitive shapes such as planes, cylinders, spheres or cones to group points together. Almost all of those methods are based on the Random Sample Consensus (RANSAC) algorithm [53]. It is an iterative algorithm which randomly selects some points to build a shape (e.g. plane, circle or line) and then calculates the error of the other points to the proposed shape. If a good model has been found, all the inliers are extracted and the next biggest shape is searched for. More about the RANSAC algorithm will also be described in section 4.4.1. There exist several improvements such as the efficient RANSAC algorithm [54], which improves the performance, or the method proposed in [55], which allows for the detection of slippable shapes (e.g. a helix). Attribute based methods are fast and very robust to outliers, the biggest drawback is that they are inaccurate when dealing with different point cloud sources. Lastly, the graph based methods consider the point cloud as a graph. Each vertex corresponds to one point in the point cloud and the edges connect to certain neighboring points. The FH algorithm [56] is one of the most well known. They are very accurate, but can usually not be run in real-time.
For the task of detecting a ramp, using a segmentation approach is sufficient, since a ramp can be seen as a planar surface. By also detecting the ground plane, the angle of the ramp and the distance to it can be estimated. Furthermore, they tend to be faster than object detection approaches. Due to the simplicity of the object of interest, the RANSAC algorithm will be used. Since it only detects planes in the point cloud, a method to robustly identify ramps in the point cloud and estimate the ramp properties must be implemented.

Cameras can also be used to generate a point cloud, on which the same methods as described above can be applied. The point clouds can be generated in different ways, some of the most common technologies are stereo cameras and ToF cameras. Stereo cameras consist of two slightly shifted cameras. The difference between the two images is used to generate a disparity map. If the baseline (the distance between the two cameras), the focal length and the image size are known, the depth can be calculated from the disparity map. A common problem is the finding of the differences between the two images, since it requires a well light environment and does not work well when the scene has very few textures. Furthermore, they tend to only work in a short range.
Those problems are not present when using ToF cameras. They are active sensors, unlike stereo cameras, and work similarly as a LiDAR, by sending out light and measuring the time it takes to reflect back. But they are scannerless, which means that they capture the entire scene with a single light pulse. Since they are active sensors, they can also be used in low light conditions and do not depend on well textured scene. But they are not as accurate as LiDAR sensors.
There is also some research done on detecting ramps specifically in a point cloud. [57] introduces a new algorithm, to extract planar maps from 3D data. The point cloud is divided into 3D cells and a histogram over the z-axis is created. Ramps can then be detected by searching for neighboring cells, where the height increases gradually. [58] used an RGB-D sensor (camera image + depth sensor) to detect ramps for wheelchairs. Ramp properties such as angle, width, length and the orientation of the ramp are determined as well.
While the image from a monocular camera does not provide any 3D information, it can also be used to detect and localize objects. A brief overview into the field of computer vision was already given in section 2.3. While traditional object detection approaches were build on handcrafted features, nowadays most are based on deep learning [59]. Due to the rapid development in deep learning, they consistently outperform traditional computer vision approaches [60]. Thanks to the rise of deep neural networks [61], and specifically the Region-based Convolutional Neural Network (R-CNN) [20] the prediction accuracy could be improved significantly. As mentioned in section 2.3.3.2, there exist several improvements of

the initially proposed R-CNN algorithm, such as the Fast R-CNN [21], Faster R-CNN [22] or the You Only Look Once (YOLO) algorithm [62], which detects objects using fixed-grid regression. While the mentioned methods allow for real time detection of objects, they only localize them using a bounding box.
Image segmentation expands the object detection, by detecting all objects in the image, and also segmenting each instance in a pixel-to-pixel manner. This allows for a more accurate localization of the objects. As described in section 2.3.3.2, one of the most popular ones is the Mask R-CNN [23], which builds on the Faster R-CNN [22] and adds a layer to the network, in which a segmentation mask is predicted. There exist several object detection libraries such as `ImageAI` [1] [63] , `GluonCV` [2] [64] or `Detectron2` [3] [65] to name a few. `Detectron2` is an open-source library developed by Facebook and supports many state-of-the-art detection and segmentation algorithms. It also supports the Mask R-CNN algorithm. Due to its very good documentation and popularity, many pretrained models are already available. Using transfer learning, they can be used as a basis for the training of a new model. This allows for a great reduction in training time and increase in accuracy. Hence, the `Detectron2` library and the Mask R-CNN network will be used in this thesis.

[1] https://github.com/OlafenwaMoses/ImageAI
[2] https://github.com/dmlc/gluon-cv
[3] https://github.com/facebookresearch/detectron2

Chapter 4

Methods

In this chapter various methods using different sensors to estimate the car's pitch angle and thus the ramp angle, as well as other properties of the ramp, such as the width and length, will be described. The problem definition is discussed in section 4.1. Before being able to use the sensor measurements, a coordinate frame transformation to the device frame is necessary. This problem is described in section 4.2. In section 4.3 multiple methods to estimate the car's pitch angle using an IMU are described. The angle is then used to identify a ramp. Furthermore, the length of the ramp can then be estimated. A novel method using a LiDAR sensor will be presented in section 4.4. It tracks the distance to the ramp and estimates the angle, width and length of the ramp. Finally, a ramp detection algorithm using a camera and a trained neural network will be presented in section 4.5.

4.1 Road Grade Definition

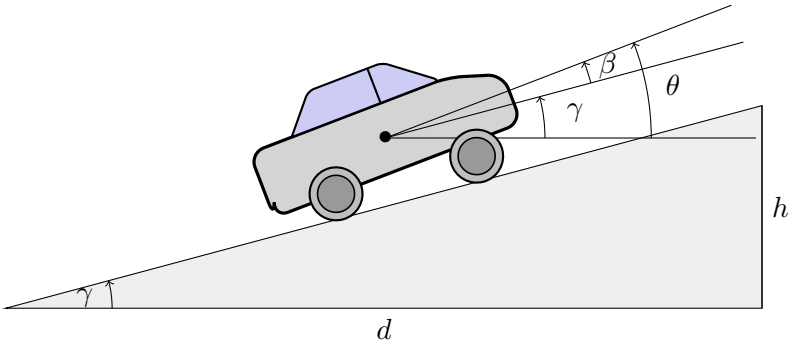


Figure 4.1: Car driving on a ramp. Due to forward acceleration the car tilts back.

The road grade γ is the angle between the road plane and the ground plane. The ground plane is perpendicular to the gravity vector. The road grade can be represented as an angle

$$\gamma = \arctan\left(\frac{h}{d}\right) \tag{4.1}$$

or in percentage

$$r = 100 \cdot \tan(\gamma). \tag{4.2}$$

The pitch angle θ of the car is defined as the angle between the ground plane and the longitudinal axis of the car. If the car accelerates or decelerates the suspension does get compressed in the back or respectively front, which makes the pitch angle not in alignment with the road grade anymore. The difference between the two angles is defined as $\beta = \theta - \gamma$

and may also occur due to rotational movement or vibrations. The mentioned variables are visualized in fig. 4.1.

4.2 Coordinate Frames

A common problem is that the coordinate frames of the sensors are usually not aligned with the device (in this case the car) frame, see fig. 4.2. To get meaningful results, the sensor frames must first be aligned with the car frame. Semi-automatic calibration methods for the IMU and LiDAR sensor will be presented in section 4.3.1 and section 4.4.1 respectively, which determine the necessary rotation to align the sensor frame with the car frame. But the translation difference between the coordinate frames can not be easily estimated or requires a more sophisticated calibration setup which was not available, and must be measured by hand. The coordinate frame of the car has the x-axis pointing forward in

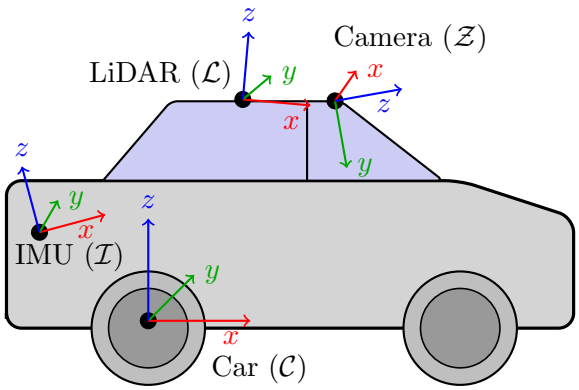


Figure 4.2: Example of a typical sensor setup. All measurements of the sensors are recorded in the sensor frame and must be transformed to the car frame.

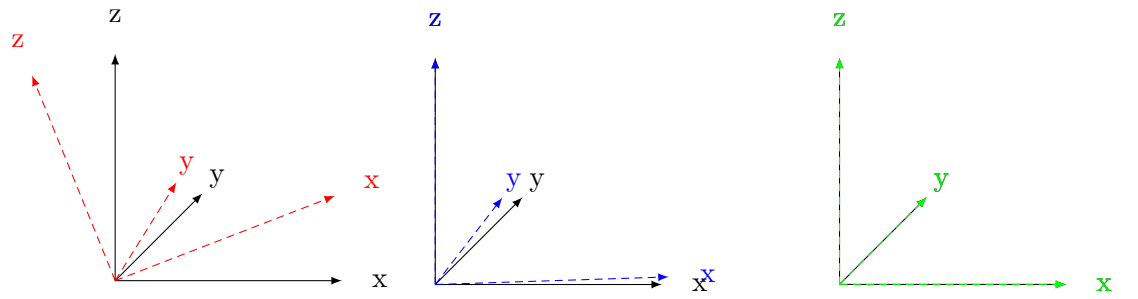
the direction of travel, the y-axis to the left and the z-axis upwards. The center of the coordinate system of the car is located at the transverse center of the rear wheel axle.

4.3 IMU-based Ramp Detection

Before being able to apply the different methods, a transformation from the sensor frame to the car frame is necessary. This is described in section 4.3.1. After the transformation an estimation of the car's pitch angle is possible, for which different methods are presented in section 4.3.2. In section 4.3.3 a ramp detection algorithm based on the estimated pitch angle is presented. Furthermore, a method to estimate the average ramp angle and length of the ramp is proposed.

4.3.1 Calibration

The IMU is usually not placed in such a way, that the coordinate frame of the device $\mathcal{I}$ aligns with that of the car $\mathcal{C}$, see fig. 4.3a. Because of that, a transformation between the two frames must be found. This can be achieved using a rotation matrix ${}^{\mathcal{I}}_{\mathcal{C}}\mathbf{M} \in \mathbb{R}^{3\times3}$ which transforms the measurements of the linear acceleration ${}_{\mathcal{I}}\mathbf{a}_k \in \mathbb{R}^{1\times3}$ and angular velocity ${}_{\mathcal{I}}\boldsymbol{\omega}_k \in \mathbb{R}^{1\times3}$ into the car frame. Note that the upper index to the left of the matrix symbol denotes the source frame, whereas the destination frame is written below it. $k \in \mathbb{N}$ is the time step of the measurement.
During standstill, the only measurable acceleration besides noise and bias is the acceleration due to gravity. Assuming the car stands on flat ground, the gravity acceleration in the car



(a) No axes are aligned (b) Z axes are aligned (c) All axes are aligned

Figure 4.3: Frame transformation from the IMU frame $\mathcal{I}$, over the intermediate frame $\mathcal{B}$ to the car frame $\mathcal{C}$.

frame is measured only in upwards z-direction. Using this information, a transformation from the IMU frame $\mathcal{I}$ to the intermediate frame $\mathcal{B}$ can be found. In the new $\mathcal{B}$ frame, both z-axes are aligned ${}_{\mathcal{B}}\mathbf{z} = {}_{\mathcal{C}}\mathbf{z}$ and thus the pitch and roll angle between the two frames become zero. Note that this is not necessarily true for the other axes, ${}_{\mathcal{B}}\mathbf{x} \neq {}_{\mathcal{C}}\mathbf{x}$ and ${}_{\mathcal{B}}\mathbf{y} \neq {}_{\mathcal{C}}\mathbf{y}$, see fig. 4.3b. According to Euler's rotation theorem [66], which says that any arbitrary rotation of a rigid body while holding one point (origin) fixed, can be achieved by a rotation around a single fixed axis passing through the origin, there exists one rotation axis $\mathbf{j}$ and rotation angle α to achieve this.
As described in section 2.1.2, the rotation axis needed for the transformation can be calculated by

$$\mathbf{j} = \frac{{}_{\mathcal{I}}\hat{\mathbf{a}} \times {}_{\mathcal{C}}\hat{\mathbf{z}}}{\|{}_{\mathcal{I}}\hat{\mathbf{a}} \times {}_{\mathcal{C}}\hat{\mathbf{z}}\|} \tag{4.3}$$

and the rotation angle with

$$\alpha = \arccos({}_{\mathcal{I}}\hat{\mathbf{a}} \cdot {}_{\mathcal{C}}\hat{\mathbf{z}}) \tag{4.4}$$

with ${}_{\mathcal{I}}\hat{\mathbf{a}} \in \mathbb{R}^{1\times 3}$ being the normalized measured linear acceleration in the IMU frame and ${}_{\mathcal{C}}\hat{\mathbf{z}}$ the (normalized) z-axis of the car.
The quaternion

$${}^{\mathcal{I}}_{\mathcal{B}}\mathbf{q} = \begin{pmatrix} \mathbf{j} \cdot \sin\left(\frac{\alpha}{2}\right) \\ \cos\left(\frac{\alpha}{2}\right) \end{pmatrix} \tag{4.5}$$

then describes the rotation between the two frames.
Now that the z-axes of the $\mathcal{I}$ and $\mathcal{C}$ frame are aligned, the x- and y-axis can be aligned by a rotation β around the z-axis. This yaw correction could usually be achieved using the magnetometer measurements, but because those are heavily obscured indoors and especially in the parking garage [67], another solution must be found. A possible solution to this problem is accelerating the car straightforward and then using the accelerometer to measure along which axis the acceleration occurred. Assuming the car tilt (pitch) during the acceleration is minimal, the acceleration is only being measured along the x- and y-axis. The resulting vector is being aligned with the forward axis of the car, such that ${}_{\mathcal{B}}\hat{\mathbf{a}} = {}_{\mathcal{C}}\mathbf{x}$, in the same way as before. This results in the rotation angle

$$\beta = \arccos({}_{\mathcal{B}}\hat{\mathbf{a}} \cdot {}_{\mathcal{C}}\hat{\mathbf{x}}) \tag{4.6}$$

and the quaternion

$${}^{\mathcal{B}}_{\mathcal{C}}\mathbf{q} = \begin{pmatrix} {}_{\mathcal{C}}\hat{\mathbf{z}} \cdot \sin\left(\frac{\beta}{2}\right) \\ \cos\left(\frac{\beta}{2}\right) \end{pmatrix}. \tag{4.7}$$

The two quaternions can then be concatenated (in reverse order) to get the final quaternion

$$\overset{\mathcal{I}}{\mathcal{C}}\mathbf{q} = \overset{\mathcal{B}}{\mathcal{C}}\mathbf{q} \otimes \overset{\mathcal{I}}{\mathcal{B}}\mathbf{q}, \tag{4.8}$$

which transforms the measurements of the IMU to the car frame. The quaternion is then converted into a rotation matrix using eq. (2.12), because it reduces the computation time. Each new measurement is then transformed from the sensor frame to the car frame using

$$_{\mathcal{C}}\mathbf{a}_k = \overset{\mathcal{I}}{\mathcal{C}}\mathbf{M} \cdot {}_{\mathcal{I}}\mathbf{a}_k \tag{4.9}$$

$$_{\mathcal{C}}\boldsymbol{\omega}_k = \overset{\mathcal{I}}{\mathcal{C}}\mathbf{M} \cdot {}_{\mathcal{I}}\boldsymbol{\omega}_k. \tag{4.10}$$

4.3.2 Road Grade Estimation

4.3.2.1 Accelerometer

If the car stands still, the only measurable acceleration beside measurement errors is the acceleration due to gravity. When standing on flat ground, only the z-axis measures an acceleration. But when the car is tilted, e.g. on a ramp, the gravity is measured also by the x-axis (which points forward), see fig. 4.4. The proportion of the acceleration measured along the x-axis $\mathbf{a}_\text{x}$ of the overall gravity $\mathbf{g}$ can then be used to determine the pitch angle in the following way

$$\theta_\text{acc} = \arcsin\left(\frac{\mathbf{a}_x}{\|\mathbf{g}\|}\right) \tag{4.11}$$

with $\|\mathbf{g}\|$ being the magnitude of the overall measured acceleration. According to the definition, the angle is zero if the car is parallel to the ground and 90° if the front of the car would be pointing straight up. Thus, the angle is positive when driving up a ramp and negative if driving down.
Disadvantages of this method are that the acceleration measurements are quite noisy and that no other accelerations are taken into account, e.g. the on track acceleration $\mathbf{a}_\text{car}$ caused by the motor. This is a problem when the car has a non-zero acceleration, because the acceleration measured by the IMU along the x-axis is given by

$$\mathbf{a}_\text{x} = \mathbf{a}_\text{car} + \mathbf{g}_\text{x} \tag{4.12}$$

and $\mathbf{a}_\text{x} = \mathbf{g}_\text{x}$ only holds true, when the car is driving with constant velocity or standing still. A better approach, which incorporates the accelerations caused by the car, is described in section 4.3.2.4.

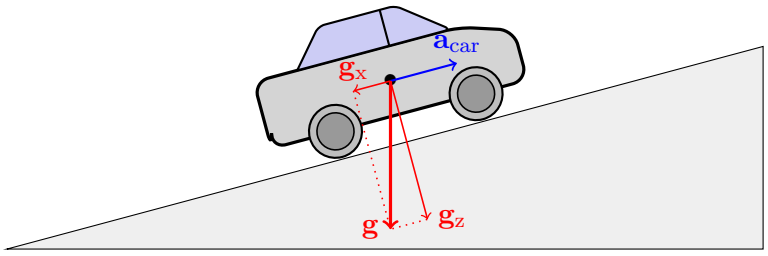


Figure 4.4: The prevailing accelerations when a car is accelerating on a ramp.

4.3.2.2 Gyroscope

The gyroscope measures the angular velocity, which must be integrated with respect to time, to get a rotation angle. The angular velocity describes the rate of change of an angle and is defined as

$$\boldsymbol{\omega} = \frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t}. \tag{4.13}$$

Assuming that the measurements of the gyroscope are continuous in time, the current angle $\boldsymbol{\theta}(t)$ at time T can be computed with the integral

$$\boldsymbol{\theta}(t = T) = \boldsymbol{\theta}_0 + \int_0^T \boldsymbol{\omega}(t)\,\mathrm{d}t \tag{4.14}$$

with $\boldsymbol{\theta}_0$ being the initial angle.
In practice, however, the IMU provides samples at discrete times k and $k-1$. Assuming that the signal remains constant during the time Δt between the two samples, the integral can be numerical approximated. An error will be introduced by the approximation, but it will be small if the sample rate is sufficiently high. The new angle after a change between two samples can be calculated with

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{\mathrm{k-1}} + \boldsymbol{\omega}_k \Delta t. \tag{4.15}$$

Similarly, the angle based on multiple consecutive measurements can be calculated by the sum of all the previous measurements

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_0 + \sum_{i=1}^{k} \boldsymbol{\omega}_i \Delta t \tag{4.16}$$

with $\boldsymbol{\theta}_0$ being the initial angle at the start of the measurement, which must be known beforehand. To get the pitch angle θ_{gyr}, only the measurements around the y-axis are of interest (see fig. 4.2). The pitch angle at time k is given by

$$\theta_{\mathrm{k,gyr}} = \theta_{0,\mathrm{gyr}} + \sum_{i=1}^{k} \boldsymbol{\omega}_{\mathrm{i,y}} \Delta t. \tag{4.17}$$

Disadvantages of using only the angular velocity are that the estimations are not reliable over a long period of time. The random walk introduced by integrating white noise or a constant bias causes the estimation to drift away from the true value.

4.3.2.3 Complementary Filter

The complementary filter uses both the linear acceleration and angular velocity measurements and combines them using sensor fusion, such that the good properties of each sensor are used to reduce the poor properties of the other. The angle estimation obtained from the linear acceleration measurements is reliable in the long-term, but is quite noisy. The estimation from the angular velocity measurements on the other hand provide good short-term accuracy, but should not be used for longer estimations due to drift.
These properties can be interpreted in the frequency domain. The error from the accelerometer data is subject to high frequency noise, whereas the estimation error from the gyroscope is mostly due to low frequency noise [68]. To minimize the error of the linear acceleration estimate, a Low-Pass Filter (LPF) should be used. A LPF passes all signals with frequency lower than a certain cut-off frequency f_0 and attenuates signals with a frequency above f_0. In contrast, a High-Pass Filter (HPF) should be used on the estimate of the gyroscope. A HPF works exactly opposite to a LPF. It blocks signals with a frequency below f_0 while allowing signals over this frequency to pass through [69]. A block diagram of the complementary filter is shown in fig. 4.5.
The pitch angle estimation obtained from the linear acceleration (eq. (4.11)) will be referred to as θ_{acc}, the pitch angle estimation from the gyroscope estimation as θ_{gyr} and the fused estimation of the complementary filter will be denoted as $\hat{\theta}$. As seen in eq. (4.17), θ_{gyr} is calculated by integrating the angular velocity $\boldsymbol{\omega}$ measured by the gyroscope.

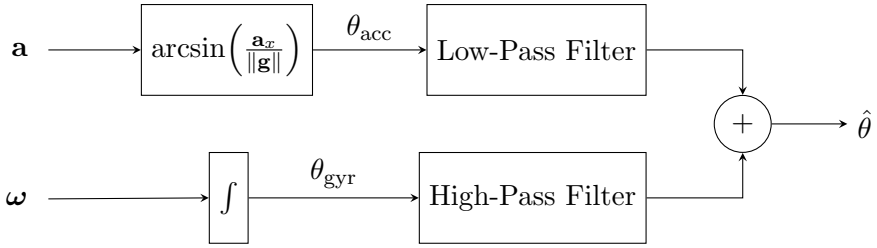


Figure 4.5: The block diagram of the complementary filter.

The Laplace transformation can be used to transform the angles from the time domain to the frequency domain. The angles θ_{acc}, θ_{gyr} and $\hat{\theta}$ will be denoted as $\Theta_{\text{acc}}(s)$, $\Theta_{\text{gyr}}(s)$ and $\hat{\Theta}(s)$ in the frequency domain, the gyroscope measurements $\boldsymbol{\omega}$ as $\Omega(s)$. s is a complex variable and is defined as $s = j\omega$. The complimentary filter estimate is computed by

$$\begin{aligned}\hat{\Theta}(s) &= G_{\text{low}}(s)\Theta_{\text{acc}}(s) + (1 - G_{\text{low}}(s))\Theta_{\text{gyr}}(s) \\ &= G_{\text{low}}(s)\Theta_{\text{acc}}(s) + (1 - G_{\text{low}}(s))\frac{1}{s}\Omega(s)\end{aligned} \tag{4.18}$$

where $G_{\text{low}}(s)$ is the transfer function of a LPF and $1 - G_{\text{low}}(s) = G_{\text{high}}(s)$ the of a HPF [3]. The sum of $G_{\text{low}}(s)$ and $1 - G_{\text{low}}(s)$ is equal to one, which means that the cut-off frequency of both filters must be the same. The transfer functions are defined as

$$G_{\text{low}}(s) = \frac{1}{1 + \alpha s} \tag{4.19}$$

for the LPF and

$$G_{\text{high}}(s) = \frac{\alpha s}{1 + \alpha s} \tag{4.20}$$

for the HPF, when using a first-order filter. α is the filter coefficient and is dependent on the cut-off frequency in the following way

$$\alpha = \frac{1}{w_0} = \frac{1}{2\pi f_0}. \tag{4.21}$$

Inserting eq. (4.19) and eq. (4.20) in eq. (4.18) and using inverse Laplace transformation as well as Euler backward discretization, the complementary filter can be written in discrete time as

$$\hat{\theta}_k = \zeta\left(\hat{\theta}_{k-1} + T\omega_{k,\text{gyr}}\right) + (1 - \zeta)\theta_{k,\text{acc}}, \tag{4.22}$$

with

$$\zeta = \frac{\alpha}{\alpha + T} \in [0, 1] \tag{4.23}$$

being the filter gain. The choice of the parameter α (and hence ζ) determines, how much each of the two measurements should be trusted. Selecting a small α (ζ close to zero) results in a high cut-off frequency. The estimation then mostly uses the accelerometer data. Contrary, a high value of α leads to a ζ close to one and low cut-off frequency, where the gyroscope estimation is trusted more [70].
Selecting the right filter coefficient is a known problem and is usually solved by calculating an initial guess followed by a fine-tuning by hand to get the desired result. One option to get an initial guess is by measuring the drift of the gyroscope and selecting the filter coefficient accordingly. The time T_{free} where the drift of the angle estimation from the gyroscope is negligible, is inversely proportional to the cut-off frequency $f_0 = \frac{1}{T_{\text{free}}}$. E.g. if the drift from the gyroscope is only negligible for 10 s then the cut-off frequency should be chosen at $f_0 = \frac{1}{10\,\text{s}} = 0.1\,\text{Hz}$ or higher. The filter gain can then be calculated using eq. (4.21) and eq. (4.23). Using the previous example and assuming a sample frequency of 100 Hz, the filter gain would then be $\zeta = 0.9938$. For the experiments done in this work different gains were tested, the best results were achieved using a filter gain of $\zeta = 0.99$.

4.3.2.4 Gravity Method

As described in section 4.3.2.1, the linear acceleration measurements can be used to determine the pitch angle. But the estimation is only valid under the condition, that there are no accelerations other than the acceleration due to gravity. This condition is not necessarily true when the car is driving, during which the car can accelerate or brake. To get the correct estimation, the car's acceleration must be subtracted from the measurement.

Car acceleration from odometer data
Using the wheel speed measurements, the velocity of the vehicle can be calculated. Because the wheel speed sensors only deliver the speed of each wheel, a model has to be used to estimate the velocity of the car. This is necessary, because during turns the left and right wheels travel at different speeds, the wheel on the inner side of the turn travels slower, than the outer wheel. A simple yet sufficiently accurate model to calculate the car velocity from the wheel speeds is the linear single track model [71]. In this model both wheels on one axis are replaced with one wheel in the middle. The linear assumption holds true for low lateral accelerations (up to $4\,\frac{\mathrm{m}}{\mathrm{s}}$), which will not be surpassed in the experiments. Using the assumptions from above, the car velocity $v_{\mathrm{car}}(t)$ in forward direction can be calculated by

$$\alpha(t) = \frac{v_{\mathrm{rl}}(t) - v_{\mathrm{rr}}(t)}{d} \tag{4.24}$$

$$\gamma(t) = \frac{\alpha(t)}{f} \tag{4.25}$$

$$v_{\mathrm{car}}(t) = \frac{v_{\mathrm{rl}}(t) + v_{\mathrm{rr}}(t)}{2} \cdot \cos(\gamma) \tag{4.26}$$

with v_{rl} and v_{rr} being the wheel speeds of the rear right and rear left wheel respectively, d the track width, f the rate of the measurements and γ is the yaw angle of the car, which can be calculated from the steering wheel angle.
To calculate the acceleration of the vehicle, the first derivative of the velocity must be taken, using

$$a_{\mathrm{car}}(t) = \frac{\mathrm{d}}{\mathrm{d}t} v_{\mathrm{car}}(t). \tag{4.27}$$

But because all measurements are discrete, numerical differentiation is necessary, it can be approximated using the backward difference

$$a_{\mathrm{k,car}} = \frac{v_{\mathrm{k,car}} - v_{\mathrm{k-1,car}}}{T} \tag{4.28}$$

with k being the current time step and $T = \frac{1}{f}$ being the step size.
Because the from the wheel speed sensors calculated velocity is discrete in time, quantized and not free of noise, a simple numerical derivation would amplify the noise. Hence, the measurements must first be filtered. One solution is to use a moving average filter, which is a simple average of the last N measurements

$$\hat{v}_{k,\mathrm{car}} = \frac{1}{N} \sum_{i=1}^{N} v_{k-i,\mathrm{car}}. \tag{4.29}$$

The finite difference of the approximation can then be calculated as

$$\hat{a}_{\mathrm{k,car}} = \frac{\hat{v}_{\mathrm{k,car}} - \hat{v}_{\mathrm{k-1,car}}}{T}. \tag{4.30}$$

Angle calculation

The prevailing accelerations during a positive acceleration can be seen in fig. 4.4. When the car brakes, the direction of $\mathbf{a}_{\text{car}}$ inverts. The car pitch angle calculation is the same as in eq. (4.11), just that now the car acceleration $\mathbf{a}_{\text{car}}$ is taken into account.

$$\theta_{\text{grav}} = \arcsin\left(\frac{\mathbf{a}_{\text{x}} - \hat{a}_{\text{car}}}{\|\mathbf{g}\|}\right) \tag{4.31}$$

with $\mathbf{a}_{\text{x}}$ being the acceleration measured by the x-axis of the IMU and $\|\mathbf{g}\|$ being the magnitude of the overall measured acceleration. It is important that both the IMU and odometer measurements are synchronized in time, otherwise a change in acceleration leads to a wrong estimation. Since $\hat{a}_{\text{car}}$ is slightly delayed due to the filtering, the same filter must be applied to the IMU measurements.

4.3.3 Ramp Detection and Classification

In the previous section different methods to estimate the pitch angle θ of the car were presented. Assuming that the car is accelerating moderately and other factors which might engage the suspension, such as road vibrations or movement in the car are minimal, the estimated pitch angle can be assumed to be close to the road grade γ. Using the estimated road grade, it can be determined whether the car is on a ramp. If the absolute number of the road grade surpasses a certain threshold, the part is classified as a ramp, otherwise it is classified as a normal road.

Using the IMU, properties of the ramp such as the angle and length of the ramp can be determined. Because the angle of the ramp is not constant, but gradually increases and decreases at the beginning and end respectively, the average angle is calculated. It is assumed that the middle part of the ramp has a constant angle. The average angle of the ramp is then calculated by measuring the pitch angle and checking whether the rate of the change is small for a certain amount of time. If this is the case, it is assumed that the car is in the middle of the ramp and the angle is then computed by averaging the last n measurements, to decrease the influence of measurement noise.

The length of the ramp can be calculated by integrating the velocity of the car over time. The velocity can be calculated in two different ways. Using the wheel speed measurements (eq. (4.26)), or by integrating the accelerometer measurements along the x-axis. The problem when using the accelerometer data is that also other accelerations than the one caused by the car are measured. For example if the car is on a ramp, the acceleration due to gravity is measured as well by the IMU. But by using eq. (4.11) and transforming it to

$$\hat{a}_{\text{car}} = \mathbf{a}_{\text{x}} - \sin(\theta)\|\mathbf{g}\| \tag{4.32}$$

the acceleration due to gravity can effectively be removed from the measurements, under the assumption, that the estimated car pitch angle is close to the true value. The length l of the ramp can then be calculated by first integrating the acceleration in x-direction with respect to the time to get the velocity

$$\hat{v}_{\text{i,car}} = \hat{a}_{0,\text{car}} + \sum_{i=1}^{k} \hat{a}_{\text{i,car}}\Delta t \tag{4.33}$$

and then integrating the velocity to get the length

$$l = \sum_{i=1}^{k} v_{\text{i,car}}\Delta t \tag{4.34}$$

where the velocity $v_{1,\text{car}}$ is the velocity at the start of the ramp and $v_{\text{k,car}}$ is the velocity at the end of the ramp. Δt is the sample time. In eq. (4.34) either the estimated velocity $\hat{v}_{\text{car}}$ using eq. (4.33) or the velocity v_{car}, calculated by using the wheel speed measurements (eq. (4.26)), can be used.

4.4 LiDAR-based Ramp Detection

This section is similarly structured as the previous section. At first the calibration process to transform the LiDAR measurements from the sensor frame to the car coordinate system is described in section 4.4.1. Then, one novel method to detect ramps before entering them is presented in section 4.4.2. Finally, it is described how different ramp properties such as the angle, width, length and distance to the ramp are estimated after a ramp has been detected.

4.4.1 Calibration

Same as for the IMU measurements, a transformation from the sensor frame to the car frame is necessary. The calibration procedure is very similar to that of the IMU. At first both z-axes will be aligned. This is achieved by detecting the ground plane in the point cloud and projecting it onto the xy-plane of the car frame. In other words, the necessary rotation to align the normal vector of the ground plane with the z-axis of the car is found. Now the only remaining rotation is the rotation around the z-axis of the car, also known as yaw angle. The yaw angle can not be easily determined and is assumed to be equal to zero, but can also be measured by hand and given as parameter.
For the ground plane detection the Random Sample Consensus (RANSAC) algorithm [53] is used. RANSAC is a non-deterministic algorithm to remove outliers and is often used in computer vision. RANSAC can also be used for plane segmentation in 3D point clouds. Consider a point cloud with n points, where point i has the coordinates x_i, y_i, z_i. In a first step, three random points from the point cloud are selected. Three, because this is the minimum number of points necessary to form a plane. Now the parameters a, b, c, d of the plane equation

$$ax + by + cz + d = 0 \tag{4.35}$$

can be calculated. Then for every other point the deviation r from the proposed plane can be calculated by

$$r = \frac{ax_i + by_i + cz_i + d}{\sqrt{a^2 + b^2 + c^2}} \tag{4.36}$$

and is then summed up. If the distance is within a certain threshold, the point counts as an inlier. After iterating through the whole point cloud, the number of inlier points and their coordinates are stored. This process is then repeated until the maximal number of iterations are reached. The plane with the greatest number of inliers is then selected.
After applying the RANSAC algorithm and receiving a plane equation, the normal vector of the proposed plane, which can be conducted from the plane equation as follows

$$\mathbf{n} = \begin{pmatrix} a & b & c \end{pmatrix}^\intercal, \tag{4.37}$$

is projected onto the z-axis of the car. The necessary rotation is then applied to the detected plane. Now that a plane has been found it must be ensured, that it really is the ground plane. Typically, either the ceiling, ground or a side wall gets detected with the RANSAC algorithm in the environment of the setup used for the experiments. The greater the plane is (or the more points lie inside a plane), the more likely is the detection of the

plane. Due to the mounting and Field of View (FOV) of the LiDAR, the ceiling usually does have the most points and is thus detected in the first iteration.
An accidental ceiling detection can be prevented by looking at the average z-values of the detected plane after the applied rotation. Because the LiDAR is mounted on the roof of the car, the z-values of the ground plane must be negative. If they are positive, the ceiling has been detected. Furthermore, it is known that due to the mounting of the LiDAR on the car, for which it was tried to keep the yaw and roll angle close to zero, the calculated roll angle should be minimal. If that is not the case a side wall has most likely been detected. If either condition has not been fulfilled, the detected plane gets removed from the point cloud and using RANSAC a new ground plane estimation is made and validated. This process gets repeated until both conditions are fulfilled.
To allow for the measuring of the distance from the car to the ramp, the translation between the sensor frame and car frame must be determined as well. The translation difference in z-direction is calculated by measuring the average z-values of the points in the ground plane. But the x- and y-translation must be measured manually. And since the distance to the ramp should specify the distance from the front of the car to the ramp, the distance from the coordinate center of the car to the front must be added. And while the pitch and roll angle between the LiDAR frame and car frame are estimated in the calibration process, the determination of the yaw angle is not easily possible. Therefore, the yaw angle is assumed to be zero.

4.4.2 Algorithm

4.4.2.1 Point Cloud Preprocessing

Because the raw LiDAR point cloud consists of many points and is too big to allow for real time processing on the available setup, preprocessing is necessary. It consists of a passthrough filter to remove unwanted points (e.g. behind the car) and a voxel grid filter to downsample the point cloud. Before the passthrough filter can be applied, the point cloud must be transformed to the car frame. The in the previous section described calibration algorithm is performed once at the start and its returned rotation is then applied to every new measurement.
The passthrough filter then removes all the points which lie outside the specified x, y and z limits. Because the car drives forward, only points in front of the car are of interest. Furthermore, the points further away than a certain threshold are neglected, because the resolution and accuracy of the measurements of the LiDAR decreases with increasing distance. The ceiling points are removed by limiting the points in z-direction.
The next step in reducing the point cloud size is the voxel grid filter [72]. The point cloud is converted into a 3D grid consisting of small cubes called voxels. Each cube can contain multiple points or none, the size of the voxels (also known as leaf size) determines the resolution. All the points inside a cube are then reduced to their most centroid point. If the cube does not contain any points, it is neglected. The exact values used for the passthrough filter and the voxel filter are listed in table 4.1.

4.4.2.2 Ramp Detection and Classification

Now that the point cloud size is reduced greatly the actual ramp detection can be performed with sufficient performance. For this task the RANSAC algorithm is used again. It usually detects the following types of planes: ceiling, ground, side wall or the desired ramp. RANSAC is applied iteratively until a plane of type ramp has been found. If a plane of different type has been found, it gets removed and the RANSAC algorithm is applied again. To prevent an infinite loop, the algorithm will exit after either a certain number of iterations has been performed, or if after the removal of a plane not enough points to

Table 4.1: The parameters used for the preprocessing of the point cloud.

Parameter	**Value**	**Unit**
Passthrough filter		
x	0 to 30	m
y	−2 to 2	m
z	−1 to 2	m
Voxel filter		
leaf size	0.1	m

form a sizeable plane are left in the point cloud. For the implementation of the RANSAC algorithm and the voxel grid filter the PCL library [73], and more specifically the python binding `python-pcl` [1], has been used.
The accidental detection of the ceiling was already prevented during the passthrough filter step, where the ceiling points have been removed from the point cloud. Wrong detections of the ground plane or side walls as ramp plane can be prevented by requiring the roll and pitch angle respectively to be in a certain range. Since the RANSAC algorithm provides the normal vector of the plane, the angle of the detected plane can be compared to the normal vector of the ground plane, which is $(0\ 0\ 1)^\intercal$ after the calibration. The normal vector of the detected plane is then projected on the ground vector and the rotation between both is calculated as described in section 2.1.2. The rotation is then converted to Euler angles. Since the LiDAR generates a 3D model of the environment, other properties of the ramp can be estimated as well. Besides using the angle of the detected plane to determine whether the detected plane is the ramp, the width of the ramp is used as well. The width of the ramp is calculated by taking the mean of the m most left and right points (y-axis) respectively and calculating the absolute difference between them. It is used to classify the ramp as a proper drivable ramp for cars. If the width is not greater than a certain threshold, it could also be a small ramp for e.g. wheelchairs and should not be labeled as a ramp. The length of the ramp is calculated similarly, by taking the mean of the n nearest and furthest points (x-axis) respectively and calculating the absolute difference between them. Furthermore, the distance to the ramp is estimated by taking the o nearest points of the ramp plane and calculating their mean. In the calculation of the width, length and distance to the ramp, the mean is used to further reduce the possibility of outliers. Even though most potential outliers should already have been removed during the RANSAC process. In the end, the angle, width and distance to the ramp are returned, if a ramp has been detected.
A visual representation of the full algorithm is depicted in fig. 4.6. At the start the calibration is performed once and the returned rotation to align the point cloud with the car frame is applied to every new measurement. The size of the point cloud is then reduced by the voxel grid filter and the passthrough filter. Using the RANSAC algorithm a plane is extracted from the point cloud and validated. If it fulfills the angle and width requirements, it is classified as a ramp and the estimated angle, width, length and distance to the ramp are returned. Otherwise, the plane is removed from the point cloud and the RANSAC algorithm is applied again. This process is repeated until either not enough points are left in the point cloud for the RANSAC algorithm to find a plane or if a certain number of iterations has been exceeded.

[1] https://github.com/strawlab/python-pcl

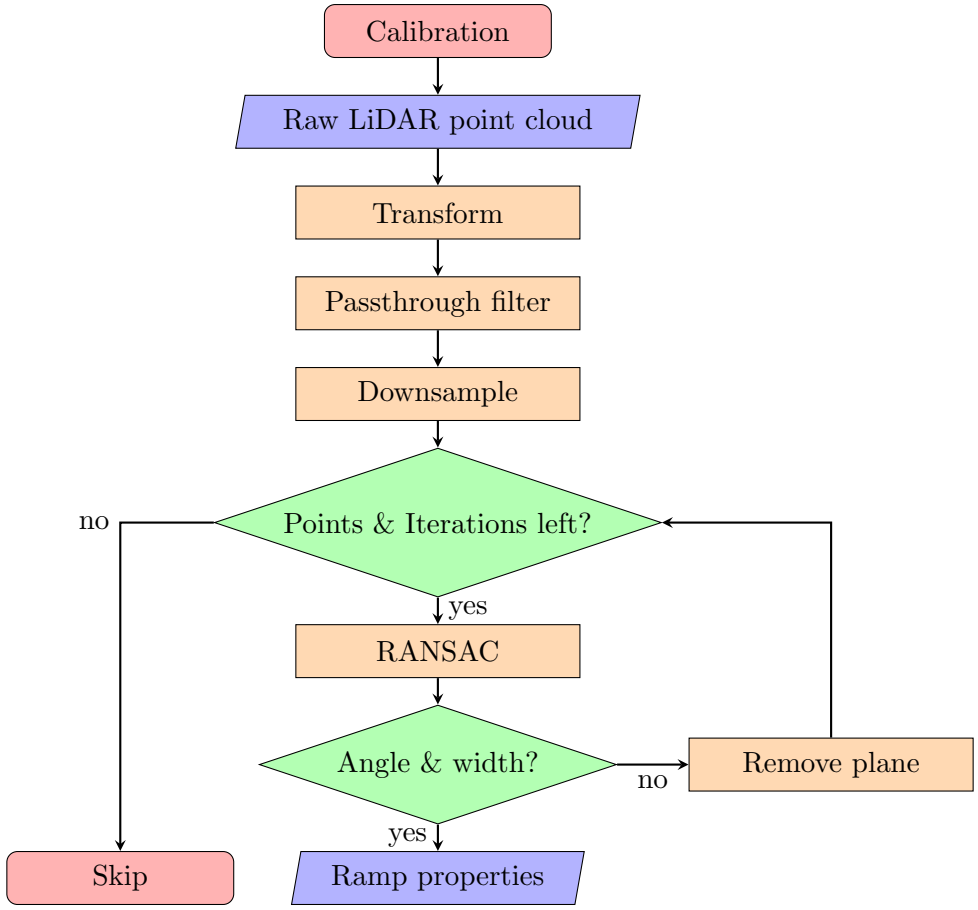


Figure 4.6: Flow chart of the ramp detection algorithm using the LiDAR sensor.

4.5 Camera-based Ramp Detection

In this section an approach using the images captured by the camera to detect a ramp is presented. At first, the choice of a deep learning network as the method of choice to solve this task is explained. In the next section the generation of the dataset used for the training is described. The training of the network and the choice of the different hyperparameters is discussed in section 4.5.3. And in the last section it is described, how the 2D prediction of the network can be used, to extract the ramp region from a point cloud.

4.5.1 Approach

Different techniques to detect objects exist, more classic methods like edge detection and thresholding or more sophisticated methods like machine learning. While less complex methods like edge detection and thresholding might work, they require a well lit environment with a high contrast between different textures, which is not the case in the parking garage. Furthermore, the features need to be chosen manually, and the algorithm would most probably only work in a very limited certain environment.
That is why a deep learning method has been chosen instead. It is more robust and can easily be adapted to other environments. The choice of the network depends on the problem. Different approaches such as object detection, instance or semantic segmentation are available. For this project, the instance semantic approach was chosen. While the object detection approach requires the least computational effort, it only provides a bounding box around the object. Both segmentation approaches on the other hand create a mask around the object, allowing for a more accurate localization of the object. For the specific task of

detecting ramps in parking garages the semantic segmentation is good enough since most times only one ramp will be visible, in which case the semantic and instance segmentation will be the same. But the instance segmentation approach is more general and can be adapted more easily to support also the detection of other objects.
That is why the Mask R-CNN architecture [23] is used in this thesis. It allows for instance segmentation, which means that multiple objects of the same class can be detected as distinct objects. While for this task only one class is used (ramp) and most times only one object will be visible, the network can be easily expanded to support multiple classes and is already supporting the detection of multiple ramps at once. Furthermore, it is well documented, and different implementations are already available for use.
For this thesis the open-source library `Detectron2` [2] [65] developed by Facebook is chosen as the framework. It is widely used and also provides a number of pretrained networks, which can be used for transfer learning. Transfer learning is the process of training a new model on top of an existing one. This provides multiple advantages over training a new model from scratch, such as a great reduction of the training time and hence also the use of resources, a better accuracy and reducing the risk of overfitting.
A model trained on the Common Objects in Context (COCO) dataset [74] is used as the basis for the training of the network. COCO is a large-scale dataset consisting of more than 200 000 labeled images with various annotations (e.g. bounding boxes, segmentation masks, image captions) of 80 different object categories (e.g. car, chair, dog, person, train). It is often used to benchmark neural network algorithms and to compare their performance. Since no class of type ramp already exists in the COCO dataset, an own dataset has to be created.

4.5.2 Dataset

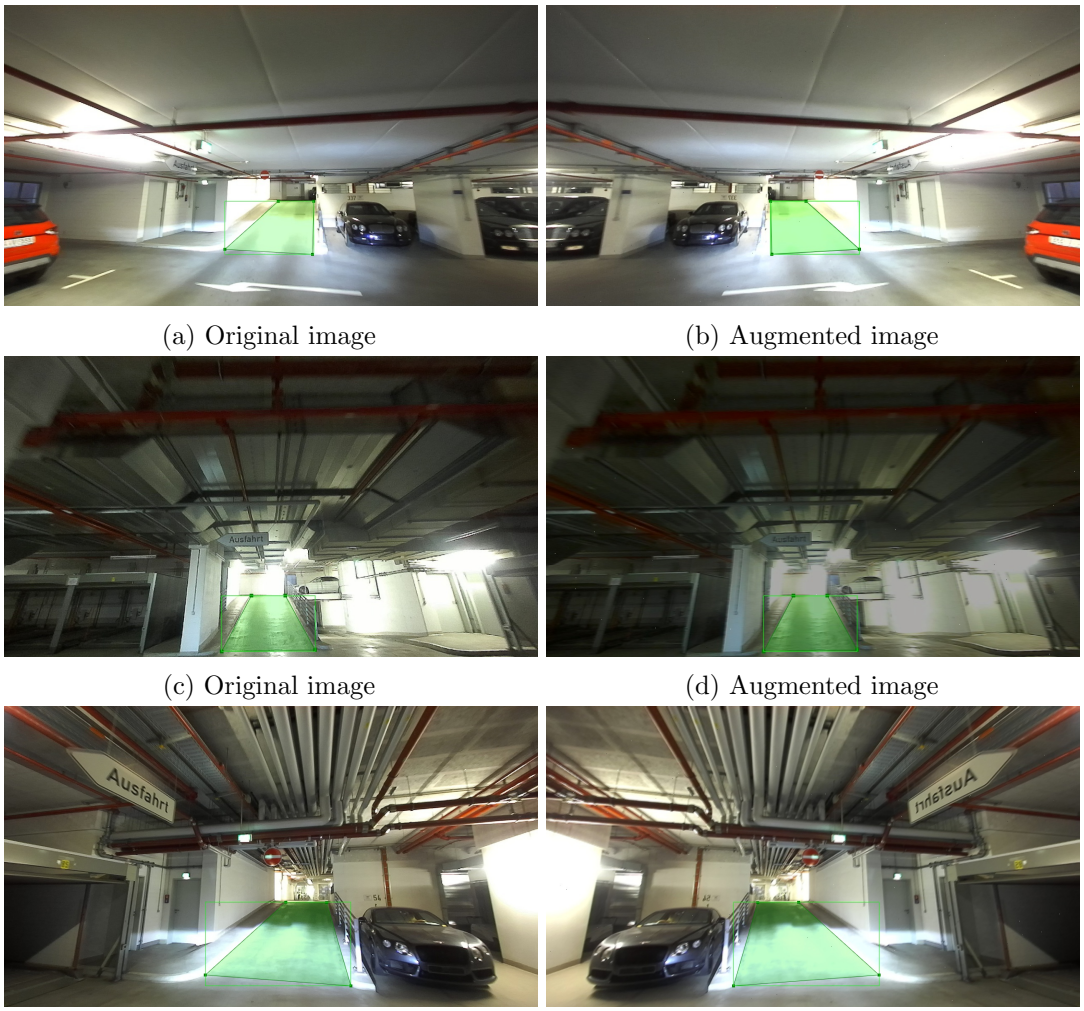
While several datasets specifically for the task of autonomous driving already exist, no dataset with ramps as a class could be found. That is why an own dataset had to be created, by taking pictures of a ramp and labeling them by hand. To reduce the workload, for the creation of the dataset only straight ramps of one specific parking garage are considered. An image of each of the three ramps used for the training are shown in fig. 4.7 on the left-hand side. Multiple recordings of each ramp were made from different distances and angles, by mounting a camera on the roof of a car and driving the car in the direction of the ramp. The images were taken from a video with a frame rate of 30, but since the car approached the ramp with a low speed of 5 $\frac{\text{km}}{\text{h}}$, the difference between two frames is very small. Since many pictures of the more or less same scenery do not provide any extra information, while increasing the training time and the potential of overfitting, only every 30th image (one every second) was used. In the end, 144 different images of three ramps were collected.
Since the training of the network is based on supervised learning, the images have to be labelled manually. For this task the tool `labelme` [3] [75] is used. Each ramp is labelled using a polygon consisting of four points, placed at the corners of the ramp. Only the drivable area of the ramp is marked, and the curbsides are not considered. The `labelme` tool creates an annotation containing the coordinate of each point and the class of the marking. The format of the annotation has to be converted to the COCO format, to allow for the training using the `Detectron2` library. The bounding box format is

$$\begin{pmatrix} x_{\min} & y_{\min} & w & h \end{pmatrix}$$

where $x_{\min}$ and $y_{\min}$ are the coordinates of the upper left corner of the bounding box and w and h are the width and height of the box. The COCO format contains information

[2] https://github.com/facebookresearch/detectron2
[3] https://github.com/wkentaro/labelme

about the segmentation mask consisting of the coordinates of the polygon, the bounding box and the class of the object.



(a) Original image (b) Augmented image

(c) Original image (d) Augmented image

(e) Original image (f) Augmented image

Figure 4.7: The three ramps used for the training. The augmented images are shown on the right-hand side and the manually created label is marked green.

4.5.3 Training

Since the size of the dataset is not very large, the training of the network is challenging, because it increases the risk of overfitting. To alleviate the problem, data augmentation is used in addition to the previously mentioned transfer learning. Data augmentation provides a good alternative to when the collection of more data is not feasible or possible [76]. It increases the size of the dataset by adding slightly modified versions of the original images. Common transformations are horizontal or vertical flipping of the image, cropping, adding noise, blur or changing the brightness and contrast of the image. The library `imgaug` [4] [77] is used for this task. In addition to editing the image, it automatically adapts the annotation according to the new position of the label, e.g. when flipping the image. The following augmentations techniques are used to artificially increase the dataset:

- Horizontal flipping of the image with a probability of 50%

[4] https://github.com/aleju/imgaug

- Changing the brightness by $-20\,\%$ to $20\,\%$
- Changing the contrast by $-20\,\%$ to $10\,\%$
- Adding motion blur with a kernel size of 3
- Adding salt and pepper (turning some pixels black or white, simulating dead pixels)

Some examples of the augmented images are shown in fig. 4.7 on the right-hand side. In each picture the annotation is shown as a mask and bounding box in green. Note that the label is automatically transformed when flipping the image. All the added changes are also common occurrences in the real world, which should help to improve the accuracy of the network in other environments. One augmentation is performed on each image which is then added to the dataset.
To evaluate the performance of the network, the dataset has to be split into a training and a validation set. The split is done before the data augmentation to ensure, that the training set does not contain any only slightly modified versions of the images used in the validation set. Due to the small size of the dataset, 80% of the dataset is used for the training and 20% for the validation set.
The accuracy of the network depends on the correct choice of multiple hyperparameters. Hyperparameters allow for the optimization of the model and are set manually before training, unlike the internal model parameters, which are estimated while training the model. Since the dataset is small and thus the training using transfer learning does not take very long, different hyperparameter configurations could be tested to find the best one. One of the most important parameters is the learning rate. It is often in the range of 0.0 to 1.0 and describes how fast the weights of the network are updated. Is the learning rate too low, the training of the network takes a longer time and might not converge to find a solution, whereas if the learning rate is too high, a suboptimal solution might be found. The learning rate is often initially set to a high value and then slowly reduced, which helps both optimization and generalization. The speed of the decay of the learning rate can be based on many things, such as the number of steps in the training, the time, or it can be reduced exponentially. Another parameter, specifically for the Mask R-CNN neural network architecture, is the number of Region of interest (ROI) proposals. As described in section 2.3.3.2, the image is divided into a grid of ROIs. The number of ROI proposals determines how many of those ROIs are taken into account for the calculation of the loss function. It can be chosen lower than the size of the grid to accelerate the training. The final parameter which will be varied is the number of epochs. It describes how often the network is trained on the entire dataset, a higher number of epochs means a longer training time, but usually also a better performance, except if it is chosen to high, in which case it can lead to overfitting. Because the dataset is small, has only one class and contains similar images, the number of epochs can be chosen fairly small.
For the hyperparameter optimization a grid search is performed. Grid search is an exhaustive search method, which tests combinations of different hyperparameters. At first, the influence of the different parameters on the score is tested by randomly testing some values. Also, the rough lower and upper bounds of each parameter, after which the performance decreases, are determined. During this process it was also found out, that a decay of the learning rate does not significantly affect the score, due to the low number of epochs, and therefore the learning rate will not be varied in the grid search. After this the grid search is performed. The lower and upper bound of each parameter together with the number of steps are specified for each parameter and then all the combinations of the parameters are tested. The tested combinations and the resulting scores will be presented in section 6.6

4.5.4 Point Cloud Extraction

The ZED 2i stereo camera used in this thesis also generates a point cloud. More information about the camera will be described in section 5.1.3. The 3D point cloud can be projected onto the 2D camera image. For the projection, the rotation vector and translation from the point cloud frame to the camera frame is necessary, as well as the intrinsic camera matrix $\mathbf{A}$. Because the ZED 2i stereo camera records both the point cloud and camera image at once, the rotation and translation between the two frames is zero. Using the `projectPoints` function of the `opencv` [5] library [78], the point cloud is projected onto the camera image. The camera image is then fed into the network, which predicts a bounding box and segmentation mask of the ramp. The bounding box is in the form $(\, x_{\min} \; x_{\max} \; w \; h \,)$ with $x_{\min}, y_{\min}$ being the coordinates of the upper left corner and w, h the width and height of the bounding box. The segmentation mask is stored as a boolean array of the same size as the image. Pixels that are part of the object are marked as `true` and the other pixels as `false`. Then, using the predicted bounding box or the segmentation mask, all the projected points outside the box or mask are removed. The points are then transformed back into the 3D space. Since it is possible that multiple points get projected onto the same pixel, some information might be lost during this step. To prevent this, the generated boolean array is artificially upscaled.
Now that only points inside the ramp region are left, the RANSAC algorithm can be applied to remove outliers and to estimate the normal vector of the plane, which is used to calculate the angle of the ramp. Different properties of the ramp can now be calculated, as described in section 4.4.2.2. Instead of using the point cloud generated by the camera, the point cloud of the LiDAR can be used as well. The LiDAR point cloud is more accurate, but the projection of the LiDAR points onto the camera image is more difficult since the rotation and translation difference between both sensor frames is not zero anymore and must be measured, which is subject to error. The speed of the RANSAC algorithm is greatly accelerated compared to the LiDAR based method, since only a subset of the points is left due to the removal of the points outside the ramp region. Hence, the full resolution can be used instead of the downsampled version, while still achieving sufficient performance.

[5] https://github.com/opencv/opencv

Chapter 5

Experimental Setup

To evaluate the performance of the proposed methods, different test drives must be conducted using a car with the appropriate sensor setup. In this chapter the properties of the sensors used during the recordings and their placement on the car are described. Information about the car and its limitations are described in section 5.4. Finally, in section 5.5, the different type of ramps and their properties are presented.

5.1 Sensors

5.1.1 IMU

(a) Withrobot myAHRS+ [79] (b) Stereolabs ZED 2i camera [80]

Figure 5.1: The two IMUs which will be used for the recordings. The Stereolabs ZED 2i camera has an integrated IMU.

Two different IMUs will be used for the experiment.
The first one being the Withrobot myAHRS+ (see fig. 5.1a), a low-cost high performance Attitude Heading Reference System (AHRS). An AHRS contains an IMU and outputs the raw measurements of the sensors, but also has an on-board processing system which estimates attitude and heading. The myAHRS+ uses an extended Kalman filter for the estimation and outputs the estimation in quaternion form and also in Euler angles.
The second IMU used during the experiment is integrated in the Stereolabs ZED 2i stereo camera and is also an AHRS. A picture of the camera can be seen in fig. 5.1b. More information about the ZED 2i camera and their other integrated sensors and functionalities will be described in section 5.1.3.
Both sensors are connected to a computer via USB. The specifications of each IMU is provided in table 5.1. Because no information about the noise density or random walk of the myAHRS+ IMU could be found, a manual estimation of the error was done using the

package `allan_variance_ros` [1]. It analyzes the Allan variance from IMU measurement data recorded over a two-hour period, during which the IMU has not been moved. The smaller values of the ZED 2i IMU indicate, that it is the better sensor.

Table 5.1: Comparison of the two used IMUs [79, 80]

Property	**myAHRS+**	**ZED 2i IMU**	**Unit**
Accelerometer range	±16	±8	g
Gyroscope range	±2000	±1000	$\frac{\circ}{s}$
Magnetometer range	±1200	±2500	µT
Rate	100	400	Hz
Accelerometer noise density	4.502×10^{-3}	1.148×10^{-3}	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$
Accelerometer random walk	7.337×10^{-5}	6.458×10^{-5}	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$
Gyroscope noise density	1.674×10^{-4}	8.254×10^{-5}	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$
Gyroscope random walk	5.042×10^{-6}	1.632×10^{-7}	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$

5.1.2 LiDAR

(a) Robosense RS-Bpearl [81] (b) Velodyne Ultra puck [82]

Figure 5.2: The two LiDARs which will be used for the recordings.

Two different LiDARs will be used during the experiment. The RS-Bpearl and the Velodyne UltraPuck, see fig. 5.2. The most relevant specifications of the two LiDARs can be seen in table 5.2. Both are mechanical LiDARs and have the same number of laser channels, but the Velodyne has a significant better vertical resolution, due to the smaller vertical Field of View (FOV). This is because the RS-Bpearl is intended to be used as near-field blind-spots detection LiDAR mounted on the side where coverage is more important than resolution. The Velodyne UltraPuck on the other hand has been specifically designed for being mounted on the roof and hence also has a greater range.
Both LiDARs need an external power supply and the data transfer to the PC is done via Ethernet connection. Due to the setup of the test car it is only possible to mount one LiDAR at a time.

[1] https://github.com/gaowenliang/imu_utils

Table 5.2: Comparison of the two used LiDARs [81, 82]

Property	**RS-Bpearl**	**Velodyne Ultra Puck**	**Unit**
Channels	32	32	
Points per second	576,000	600,000	
Laser wavelength	905	903	nm
Frame rate	10 to 20	5 to 20	Hz
Range	100	200	m
Range accuracy	±3	±3	cm
Horizontal FOV	360	360	deg
Vertical FOV	90	40 (−25 to 15)	deg
Horizontal resolution	0.2 to 0.4	0.1 to 0.4	deg
Vertical resolution	2.81	0.33	deg

5.1.3 Camera

The ZED 2i camera from Stereolabs as seen in fig. 5.1b will be used during the experiment to record the camera image. The camera has two horizontally displaced lenses, allowing for stereo vision and thus also depth estimation. A barometer, temperature sensor and an IMU are integrated as well. It also provides many more features such as 3D positional tracking, mapping, object detection and point cloud generation.
Because only one image is needed, only the image of the left camera is used. The resolution is set to 1280x720 and a frame rate of 30 Frames per Second (FPS) is used. The camera is connected to the PC through USB.

5.2 Sensor Placement

To ensure the best possible performance of the system, the sensors must be placed in a specific way. The IMU must be placed on a rigid point of the car, such that the IMU's position always stays the same relative to the car. Other than that it should also be placed in the transversal center of the car to guarantee that the centripetal acceleration is not skewed towards one side when driving around a corner. The placement of the IMU integrated in the ZED 2i camera is limited, because the camera images are needed as well. It was placed on the roof of the car, which is sturdy enough to not be suspect to flexion, while also allowing for a good FOV for the camera. The myAHRS+ IMU was mounted on the floor of the car trunk.
The LiDAR is placed on top of the car, to get a greater FOV. The pitch angle β at which the LiDAR will be mounted should be chosen such that the number of points in the area at the beginning of the ramp are maximized. This allows for the most accurate distinction between planes of different inclination angles. Because the distance to the ramp is not constant due to the movement of the car, the optimization can only be done for a specific distance. The coordinates at which the lasers hit the ground and ramp depend on the height of the LiDAR h_l, the distance to the ramp d, the angle of the ramp γ, the angle β at which the LiDAR has been mounted on the car and finally on the vertical resolution ϵ and FOV of the LiDAR. A visualization of the different variables can be seen in fig. 5.3. The coordinates at which the laser lines of the LiDAR hit the ground can be calculated in the following way.
The angle κ between the plane parallel to the ground at LiDAR height and each laser wave

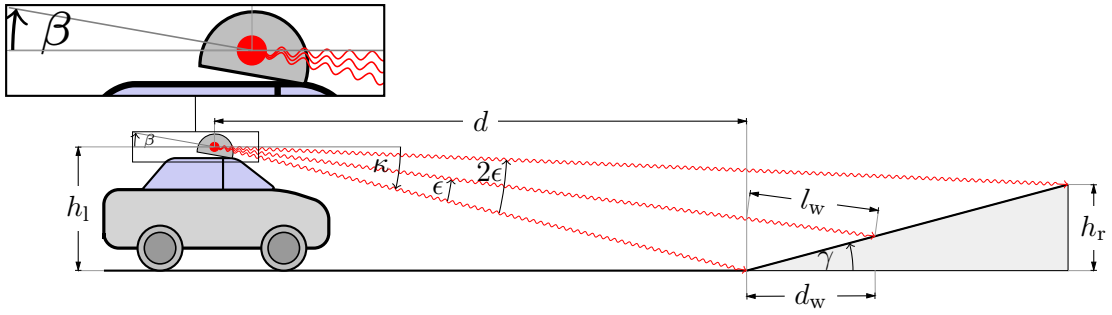


Figure 5.3: Calculation of the best mounting angle of the LiDAR on the car. Variable description in table 5.3.

is defined as

$$\kappa = \beta - i\epsilon \tag{5.1}$$

with $i \in [0, 1, 2, \ldots, n]$ being the laser channel ID starting from the lowest opening angle and going to the highest and n being the number of laser channels. On flat ground the distance at which the laser waves hit the ground can be calculated by

$$d_{\text{hit,ground}} = \tan(90° - \kappa)h_l. \tag{5.2}$$

With a ramp, the assumption from eq. (5.2) does not hold anymore. The light does not travel as far. The height above ground, when the light is at the beginning of the ramp can be calculated by

$$h_{\text{w,start}} = h_l - d\tan(\kappa). \tag{5.3}$$

The distance l_w which the light travels from the beginning of the ramp to the contact point with the ramp can be calculated using the law of sines

$$\begin{aligned} l_w &= \frac{h_{\text{w,start}}}{\sin(\gamma + \kappa)} \sin(90° - \gamma) \\ &= -\frac{h_{\text{w,start}}}{\sin(\gamma + \kappa)} \cos(\gamma). \end{aligned} \tag{5.4}$$

The traveled distance along the x-axis from the start of the ramp to the contact point is then

$$\begin{aligned} d_w &= l_w \sin(90° - 2\gamma - \kappa) \\ &= -l_w \cos(2\gamma + \kappa). \end{aligned} \tag{5.5}$$

Putting everything together, the total x distance from the LiDAR to the contact point on the ramp can be calculated by

$$\begin{aligned} d_{\text{hit,ramp}} &= d + d_w \\ d_{\text{hit,ramp}} &= d + \frac{h_l - d\tan(\kappa)}{\sin(\gamma + \kappa)} \cos(\gamma)\cos(2\gamma + \kappa). \end{aligned} \tag{5.6}$$

Using eq. (5.2) and eq. (5.6) and optimizing β such that the number of points in the area at the start of the ramp are maximized, the optimal mounting pitch angle β for the two LiDARs has been found with $\beta_{\text{velodyne}} = 0°$ and $\beta_{\text{robos}} = 20°$. The optimization was done for a distance of 10 m to the ramp. The angle between the two LiDARs differs due to the different starting opening angle of $-25°$ and $0°$ for the Velodyne and Robosense respectively, as well as due to the different vertical resolution. A detailed description of all the variables used for the calculation can be found in table 5.3.

Table 5.3: Variables used to calculate the optimal mounting angle of the LiDAR.

Variable	Description	Unit
h_l	LiDAR height above ground	m
d	Distance to ramp	m
h_r	Height of ramp	m
l_w	Light travel distance from ramp start to contact point	m
d_w	Distance from ramp start to contact point	m
γ	Ramp angle	deg
β	LiDAR mount angle	deg
κ	Laser line angle	deg
ϵ	LiDAR vertical resolution	deg
n	Number of laser channels	

5.3 Data Recording

All the sensors are connected to a PC in the car, located in the booth. A notebook is connected to the PC using a Secure Shell Protocol (SSH) connection and is used to start the different sensor streams. For the communication between the PC and sensors and the recording of the sensor data the Robot Operating System (ROS) framework is used. Each sensor publishes its data to a specific topic, which can be subscribed to. While ROS allows for real-time running of nodes it is also possible to record all sensor streams and store them in a rosbag. A rosbag contains the serialized message data and can later be used as a dataset. The rosbag can be played back in ROS, simulating the same conditions as during the recording, by publishing the same topics again which were initially subscribed to during the recording. The different algorithms can then be tested and perhaps improved, without having to physically do the test drive again.

5.4 Car

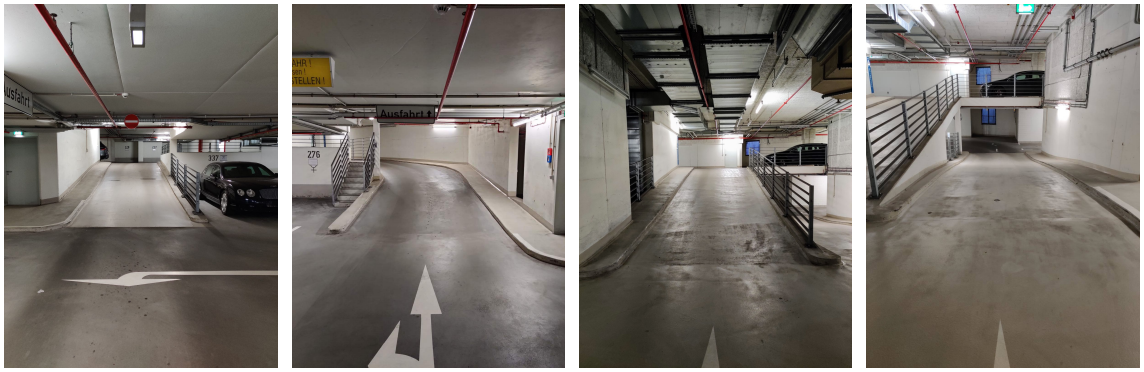
The car used for the recordings is an eGolf 2017. Being an electric car it provides better vibration properties than a car with an internal combustion engine, which means that the IMU measurements are less influenced by external noise. The car has built in wheel speed sensors, which only provide a signal if used in a special driving mode, in which the output power of the motor is limited. In this mode, the maximum speed is capped at $5\,\frac{\mathrm{km}}{\mathrm{h}}$ and the torque is limited, such that is not possible to drive a ramp all the way up. The car can only make it about halfway up. Because of that, the normal mode was used to drive between different levels. Before driving down, the mode was switched again to also provide the wheel speed measurements. A picture of the car with the mounted LiDAR and ZED 2i camera can be found in fig. 5.4.

5.5 Garage

To prevent an overfitting of the model it is important to have different test scenarios. The garage in which the car is normally parked has different types of ramps, which are shown in fig. 5.5. The properties of each ramp are described in table 5.4. The values were measured by inspecting the point cloud generated by the LiDAR. The width of the ramp is measured between the side wall and the railing, and not between the two curbsides. Because the

Figure 5.4: The car used for the recordings with the full sensor setup mounted on the roof.

ramps do not have a constant angle, because the change of the angle gradually increases, decreases and then increases again, the average angle of the ramps is used.



(a) (b) (c) (d)

Figure 5.5: On the ramps A, B and C the car will be driven up. On the ramp D as well as also on the ramp A the car will be driven down.

Table 5.4: The measured ramp properties.

Ramp	**Angle** [°]	**Width** [m]	**Length** [m]
A	7.2	3.94	11.97
B	6.5	3.96	14.15
C	7.4	3.90	11.85
D	−8.3	3.85	11.89

Add curb ramp width

Chapter 6

Results

In this chapter the results of the different proposed methods will be presented. At first, the collection of the reference data will be explained. The different metrics used for the evaluation of the results will be described in section 6.3. The different methods using the IMU are analyzed in section 6.4. Afterwards the performance of the LiDAR algorithm will be discussed in section 6.5 and finally the results of the object detection using the camera will be presented in section 6.6.

6.1 Evaluation Concept

During each test drive the measurements of the accelerometer and gyroscope of the IMU (of both IMUs, myAHRS+ and of the IMU integrated in the ZED 2i camera), the camera image of the ZED 2i camera and the point cloud generated by the LiDAR are recorded. Only one LiDAR could be mounted at a time, so the Velodyne UltraPuck was used for most test drives, but two recordings were also made using the Robosense. Furthermore, the wheel speeds were recorded when available, which was only the case when driving a ramp down or only half-way up, as mentioned in section 5.4.
To prevent an overfitting of the model it is important to have different test scenarios. As mentioned in section 5.5, four different ramps were available to test the model. For the evaluation of the IMU only the ramps A B, and D will be used. Multiple test drives were made for each scenario, the most drives where performed on ramp A. Because the wheel speed sensor measurements are needed for some methods, which are only available in a special mode where the power of the car is very limited, the ramps A and B could only be driven halfway up. But a full recording including the wheel speed measurements could be done for ramp D and A, when driving down.
The algorithms using the LiDAR and camera only detect ramps going up, so only the ramps A, B and C were used. Furthermore, a test drive without any ramps in sight was made to test if false positives are being detected.

6.2 Reference Data

To evaluate the performance of the different algorithms a reference is necessary. The open-source ROS package `hdl_graph_slam` [1] [83] is used for this task. It is based on 3D graph SLAM and uses the LiDAR data to map the environment and estimate the pose (position and orientation) of the car. It uses Normal Distributions Transform (NDT) scan matching-based odometry estimation with loop detection. The point cloud of common features at time t is compared to the point cloud from the time $t-1$ and matched against

[1] https://github.com/koide3/hdl_graph_slam

each other. The algorithm then estimates the translation and orientation difference between those two point clouds. In [84] the accuracy of the HDL Graph SLAM was tested and a mean error of 4 cm and a standard deviation of 5 cm was measured for an indoor scenario. From the pose information of the HDL Graph SLAM the pitch angle of the car can be calculated and be used as a reference for the road grade, to evaluate the performance of the different IMU-based methods. Because the LiDAR only records at 10 Hz and thus the estimation of the HDL Graph SLAM also only updates at a rate of 10 Hz, but the other sensors record from 100 Hz to 400 Hz, the estimate was upsampled using a Fourier method. Beside estimating the road grade, the IMU is used to estimate the angle and length of the ramp. The reference value for the length can be extracted from the generated point cloud map, by measuring the distance between the corresponding points. Because the ramp angle is not constant, the average angle of the ramp is used as reference. The average angle γ is calculated by measuring the length and height of the ramp and using the law of sines

$$\gamma = \arcsin\left(\frac{h_{\text{ramp}}}{l_{\text{ramp}}}\right). \tag{6.1}$$

The LiDAR is used to detect and track the distance to the ramp and also to estimate the angle, width and length. For the evaluation of the tracking accuracy, the generated map and pose provided by the HDL Graph SLAM is used again. In the generated point cloud map, the ramp region was marked manually by visual inspection. Then, using the position of the car, provided by the SLAM, the true distance to the beginning of the ramp could be calculated, by measuring the distance of the current position to the beginning of the ramp for each frame.
An example of the point cloud map generated by the `hdl_graph_slam` package is shown in fig. 6.1. The color of the points gives information about the z-value (height information) of the points. The ramp region is marked by the greater points in rainbow color and the black squares visualize the trajectory of the car. In this example the car was driven only half-way up the ramp. As mentioned in section 4.5, the camera is also used to identify the ramp. Because manual labeling of all the images was necessary for the training of the network, the labels can be used as a reference.

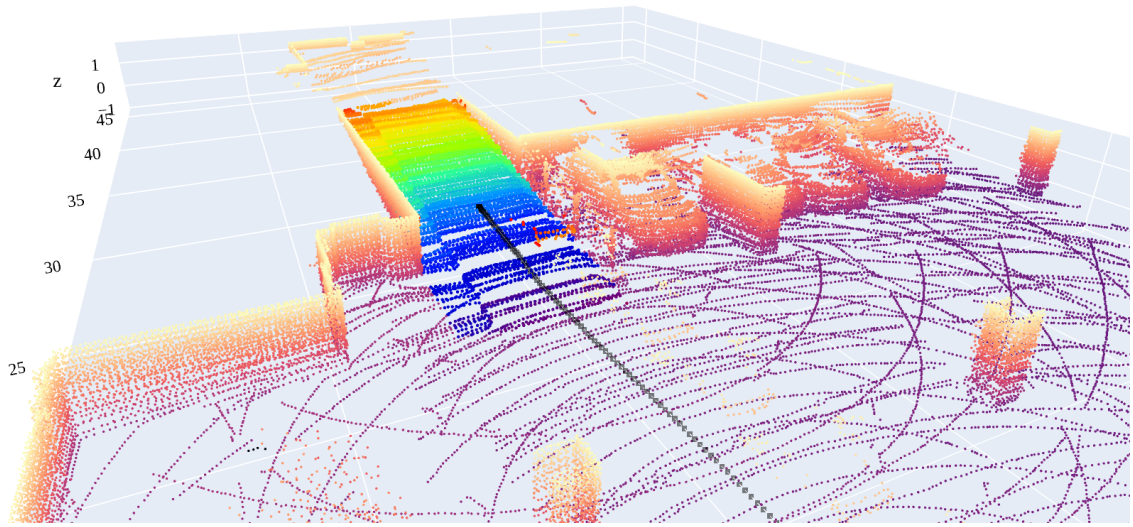


Figure 6.1: The point cloud map generated by the `hdl_graph_slam` package. The ramp region was selected manually by visual inspection and is marked by the rainbow-colored points. The black squares visualize the trajectory of the car.

6.3 Performance Measures

Using the IMU, the pitch angle of the car is estimated over time. The goodness of the fit between the estimation $\hat{y} = (\hat{y}_i, \ldots, \hat{y}_n)^\intercal$ and the reference $y = (y_i, \ldots, y_n)^\intercal$ can then be described by the Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}, \tag{6.2}$$

which quantifies by how much the predicted values differ from the reference value on average. It is defined in the range $[0, \infty)$, with a value of 0 indicating a perfect fit.
The same can be applied to the estimation for the angle, width, length and distance to the ramp by the LiDAR-based method. The only difference is that the reference angle, width and length are constant and thus the RMSE is basically the same as the standard deviation in this case. Furthermore, the coefficient of determination R^2 is used to evaluate the pitch angle estimation

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2}, \tag{6.3}$$

where $\overline{y}$ indicates the mean of the reference. The goodness of the fit is described in the range from 0 to 1, where 1 describes a perfect fit.
The LiDAR is used to detect a ramp, and to calculate its properties if a ramp is detected. The detection rate is evaluated by using the number of True Positives (TPs) and False Negatives (FNs). What is counted as TP or FN always depends on the use case. In the LiDAR scenario, a frame is labeled as TP if the ramp is visible to the LiDAR, the algorithm detected a ramp and at least 70% of the detected points actually lie inside the ramp region. Analogously, a frame is classified as FN, if a ramp is visible but less than 70% of the detected points lie inside the ramp region. Furthermore, when a ramp was detected even though none was visible, the frame is labeled as False Positive (FP).
The performance of the neural network can be evaluated in several ways. One commonly used metric in the field of computer vision is the Average Precision (AP). It is a detection evaluation metric, commonly used by the Common Objects in Context (COCO) dataset. It is based on the precision and recall score, which can be calculated by

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6.4}$$

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{6.5}$$

The precision gives information about how accurate the predictions are and the recall gives information about how much of the ground truth is detected. The area under the precision-recall curve is the AP score. So, unlike the name might suggest, the AP score is not actually just the average of the precision score. Otherwise, a model which correctly detects some objects in the image, but misses others, would still achieve a score of 1. Similarly to the LiDAR, it first must be defined what classifies as a TP and a FP. For this, a new metric is introduced, the Intersection of Union (IoU), which is defined as

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}. \tag{6.6}$$

The area of overlap is defined as the intersection between the predicted and ground truth bounding box, and the area of union is the area of both boxes added together. Instead

of a bounding box, a segmentation mask can be used as well. A perfect detection would have an IoU score of 1. Because an IoU score of 1 is nearly impossible, a frame is labeled as TP instead, if the IoU score is greater than a certain threshold m. The score is then written as AP_m. For example, AP_{50} means how many detections have been made with an IoU score of at least 50%. Since selecting a meaningful threshold for the AP score depends on the dataset, the Mean Average Precision (mAP) is often used instead. The mAP score is the average over multiple IoU thresholds for all the different classes. It is calculated by averaging all AP scores in the range of 50% to 95% with a step size of 5%, meaning that the average of 10 different values is used.

6.4 IMU-based measurements

In this section the different methods to estimate the road grade using the IMU are being evaluated. The results for two different drives are visualized. Lastly, the estimated average ramp angle and length are being compared to the ground truth.

6.4.1 Road Grade Estimation

The ramp detection using the IMU relies on the correct estimation of the road grade angle. Hence, the evaluation of the quality of the estimation is necessary to determine the performance of the ramp detection algorithm.
Different recordings of different ramps were made, but the results will be discussed on only two test drives. Furthermore, two different IMUs were used, but only the measurements of one IMU will be used to present the results, since both produced similar results. Because the IMU of the ZED camera is slightly more accurate as shown in table 5.1, only those measurements will be used.
In the first drive the car was accelerated from stand still and drove up ramp A about half-way up. As explained in section 5.4, the ramp could not be driven completely, due to the need of the odometer readings, which are only available when the motor output is limited. The result of using only the raw measurements of the IMU for the pitch angle calculation is illustrated in fig. 6.2. The measurement by the accelerometer is very noisy

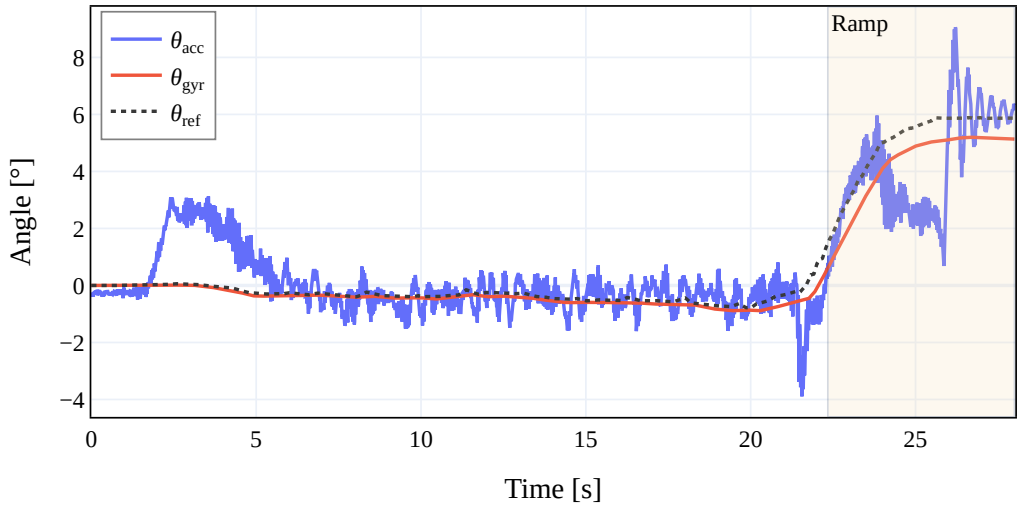


Figure 6.2: Pitch angle estimation from the raw accelerometer and gyroscope measurements. The drive did start from stand still and did end in the middle of the ramp.

and is easily influenced by accelerations other than gravity, which can be seen at time 2 s to 4 s, where the car started driving. The gyroscope on the other hand provides good

short-term accuracy and is not influenced by other accelerations, but is slowly drifting over time. The reference is taken from the orientation estimation of the `hdl_graph_slam` package, which uses the LiDAR data. The time frame during which the car was on the ramp is marked by the yellow coloring. The beginning of the ramp is classified as the point, where the reference data surpasses 1.5°.
The gravity method tries to overcome the problem of the accelerometer of also detecting other accelerations than gravity, by subtracting the car's acceleration from the accelerometer measurement. The car's acceleration $\mathbf{a}_{\text{odom,x}}$ was calculated by calculating the derivate of the low-pass filtered car velocity v_{car}, which was calculated from the wheel speed measurements. Figure 6.3 shows the (low-pass filtered) acceleration measured by the IMU along the x-axis and the (low-pass filtered) car's acceleration. And $\mathbf{a}_{\text{grav,x}}$ is the acceleration

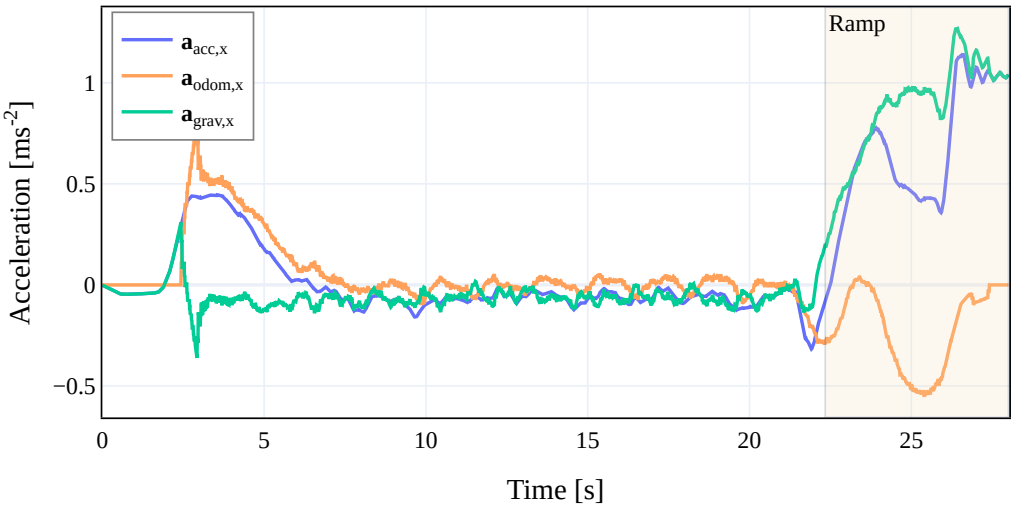


Figure 6.3: The measured acceleration in x-direction by the IMU and the acceleration derived from the wheel speed measurements and the difference between both.

measured by the IMU from which the car acceleration $\mathbf{a}_{\text{odom,x}}$ was subtracted. It can be seen, that especially at the beginning of the ramp (21 s to 23 s) the gravity method shows it advantages. The deceleration before entering the ramp is measured by both sensors and thus cancels out each other. The same can be seen in the initial acceleration phase, where the car starts to drive from still stand (2 s to 4 s). Although both sensors are synchronized in time, the IMU senses the acceleration earlier than the wheel speed sensors which leads to a slight pike. This could be due to the wheel speed sensors having a certain velocity threshold, below which they do not pick up any changes. Other reasons for the difference could be, that other forces than the one from the car are present, e.g. from the suspension of the car, vibrations due to the road quality or movement in the car, which are not measured by the wheel speed sensors. Moreover, the approximations made by calculating the finite difference of the car velocity to get the car acceleration have a negative influence on the result.
The resulting angles calculated from the accelerations can be seen in fig. 6.4. It can be seen that the gravity method improves the estimation accuracy, compared to when only the accelerometer data is used. Especially the angle at the start and on the ramp is more accurately described when subtracting the odometer data from the accelerometer data for the calculation. But it can also be seen that the time synchronization between both signals is important, which becomes apparent during the initial acceleration phase. Here, the subtraction leads to a negative value, which neither sensor had measured.
Another way to improve the estimation is using a complementary filter. The results of which, together with all other methods, are shown in fig. 6.5. Those are the results of

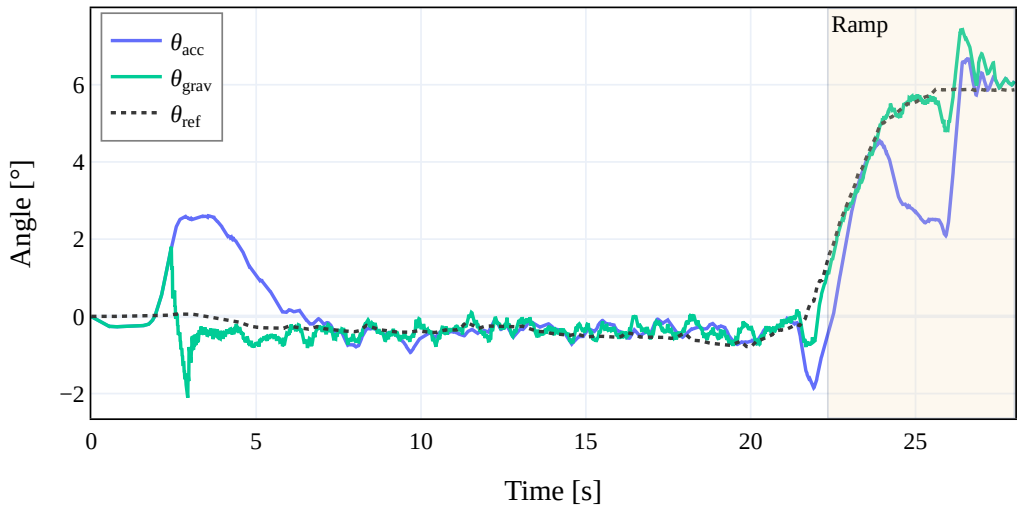


Figure 6.4: Pitch angle estimation using only the accelerometer compared to the gravity method, which additionally uses the wheel speed measurements.

a new drive, where the car was driven the ramp D and afterwards ramp A down. The

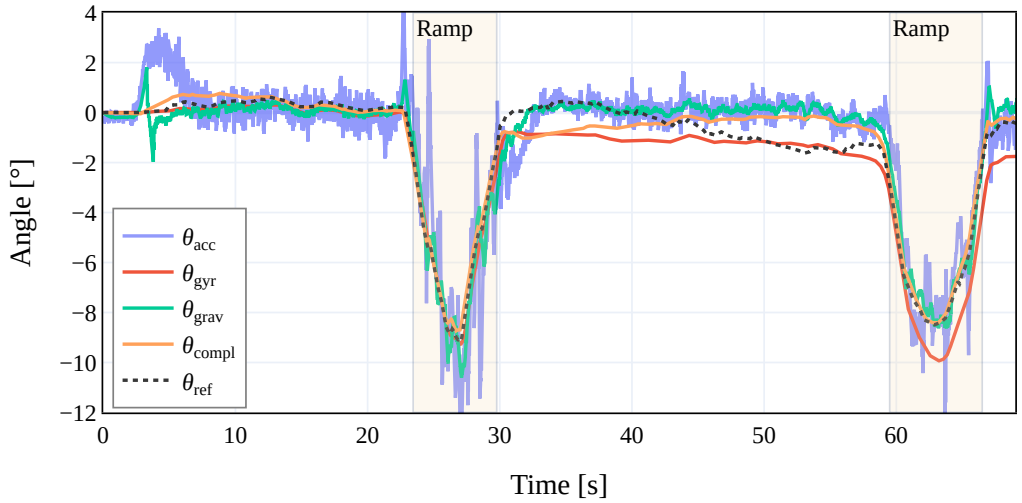


Figure 6.5: Comparison of different methods to estimate the pitch angle. Two different ramps were driven down successively.

complementary filter uses the estimation of the gyroscope measurements and corrects them using the accelerometer measurements to prevent drift. It can be seen that the estimation using the complementary filter closely follows the reference except for the part between the two ramps (30 s to 60 s). Also, it has no offset at the end, unlike the estimation from the gyroscope data. The results of the other methods applied to the recordings of the same ride are also shown in the same figure. Using the gyroscope measurements, the angle estimation follows the reference data very closely for the first ramp. But at the end a drift can be observed. The gravity method reduces the spikes of the raw accelerometer estimation, but introduces a new error at the beginning, due to the odometer readings being slightly shifted in time in regard to the accelerometer readings.
While a visual inspection gives a first impression on which method performs the best, the use of different metrics allows for a more accurate evaluation. The RMSE, R^2 as well as the maximal error of the different methods when driving up, for two kinds of ramps, are shown in table 6.1. Multiple recordings have been made for each ramp, but in all of them

the car started from stand still and ended in the middle of the ramp. The average of all drives was then calculated.
The estimation using the raw accelerometer data performs the worst and has the greatest maximal error. Using the gyroscope the deviation from the reference data is fairly small, which can be seen in the maximal error. But due to the drifting the errors cumulate, resulting in a high RMSE value. The acceleration method performs well, but is negatively influenced by the acceleration from stand still, during which the accelerometer and wheel speed data are not properly aligned. The best results are achieved using the complementary filter. In table 6.2 the average results for down drives are shown. The data are from a test

Table 6.1: Road grade estimation when driving ramps up, using different methods.

	Ramp A			**Ramp B**		
Method	**RMSE** [°]	$\mathbf{R^2}$	$\mathbf{Error_{max}}$ [°]	**RMSE** [°]	$\mathbf{R^2}$	$\mathbf{Error_{max}}$ [°]
Accelerometer	0.875	0.675	4.944	0.650	0.787	3.378
Gyroscope	0.508	0.905	1.373	0.768	0.802	**1.594**
Acceleration method	0.357	0.954	2.482	0.351	**0.955**	1.707
Complementary	**0.273**	**0.976**	**1.231**	**0.324**	0.952	1.647

drive in which at first the ramp C was driven down, followed by ramp A in one consecutive drive. The drive was performed multiple times and the average of all results was calculated again. The estimations for one exemplary ride were illustrated in fig. 6.5. The values for ramp C correspond to the measurements from 0 s to 31 s and the measurements from 31 s until the end are attributed to ramp A. It can be seen that at the beginning (ramp C) the gyroscope performs the best, but with increasing time the drift negatively impacts the results. All methods perform significantly worse in the second part of the drive (ramp A), due to the deviation of the reference value in the time frame between the two ramps. But when looking at the graph in fig. 6.5 it can be seen, that the gravity method and especially the complementary filter closely follows the reference data.

Table 6.2: Road grade estimation when driving ramps down, using different methods.

Method	**Ramp C**			**Ramp A**		
	RMSE [°]	$\mathbf{R^2}$	$\mathbf{Error_{max}}$ [°]	**RMSE** [°]	$\mathbf{R^2}$	$\mathbf{Error_{max}}$ [°]
Accelerometer	0.894	0.746	8.291	0.938	0.784	3.967
Gyroscope	**0.219**	**0.990**	**0.736**	0.784	0.867	**1.481**
Acceleration method	0.369	0.964	2.211	**0.719**	**0.872**	2.136
Complementary	0.327	0.977	1.184	0.753	0.862	2.131

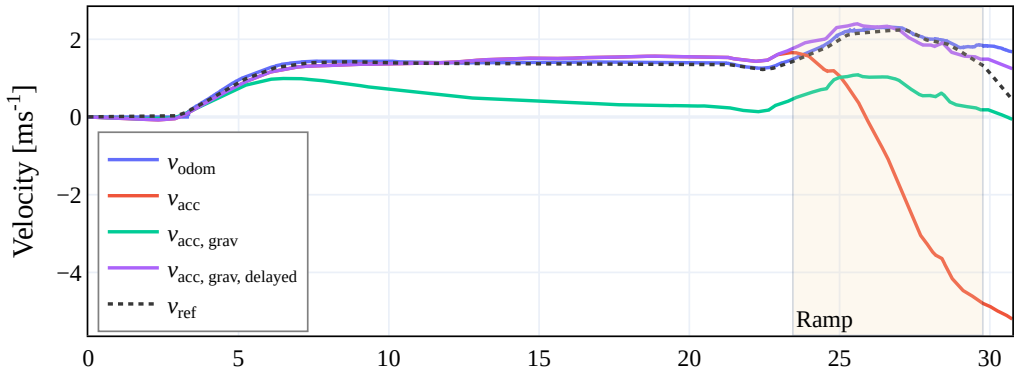
6.4.2 Estimation of Ramp Properties

Now that the estimation of the road grade is done, the ramp properties can be estimated. The estimated ramp angles and the corresponding deviation from the reference value for three different ramps are listed in table 6.3. All the different methods seem to underestimate the angle when driving up (ramp A and B) or overestimate it when driving down (ramp D). Because ramp A and B could only be driven half-way up, it could be possible that the average angle had not been reached yet. Overall it can be seen that the complementary filter and the acceleration method perform the best.

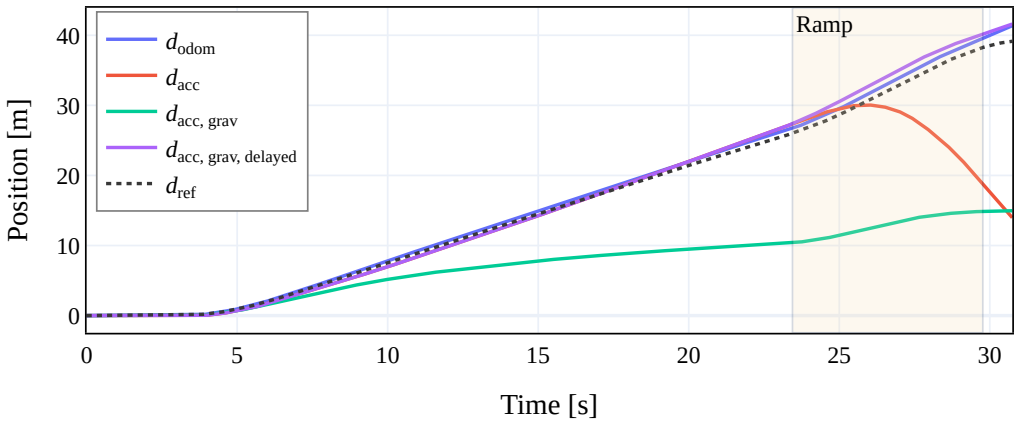
Table 6.3: Estimation of ramp angle.

	Ramp A		Ramp B		Ramp D	
Method	**Angle** [°]	**Error** [°]	**Angle** [°]	**Error** [°]	**Angle** [°]	**Error** [°]
Accelerometer	4.14	3.06	4.75	1.75	-10.41	2.11
Gyroscope	4.97	2.23	4.30	2.20	-9.11	0.81
Acceleration method	6.15	1.05	6.18	0.32	-9.52	1.22
Complementary	5.42	1.78	6.12	0.38	-9.01	0.71

As discussed in section 4.3.3, the length of the ramp can be estimated by integrating the velocity of the car over the time interval between the start and end of the ramp. The velocity can be estimated in two different ways. Either by using the wheel speed measurements, or by integrating the accelerometer measurements. For the evaluation of the accuracy of the estimation, recordings of a drive with a full traverse of a ramp is necessary. As mentioned in section 5.4, this is only possible when driving down. Ramp D is used for this evaluation, the first part of the drive is show in fig. 6.5.
In fig. 6.6a different methods to estimate the velocity of the car are shown. v_{odom} is the



(a) Estimated car velocity calculated from the wheel speed measurements and different methods using the accelerometer data.



(b) The integration of the velocity leads to the travelled distance. The length of the ramp can then be calculated from the difference between the position at the end and start of the ramp.

Figure 6.6: Comparison of different methods to estimate the length of the ramp.

velocity calculated from the wheel speed measurements using eq. (4.26). The other three methods are based on the integration of the accelerometer measurements. v_{acc} is the

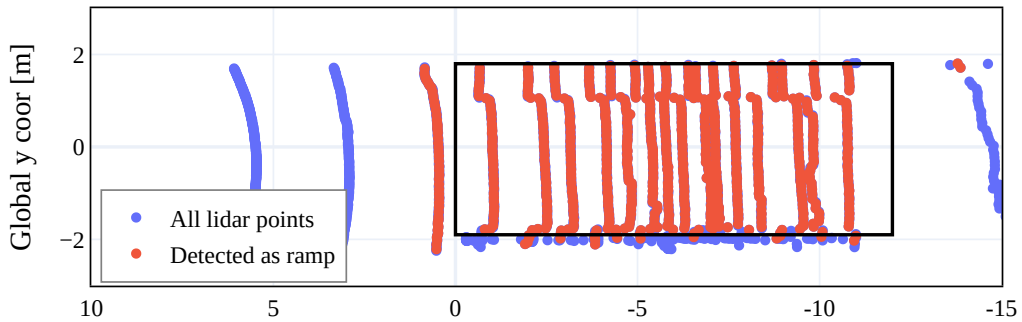
result of integrating the raw accelerometer measurements $\mathbf{a}_x$. Because the raw signal also contains the gravity component, the estimation is not correct anymore when entering the ramp. Up until the point where the car enters the ramp both estimations are very similar, but during the phase on the ramp the acceleration due to gravity disturbs the estimation. This problem is partially solved by $v_{\text{acc,grav}}$, which eliminates the gravity component by subtracting it using the pitch angle of the car, according to eq. (4.32). Different angles can be used in the equation, the angle estimation using the complementary filter was chosen, since it achieved the best results. Nonetheless, because the angle estimation of the angle is not correct at the start of the acceleration (3 s to 5 s), an error is introduced. As it can be seen in fig. 6.5 the angle is slightly overestimated by the complementary filter during the acceleration phase. This leads according to eq. (4.32) to an underestimation of the acceleration, and thus also an underestimation of the velocity. Due to the integration the error is cumulated over time and increases with the time.
This error can be fixed by introducing a new method, which ignores the angle estimation during the acceleration phase and instead assumes an angle of zero during this phase. The result is visualized by $v_{\text{acc,grav,delayed}}$. For the sake of consistency, the reference data is once again calculated from the pose provided by the `hdl_graph_slam` package. But it must be noted, that the data from the wheel sensors is most likely more accurate, but the results will still be compared to the reference data.
The estimation of the travelled distance for the different methods is shown in fig. 6.6b. Since the distance is calculated by integrating the velocity, an error is cumulated over time. This can be seen in both d_{acc} and $d_{\text{acc,grav}}$. The only usable results are produced by the method using the odometer measurements and by $v_{\text{acc,grav,delayed}}$.
The actual length of the ramp is calculated by subtracting the position of the car at the end of the ramp from the position at the start of the ramp. The calculated length for the different methods and the corresponding error are shown in table 6.4.

Table 6.4: The estimation of the ramp length using different methods.

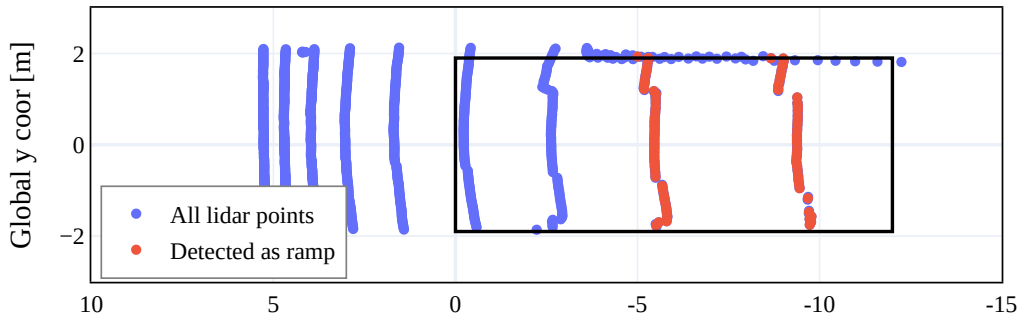
Method	**Length** [m]	**Error** [m]
d_{odom}	11.88	0.63
d_{acc}	−8.44	20.69
$d_{\text{acc,grav},}$	4.52	7.73
$d_{\text{acc,grav,later}}$	11.85	0.60
d_{ref}	11.25	0

6.5 Ramp Detection by LiDAR

The LiDAR is used to detect if a ramp is visible, track the position to the ramp as well as to estimate the angle, width and length of the ramp. Due to the setup it was only possible to mount one LiDAR at a time. To get the best possible results the LiDAR with the best resolution, especially in vertical direction, should be used. A comparison of the resolution of the two LiDARs can be seen in fig. 6.7. Here, a top-down view on the ramp of the points measured by the LiDAR at a distance of 7 m to the ramp is shown. The blue points visualize all the points which were visible to the ramp detection algorithm after the downsampling and pass through filter has been applied. The points marked red symbolize the points that were then detected as part of the ramp by the algorithm. The ramp region is framed in black and was drawn manually by visually inspecting the by the `hdl_graph_slam` package generated point cloud. The Velodyne LiDAR, shown in fig. 6.7a,



(a) Velodyne



(b) Robosense

Figure 6.7: Resolution comparison of the two LiDARs. The ramp region is framed in black.

provides many more lines compared to the Robosense, shown in fig. 6.7b. Hence, only the results produced using the Velodyne LiDAR will be discussed from here on.
The detection rate is evaluated by looking at the number of TPs and FNs, as described in section 6.3. For all the other properties, the RMSE between the predicted values and the values measured in the point cloud, are calculated. Since the resolution improves the closer the car gets to the ramp, the evaluation is divided into distance intervals of 5 m length.
The results for three different ramps are shown in table 6.5. Each drive started about 30 m from the ramp. It can be seen that the detection works very well if the distance to the ramp is 20 m or less. For the ramp C it can be seen, that the detection is only reliable when the car is less than 15 m away from the ramp. This can be explained by the taken path during the recording, which was started at an offset in y-direction to the ramp. Due to the passthrough filter the ramp was thus not visible to the algorithm, and could not be detected at the beginning. It can also be observed, that the deviation of the width estimation is significantly smaller than the deviation of the distance and length estimation. This can be explained by the working principle of the LiDAR, which sends out laser points in horizontal lines, as depicted in fig. 6.7. The horizontal resolution is thus greater than the vertical resolution. Furthermore, it must be noted that all calculations are done one the downsampled point cloud, which reduced the resolution to 10 cm.
Beside the three ramps, data consisting of 252 frames were recorded of an environment where no ramp was visible. One frame was wrongly labeled as a ramp (FP), which results together with the other recordings in a precision score of 99.97 % and recall score of 96.54 %.
The estimated ramp properties and distance to the ramp for one exemplary ride are shown in fig. 6.8. All values are plotted against the distance to the ramp, which is almost linear to the time, since after the initial acceleration from stand still, the car moved at a constant speed. In fig. 6.8a the estimated distance from the car to the ramp is shown in comparison to the reference distance provided by the `hdl_graph_slam` package. The error reduces

Table 6.5: Ramp detection and property estimation evaluation of the LiDAR algorithm. The precision increases the closer the car gets to the ramp.

Ramp	**Distance** [m]	**Frames**	**TP**[%]	**FN**[%]	**RMSE**			
					θ[°]	d[m]	w[m]	l[m]
A	0 to 5	116	100.00	0.00	0.31	0.70	0.03	1.27
	5 to 10	117	100.00	0.00	0.30	0.77	0.04	0.94
	10 to 15	116	100.00	0.00	0.34	0.81	0.07	1.03
	15 to 20	123	100.00	0.00	0.27	1.01	0.05	1.84
	20 to 25	140	99.23	0.77	0.65	1.70	0.12	6.28
	25 to 30	50	59.78	40.22	1.79	1.21	0.25	10.04
B	0 to 5	62	100.00	0.00	0.52	0.71	0.02	0.81
	5 to 10	62	100.00	0.00	0.53	0.77	0.02	0.78
	10 to 15	59	100.00	0.00	0.51	0.86	0.02	0.85
	15 to 20	61	97.92	2.08	0.49	1.23	0.02	3.30
	20 to 25	61	97.83	2.17	0.83	2.18	0.12	9.33
	25 to 30	59	42.75	57.25	1.82	4.35	0.90	11.98
C	0 to 5	21	100.00	0.00	0.64	0.87	0.05	2.58
	5 to 10	23	100.00	0.00	0.50	0.89	0.11	4.40
	10 to 15	28	100.00	0.00	0.44	0.70	0.10	1.23
	15 to 20	27	37.04	62.96	0.66	0.79	0.19	1.27
	20 to 25	29	0.00	100.00				
	25 to 30	28	10.71	89.29	1.91	1.40	0.25	10.78

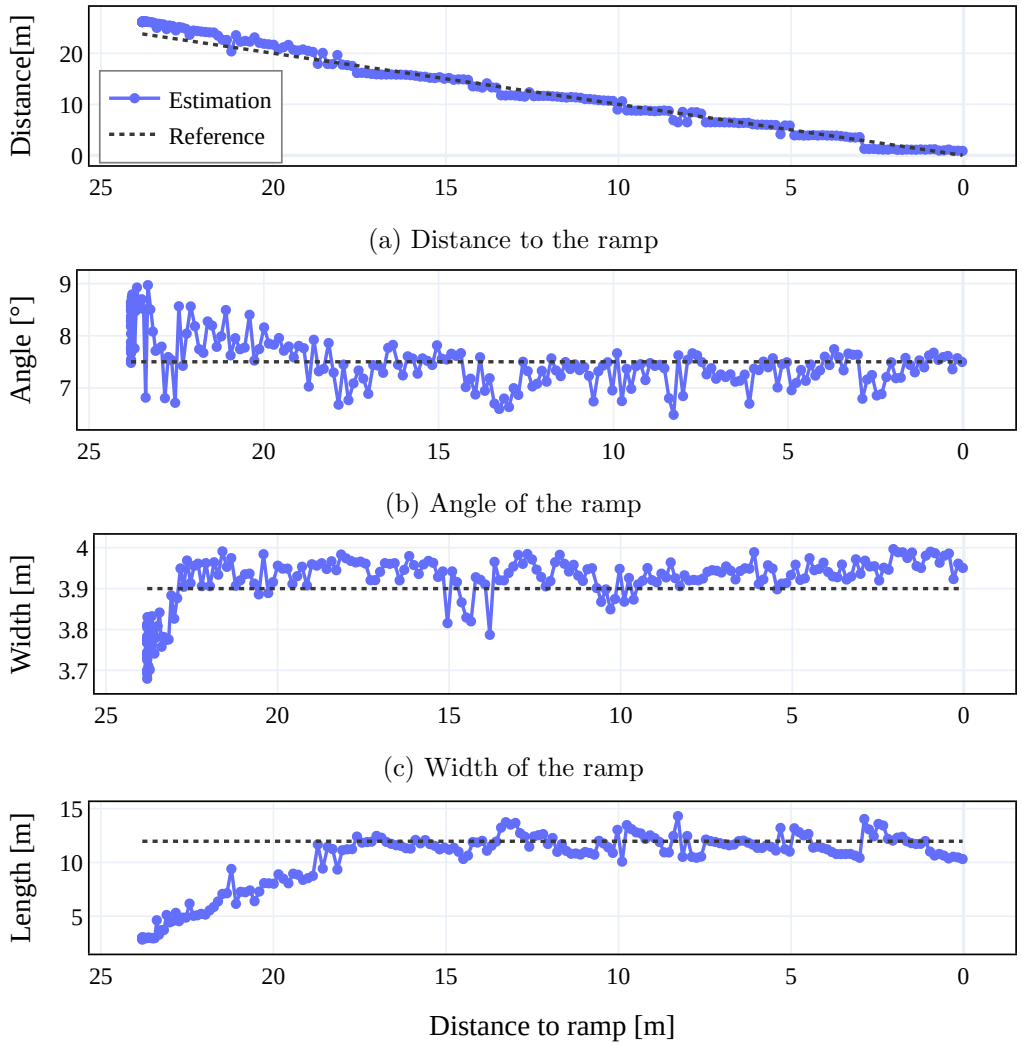
when the car is closer to the ramp. Interestingly the value of the estimated distance seems to hold itself for several m. This is most probably due to the vertical resolution of the LiDAR. Because the vertical resolution is not linear, the most lines are centered around the middle of the opening angle of the LiDAR, which is -5° for the Velodyne. Hence, only few lines fall into the region at the start of the ramp. The distance to the ramp is calculated by measuring the distance from the car to the n closest points which have been identified as part of the ramp. Therefore, the distance can only be updated if a line which has previously hit the ground now hits the ramp.

Figure 6.8b shows the difference between the estimated angle and the measured average angle. It can be seen that the estimation varies by about 1° if the distance is less than 20 m. The angle is almost exclusively underestimated, which could be due to the fact that the reference value measurement is not perfect.

The estimated width at different distances to the ramp can be seen in fig. 6.8c. The error is very small compared to the tracking error and lies in the order of 10 cm. This is because the horizontal resolution is significantly better than the vertical resolution. A deviation of up to 10 cm is expected, since this is the resolution of the downsampled point cloud. Note that the estimated width is the width of the whole ramp and not only the width of the drivable part.

Finally, the estimated length of the ramp is shown in fig. 6.8d. At a large distance not enough lines fall into the ramp region, which leads to an underestimation of the length at first. However, the estimation improves at a distance of about 17 m to the ramp, after which the estimation fluctuates around the measured value.

A visualization of the detection algorithm can be seen in fig. 6.9. Here, the 3D point cloud generated by the LiDAR is projected onto the 2D camera image. The quality of the



(a) Distance to the ramp

(b) Angle of the ramp

(c) Width of the ramp

(d) Length of the ramp

Figure 6.8: Estimated ramp properties and tracking of the distance to the ramp at different distances to the ramp.

Figure 6.9: Lidar points projected onto the camera image. The green points were identified as part of a ramp by the algorithm.

projection depends on the accuracy of the measured translation and orientation difference between both sensors. It can be seen that it is not perfect, e.g. the points do not quite match the camera image at the left pillar or the pipe on the ceiling. Nonetheless, it gives a good indication of what the LiDAR actually sees. The coloring of the point indicates the distance from the car to the points. Objects far away are marked by yellow points and nearby objects by blue points. The green points were identified as part of the ramp by the algorithm. It mostly fits the actual ramp very well. The previously mentioned problem of the vertical resolution is clearly visible here. While the density of the laser lines is sufficient in the ramp region, the start of the ramp and especially the ground is covered by very few lines. This makes the precise tracking of the distance to the ramp difficult.

6.6 Camera

6.6.1 Object Recognition

The camera was used to detect the ramp and mask the corresponding region in the image. As described in section 6.3, the mAP score is used to evaluate the performance of the neural network. Since transfer learning is used and the dataset is fairly small, different hyperparameters could be tested. As described in section 4.5.3, grid search was used to find the best hyperparameters. In table 6.6 the mAP scores for different hyperparameter combinations are listed. LR is the learning rate and SR is the sample rate and determines the number of ROI proposals, as described in section 4.5.3.

Table 6.6: Testing of different training hyperparameters and their influence on the AP score.

Parameter			**Scores**	
Epochs	**LR**	**SR**	**Box mAP**	**Mask mAP**
1	0.001	128	69.3	83.4
1	0.001	512	84.8	85.7
1	0.003	128	73.3	76.8
1	0.003	512	83.1	89.6
3	0.001	128	77.6	90.1
3	0.001	512	87.3	90.8
3	0.003	128	83.9	89.0
3	0.003	512	84.9	87.6

Since the configuration with a learning rate of 0.001, sample rate of 512 and 3 epochs achieved the best results, all the following images and calculations were generated using this configuration. A visualization of the prediction for some example frames when driving straight and from an angle at a ramp can be seen in fig. 6.10a and fig. 6.10b respectively. In all those cases the detection of the ramp was successful and seems to be fairly accurate. The annotation inside the image describes how confident the network is about the detection. All images used for the training and evaluation are colored, but are visualized as grayscale image in this case, to allow for an easier distinction between the segmented region and the background.

(a) Driving straight at the ramp.

(b) Driving at an angle to the ramp.

Figure 6.10: Some examples of the predicted bounding box and mask when driving at a ramp. The class of the object and the confidence score are annotated as well. All those images were not used for the training of the network.

6.6.2 Point Cloud Extraction

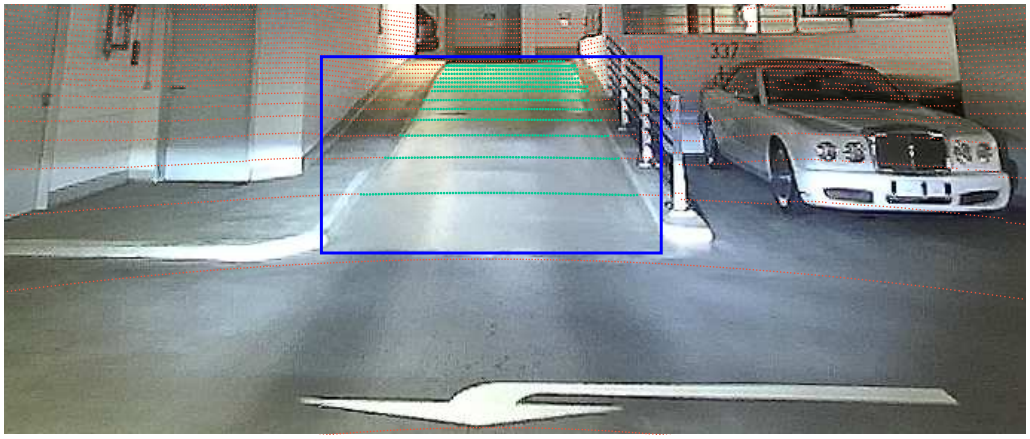
As described in section 4.5.4, the predicted segmentation mask can be used to only extract the points that are part of the ramp from the point cloud. Different ramp properties can then be calculated from the point cloud. In a first step, the points must be projected from 3D space into 2D space. In fig. 6.11a the projected point cloud of the ZED 2i camera are shown as orange points. The green points symbolize the points that were detected by the neural network as part of the ramp. The blue box is the corresponding bounding box. It can be seen that detection works very well and also only the actual drivable part of the ramp is marked, unlike when just the LiDAR is used (see fig. 6.9). While most of the area is covered by the point cloud, objects further away than 20 m are not detected. This can be seen in the area above the ramp, which is too far away for the sensor to detect. But since the car is not far away from the ramp, most of the ramp could be covered by points. This problem is not present, when the point cloud of the LiDAR is used instead. The projection is shown in fig. 6.11b. It can be seen that the vertical resolution is worse at most opening angles, but the horizontal resolution is better. This is especially apparent at the beginning of the ramp. The neural network marked the pixels where the bounding box ends as part of the ramp, but because the LiDAR does not have any points near this line, the nearest marked points are multiple cm further away.

Maybe add specific range

This leads to a wrong distance and perhaps length estimation.
Projecting the point cloud back into 3D space allows for the calculation of the properties of the ramp. The results for both sensors are shown in table 6.7. The box method uses the bounding box to extract the points from the point cloud and the mask method uses the mask to extract the points from the point cloud. The RANSAC algorithm has been applied in both cases to the extracted points, to remove outliers. The used sensor is written in

the subscript of the method name. Since in the neural network is trained to only detect the drivable part of the ramp, the width had to be measured again. For the ramp A, the width between the curbs was measured as 2.79 m.
It can be seen that the point cloud generated by the ZED camera produces worse results than the point cloud generated by the lidar. While it can not be seen very well in fig. 6.11a, the point cloud was very inaccurate. The points on the ramp do not build a planar surface, that is why the RANSAC algorithm had trouble to find an optimal plane. The LiDAR on the other hand produced an accurate point cloud and achieved good results.

(a) Projection using the by the ZED camera generated point cloud.



(b) Projection using the by the LiDAR generated point cloud.

Figure 6.11: Predicted bounding ion of the 3D point cloud onto a 2D image. Using the predicted segmentation mask, the points that lie in the ramp regions can now be extracted from the point cloud.

Table 6.7: Estimation of different ramp properties using the extracted point cloud.

Method	**Angle** [°]	**Error** [°]	**Width** [m]	**Error** [m]	**Length** [m]	**Error** [m]
Box_{Zed}	8.84	1.64	3.74	0.95	13.18	1.21
Mask_{Zed}	8.72	1.52	2.53	0.26	12.61	0.64
$\text{Box}_{\text{LiDAR}}$	7.78	0.58	5.10	2.31	10.98	0.99
$\text{Mask}_{\text{LiDAR}}$	7.64	0.44	2.82	0.03	10.19	1.78

Chapter 7

Conclusion

7.1 Summary

The detection of ramps in parking garages for AVP is a problem...

One sentence about why this thesis was done

In this thesis different sensors and methods for the detection and classification of ramps in parking garages were developed and evaluated. Using an IMU, a ramp could be detected by measuring the tilt of the car. Different online methods were evaluated, the best results were achieved using a complementary filter, which fuses the accelerometer data and gyroscope data of the IMU. By also using the data from the wheel speed sensors, an accurate estimation of the length of the ramp was possible, in addition to the average angle estimation of the ramp.
To be able to detect a ramp before entering it, different sensors were used. At first a LiDAR was tested. Using the point cloud generated by the LiDAR the planar ramp region could be extracted using the RANSAC algorithm. The length and width of the ramp could be measured, as well as the average angle of the ramp and the distance of the car to the ramp. The ramp was successively detected in more than 95% of the cases, if the distance to the ramp was less than 25 m. For both the IMU and LiDAR a novel calibration method was implemented, which automatically transforms the sensor measurements into the coordinate system of the car.
Lastly, a camera was used to detect the ramp. The object detection was done using the deep learning Mask R-CNN, which generates a segmentation mask of the object. An own dataset was created for the training. Due to the small size, augmented images were added to the dataset and an on the COCO dataset pretrained network was used. The network proofed to be able to detect ramps well and achieved a mAP score of over 90%.
Lastly the predicted segmentation mask was elevated into 3D space, by projection a 3D point cloud onto an image, removing points outside the mask, and then transforming them back into 3D space. This was done for the point cloud generated by a stereo-camera, and the point cloud of the LiDAR. This allowed for the accurate detection of the drivable part of the ramp in 3D space, whereas the LiDAR algorithm did not make a distinction between the drivable part and the curb sides.

7.2 Conclusion

The IMU methods were suspect. Due to the significant drift of the gyroscope, the best results were achieved by using a complementary filter, but using the gravity method, also good results were achieved.

Since the LiDAR used in the experiments is intended to be used as a 360 top LiDAR, the vertical resolution was limited. This leads to length and distance fail and could be solved by mems lid?

7.3 Future Work

In this thesis different sensors were used to detect a ramp and measure its properties. While relatively good results could be achieved, each method could be improved further. For the IMU other algorithms such as a Kalman filter could be tested. Furthermore, the problem of the delay between the wheel speed sensor and IMU measurements could be investigated. The same goes for the LiDAR, other algorithms than RANSAC could be tested. There exist newer algorithms which are more efficient, which means that the downsampling and passthrough filtering could be reduced. The neural network for the object detection could also be made more robust by using more data, preferably of other environments, in the training.
To further improve the results, the different methods could also be combined. In the thesis the online implementation of the objection detection was not carried out, this could be done in the future. Then, the LiDAR point cloud extraction using the instance segmentation prediction could also be properly evaluated. Another option of sensor fusion would be to use the estimated ramp properties using the LiDAR data and correct them after driving on the ramp using the IMU data.
For the evaluation of the different methods the LiDAR data was used, which is subject to error. Additionally, the testing in a more controlled environment would be useful. This could be done in a simulated environment, e.g. in `CARLA` [1] [85].
Another potential future task would be to use the implemented methods to create a meta map of the environment. If a map is created, e.g. using the LiDAR and the `hdl_slam` library, the location of the ramp could be labeled in the map, together with the properties of the ramp.

7.4 Advantages and disadvantages of different methods

Just a really short sketch of what could be written here

In this section the different sensors and methods will be compared. At first the estimation using an IMU was tested. An advantage is that almost all modern cars are already equipped with an IMU and wheel encoders. ... disadvantage?
To detect a ramp before entering it, a LiDAR and a camera were used. Using a LiDAR, a very accurate measurement estimation of the width and angle of the ramp was possible. But the distance estimation and length estimation was limited by the vertical resolution. The LiDAR While good results were achieved in the one garage, other garages? ... Furthermore, LiDARs are only available in premium vehicles SOURCE due to its high price, but with the ... in Microelectromechanical Systems (MEMS) LiDARs this could change in the future.
A camera on the other hand is fairy cheap and in most cars.
Using the IMU and odometer, no additional hardware is needed, since both sensors already available in most currently available cars. The LiDAR provides great accuracy and range, but is not used in many cars due to its price. But this could change in the future, when MEMS LiDARs become cheaper. The camera is available in most cars, but for a robust neural network, a bigger dataset is needed.

[1] https://github.com/carla-simulator/carla

Bibliography

[1] A. SE, "A sudden bang when parking," 2015. [Cited on page 1]

[2] H. Banzhaf, D. Nienhuser, S. Knoop, and J. Marius Zollner, "The future of parking: A survey on automated valet parking with an outlook on high density parking," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 1827–1834, 2017. [Cited on page 1]

[3] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors for position and orientation estimation," *Foundations and Trends in Signal Processing*, vol. 11, no. 1-2, pp. 1–153, 2017. [Cited on pages 4, 5, and 24]

[4] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, vol. 2, aug 2005. [Cited on page 5]

[5] O. J. Woodman, "An introduction to inertial navigation," Research report 696, University of Cambridge, aug 2007. [Cited on pages 6, 7, and 9]

[6] D. K. Shaeffer, "MEMS inertial sensors: A tutorial overview," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 100–109, 2013. [Cited on page 7]

[7] M. Perlmutter and S. Breit, "The future of the MEMS inertial sensor performance, design and manufacturing," *2016 DGON Inertial Sensors and Systems, ISS 2016 - Proceedings*, pp. 1–12, 2016. [Cited on pages 7 and 8]

[8] E. Lefferts, F. Markley, and M. Shuster, "Kalman Filtering for Spacecraft Attitude Estimation," *Journal of Guidance, Control, and Dynamics*, vol. 5, pp. 417–429, sep 1982. [Cited on pages 7 and 8]

[9] M. Kok and T. B. Schon, "Magnetometer Calibration Using Inertial Sensors," *IEEE Sensors Journal*, vol. 16, pp. 5679–5689, jul 2016. [Cited on page 8]

[10] Pengfei Guo, Haitao Qiu, Yunchun Yang, and Zhang Ren, "The soft iron and hard iron calibration method using extended kalman filter for attitude and heading reference system," in *2008 IEEE/ION Position, Location and Navigation Symposium*, pp. 1167–1174, IEEE, 2008. [Cited on page 8]

[11] P. Zhang, X. Zhan, X. Zhang, and L. Zheng, "Error characteristics analysis and calibration testing for MEMS IMU gyroscope," *Aerospace Systems*, vol. 2, pp. 97–104, dec 2019. [Cited on page 8]

[12] Y. Li and J. Ibanez-Guzman, "Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, 2020. [Cited on page 9]

[13] T. Fujii and T. Fukuchi, *Laser Remote Sensing*. CRC Press, 2005. [Cited on page 10]

[14] D. Wang, C. Watkins, and H. Xie, "MEMS Mirrors for LiDAR: A Review," *Micromachines*, vol. 11, p. 456, apr 2020. [Cited on page 10]

[15] K. Reif, *Bosch Autoelektrik und Autoelektronik*. Springer-Verlag, 2011. [Cited on pages 10 and 11]

[16] K. Reif, *Automobilelektronik: eine Einführung für Ingenieure*. Springer-Verlag, 2014. [Cited on page 11]

[17] R. Szeliski, *Computer Vision*. Texts in Computer Science, Springer International Publishing, 2022. [Cited on page 11]

[18] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image Segmentation Using Deep Learning: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8828, no. c, pp. 1–1, 2021. [Cited on page 11]

[19] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A Review on Deep Learning Techniques Applied to Semantic Segmentation," pp. 1–23, apr 2017. [Cited on page 12]

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 580–587, IEEE, jun 2014. [Cited on pages 12, 13, and 17]

[21] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, vol. 2015 Inter, pp. 1440–1448, IEEE, dec 2015. [Cited on pages 12 and 18]

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, jun 2017. [Cited on pages 12 and 18]

[23] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, IEEE, oct 2017. [Cited on pages 13, 18, and 31]

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Oote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, p. 5, 2009. [Cited on page 13]

[25] J. Jauch, J. Masino, T. Staiger, and F. Gauterin, "Road grade estimation with vehicle-based inertial measurement unit and orientation filter," *IEEE Sensors Journal*, vol. 18, no. 2, pp. 781–789, 2018. [Cited on page 15]

[26] P. Sahlholm, H. Jansson, E. Kozica, and K. H. Johansson, "A SENSOR AND DATA FUSION ALGORITHM FOR ROAD GRADE ESTIMATION," *IFAC Proceedings Volumes*, vol. 40, no. 10, pp. 55–62, 2007. [Cited on page 15]

[27] P. Sahlholm and K. Henrik Johansson, "Road grade estimation for look-ahead vehicle control using multiple measurement runs," *Control Engineering Practice*, vol. 18, pp. 1328–1341, nov 2010. [Cited on page 15]

[28] K. Maleej, S. Kelouwani, Y. Dube, and K. Agbossou, "Event-Based Electric Vehicle Mass and Grade Estimation," in *2014 IEEE Vehicle Power and Propulsion Conference (VPPC)*, pp. 1–6, IEEE, oct 2014. [Cited on page 15]

[29] N. Kidambi, R. L. Harne, Y. Fujii, G. M. Pietron, and K. W. Wang, "Methods in Vehicle Mass and Road Grade Estimation," *SAE International Journal of Passenger Cars - Mechanical Systems*, vol. 7, pp. 981–991, apr 2014. [Cited on page 15]

[30] J. Ryu and J. C. Gerdes, "Integrating Inertial Sensors With Global Positioning System (GPS) for Vehicle Dynamics Control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 126, pp. 243–254, jun 2004. [Cited on page 15]

[31] B. Yazdani Boroujeni and H. C. Frey, "Road grade quantification based on global positioning system data obtained from real-world vehicle fuel use and emissions measurements," *Atmospheric Environment*, vol. 85, pp. 179–186, mar 2014. [Cited on page 15]

[32] L. Mainetti, L. Patrono, and I. Sergi, "A survey on indoor positioning systems," in *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 111–120, IEEE, sep 2014. [Cited on page 15]

[33] B. Ozdenizci, V. Coskun, and K. Ok, "NFC Internal: An Indoor Navigation System," *Sensors*, vol. 15, pp. 7571–7595, mar 2015. [Cited on page 15]

[34] M. Nilsson and E. Öhlund, "Estimation of road inclination," 2012. [Cited on page 15]

[35] N. Palella, L. Colombo, F. Pisoni, G. Avellone, and J. Philippe, "Sensor fusion for land vehicle slope estimation," in *2016 DGON Intertial Sensors and Systems (ISS)*, pp. 1–20, IEEE, sep 2016. [Cited on page 15]

[36] C. Sentouh, S. Mammar, and S. Glaser, "Simultaneous vehicle state and road attributes estimation using unknown input proportional-integral observer," in *2008 IEEE Intelligent Vehicles Symposium*, pp. 690–696, IEEE, jun 2008. [Cited on page 15]

[37] W. He and J. Xi, "A Quaternion Unscented Kalman Filter for Road Grade Estimation," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1635–1640, 2020. [Cited on page 15]

[38] J. Wu, Z. Zhou, J. Chen, H. Fourati, and R. Li, "Fast Complementary Filter for Attitude Estimation Using Low-Cost MARG Sensors," *IEEE Sensors Journal*, vol. 16, pp. 6997–7007, sep 2016. [Cited on page 16]

[39] M. Euston, P. Coote, R. Mahony, Jonghyuk Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 340–345, IEEE, sep 2008. [Cited on page 16]

[40] W. Higgins, "A Comparison of Complementary and Kalman Filtering," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-11, pp. 321–325, may 1975. [Cited on page 16]

[41] 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*, pp. 4112–4117, IEEE, sep 2014. [Cited on page 16]

[42] B. Li, T. Zhang, and T. Xia, "Vehicle Detection from 3D Lidar Using Fully Convolutional Network," *Robotics: Science and Systems*, vol. 12, aug 2016. [Cited on page 16]

[43] S. Song and J. Xiao, "Sliding Shapes for 3D Object Detection in Depth Images," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 634–651, Springer International Publishing, 2014. [Cited on page 16]

[44] S. Song and J. Xiao, "Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2016-Decem, pp. 808–816, IEEE, jun 2016. [Cited on page 16]

[45] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 77–85, dec 2016. [Cited on page 16]

[46] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, pp. 5100–5109, jun 2017. [Cited on page 16]

[47] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4490–4499, 2018. [Cited on page 16]

[48] A. Nguyen and B. Le, "3D point cloud segmentation: A survey," in *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pp. 225–230, IEEE, nov 2013. [Cited on page 16]

[49] A. Sappa and M. Devy, "Fast range image segmentation by an edge detection strategy," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, vol. 2001-Janua, pp. 292–299, IEEE Comput. Soc, 2001. [Cited on page 16]

[50] P. Besl and R. Jain, "Segmentation through variable-order surface fitting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 167–192, mar 1988. [Cited on page 16]

[51] Gabriel Taubin, "Estimation of Planar Curves, Surfaces,and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 11, pp. 1115–1138, 1991. [Cited on page 16]

[52] J. Chen and B. Chen, "Architectural Modeling from Sparsely Scanned Range Data," *International Journal of Computer Vision*, vol. 78, pp. 223–236, jul 2008. [Cited on page 16]

[53] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, jun 1981. [Cited on pages 17 and 27]

[54] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for Point-Cloud Shape Detection," *Computer Graphics Forum*, vol. 26, pp. 214–226, jun 2007. [Cited on page 17]

[55] Y. Li, X. Wu, and Y. Chrysathou, "Globfit: Consistently fitting primitives by discovering global relations," in *ACM SIGGRAPH 2011 papers*, pp. 1–12, 2011. [Cited on page 17]

[56] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Graph-Based Image Segmentation," *International Journal of Computer Vision*, vol. 59, pp. 167–181, sep 2004. [Cited on page 17]

[57] V. Šakénas, O. Kosuchinas, M. Pfingsthorn, and A. Birk, "Extraction of semantic floor plans from 3D point cloud maps," *SSRR2007 - IEEE International Workshop on Safety, Security and Rescue Robotics Proceedings*, no. September, 2007. [Cited on page 17]

[58] M. Nejati and B. D. Argall, "Automated incline detection for assistive powered wheelchairs," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 1007–1012, IEEE, aug 2016. [Cited on page 17]

[59] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object Detection With Deep Learning: A Review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, pp. 3212–3232, nov 2019. [Cited on page 17]

[60] N. O. Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. Velasco-Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep Learning vs. Traditional Computer Vision," *Advances in Intelligent Systems and Computing*, vol. 943, pp. 128–144, oct 2019. [Cited on page 17]

[61] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, may 2017. [Cited on page 17]

[62] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, IEEE, jun 2016. [Cited on page 18]

[63] M. Olafenwa and J. Olafenwa, "ImageAI, an open source python library built to empower developers to build applications and systems with self-contained Computer Vision capabilities." https://github.com/OlafenwaMoses/ImageAI, 2018. [Cited on page 18]

[64] J. Guo, H. He, T. He, L. Lausen, M. Li, H. Lin, X. Shi, C. Wang, J. Xie, S. Zha, A. Zhang, H. Zhang, Z. Zhang, Z. Zhang, S. Zheng, and Y. Zhu, "GluonCV and gluon NLP: Deep learning in computer vision and natural language processing," *Journal of Machine Learning Research*, vol. 21, pp. 1–7, 2020. [Cited on page 18]

[65] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." https://github.com/facebookresearch/detectron2, 2019. [Cited on pages 18 and 31]

[66] L. Euler, "Formulae generales pro translatione quacunque corporum rigidorum," *Novi Commentarii academiae scientiarum Petropolitanae*, pp. 189–207, 1776. [Cited on page 21]

[67] B. Li, T. Gallagher, A. G. Dempster, and C. Rizos, "How feasible is the use of magnetic field alone for indoor positioning?," in *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–9, IEEE, nov 2012. [Cited on page 21]

[68] S. Colton, "The balance filter: a simple solution for integrating accelerometer and gyroscope measurements for a balancing platform," *Chief Delphi white paper*, vol. 1, 2007. [Cited on page 23]

[69] R. G. Lyons, *Understanding Digital Signal Processing*. Addison-Wesley Longman Publishing Co., Inc., 1996. [Cited on page 23]

[70] A.-j. Baerveldt and R. Klang, "A Low-cost and Low-weight Attitude Estimation System for an Autonomous Helicopter," in *Proceedings of IEEE International Conference on Intelligent Engineering Systems*, pp. 391–395, 1997. [Cited on page 24]

[71] M. Mitschke and H. Wallentowitz, *Dynamik der Kraftfahrzeuge*. Springer-Verlag, 2014. [Cited on page 25]

[72] G. Vosselman, B. Gorte, G. Sithole, and T. Rabbani, "Recognising structure in laser scanner point clouds," *International archives of photogrammetry, remote sensing and spatial information sciences*, vol. 46, no. 8, pp. 33–38, 2004. [Cited on page 28]

[73] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1–4, IEEE, may 2011. [Cited on page 29]

[74] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3686–3693, may 2014. [Cited on page 31]

[75] K. Wada, "Labelme: Image Polygonal Annotation with Python." https://github.com/wkentaro/labelme, 2016. [Cited on page 31]

[76] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, p. 60, dec 2019. [Cited on page 32]

[77] A. B. Jung, "imgaug." https://github.com/aleju/imgaug, 2018. [Cited on page 32]

[78] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000. [Cited on page 34]

[79] Withrobot, "myAHRS+ User Manual," 2017. [Cited on pages 35 and 36]

[80] Stereolabs, "ZED2i User Manual," 2019. [Cited on pages 35 and 36]

[81] RoboSense, "RS-Bpearl User Manual," 2020. [Cited on pages 36 and 37]

[82] Velodyne, "VLP-32C User Manual," 2018. [Cited on pages 36 and 37]

[83] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement," *International Journal of Advanced Robotic Systems*, vol. 16, mar 2019. [Cited on page 41]

[84] B. Akpnar, "Performance of different slam algorithms for indoor and outdoor mapping applications," *Applied System Innovation*, vol. 4, no. 4, 2021. [Cited on page 42]

[85] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," pp. 1–16, nov 2017. [Cited on page 57]