



(expleo)



Multi Sensor Ramp Detection and Localization for Autonomous Valet Parking

Master thesis

Felix Saalfrank
368199

April 28, 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

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:

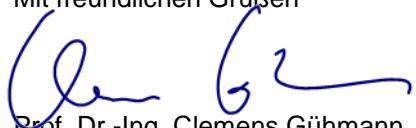
- Research of current methods to determine road grade angle using IMU, LiDAR or camera
- Comparison and selection of the most appropriate method for each sensor
- Implementation of a ramp detection algorithm
 - using an IMU
 - using a LiDAR sensor
 - using a camera
- Testing and optimizing of the methods
- 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	ix
List of Tables	x
List of Acronyms	xi
List of Symbols	xii
1 Introduction	1
1.1 Motivation	1
1.2 Outline	1
2 Background	3
2.1 Mathematical Background	3
2.1.1 Mathematical Representations of 3D Orientations	3
2.1.1.1 Rotation matrix	3
2.1.1.2 Euler angles	3
2.1.1.3 Quaternions	4
2.1.2 Vector projection	5
2.2 Sensors	5
2.2.1 Inertial Measurement Unit (IMU)	6
2.2.1.1 MEMS Accelerometer	7
2.2.1.2 MEMS Gyroscopes	7
2.2.1.3 (MEMS) Magnetometer	8
2.2.1.4 Typical MEMS errors	8
2.2.2 Light Detection and Ranging (LiDAR)	9
2.2.3 Wheel speed sensor	10
2.3 Computer Vision	11
2.3.1 Object detection	11
2.3.2 Image segmentation	11
2.3.3 Neural Networks	11
2.3.3.1 Convolutional Neural Network (CNN)	12
2.3.3.2 Region-based Convolutional Neural Network (R-CNN) . .	12
2.4 Robot Operating System (ROS)	13
3 State of the art	15
3.1 IMU	15
3.2 LiDAR	16
3.3 Camera	16
3.4 Outlook?	16

4 Methods	17
4.1 Road grade definition	17
4.2 Coordinate frames	18
4.3 IMU based	18
4.3.1 Calibration	18
4.3.2 Road grade estimation	20
4.3.2.1 Accelerometer	20
4.3.2.2 Gyroscope	21
4.3.2.3 Complementary filter	21
4.3.2.4 Gravity method	23
4.3.3 Ramp detection and classification	24
4.4 LiDAR based	25
4.4.1 Calibration	25
4.4.2 Algorithm	26
4.4.2.1 Point cloud preprocessing	26
4.4.2.2 Ramp detection and classification	27
4.5 Camera based	27
4.5.1 Approach	28
4.5.2 Dataset	29
4.5.3 Training	30
4.5.4 Point cloud extraction	31
5 Experimental Setup	33
5.1 Sensors	33
5.1.1 IMU	33
5.1.2 LiDAR	34
5.1.3 Camera	35
5.2 Sensor placement	35
5.3 Data recording	37
5.4 Car	37
5.5 Garage	37
6 Results	39
6.1 Evaluation concept	39
6.2 Reference data	39
6.3 Performance measures	41
6.4 IMU	42
6.4.1 Road grade estimation	42
6.4.2 Ramp properties estimation	45
6.5 Ramp detection (LiDAR)	47
6.6 Camera	49
6.6.1 Point cloud	50
7 Conclusion	55
7.1 Advantages and disadvantages of different methods	55
7.2 Outlook	55
7.3 Conclusion	55
Bibliography	56
8 Appendix	60

List of Figures

2.1	zxy Euler convention	4
2.2	Setup of a stable platform IMU	6
2.3	Micro structure of a MEMS accelerometer and gyroscope	7
2.4	Setup of a mechanical spinning LiDAR	10
2.5	Setup of an active wheel speed sensor	10
2.6	R-CNN framework	12
2.7	Mask R-CNN framework	13
2.8	ROS communication	14
4.1	Ramp angle definition	17
4.2	Sensor coordinate frames	18
4.3	Frame transformation	19
4.4	Accelerations on a ramp	20
4.5	Block diagram of the complementary filter	22
4.6	Flow chart of the LiDAR algorithm	28
4.7	Augmented images	30
5.1	The IMUs used in the experiment	33
5.2	The two LiDARs which will be used for the recordings.	34
5.3	LiDAR placement on the car	36
5.4	Car with mounted sensors	38
5.5	Ramps of the garage	38
6.1	Generated point cloud map	40
6.2	Angle estimation using raw measurements	43
6.3	Acceleration from IMU and odometer	43
6.4	Angle estimation using the gravity method	44
6.5	Angle estimation using the gravity method	44
6.6	Comparison of different methods to estimate the length of the ramp.	47
6.7	Resolution comparison of the two Light Detection and Rangings (LiDARs). The ramp region is framed in black. The Velodyne LiDAR has a higher resolution.	49
6.8	Estimated ramp properties and tracking of the distance to the ramp at different distances to the ramp.	52
6.9	Lidar points projected onto the camera image. The green points were iden- tified as part of a ramp by the algorithm.	53
6.10	Predicted bounding box and mask when driving at a ramp. The class of the object and the confidence are shown in the top left corner of the box. .	53
6.11	The projection 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.	54

List of Tables

4.1	Parameters for the preprocessing of the point cloud	26
5.1	Comparison of the two used Inertial Measurement Units (IMUs) [1, 2]	34
5.2	Comparison of the two used LiDARs [3, 4]	35
5.3	Variables used to calculate the optimal mounting angle of the Light Detection and Ranging (LiDAR).	37
5.4	Measured ramp properties	38
6.1	Road grade estimation when driving ramps up, using different methods.	45
6.2	Road grade estimation when driving ramps down, using different methods.	45
6.3	Estimation of ramp angle.	46
6.4	Ramp length estimation	48
6.5	Ramp detection and property estimation evaluation of the LiDAR algorithm. The precision increases the closer the car gets to the ramp.	50
6.6	Detection evaluation	51
6.7	Ramp length	51
8.1	Performance evaluation	60
8.2	Used parameters for lidar algo	61

List of Acronyms

AHRS Attitude Heading Reference System

AVP Automated Valet Parking

CNN Convolutional Neural Network

FOV Field of View

GPS Global Positioning System

HPF High-Pass Filter

IMU Inertial Measurement Unit

IoU Intersection of Union

LiDAR Light Detection and Ranging

LPF Low-Pass Filter

MEMS Microelectromechanical Systems

R-CNN Region-based Convolutional Neural Network

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

- a** Linear acceleration
- θ Pitch angle of the car
- g** Gravitational acceleration
- j** Rotation axis
- n** Normal vector
- ${}_{\mathcal{B}}^{\mathcal{A}} \mathbf{q}$ Quaternion to transform a vector from frame \mathcal{A} to \mathcal{B}
- \otimes Quaternion multiplication
- ${}_{\mathcal{B}}^{\mathcal{A}} \mathbf{M}$ Rotation matrix to transform a vector from frame \mathcal{A} to \mathcal{B}
- R^2 Coefficient of determination
- γ Ramp angle
- α Rotation angle
- v** Vector
- ${}_{\mathcal{A}} \mathbf{v}$ Vector in coordinate frame \mathcal{A}
- $\|\mathbf{v}\|$ Norm of vector
- $\hat{\mathbf{v}}$ Unit vector
- ω** Angular velocity

Todo list

2do	iv
2do	v
This is TODO note by my self	1
This is an improvement note	1
This is a question I have	1
This is a suggestion from others	1
Just copied this from the expose, needs improvement maybe some more sections about the specific topics or about goals	1
2do	1
Add short overview	15
Move everything from this chapter to the corresponding subsection in chapter 3 (e.g. Inertial Measurement Unit (IMU) state of the art after IMU background)	15
Or just switch this chapter with Background chapter	15
Section also contains other methods without imu	15
2do	16
2do	16
Should a part of this moved to Evaluation concept (results)?	37
Measure ramp D and check lengths again	38
Add curb ramp width	38
Chapter description	39
What exactly is a false positive and false negative in my case?	41
Add figure with true and predicted bounding box and corresponding iou score.	42
Brief overview	42
Some more text	45
What about measured length of the ramp? Should I use this value instead?	46
What is a fp and fn in my case?	48
Explain choice of parameters	49
This only explains length, and also why does it look good in 2D image?	51
Describe problem of quantization	51
Just a really short sketch of what could be written here	55

Chapter 1

Introduction

Note explanation

This is TODO note by my self

This is an improvement note

This is a question I have

This is a suggestion from others

1.1 Motivation

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

Parking is one of the most challenging driving tasks and the cause of almost half of the car accidents [5]. 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 [6]. 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.

1.2 Outline

Brief overview over structure of thesis

... following way. In chapter 2 the theoretical foundation needed for the methods used in this thesis is 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 are presented using the Inertial Measurement Unit (IMU) in section 4.3.

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, I will briefly describe the ones used in this thesis.

2.1.1.1 Rotation matrix

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

$$\mathbf{M}\mathbf{M}^\top = \mathbf{M}^\top\mathbf{M} = I_3, \quad \det \mathbf{M} = 1. \quad (2.1)$$

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.

$${}^C_A \mathbf{p} = {}^A_C \mathbf{M}_A \mathbf{p} = {}^B_C \mathbf{M}_B {}^A_C \mathbf{M}_A \mathbf{p} \quad (2.2)$$

to transform the point ${}_A \mathbf{p}$ 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.

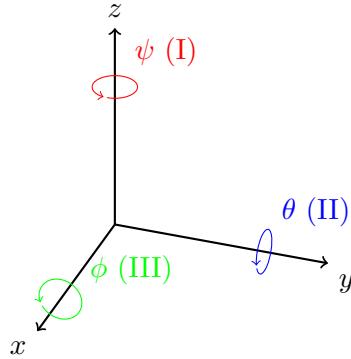


Figure 2.1: Definition of the zyx Euler convention. The rotation starts with a rotation ψ around the z-axis and ends with a rotation ϕ around the x-axis.

The angles are called yaw (or heading), pitch and roll respectively. Roll 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 rotation axes are parallel to each other [7].

2.1.1.3 Quaternions

A quaternion is defined as

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

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

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

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 $\mathbf{q}_v = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^\top$

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_v \\ q_w \end{bmatrix} = \begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}^\top. \quad (2.5)$$

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

$$|\mathbf{q}| = \sqrt{\mathbf{q}^\top \mathbf{q}} = \sqrt{|\mathbf{q}_v|^2 + q_w^2} = 1 \quad (2.6)$$

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

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

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

$${}_{\mathcal{B}}\mathbf{p} = {}_{\mathcal{B}}^{\mathcal{A}}\mathbf{q} \otimes {}_{\mathcal{A}}\bar{\mathbf{p}} \otimes {}_{\mathcal{B}}^{\mathcal{A}}\mathbf{q}^*, \quad (2.8)$$

where

$${}_{\mathcal{A}}\bar{\mathbf{p}} = \begin{bmatrix} 0 & {}_{\mathcal{A}}\mathbf{p}^\top \end{bmatrix}^\top, \quad (2.9)$$

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

$$\mathbf{q}^* = \begin{bmatrix} -\mathbf{q}_v \\ q_w \end{bmatrix}. \quad (2.10)$$

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

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

A quaternion can be converted to a rotation matrix with

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

More information about quaternions can be found in [7, 8].

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

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

The angle between two vectors can be calculated by

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

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 and position 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.

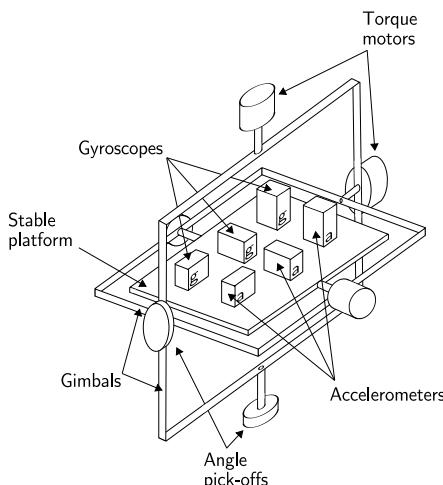


Figure 2.2: Setup of a stable platform IMU [9].

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 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 [9].

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 [10]. 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 [11].

2.2.1.1 MEMS Accelerometer

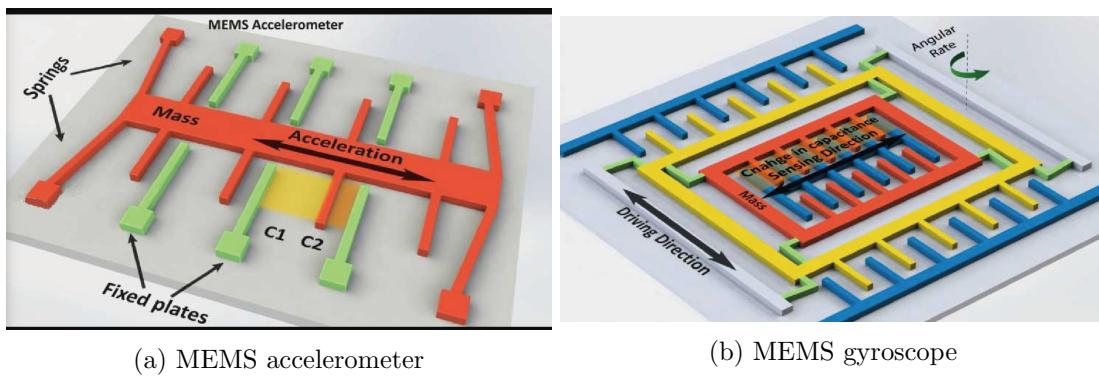


Figure 2.3: Micro structure of a MEMS accelerometer and gyroscope. **TODO: Better images (with ref or make own)**

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. 2.3a. A mass is 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 perpendicular to the axis along which the movement occurs, which affect the capacity between the fixed plates. 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 [12] in a simplified way as

$$\hat{\mathbf{a}} = \mathbf{a} + \mathbf{b}_a + \mathbf{n}_a. \quad (2.15)$$

The measurement differs from the true acceleration \mathbf{a} due to random noise \mathbf{n}_a and a slowly time-varying bias \mathbf{b}_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 ω of mass m and velocity \mathbf{v} experiences a force

$$\mathbf{F}_C = -2m(\omega \times \mathbf{v}). \quad (2.16)$$

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 [11].

Similar to the accelerometer, the measurements $\hat{\omega}$ of the MEMS gyroscope are influenced by errors, which can be modeled according to [12] as

$$\hat{\omega} = \omega + \mathbf{b}_g + \mathbf{n}_g. \quad (2.17)$$

The measurement differs from the true angular velocity ω due to the slowly time-varying bias vector \mathbf{b}_g and the random noise vector \mathbf{n}_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 ($25 \mu\text{T}$ to $65 \mu\text{T}$), the magnetometer readings can easily be influenced by other objects [13].

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 alterations 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 [14]. 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 [15]. 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 than 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 [9].

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 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

$$s = c \frac{\Delta t}{2}. \quad (2.18)$$

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 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 Field of View (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 [17].

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 [18].

After transmitting the laser signal the reflected light passes through the receiving optic

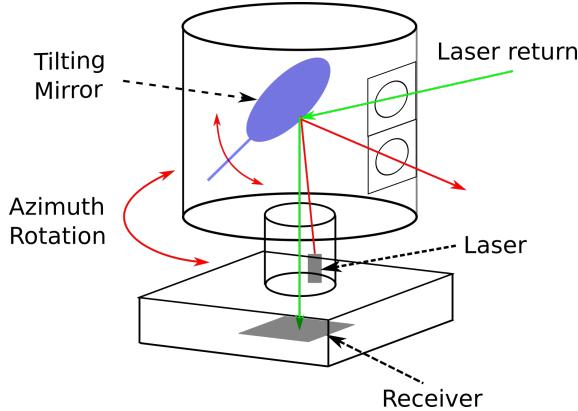


Figure 2.4: Setup of a mechanical spinning LiDAR [16]. The laser is reflected off a tilting mirror and is received through a transmission optic by the receiver. A rotation of the base independently of the mirror allows for a 360° horizontal FOV.

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 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.

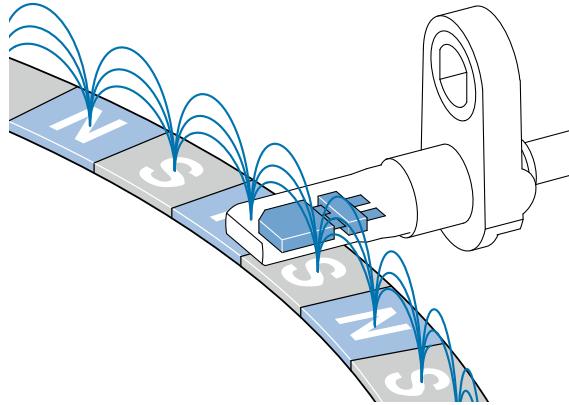


Figure 2.5: Setup of an active wheel speed sensor [19]. The sensor head measures the alternating magnetic field. The amplitude of the induced voltage can then be used to determine the wheel speed.

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 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 [20]. 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 [19]. 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 ref. [21], 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 [22].

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.

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 [23], which can detect up to 80 different type 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. 2.6. 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

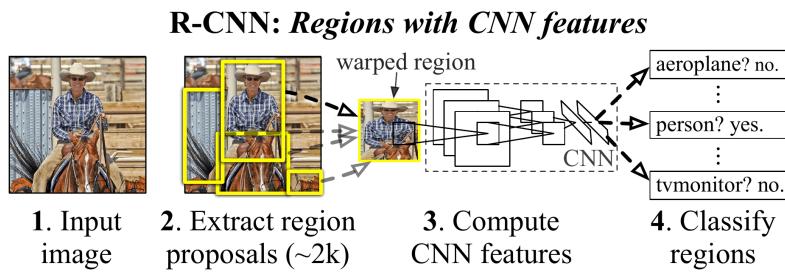


Figure 2.6: The R-CNN framework [23]. The image is divided into subregions and each subregion is classified as one of the object classes or the background independently.

[24] and Faster R-CNN [25]. Fast R-CNN greatly improves the 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 [26] 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. 2.7. 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.

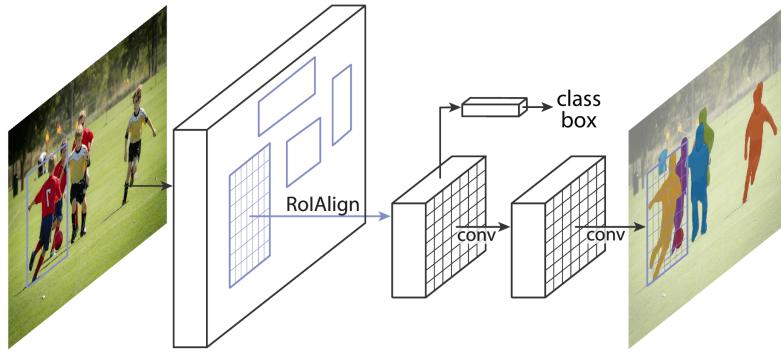


Figure 2.7: The Mask R-CNN framework [26]. It also generates a segmentation mask in addition to the bounding box and class label.

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 [27]. 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. 2.8. 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.

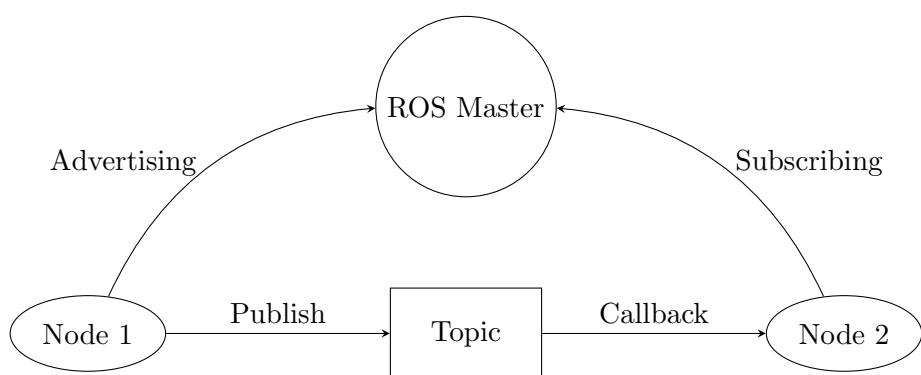


Figure 2.8: ROS communication between two nodes.

Chapter 3

State of the art

Add short overview

Move everything from this chapter to the corresponding subsection in chapter 3 (e.g. Inertial Measurement Unit (IMU) state of the art after IMU background)

Or just switch this chapter with Background chapter

... In this chapter an overview of different existing methods to detect ramps or measure the road grade is given. It is divided into three sections, each focusing on work which relies mainly on the IMU, Light Detection and Ranging (LiDAR) or camera sensor.

3.1 IMU

Section also contains other methods without imu

In ref. [28] 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 [29]. A Kalman filter is also used in ref. [30], 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 [30, 31]. More methods such as recursive least squares, extended Kalman filtering and a dynamic grade observer are discussed in ref. [32]. Another method using GPS data and IMUs to calculate the vertical and horizontal velocity change respectively and thereby the road grade is proposed in ref. [33]. [34] 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 [35]. 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 [36].

Most methods mentioned above do not seem fit for the task, 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 use GPS, but only accelerometers and wheel odometers instead is described in ref. [37] and [38]. 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 not anymore if driving on an elevated road, and can

be used to calculate the road grade angle. A similar approach is used in ref. [39]. [40] 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. [41] 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. [42, 28] also use a complementary, but fuse the estimated angle from the accelerometer and gyroscope instead of the quaternions.

3.2 LiDAR

While it does seem that no research exists, where a LiDAR is used to detect ramps, several papers discuss the detection of planes in point cloud data and estimate road grade? Different methods exist where the LiDAR recordings from a plane are used to map the road grade. In ref.[43] a new technique for plane detection is proposed. The point cloud is down-sampled using octree into small cubes, which are then down-sampled based on their local density. Principal component analysis (PCA) is used to find planar surfaces. It is able to process large point clouds faster than the efficient Random Sample Consensus (RANSAC) algorithm, while achieving a higher accuracy. Ref. [44] 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.

- LiDAR methods for plane detection
- If found, methods for elevation estimation

3.3 Camera

- Something simple
- Stereo vs mono

Ref. [45] 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 were determined as well. A ground plane estimation using a disparity map, which can be generated from a stereo camera, is proposed in [46].

3.4 Outlook?

Due to the available sensor stack, not all mentioned methods can be tested. While a Kalman filter achieves very good results, it is generally complex and the precise knowledge of process and measurement noise is necessary [47] The gravity method and complementary filter will be tested for the IMU. Because no method to detect ramps using a LiDAR could be found, an own method has to be implemented. For the plane detection the RANSAC algorithm will be used, since it is already implemented in the PCL library ADD LINK.

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 Inertial Measurement Unit (IMU) are described. The angle is then used to identify a ramp. A novel method using a Light Detection and Ranging (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

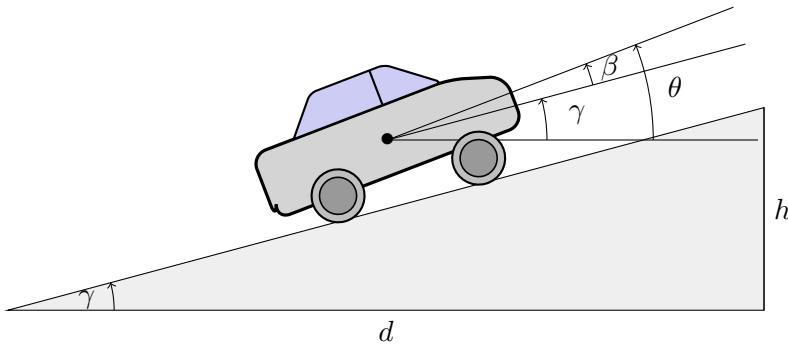


Figure 4.1: Car driving on a ramp. Due to forward acceleration the car tilts back.

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

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

or in percentage

$$r = 100 \cdot \tan(\gamma). \quad (4.2)$$

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 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 center in the front of the car at the height of the wheel axle.

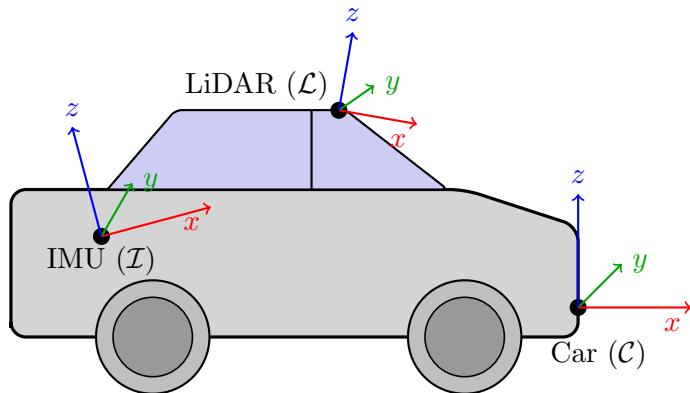


Figure 4.2: Example of a typical sensor setup. All measurements of the sensors are recorded in the sensor frame and must be transformed to the car frame.

4.3 IMU based

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 $\mathcal{I}^{\mathcal{C}}\mathbf{M} \in \mathbb{R}^{3 \times 3}$ which transforms the measurements of the linear acceleration $\mathcal{I}\mathbf{a}_k \in \mathbb{R}^{1 \times 3}$ and angular velocity $\mathcal{I}\boldsymbol{\omega}_k \in \mathbb{R}^{1 \times 3}$ 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.

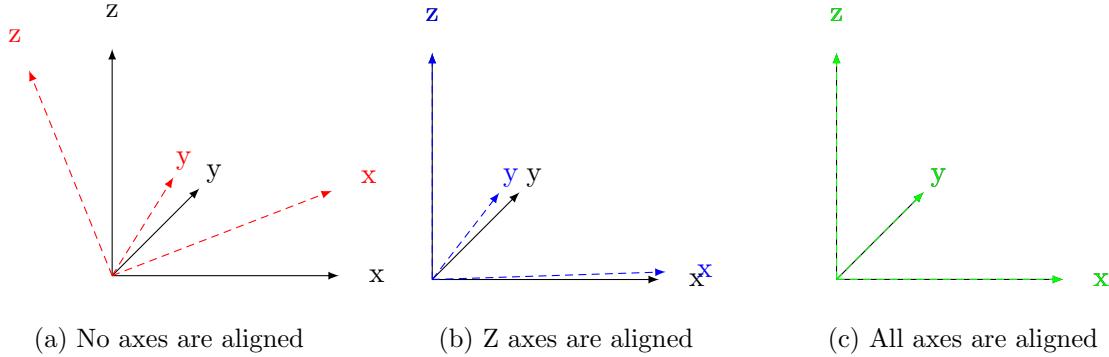


Figure 4.3: Frame transformation from the IMU frame \mathcal{I} , over the intermediate frame \mathcal{B} to the car frame \mathcal{C} .

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 frame is measured only in upwards z-direction. Using this, a transformation from 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, 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

$$\mathbf{j} = \frac{\mathcal{I}\hat{\mathbf{a}} \times \mathcal{C}\hat{\mathbf{z}}}{\|\mathcal{I}\hat{\mathbf{a}} \times \mathcal{C}\hat{\mathbf{z}}\|} \quad (4.3)$$

and the rotation angle with

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

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

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

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 [48], 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. Resulting in the rotation angle

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

and the quaternion

$$\overset{\mathcal{B}}{c}\mathbf{q} = \begin{bmatrix} c\hat{\mathbf{z}} \cdot \sin\left(\frac{\beta}{2}\right) \\ \cos\left(\frac{\beta}{2}\right) \end{bmatrix}. \quad (4.7)$$

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

$$\overset{\mathcal{I}}{c}\mathbf{q} = \overset{\mathcal{B}}{c}\mathbf{q} \otimes \overset{\mathcal{I}}{B}\mathbf{q} \quad (4.8)$$

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

$$c\mathbf{a}_k = \overset{\mathcal{I}}{c}\mathbf{M} \cdot \overset{\mathcal{I}}{c}\mathbf{a}_k \quad (4.9)$$

$$c\boldsymbol{\omega}_k = \overset{\mathcal{I}}{c}\mathbf{M} \cdot \overset{\mathcal{I}}{c}\boldsymbol{\omega}_k. \quad (4.10)$$

4.3.2 Road grade estimation

4.3.2.1 Accelerometer

If the car stands still, the only measurable acceleration besides 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}_x of the overall gravity \mathbf{g} can then be used to determine the pitch angle in the following way

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

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. 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}_{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

$$\mathbf{a}_x = \mathbf{a}_{car} + \mathbf{g}_x \quad (4.12)$$

and $\mathbf{a}_x = \mathbf{g}_x$ 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.

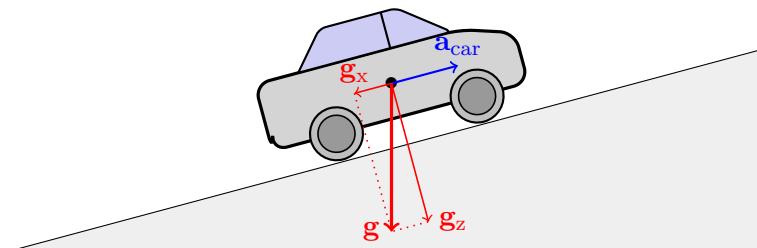


Figure 4.4: The prevailing accelerations when a car is accelerating on a ramp.

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

$$\boldsymbol{\omega} = \frac{d\boldsymbol{\theta}}{dt}. \quad (4.13)$$

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

$$\boldsymbol{\theta}(t = T) = \boldsymbol{\theta}_0 + \int_0^T \boldsymbol{\omega}(t) dt \quad (4.14)$$

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

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{\omega}_k \Delta t. \quad (4.15)$$

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

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

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 along the y-axis are of interest (see fig. 4.2). The pitch angle at time k is given by

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

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 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 [49]. 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

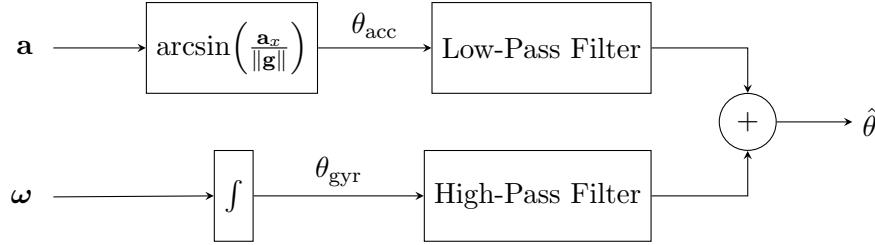


Figure 4.5: The block diagram of the complementary filter.

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 [50]. A block diagram of the complementary filter is shown in fig. 4.5.

The angle estimation obtained from the linear acceleration (eq. (4.11)) will be referred to as θ_{acc} , the 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 ω_{gyr} measured by the gyroscope. 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 ω_{gyr} as $\Omega_{\text{gyr}}(s)$. The complimentary filter estimate is computed by

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

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 [7]. 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

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

for the LPF and

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

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

$$\alpha = \frac{\tau}{\tau + T} = \frac{1}{2\pi f_0} \quad (4.21)$$

with τ being the time constant of the filter and T being the sample period. 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

$$\hat{\theta}_k = \gamma (\hat{\theta}_{k-1} + T\omega_{k,\text{gyr}}) + (1 - \gamma)\theta_{k,\text{acc}}, \quad (4.22)$$

with

$$\gamma = \frac{\alpha}{\alpha + T} \in [0, 1] \quad (4.23)$$

being the filter gain. The choice of the parameter α (and hence γ) determines, how much each of the two measurements should be trusted. Selecting a small α (γ 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 γ close to one and low cut-off frequency,

where the gyroscope estimation is trusted more [51].

Selecting the right filter coefficient is a known problem and is usually solved by calculating an initial guess followed by 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 $\gamma = 0.9938$. For the experiment a cut-off frequency of $f_0 = 0.16076$ was chosen, resulting in a filter gain of $\gamma = 0.99$.

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.

4.3.2.4.1 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 [52]. 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 $4\frac{\text{m}}{\text{s}^2}$), which will not be surpassed in the experiments. Using the assumptions from above, the car velocity $v_{\text{car}}(t)$ can be calculated by

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

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

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

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

$$a_{\text{car}}(t) = \frac{d}{dt} v_{\text{car}}(t). \quad (4.27)$$

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

$$a_{k,\text{car}} = \frac{v_{k,\text{car}} - v_{k-1,\text{car}}}{T} \quad (4.28)$$

with $T = \frac{1}{f}$ 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

$$\hat{v}_{k,\text{car}} = \frac{1}{N} \sum_{i=1}^N v_{k-i,\text{car}}. \quad (4.29)$$

The finite difference of the approximation can then be calculated as

$$\hat{a}_{k,\text{car}} = \frac{\hat{v}_{k,\text{car}} - \hat{v}_{k-1,\text{car}}}{T}. \quad (4.30)$$

4.3.2.4.2 Gravity method

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

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

with \mathbf{a}_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}_{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 increases and decreases at the beginning and end respectively, the average angle is calculated. It is assumed that the average angle is equal to angle in the middle of the ramp. The average angle of the ramp is then calculated by measuring the pitch angle and checking whether the rate of the change is near zero for a certain 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 outliers.

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. If the car is on a ramp, the acceleration due to gravity is measured as well. But by using eq. (4.11) and transforming it to

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

the acceleration due to gravity can 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 with respect to the time to get the velocity

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

and then integrating the velocity to get the length

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

where the velocity $v_{1,\text{car}}$ is the velocity at the start of the ramp and $v_{k,\text{car}}$ is the velocity at the end of the ramp. Δt is the sample time. In eq. (4.34) either the estimated velocity \hat{v}_{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

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, a transformation from the LiDAR frame to the car frame is necessary. The calibration 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 finding the transform, such that the normal vector of the plane aligns with the z-axis of the car (the ground plane gets projected onto the xy-plane of the car frame). This results in the correct pitch and roll 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

$$ax + by + cz + d = 0 \quad (4.35)$$

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

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

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.

Then the normal vector of the plane, which can be conducted from the plane equation as follows

$$\mathbf{n} = \begin{pmatrix} a & b & c \end{pmatrix}^\top \quad (4.37)$$

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 RANSAC. 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. 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, that the calculated roll angle should be minimal. If that is not the case, most likely a side wall has 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. The yaw angle and the x- and y-translation from the LiDAR to the centered front of the car must be entered manually, but the pitch and roll angle and the distance from the LiDAR to the ground are estimated in the calibration process.

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 exact values used for the passthrough filter can be seen along the other parameters in table 4.1. The next step in reducing the point cloud size is the voxel grid filter [54]. 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.

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

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 are left in the point cloud. For the implementation of the RANSAC algorithm and the voxel grid filter the PCL library [55] 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 $(0 \ 0 \ 1)^\top$ 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. Beside 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 after the RANSAC algorithm. 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.

4.5 Camera based

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

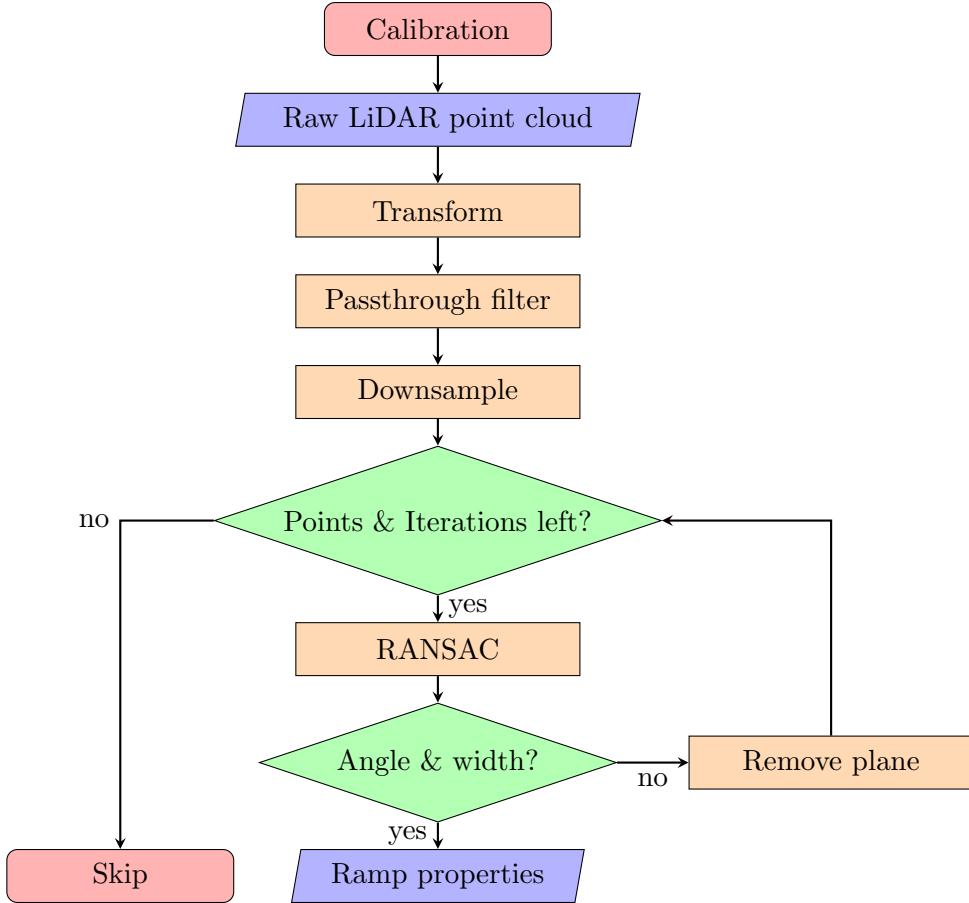


Figure 4.6: Flow chart of the ramp detection algorithm using the LiDAR sensor.

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 would probably be 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 Region-based Convolutional Neural Network (R-CNN) architecture 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**¹ [56] 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 [57] 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 the three ramps used for the training is 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**² [58] 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 curbside is 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 had to be converted to the COCO format, to allow for the training using the **Detectron2** library. The COCO format contains information about the segmentation mask consisting of the coordinates of the polygon, the bounding box and the class of the object.

¹<https://github.com/facebookresearch/detectron2>

²<https://github.com/wkentaro/labelme>

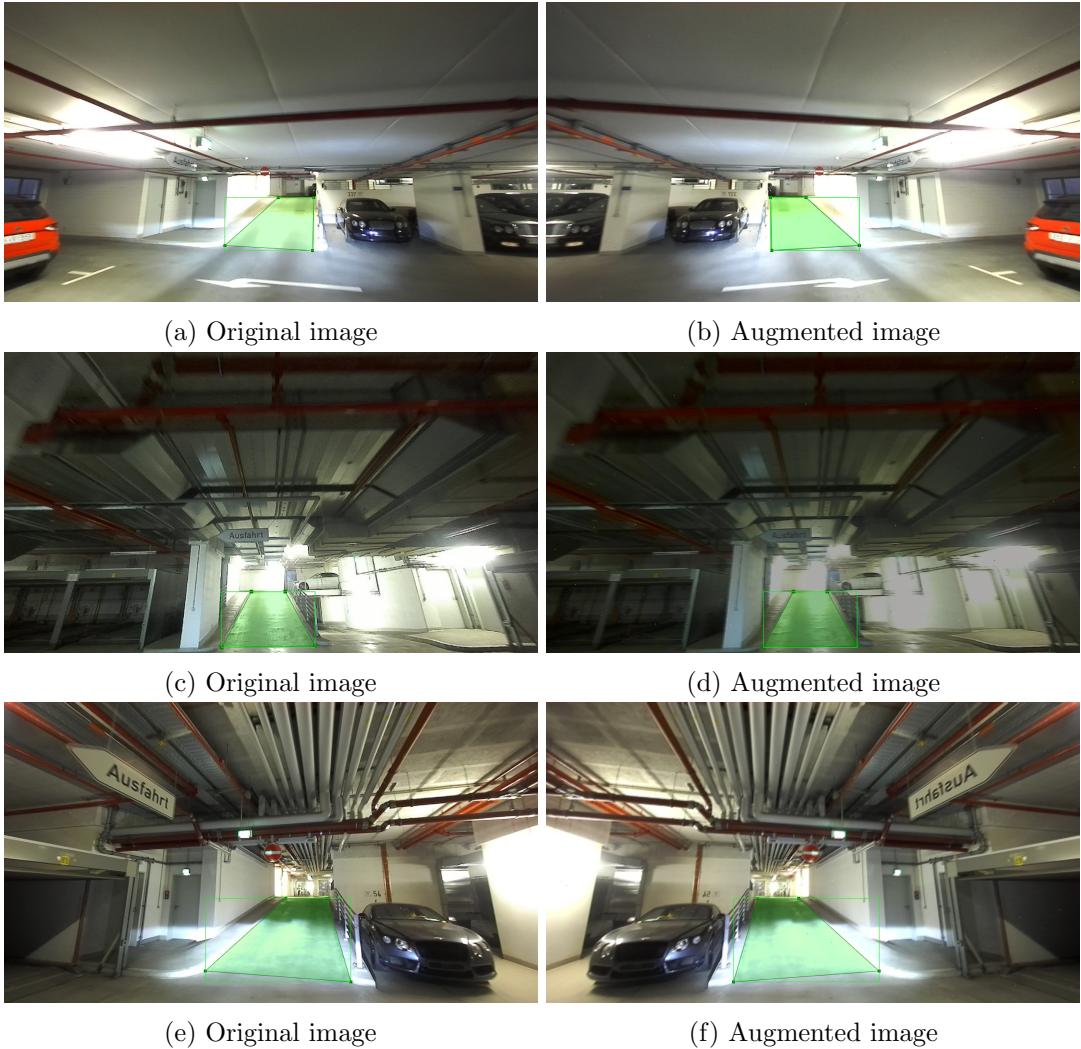


Figure 4.7: The three ramps used for the training. The augmented images are shown on the right-hand side and the manually created label is marked green.

4.5.3 Training

Since the size of the dataset is not very great, 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 [59]. 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`³ [60] 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%), changing the brightness by -20 % to 20 %, changing the contrast by -20 % to 10 %, adding motion blur with a kernel size of 3 and 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

³<https://github.com/aleju/imgaug>

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. 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. If the learning rate is 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.

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 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` library [61], 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_0, y_0, w, h) with x_0, y_0 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 binary image with the same size as the camera image. 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.

Now that only points inside the ramp region are left, the RANSAC algorithm can be applied. It is used to find the best fitting plane of all the inliers and removes potential outliers. 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 by hand, which is subject to error.

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+ [1]

(b) Stereolabs ZED 2i camera [2]

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 can be read 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 [1, 2]

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

5.1.2 LiDAR



(a) Robosense RS-Bpearl [3]

(b) Velodyne Ultra puck [4]

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.

¹https://github.com/gaowenliang imu_utils

Table 5.2: Comparison of the two used LiDARs [3, 4]

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

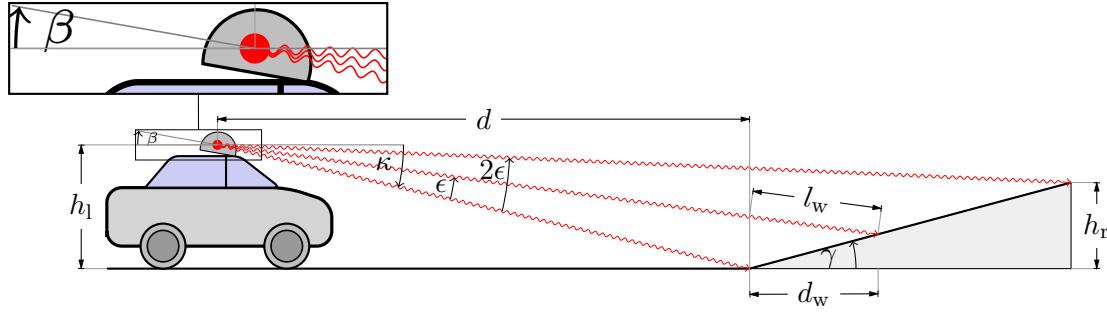


Figure 5.3: Calculation of the best mounting angle of the LiDAR on the car. Variable description in table 5.3.

wave is defined as

$$\kappa = \beta - i\epsilon \quad (5.1)$$

with $i \in [0, 1, 2, \dots, n]$ 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

$$d_{\text{hit,ground}} = \tan(90^\circ - \kappa) h_l. \quad (5.2)$$

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

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

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

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

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

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

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

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

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 $\beta_{\text{velodyne}} = 0^\circ$ and $\beta_{\text{robos}} = 20^\circ$. 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

Should a part of this moved to Evaluation concept (results)?

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 $5 \frac{\text{km}}{\text{h}}$ 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



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

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 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.

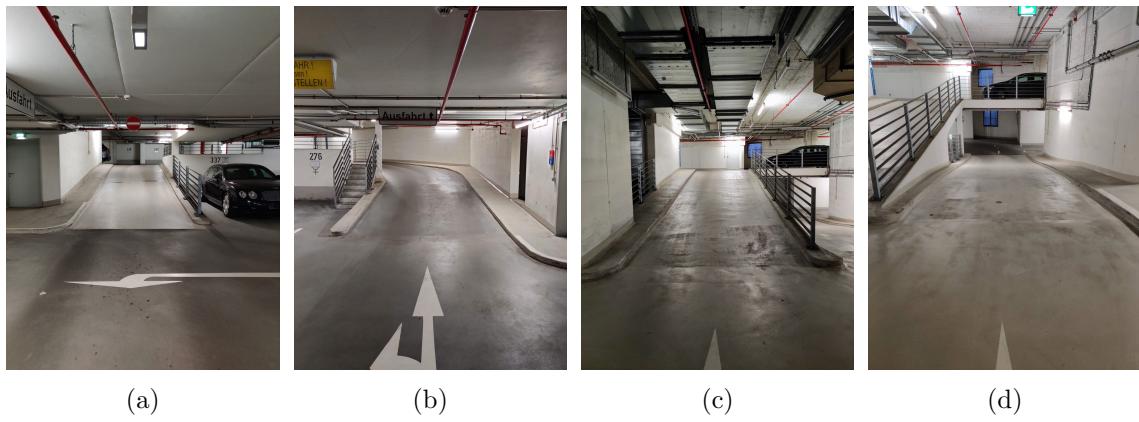


Figure 5.5: On the ramps A, B and C the car will be driven up. On the ramp D as well as also on the ramp A the car will be driven down.

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	-7.3?	3.85	11.89

Measure ramp D and check lengths again

Add curb ramp width

Chapter 6

Results

Chapter description

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 were 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 Robot Operating System (ROS) package `hdl_graph_slam`¹ is used for this task. It is based on 3D graph Simultaneous Localization and Mapping (SLAM) and uses the LiDAR data to map the environment and estimate the pose (position and orientation) of the car. It uses 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 each other. The algorithm then estimates the translation and orientation difference between those two point clouds. In ref. [62] 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.

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

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

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

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 and brighter points and the white arrows 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.

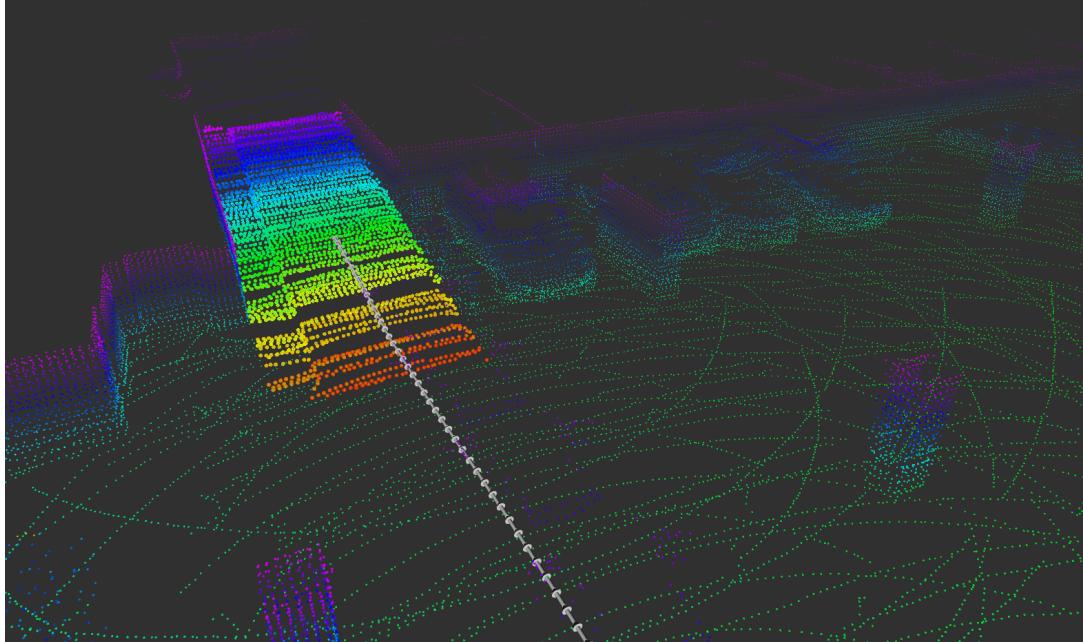


Figure 6.1: The by the `hdl_graph_slam` package generated point cloud map. The ramp is marked by the brighter points. The white arrows visualize the trajectory of the car.

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 $\hat{y} = (\hat{y}_1, \dots, \hat{y}_n)^\top$ and the reference $y = (y_1, \dots, y_n)^\top$ can then be described by the Root Mean Square Error (RMSE)

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

which quantifies 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

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

where \bar{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 Positives (FPs). What is counted as TP or FPs 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.

What exactly is a false positive and false negative in my case?

Analogously, a frame is classified as FP, when a ramp was detected even though it was not visible or if less than 70% of the detected points lie inside the ramp region.

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 COCO dataset. It is based on the precision and recall score, which also considers the number of False Negatives (FNs) in addition to the TPs and FPs. The scores are calculated by

$$\text{Precision} = \frac{TP}{TP + FP} \quad (6.4)$$

$$\text{Recall} = \frac{TP}{TP + FN}. \quad (6.5)$$

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

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}. \quad (6.6)$$

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. 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%. A visualization of the IoU score calculation is shown in FIG.

Since selecting a meaningful threshold for the AP score depends on the dataset, the Mean Average Precision (mAP) is 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.

Add figure with true and predicted bounding box and corresponding iou score.

6.4 IMU

Brief overview

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 goodness 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 and is easily influenced by accelerations other than gravity, which can be seen at time 2s to 4s, 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 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 (21s to 23s) the gravity method shows its advantages. The deceleration before entering the ramp is measured by both sen-

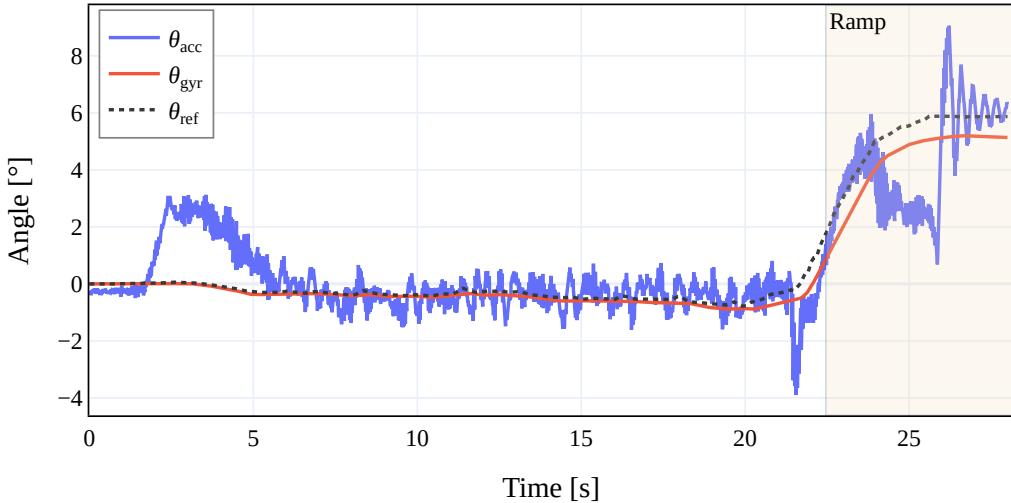


Figure 6.2: Pitch angle estimation from the raw accelerometer and gyroscope measurements. The drive did start from stand still and did end in the middle of the ramp.

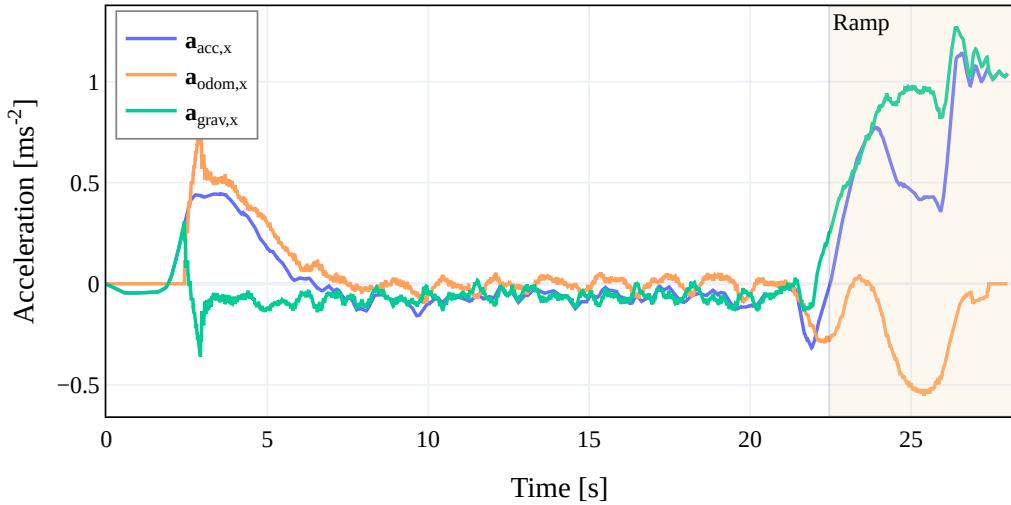


Figure 6.3: The measured acceleration in x-direction by the IMU and the acceleration derived from the wheel speed measurements and the difference between both.

sors 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 (2s to 4s). Although both sensor 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

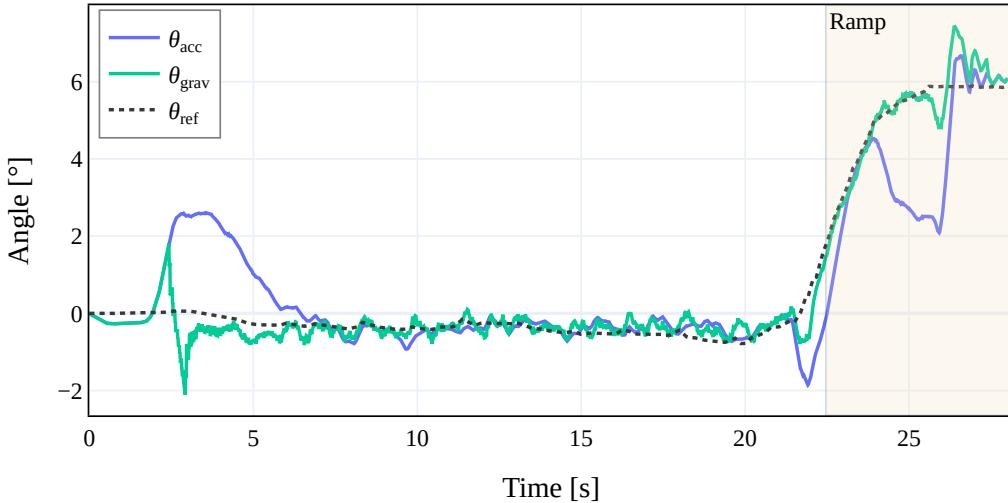


Figure 6.4: Pitch angle estimation using only the accelerometer compared to the gravity method, which additionally uses the wheel speed measurements.

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

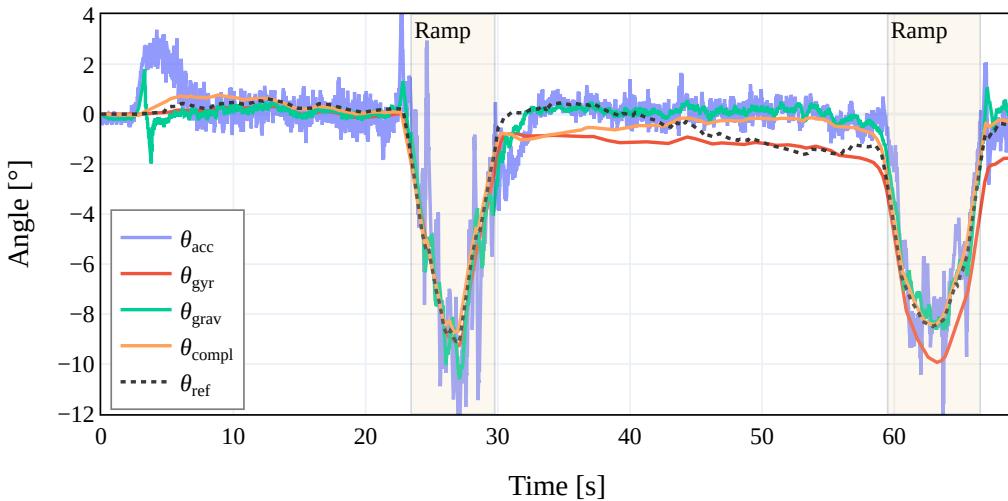


Figure 6.5: Comparison of different methods to estimate the pitch angle. Two different ramps were driven down successively.

plementary 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 different methods for two different 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

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

Method	Ramp A			Ramp B		
	RMSE [°]	R^2	Error _{max} [°]	RMSE [°]	R^2	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

a test 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 [°]	R^2	Error _{max} [°]	RMSE [°]	R^2	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 Ramp properties estimation

Now that the estimation of the road grade is done, the ramp properties can be estimated. The estimated ramp angle for three different ramps is listed in table 6.3.

Some more text

Table 6.3: Estimation of ramp angle.

Method	Ramp A		Ramp B		Ramp D	
	Angle [°]	Error [°]	Angle [°]	Error [°]	Angle [°]	Error [°]
Accelerometer	4.14	3.06	4.75	1.75		
Gyroscope	4.97	2.23	4.30	2.20		
Acceleration method	6.15	1.05	6.18	0.32		
Complementary	5.42	1.78	6.12	0.38		

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 show in fig. 6.5.

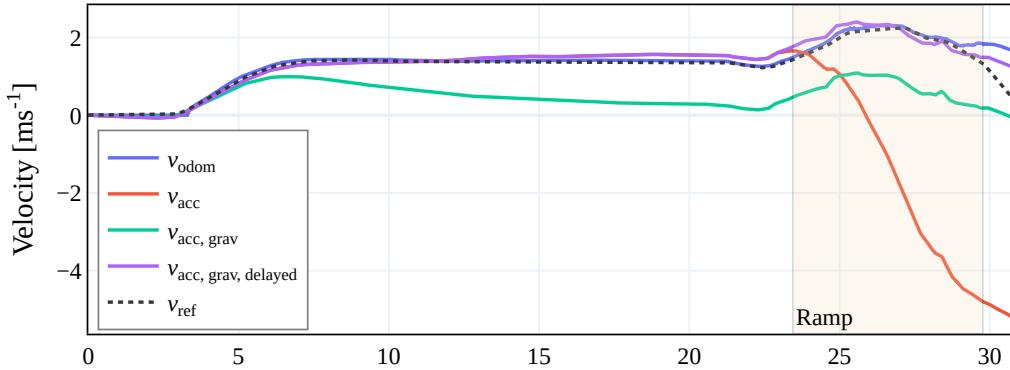
In fig. 6.6a different methods to estimate the velocity of the car are shown. v_{odom} is the velocity calculated from the wheel speed measurements using eq. (4.26). The other three methods are based on the integration of the accelerometer measurements. v_{acc} is the result of integrating the raw accelerometer 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.

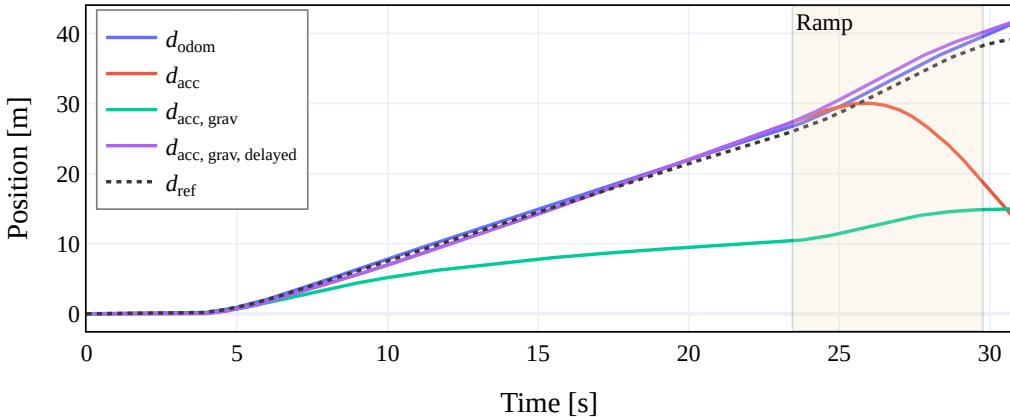
The estimation of the travelled distance for the different methods is shown in fig. 6.6b. Since the distance is calculated by integrating the velocity, an error is cumulated over time. This can be seen in both d_{acc} and $d_{\text{acc,grav}}$. The only usable results are produced by the method using 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.

What about measured length of the ramp? Should I use this value instead?



(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.

6.5 Ramp detection (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, 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 FPs, 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 of length.

The results for three different ramps are shown in table 6.5. Each drive started about

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

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.

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 ramp, which results together with the other recordings in a precision score of 99.54 % or 99.91 %.

What is a fp and fn in my case?

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 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.

Fig. 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. 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 great 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

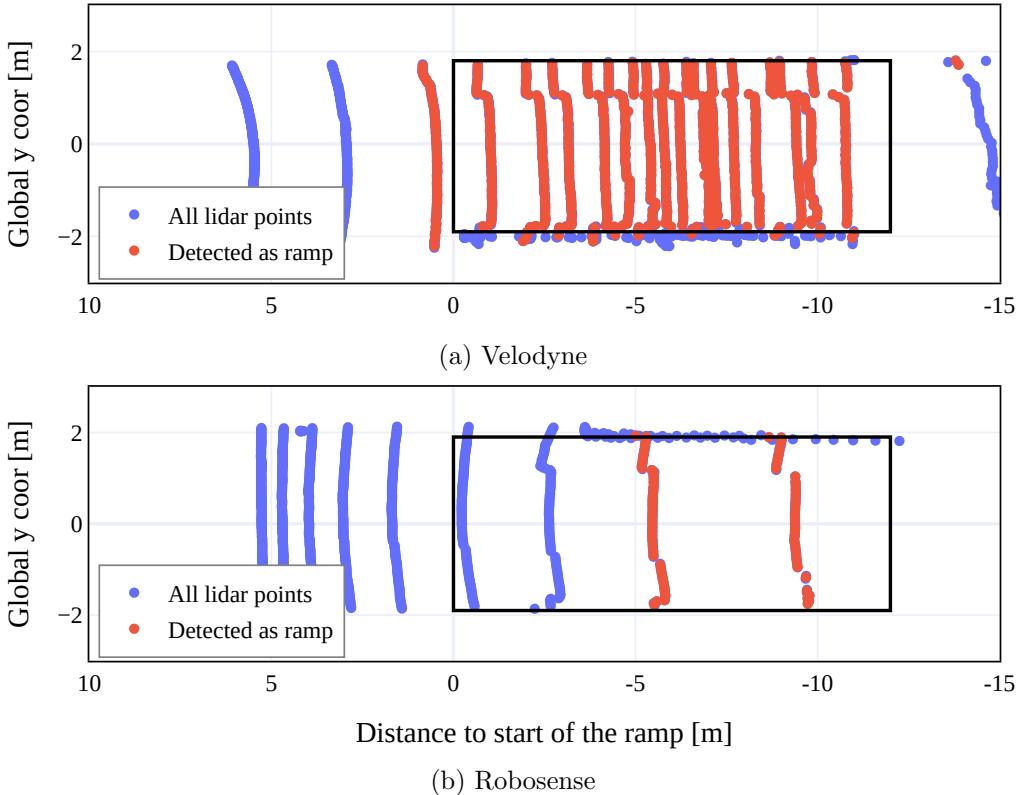


Figure 6.7: Resolution comparison of the two LiDARs. The ramp region is framed in black. The Velodyne LiDAR has a higher resolution.

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 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

The camera was used to detect the ramp and mask the region. 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. The influence of the different hyperparameters on the score was ?

Explain choice of parameters

In table 6.6 the mAP scores for different hyperparameters are listed. Since ... achieved the best results, all the following images and calculations were generated using this configuration.

A visualization of the prediction for when driving straight and from an angle at a ramp

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[%]	FP[%]	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	0.00	0.65	1.70	0.12	6.28
	25 to 30	50	59.78	0.00	1.79	1.21	0.25	10.04
B	0 to 5	21	100.00	0.00	0.64	0.87	0.05	2.65
	5 to 10	23	100.00	0.00	0.50	0.89	0.11	4.10
	10 to 15	28	100.00	0.00	0.44	0.70	0.10	1.48
	15 to 20	27	37.04	3.70	1.11	4.82	0.18	2.75
	20 to 25	29	0.00	3.45	3.28	20.34	0.04	3.39
	25 to 30	28	10.71	0.00	1.91	1.40	0.25	11.48
C	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	0.68	2.58	0.02	3.44
	20 to 25	61	97.83	2.17	0.90	3.40	0.12	9.31
	25 to 30	59	42.75	0.00	1.82	4.35	0.90	11.98

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.

6.6.1 Point cloud

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, object 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,

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

the nearest marked points are multiple cm further away. 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. Using the point cloud of the ZED camera, both the width and length of the ramp are underestimated. This can attribute to the fact, that the range is limited. The angles are both bad because the transformation are bad... Also, as mentioned in section 4.5.4, the accuracy of the back into the 3D projected point cloud depends on the resolution of the image. Since during the projection from 3D to 2D multiple points can be projected onto the same pixel, information is lost when transforming back.

This only explains length, and also why does it look good in 2D image?

Describe problem of quantization

Table 6.7: The estimation of the ramp length using different methods.

Method	Angle [°]	Width [m]	Length [m]
ZED Box	8.84	2.80	6.23
ZED Mask	8.72	2.60	6.25
Lidar Box	7.78	4.40	9.49
Lidar Mask	7.64	3.30	10.19

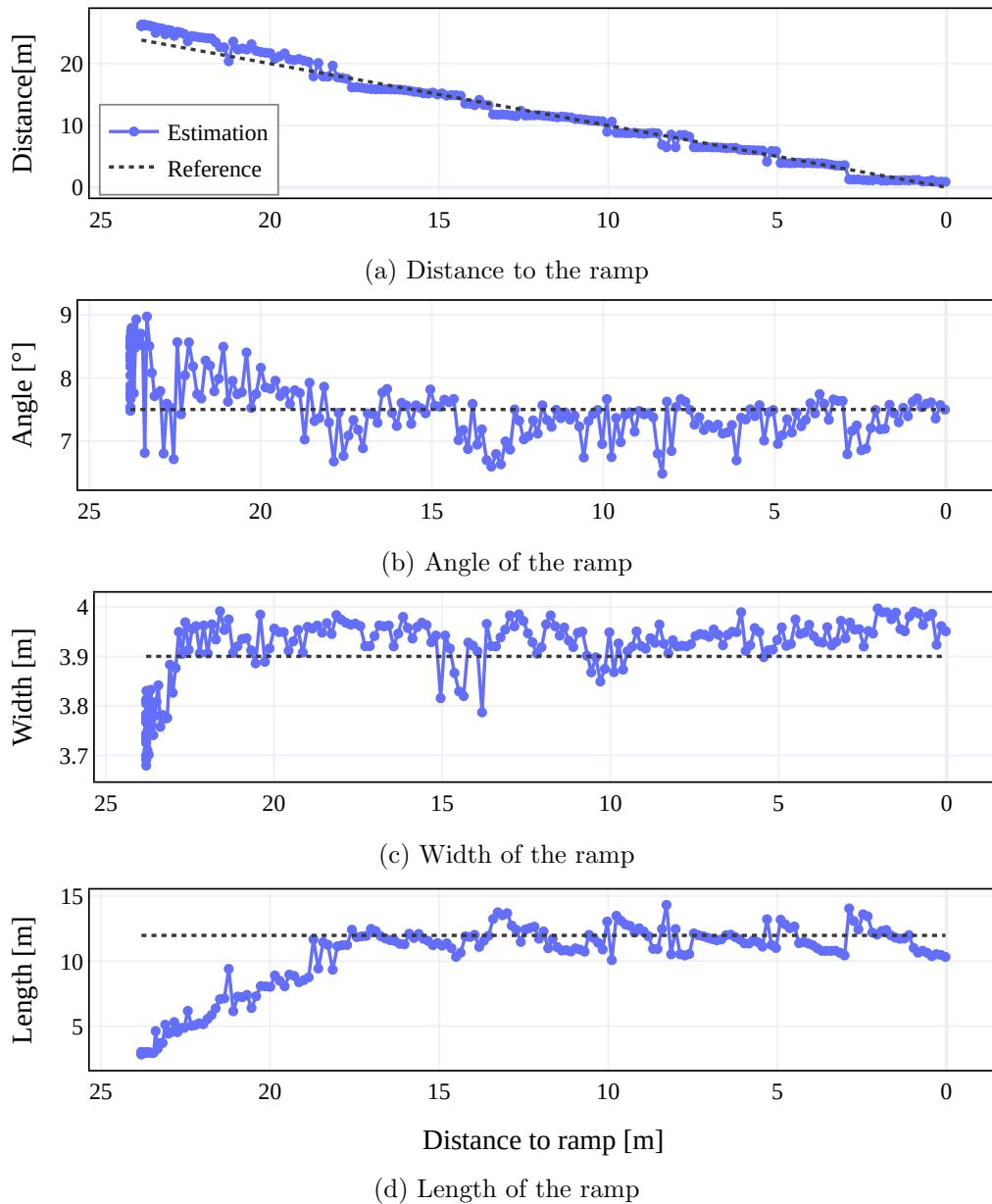


Figure 6.8: Estimated ramp properties and tracking of the distance to the ramp at different distances to the ramp.



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

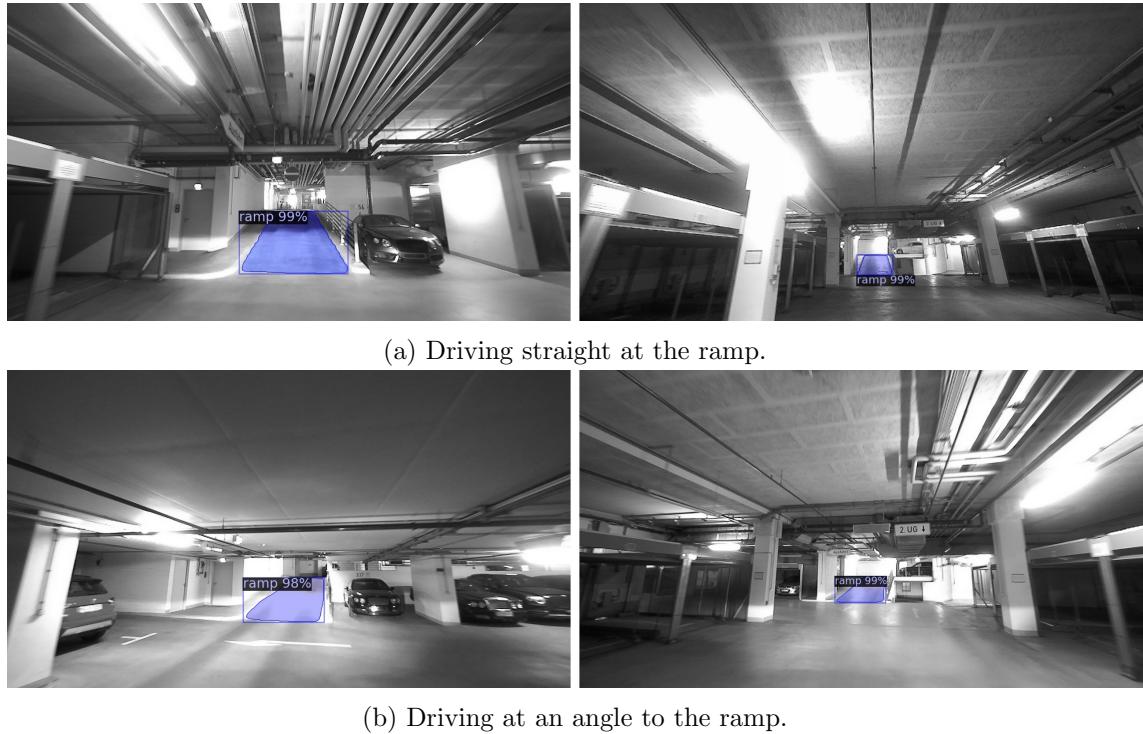
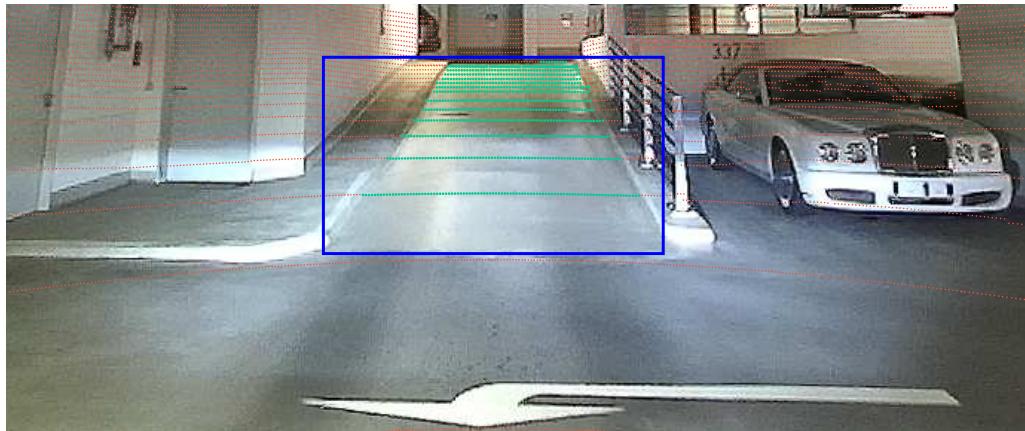


Figure 6.10: Predicted bounding box and mask when driving at a ramp. The class of the object and the confidence are shown in the top left corner of the box.



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



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

Figure 6.11: The projection 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.

Chapter 7

Conclusion

7.1 Advantages and disadvantages of different methods

Just a really short sketch of what could be written here

Using the Inertial Measurement Unit (IMU) and odometer, no additional hardware is needed, since both sensors already available in most currently available cars. The Light Detection and Ranging (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 Microelectromechanical Systems (MEMS) LiDARs become cheaper. The camera is available in most cars, but for a robust neural network, a bigger dataset is needed.

7.2 Outlook

- Sensors could be fused
- Other IMU methods such as kalman filter
- use camera and extract point cloud from LiDAR
- use LiDAR to detect ramp and use it to label ramp in image
- octomap
-

7.3 Conclusion

Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Bibliography

- [1] Withrobot, “myAHRS+ User Manual,” 2017.
- [2] Stereolabs, “ZED2i User Manual,” 2019.
- [3] RoboSense, “RS-Bpearl User Manual,” 2020.
- [4] Velodyne, “VLP-32C User Manual,” 2018.
- [5] A. SE, “A sudden bang when parking,” 2015.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] O. J. Woodman, “An introduction to inertial navigation,” Research report 696, University of Cambridge, aug 2007.
- [10] D. K. Shaeffer, “MEMS inertial sensors: A tutorial overview,” *IEEE Communications Magazine*, vol. 51, no. 4, pp. 100–109, 2013.
- [11] 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.
- [12] 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.
- [13] M. Kok and T. B. Schon, “Magnetometer Calibration Using Inertial Sensors,” *IEEE Sensors Journal*, vol. 16, pp. 5679–5689, jul 2016.
- [14] 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.
- [15] 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.

- [16] 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.
- [17] T. Fujii and T. Fukuchi, *Laser Remote Sensing*. CRC Press, 2005.
- [18] D. Wang, C. Watkins, and H. Xie, “MEMS Mirrors for LiDAR: A Review,” *Micro-machines*, vol. 11, p. 456, apr 2020.
- [19] K. Reif, *Bosch Autoelektrik und Autoelektronik*. Springer-Verlag, 2011.
- [20] K. Reif, *Automobilelektronik: eine Einführung für Ingenieure*. Springer-Verlag, 2014.
- [21] R. Szeliski, *Computer Vision*. Texts in Computer Science, Springer International Publishing, 2022.
- [22] 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.
- [23] 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.
- [24] R. Girshick, “Fast R-CNN,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, vol. 2015 Inter, pp. 1440–1448, IEEE, dec 2015.
- [25] 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.
- [26] 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.
- [27] 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.
- [28] 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.
- [29] 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.
- [30] 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.
- [31] 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.
- [32] 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.

- [33] 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.
- [34] 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.
- [35] 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.
- [36] B. Ozdenizci, V. Coskun, and K. Ok, "NFC Internal: An Indoor Navigation System," *Sensors*, vol. 15, pp. 7571–7595, mar 2015.
- [37] M. Nilsson and E. Öhlund, "Estimation of road inclination," 2012.
- [38] 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.
- [39] 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.
- [40] 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.
- [41] 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.
- [42] 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.
- [43] E. El-Sayed, R. F. Abdel-Kader, H. Nashaat, and M. Marei, "Plane detection in 3D point cloud using octree-balanced density down-sampling and iterative adaptive plane extraction," *IET Image Processing*, vol. 12, no. 9, pp. 1595–1605, 2018.
- [44] 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.
- [45] 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.
- [46] N. Chumerin and M. M. Van Hulle, "Ground Plane Estimation Based on Dense Stereo Disparity," *The Fifth International Conference on Neural Networks and Artificial Intelligence (ICNNNAI)*, no. May, pp. 209–213, 2008.
- [47] W. Higgins, "A Comparison of Complementary and Kalman Filtering," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-11, pp. 321–325, may 1975.
- [48] 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.

- [49] 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.
- [50] R. G. Lyons, *Understanding Digital Signal Processing*. Addison-Wesley Longman Publishing Co., Inc., 1996.
- [51] 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.
- [52] M. Mitschke and H. Wallentowitz, *Dynamik der Kraftfahrzeuge*. Springer-Verlag, 2014.
- [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.
- [54] 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.
- [55] 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.
- [56] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2.” <https://github.com/facebookresearch/detectron2>, 2019.
- [57] 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.
- [58] K. Wada, “Labelme: Image Polygonal Annotation with Python.” <https://github.com/wkentaro/labelme>, 2016.
- [59] 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.
- [60] A. B. Jung, “imgaug.” <https://github.com/aleju/imgaug>, 2018.
- [61] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [62] B. Akpnar, “Performance of different slam algorithms for indoor and outdoor mapping applications,” *Applied System Innovation*, vol. 4, no. 4, 2021.

Chapter 8

Appendix

Code extracts and extra plots etc

Table 8.1: Performance evaluation

Structure	Distance	Frames	TP	FP
uc	0 m to 5 m	62	100.00%	0.00%
uc	5 m to 10 m	62	100.00%	0.00%
uc	10 m to 15 m	59	100.00%	0.00%
uc	15 m to 20 m	61	97.92%	2.08%
uc	20 m to 25 m	61	97.83%	2.17%
uc	25 m to 30 m	59	42.75%	0.00%
us	0 m to 5 m	116	100.00%	0.00%
us	5 m to 10 m	117	100.00%	0.00%
us	10 m to 15 m	116	100.00%	0.00%
us	15 m to 20 m	123	100.00%	0.00%
us	20 m to 25 m	140	99.23%	0.00%
us	25 m to 30 m	50	59.78%	0.00%
us2	0 m to 5 m	21	100.00%	0.00%
us2	5 m to 10 m	23	100.00%	0.00%
us2	10 m to 15 m	28	100.00%	0.00%
us2	15 m to 20 m	27	37.04%	3.70%
us2	20 m to 25 m	29	0.00%	3.45%
us2	25 m to 30 m	28	10.71%	0.00%
us3	0 m to 5 m	25	0.00%	40.00%
us3	5 m to 10 m	9	0.00%	0.00%
us3	10 m to 15 m	0	nan%	nan%
us3	15 m to 20 m	0	nan%	nan%
us3	20 m to 25 m	0	nan%	nan%
us3	25 m to 30 m	0	nan%	nan%

Table 8.2: Used parameters for lidar algo

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	m
RANSAC		
max_iter	100	m
distance_threshold	0.11 m	m
normal_distance_w	0.01	m
rd		
angle	3° to 9°	m
width	2 m to 6 m	m
<i>o</i>	4	m