Traffic Scene Perception for Automated Driving with Top-View Grid Maps

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)
angenommene

DISSERTATION

von

Sascha Wirges, M.Sc.

Tag der mündlichen Prüfung: 26.03.2021

Hauptreferent: Prof. Dr.-Ing. Christoph Stiller
Korreferent: Prof. Dr.-Ing. Michael Heizmann

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Mess- und Regelungstechnik (MRT) des Karlsruher Instituts für Technologie (KIT). Sie wäre nicht möglich gewesen ohne die vielfältige Unterstützung meiner Kollegen, Familie und Freunde.

Mein erster Dank gilt Herrn Prof. Dr.-Ing. Christoph Stiller, der für perfekte Rahmenbedingungen zur freien Forschung und damit zur Promotion (auch über die Dauer von $4 + x, x < 1$ Jahren hinaus) gesorgt hat. Weiterhin möchte ich mich bei Prof. Dr.-Ing. Michael Heizmann für die Übernahme des Korreferats und das damit verbundene Interesse an meiner Arbeit bedanken.

Das MRT lebt von seinen Mitarbeitern. Ohne die vielen fachlichen Diskussionen, Sommerseminare, Konferenzbesuche, Skiausflüge, Sport- und Pubquizabende, Social Tuesdays und sonstigen Kneipenaufenthalte hätte ich vermutlich nicht so lange durchgehalten! Besonders hervorheben möchte ich dabei meine werten (Ex-)kollegen Johannes B., Philipp, Claudio, Frank, Christoph, Johannes G., Andi, Hao, Hendrik, Max, Pio, Fabi, Eike, Sven, Marc und Flo, die meine Zeit am MRT in vielerlei Hinsicht bereichert haben. Auch dem Sekretariat, das in besonders unkomplizierter und sympathischer Weise stets unangenehme Aufgaben von den Doktoranden fern hält, möchte ich herzlich danken. Mein Dank gilt auch den Werkstätten für ihre besonders schnelle und gute Arbeit, aus denen schöne und sichere Versuchsaufbauten entstanden sind. Nicht zuletzt danke ich Werner, der immer für frischen Speicherplatz und Nachschub an Studentenaccounts gesorgt hat.

Während meiner Zeit am MRT habe ich einige Studenten betreut, deren Arbeiten wichtige Impulse für meine Forschung gegeben haben. Besonders danke ich hierbei Shuxiao und Tom, die mich ebenfalls als Hiwis durch ihre Zuarbeit unterstützt haben.

Abschließend möchte ich mich bei meinen Eltern bedanken, die mir so viel ermöglicht und mich damit zu dem Menschen gemacht haben, der ich heute bin.

Mein größter Dank gilt jedoch meiner Frau Lisa, die mich auch in schwierigen Zeiten stets ertragen, motiviert und uns dabei noch den kleinen Fritz geschenkt hat! – Ohne dich wäre ich nicht hier.

Karlsruhe im Dezember 2020, *Sascha Wirges*

Kurzfassung

Ein automatisiertes Fahrzeug muss sichere, sinnvolle und schnelle Entscheidungen auf Basis seiner Umgebung treffen. Dies benötigt ein genaues und recheneffizientes Modell der Verkehrsumgebung. Mit diesem Umfeldmodell sollen Messungen verschiedener Sensoren fusioniert, gefiltert und nachfolgenden Teilsysteme als kompakte, aber aussagekräftige Information bereitgestellt werden.

Diese Arbeit befasst sich mit der Modellierung der Verkehrsszene auf Basis von Top-View Grid Maps. Im Vergleich zu anderen Umfeldmodellen ermöglichen sie eine frühe Fusion von Distanzmessungen aus verschiedenen Quellen mit geringem Rechenaufwand sowie eine explizite Modellierung von Freiraum.

Nach der Vorstellung eines Verfahrens zur Bodenoberflächenschätzung, das die Grundlage der Top-View Modellierung darstellt, werden Methoden zur Belegungs- und Elevationskartierung für Grid Maps auf Basis von mehreren, verrauschten, teilweise widersprüchlichen oder fehlenden Distanzmessungen behandelt. Auf der resultierenden, sensorunabhängigen Repräsentation werden anschließend Modelle zur Detektion von Verkehrsteilnehmern sowie zur Schätzung von Szenenfluss, Odometrie und Tracking-Merkmalen untersucht.

Untersuchungen auf öffentlich verfügbaren Datensätzen und einem Realfahrzeug zeigen, dass Top-View Grid Maps durch on-board LiDAR Sensorik geschätzt und verlässlich sicherheitskritische Umgebungsinformationen wie Beobachtbarkeit und Befahrbarkeit abgeleitet werden können. Schließlich werden Verkehrsteilnehmer als orientierte Bounding Boxen mit semantischen Klassen, Geschwindigkeiten und Tracking-Merkmalen aus einem gemeinsamen Modell zur Objektdetektion und Flussschätzung auf Basis der Top-View Grid Maps bestimmt.

Abstract

An automated driving system needs to make safe, reasonable and quick decisions based on its surroundings. This requires an accurate and computationally inexpensive model of the traffic environment. Using this model, measurements of multiple sensors are fused, filtered and forwarded to subsequent systems in a condensed but meaningful way.

This work concludes research on top-view grid maps as one way to model the traffic scene. Compared to other environment models, top-view grid maps enable early-stage range sensor data fusion at low computational cost and an explicit modeling of free-space.

After we revisited ground surface estimation from range sensors as a prerequisite for top-view modeling, we present methods for occupancy and elevation grid mapping from multiple noisy, possibly contradicting or missing range sensor measurements. Given the resulting sensor-agnostic representation, we then investigate models for traffic participant detection as well as scene flow, odometry and tracking feature estimation.

Experiments on publicly available data sets and our experimental vehicle show that we are able to reliably estimate safety-critical information such as the observability and drivability from top-view grid maps estimated from on-board LiDAR sensors. In addition, we are able to estimate traffic participants as oriented bounding boxes with semantic classes, velocities and tracking features from a joint grid map-based object detection and scene flow estimation.

Table Of Contents

Abbreviations and Notations

Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
AD	Automated Driving
ADAM	Adaptive Moment Estimation
ANN	Artificial Neural Network
AOE	Average Orientation Error
AORE	Average Odometry Rotation Error
AOTE	Average Odometry Translation Error
AP	Average Precision
AR	Average Recall
ASE	Average Scale Error
ATE	Average Translation Error
BBA	Basic Belief Assignment
BCE	Binary Cross Entropy
BiFPN	Bidirectional Feature Pyramid Network
BN	Batch Normalization
CCL	Connected-Components Labeling

CE	Cross Entropy
CNN	Convolutional Neural Network
DBSCAN	Density-Based Spatial Clustering Of Applications With Noise
DL	Deep Learning
FFNN	Feedforward Neural Network
FOV	Field Of View
FPN	Feature Pyramid Network
GMC	Geman McClure
GNC	Graduated Non-Convexity
GPU	Graphics Processing Unit
IoU	Intersection Over Union
KLD	Kullback-Leibler Divergence
LiDAR	Light Detection And Ranging
LLS	Linear Least-Squares
LS	Least-Squares
mAOE	Mean Average Orientation Error
mAORE	Mean Average Odometry Rotation Error
mAOTE	Mean Average Odometry Translation Error
mAP	Mean Average Precision
mASE	Mean Average Scale Error
mATE	Mean Average Translation Error
mAVE	Mean Average Velocity Error
ML	Machine Learning
NMS	Non-Maximum Suppression

OLS	Ordinary Least-Squares
PDF	Probability Density Function
RADAR	Radio Detection And Ranging
ReLU	Rectified Linear Unit
RFS	Random Finite Set
RPN	Region Proposal Network
SGD	Stochastic Gradient Descent
TLS	Truncated Least-Squares
UBS	Uniform B-Spline
WLS	Weighted Least-Squares

Notations

v	Scalar variable
$\boldsymbol{v}$	Vector variable
$\boldsymbol{V}$	Matrix or tensor variable
f	Scalar-valued function
$\mathbf{f}, \mathbf{F}$	Vector-, matrix- or tensor-valued function

1 Introduction

Automated driving (AD) has the potential to reduce traffic accidents, congestion and CO_2 emissions while increasing transport capacity, personal travel comfort and availability of transportation [Gol18]. Whereas the development of driver assistance systems 40 years ago aimed at supporting the driver in dangerous situations (SAE-L1), recent research is focused on automating all driving tasks and thus entirely replacing the driver (SAE-L5) [SAE18].

As illustrated in Fig. 1.1, an AD system can be divided into several interacting components. Driving decisions and actions are made and conducted by behavior generation and motion planning modules based on the surrounding traffic and the vehicle state, which is estimated in a separate module. Input for many components is an environment model which contains the state (e.g. position, orientation, velocity, shape, semantics) of other traffic participants and regulatory elements (e.g. traffic lights, signs) in the vehicle's surroundings. An efficient and accurate environment representation (and its interpretation) is one of the key challenges in AD.

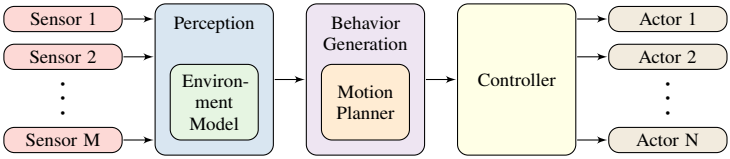


Figure 1.1: (Simplified) information flow in an automated vehicle. The perception module uses sensory inputs to estimate its environment model. This information is passed to behavior generation, where it is used by the motion planner to compute a desired trajectory. From this trajectory, the controller then generates actor commands.

Relevant traffic information needs to be detected, filtered and condensed to provide a more meaningful, abstract representation to subsequent modules such as behavior generation or motion planning. To achieve robustness w.r.t. sensor errors (e.g. noise, false/missed detections, outage), environmental conditions

(e.g. weather, daytime, but also adversarial attacks), different sensor modalities and technologies (e.g. cameras, ultrasound, LiDAR, RADAR) are used to maintain diversity and redundancy. These sensor measurements may originate from the same or different sensors at different times and may be contradicting. Due to the restricted field of view (FOV) and mounting position of sensors and because of occlusions, we aim to know if space can be considered free, occupied (e.g. by other traffic participants or obstacles) or uncertain. Additionally, all sensory information needs to be processed at minimum delay subject to memory and computational constraints on the mobile platform. Therefore, our goal is to develop an environment model meeting these requirements and to efficiently estimate the state of such.

1.1 Environment Models

In the following we introduce selected environment models from a low to a high level of abstraction. As these models only denote a subset of models presented in literature, we refer to Schreier [Sch18] for a more detailed overview.

1.1.1 Range Images & Point Sets

Many range sensors contain a structured sensing array or record measurements in a structured fashion (e.g. spinning range sensors). This means that sensor readings can be laid out on a 2D grid, yielding a compact representation called *range image*. Range images have the advantage that operations working on structured data such as 2D convolutions can be applied. However, note that the neighborhood relation between two adjacent range image readings is different to a neighborhood relation which considers point distances in 3D Cartesian space. A disadvantage of range images is that the grid structure cannot be retained during sensor data fusion, i.e. when multiple range images should be combined.

For this reason, *point sets* (also called point clouds) are often used as sensor-agnostic data representation. However, point sets do not model the neighborhood relation between points. While data from multiple sensors can be combined easily by the union of point sets, the number of points grows with time and a

reduction strategy needs to be applied. In the perception context, it is not possible to distinguish between observable space, i.e. the free-space between sensor origin and reflections and non-observable space (e.g. outside the sensors' FOV). Furthermore, it is hard to detect and combine contradicting measurements (e.g. noisy or false measurements). Finally, operations working on structured data such as convolutions cannot be applied and it is computationally expensive to establish any neighborhood relations, e.g. by building kd-trees [ML14].

1.1.2 Volume Grids

Regular volume grids tesselate the Euclidean space by volumetric elements called voxels[1]. Compared to polygons (Section 1.1.3), the voxel state does not explicitly store its coordinates. Instead, it can be inferred based on its relative position on the grid. Depending on the application, voxels can be stored in dense structures (contiguous memory) or sparse hierarchical structures (e.g. quad-/octrees), which may reduce the storage demands and enable efficient branching methods at the cost of higher hierarchy traversal times. In the context of sensor data fusion, voxel states are usually modeled mutually independent, enabling data fusion on a per-voxel level. If a mapping from coordinates in an arbitrary coordinate frame to voxel states is implied, grids are often called *grid maps*[2].

Top-view grid maps are a specialization of volume grids which map 2D ground coordinates to cell states from a top-down (nadir) view of the scene. Although many graphical projections can be used to reduce the dimensionality from 3D to 2D, orthographic projections are the most commonly used. The dimensionality reduction implies a storage reduction and a reduction of computational complexity for many algorithms, e.g. convolutions. A popular storage layout are regular grids where the cell state dimensions denote the outer storage dimension, called *multi-layer* grids.

[1] Similar to pixel and texel, a voxel is a portmanteau of **vo**lume and **el**ement.

[2] The nomenclature in literature is sometimes not consistent to this definition, where grid maps denote a 2D and voxel maps a 3D mapping.

1.1.3 Surfaces

Polygon meshes usually consist of vertices, edges and convex faces and can be used to approximate arbitrary 2D/3D surfaces. Here, vertices denote 2D/3D points (possibly with additional information), edges denote the connection between two points and faces denote a closed set of edges, often chosen to be triangles. However, algorithms to estimate polygon meshes such as marching squares/cubes often work on volume grids so that these need to be estimated beforehand.

Rectangles and cuboids/boxes are a convex specialization of polygon meshes with rectangular faces for which efficient algorithms exist, such as volume or intersection computation (cf. Appendix A.8). Images/3D space can be easily labeled with boxes by human annotators, which makes them a popular object shape representation (e.g. pedestrians, cyclists or cars) in many publicly available data sets (cf. [GLU12; Cae+20]).

Another surface representation are splines (cf. Section 2.3), which are often used to model smooth surfaces. However, as splines impose smoothness, closed surfaces with sharp edges usually need to be constructed using multiple spline surfaces.

1.2 Goals & Outline of this Work

We aim to develop an environment model that represents the traffic scene by observable and drivable space as well as different traffic participants in a way that is meaningful for subsequent AD modules.

In AD, multiple noisy and possibly contradicting on-board sensor measurements need to be combined with low latency and subject to the bounded memory and computational power of a mobile platform. Additionally, the description of occluded and drivable areas is a prerequisite for safe driving. As in road scenarios all traffic participants move on a common ground surface, we may store relevant information in a 2D structure relative to it. Thus, we believe that top-view grid maps (cf. Section 1.1.2) are a suitable way of modeling a traffic scenario as they leverage simple sensor data fusion and explicit cell state modeling (e.g. for occupied, free and unknown space or elevation).

However, subsequent AD modules such as motion planning or behavior generation may require a more condensed representation such as discrete objects. Here, we detect, classify and estimate the shape of relevant traffic participants from top-view grid maps and represent them by oriented cuboids placed on the ground surface. For tracking applications, the motion and reidentification of traffic participants may be required as well. Therefore, we also estimate object velocities with the vehicle odometry as by-product and a shape descriptor for each traffic participant which can be used to identify the same object across different frames.

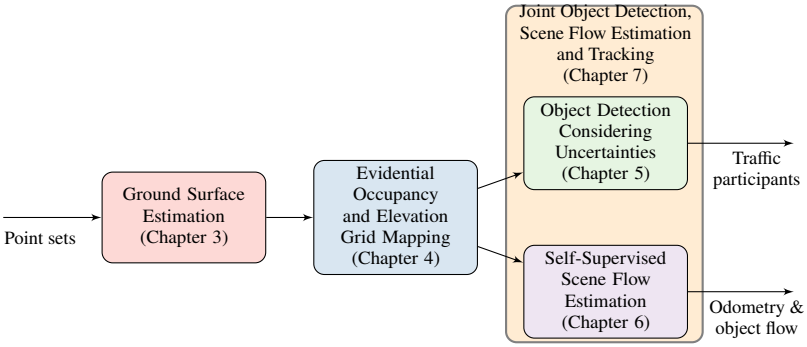


Figure 1.2: We first estimate the ground surface based on multiple range sensor measurements represented as point sets before estimating occupancy and elevation grid maps. These grid maps are input for the subsequent object detection and flow estimation. We also suggest a joint approach including the estimation of shape descriptors for tracking applications.

Figure 1.2 illustrates the dependencies between different processing stages presented in this thesis. In Section 1.1.2 we introduced top-view grid maps as an environment model which is well-suited for sensor data fusion, mapping and machine learning (ML) applications.

One of the assumptions for using top-view grid maps is that the ground surface is known a priori. Therefore, we model the ground surface as a uniform B-spline and propose a robust estimation approach from available range measurements represented as point sets in Chapter 3. We are able to estimate the ground surface with low computational cost and in real-time on our experimental

vehicle platform and find a high accuracy by evaluating our approach on the SemanticKITTI data set [Beh+19].

To perform sensor data fusion, we collect evidence from different range sensor measurements, combine this information into an evidential occupancy grid representation using available height information and propose a method for mapping static environment with known poses (Chapter 4). Compared to related work, we focus on accurately describing the evidential mapping process and taking height information into account which only becomes available by the prior ground surface estimation. We validate our grid mapping approach on our experimental vehicle.

Based on the resulting evidential layers we then introduce a deep model to detect, classify and estimate the shape of relevant traffic participants assuming oriented bounding boxes in Chapter 5. Leveraging self-supervision, we then develop a model to estimate the scene flow in grid maps, which yields ego motion and the motion of other traffic participants (Chapter 6). Subsequently, we combine the object detection and flow estimation models into a joint model with an additional tracking feature estimation in Chapter 7 and show that object detection performance can be improved. In contrast to many other works, we apply our models directly on the grid maps instead of the original point sets which makes our approach more modular and extensible, regarding not only multiple range sensors but also different sensor modalities.

Finally, we conclude our work and point to future work in the research field (Chapter 8).

2 Fundamentals

Here, we briefly present the fundamental concepts used throughout this thesis. In Chapter 4, we model and fuse sensor measurements by means of evidence theory, presented in Section 2.1. The parameter optimization techniques introduced in Section 2.2 are used for ground surface estimation (Chapter 3), learning deep models from data (Chapters 5 to 7) and odometry estimation in Chapter 6. In Section 2.3, we define uniform B-splines, which we use to model the ground surface in Chapter 3. Finally, we provide an overview on machine learning and deep learning techniques, that are the foundation for the approaches presented in Chapters 5 to 7.

2.1 Evidence Theory

In evidence or Dempster-Shafer theory (DST), a generalization of Bayesian theory, evidence from different sources is not only assigned to single hypotheses under consideration but to all possible combinations. It is well-suited for sensor data fusion as different, possibly conflicting sources can be combined using combination rules.

Let Ω be the frame of discernment, i.e. the set of hypotheses under consideration. The basic belief assignment (BBA)

$$\mathrm{m}\colon 2^{\Omega} \to [0,1], \quad \mathrm{m}(\emptyset) = 0, \quad \sum_{X \in 2^{\Omega}} \mathrm{m}(X) = 1 \tag{2.1}$$

assigns evidence mass to each element of its power set 2^{Ω}. The elements in 2^{Ω} that yield a non-zero value are called focal elements.

Two sources of evidence m_1 and m_2 can be combined using different combination rules. Dempster's rule of combination

$$(m_1 \oplus m_2)(X) = \frac{1}{1-C} \sum_{\Psi_1 \cap \Psi_2 = X} m_1(\Psi_1) m_2(\Psi_2), \tag{2.2}$$

$$C = \sum_{\Psi_1 \cap \Psi_2 = \emptyset} m_1(\Psi_1) m_2(\Psi_2), \tag{2.3}$$

normalizes the combined result by a measure C of conflicting sources. Evidence about a proposition X lies within the interval $[\mathrm{bel}(X), \mathrm{pl}(X)]$, where

$$\mathrm{bel}(X) = \sum_{\Psi \subseteq X} m(\Psi) \tag{2.4}$$

$$\mathrm{pl}(X) = \sum_{\Psi \cap X \notin \emptyset} m(\Psi) \tag{2.5}$$

denote the summed evidence that other propositions Ψ strictly support X (belief) or do not conflict with X (plausibility). The difference

$$\mathrm{U}(X) = \mathrm{pl}(X) - \mathrm{bel}(X) \tag{2.6}$$

can be interpreted as the uncertainty for this proposition. If a decision between the hypotheses in Ω has to be made, the pignistic transform [DSD04]

$$\mathrm{prob}(\omega) = \sum_{\omega \in \Psi} \frac{m(\Psi)}{|\Psi|}, \quad \mathrm{bel}(\omega) \leq \mathrm{prob}(\omega) \leq \mathrm{pl}(\omega) \tag{2.7}$$

distributes evidence mass of propositions Ψ equally between the hypotheses $\omega \in \Omega$.

Example 2.1: Crime Case

Appendix A.1 shows how to calculate evidential measures based on a crime case example.

2.1.1 Comparison to Bayesian Theory

A Bayesian belief function

$$\mathrm{bel}\colon 2^{\Omega} \to [0,1], \quad \mathrm{bel}(\emptyset) = 0, \quad \mathrm{bel}(\Omega) = 1 \tag{2.8}$$

can be defined similar to belief functions in evidence theory. However, its property

$$\mathrm{bel}(X \cup \Psi) = \mathrm{bel}(X) + \mathrm{bel}(\Psi) \quad \text{if } X \cap \Psi = \emptyset \tag{2.9}$$

is relaxed in evidence theory as it is also possible to assign evidence to super sets (e.g. $X \cup \Psi$) directly. If all focal elements of the BBA are elements of Ω, then

$$\mathrm{bel}(X) + \mathrm{bel}(\overline{X}) = \mathrm{bel}(X) + \mathrm{bel}(\Omega \setminus X) = 1 \tag{2.10}$$

holds and yields a Bayesian belief assignment as special case of evidence theory.

2.2 Continuous Parameter Optimization

In continuous parameter optimization, we aim to find the N_p *optimal* parameters

$$\boldsymbol{p}^* = \arg\min_{\boldsymbol{p}} \mathrm{c}(\boldsymbol{p}), \quad \boldsymbol{p} \in \mathbb{R}^{N_\mathrm{P}}, \tag{2.11}$$

that minimize a cost function $\mathrm{c}\colon \mathbb{R}^{N_\mathrm{P}} \to \mathbb{R}$ on a continuous parameter and value space.

2.2.1 Least-Squares Optimization

If the cost function

$$\mathrm{c}(\boldsymbol{p}) = \mathbf{r}^\top(\boldsymbol{p})\, \boldsymbol{W}\, \mathbf{r}(\boldsymbol{p}) = \left\| \boldsymbol{W}^{\frac{1}{2}} \mathbf{r}(\boldsymbol{p}) \right\|^2 \tag{2.12}$$

is a linear combination of squared residuals $\mathbf{r}\colon \mathbb{R}^{N_\mathrm{P}} \to \mathbb{R}^{N_\mathrm{R}}$ weighted by a positive semi-definite matrix $\boldsymbol{W} \in \mathbb{R}^{N_\mathrm{R} \times N_\mathrm{R}}$, it is called Least-Squares (LS) problem. If $\boldsymbol{W}$ is a diagonal matrix, the problem is called weighted Least-

Squares (WLS) and ordinary Least-Squares (OLS) if $\boldsymbol{W}$ is the identity matrix. Note that in the context of LS optimization it is often referred to the cost function

$$\mathrm{c}(\boldsymbol{p}) = \sum_{k=1}^{N_\mathrm{B}} w_k \|\mathbf{r}_k(\boldsymbol{p})\|^2 \,, \quad w_k \geq 0 \tag{2.13}$$

with $N_\mathrm{B} \leq N_\mathrm{R}$ individual residual block functions $\mathbf{r}_k : \mathbb{R}^{N_\mathrm{P}} \to \mathbb{R}^{N_k}$ and non-negative weights w_k. However, it is a specialization of Eq. (2.12) for the WLS case.

Additionally, the LS problem is called linear Least-Squares (LLS) problem if the residual function

$$\mathbf{r}(\boldsymbol{p}) = \underbrace{\boldsymbol{H}\boldsymbol{p}}_{\hat{\boldsymbol{y}}} - \boldsymbol{y}, \quad \boldsymbol{H} \in \mathbb{R}^{N_\mathrm{R} \times N_\mathrm{P}}, \boldsymbol{y} \in \mathbb{R}^{N_\mathrm{R}} \tag{2.14}$$

is affine so that the cost function becomes convex. This means that if a strict local minimum

$$\left.\frac{\partial \mathrm{c}}{\partial \boldsymbol{p}}\right|_{\boldsymbol{p}=\boldsymbol{p}^*} = \mathbf{0}, \quad \left.\frac{\partial^2 \mathrm{c}}{\partial \boldsymbol{p}^2}\right|_{\boldsymbol{p}=\boldsymbol{p}^*} \text{ is positive-definite} \tag{2.15}$$

exists, it is also a global minimum of c. In this case, a closed-form solution

$$\boldsymbol{p}^* = \left(\boldsymbol{H}^\top \boldsymbol{W} \boldsymbol{H}\right)^{-1} \boldsymbol{H}^\top \boldsymbol{W} \boldsymbol{y} \tag{2.16}$$

exists.

2.2.2 Gradient Descent Methods

In general, cost functions are non-convex with no guarantee for a local minimum (such as Eq. (2.15)) being the global minimum. This means that usually no closed-form solutions to compute the optimum exist. However, iterative optimization methods may be used that determine updated parameters

$$\boldsymbol{p}_{k+1} = \mathbf{u}\left(\boldsymbol{p}_k\right) \tag{2.17}$$

at step k using an update function $\mathbf{u}$, that reduces the cost such that

$$\mathrm{c}(\boldsymbol{p}_{k+1}) < \mathrm{c}(\boldsymbol{p}_k). \tag{2.18}$$

A popular family of optimization algorithms (e.g. often used in machine learning applications) are gradient descent methods. Here, the update function

$$\mathbf{u}(\boldsymbol{p}_k) = \boldsymbol{p}_k + \lambda \mathbf{d}(\boldsymbol{g}_k), \quad \boldsymbol{g}_k = \left.\frac{\partial \mathrm{c}}{\partial \boldsymbol{p}}\right|_{\boldsymbol{p}=\boldsymbol{p}_k} \tag{2.19}$$

evaluates the cost gradient $\boldsymbol{g}_k$ given the current parameters $\boldsymbol{p}_k$ and updates the parameters with learning rate λ in a direction determined by the direction function $\mathbf{d}$.

Example 2.2: Steepest Descent

In the simplest case, the direction

$$\mathbf{d}_{\mathrm{SD}}(\boldsymbol{g}_k) = -\boldsymbol{g}_k \tag{2.20}$$

points to the steepest descent of the cost surface.

In many cases, the number of residuals or computations during optimization (e.g. gradients) is too large to fit into memory so that the optimization needs to be split into smaller fractions called *batches*. Then, a smaller number of samples is repeatedly drawn randomly from the data set as an approximation of the whole data set and used in optimization. Gradient descent variants that approximate the gradient based on a subset of data are called stochastic gradient descent (SGD) methods.

Example 2.3: Adaptive Moment Estimation

A popular SGD method with the descent direction

$$\mathbf{d}_{\text{ADAM}}(\boldsymbol{g}_k) = -\frac{\hat{\boldsymbol{\mu}}_k(\boldsymbol{g}_k)}{\sqrt{\hat{\boldsymbol{\sigma}}_k^2(\boldsymbol{g}_k)} + \epsilon}\,, \quad \epsilon > 0 \tag{2.21}$$

is called Adaptive Moment Estimation (ADAM) [KB14]. During optimization, the first and second gradient moments

$$\boldsymbol{\mu}_k(\boldsymbol{g}_k) = \beta_1 \boldsymbol{\mu}_{k-1}(\boldsymbol{g}_{k-1}) + (1-\beta_1)\boldsymbol{g}_k\,, \quad \hat{\boldsymbol{\mu}}_k(\boldsymbol{g}_k) = \frac{\boldsymbol{\mu}_k(\boldsymbol{g}_k)}{1-\beta_1^k} \tag{2.22}$$

$$\boldsymbol{\sigma}_k^2(\boldsymbol{g}_k) = \beta_2 \boldsymbol{\sigma}_{k-1}^2(\boldsymbol{g}_{k-1}) + (1-\beta_2)\boldsymbol{g}_k^2\,, \quad \hat{\boldsymbol{\sigma}}_k^2(\boldsymbol{g}_k) = \frac{\boldsymbol{\sigma}_k^2(\boldsymbol{g}_k)}{1-\beta_2^k} \tag{2.23}$$

are estimated by exponential smoothing controlled by the parameters $0 \leq \beta_1, \beta_2 \leq 1$. Note that all operations on vectors in Eqs. (2.21) to (2.23) are element-wise and a small ϵ is added for numerical stability.

2.2.3 Robust Estimation

In presence of outliers, OLS optimization problems such as

$$\min_{\boldsymbol{p}} \sum_{k=1}^{N_{\text{R}}} \|\mathbf{r}_k(\boldsymbol{p})\|^2 \tag{2.24}$$

yield suboptimal results as outliers gain large values and thus highly influence the function minimum. To mitigate this issue, a non-linear robustifier $\rho\colon \mathbb{R} \to \mathbb{R}$ is introduced and the problem reformulated as the non-linear LS problem

$$\min_{\boldsymbol{p}} \sum_{k=1}^{N_{\text{R}}} \rho(\|\mathbf{r}_k(\boldsymbol{p})\|). \tag{2.25}$$

Example 2.4: Truncation and Huber Loss

Popular choices for robustifiers are truncations

$$\rho(x) = \begin{cases} x^2 & \text{if } x \in [-c, c] \\ c^2 & \text{otherwise} \end{cases} \tag{2.26}$$

and specializations of the Generalized Charbonnier loss (cf. Appendix A.2) such as the Huber loss function

$$\mathcal{H}_c(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| \leq c \\ c\left(|x| - \frac{c}{2}\right) & \text{otherwise.} \end{cases} \tag{2.27}$$

In the context of LS optimization, problems with truncation robustifiers as in Eq. (2.26) are also called truncated Least-Squares (TLS) problems.

Using non-linear robustifiers makes the cost function non-convex, which leads to convergence problems if the initial parameters are far from the optimal parameters. Yang et al. [Yan+20] propose an iterative algorithm for robust estimation based on the concept of graduated non-convexity and the Black-Rangarajan duality, summarized in Appendix A.3.

2.3 Uniform B-Splines

We briefly introduce univariate splines, extend them to spline surfaces and highlight properties of uniform B-splines (UBSs). A more detailed overview on different splines is presented by Beck [Bec20].

A spline

$$\mathrm{s}\colon \mathbb{R} \to \mathbb{R}\,, \quad \mathrm{s}(x) = \begin{cases} \mathrm{s}_1(x) & x_1 \leq x < x_2 \\ \mathrm{s}_2(x) & x_2 \leq x < x_3 \\ & \vdots \\ \mathrm{s}_N(x) & x_N \leq x < x_{N+1} \end{cases} \tag{2.28}$$

is defined piece-wise by polynomial segment functions

$$\begin{aligned} \mathrm{s}_i(x) &= \sum_{k=1}^{N_\mathrm{C}} \mathrm{b}_{i,k}(x) p_k = \langle \mathbf{b}_i(x), \boldsymbol{p} \rangle = \langle \boldsymbol{B}_i \mathbf{a}(x), \boldsymbol{p} \rangle \,, \\ \mathbf{a}(x) &= \left[1, x, x^2, \ldots, x^d\right]^\top \in \mathbb{R}^{d+1} \end{aligned} \tag{2.29}$$

of degree d with N_C control points, where $\boldsymbol{B}_i \in \mathbb{R}^{N_\mathrm{C} \times (d+1)}$ denotes the spline basis, $\mathbf{a}(x)$ the polynomial vector and $\boldsymbol{p} \in \mathbb{R}^{N_\mathrm{C}}$ the control point vector.

B-splines are based on B-spline polynomials and can be defined by the Cox-de Bor recursion formula [De 78, p. 90]:

$$\mathrm{b}_{i,j,0}(x) = \begin{cases} 1 & \text{if } k_j \leq x < k_{j+1} \\ 0 & \text{otherwise} \end{cases} \,, \tag{2.30}$$

$$\mathrm{b}_{i,j,k}(x) = \frac{x - k_j}{k_{j+k} - k_k} \mathrm{b}_{i,j,k-1}(x) + \frac{k_{j+k+1} - x}{k_{j+k+1} - k_{j+1}} \mathrm{b}_{i,j+1,k-1}(x) \,, \tag{2.31}$$

$$\mathrm{b}_{i,j}(x) = \mathrm{b}_{i,j,d}(x) \,, \tag{2.32}$$

where $k_1, \ldots, k_{N_\mathrm{C}+d+1}$ denote the monotonically increasing knot values. Equation (2.30) shows that a spline has bounded support as every $b_{i,j,0}$ is only non-zero within two knots. Choosing the knot values $k_j = \frac{j-1-d}{N_\mathrm{C}-d}$ to be uniformly spaced leads to UBSs in which the same basis for each segment can be used and precomputed, yielding computational benefits.

Equations (2.28) and (2.29) define splines in the univariate case. However, we can define segment functions $\mathrm{s}_i \colon \mathbb{R}^2 \to \mathbb{R}$,

$$\mathrm{s}_i(x, y) = \sum_{j=1}^{N_{\mathrm{C}_1}} \sum_{k=1}^{N_{\mathrm{C}_2}} \mathrm{b}_{1,i,j}(x) \mathrm{b}_{2,i,k}(y) p_{j,k} \tag{2.33}$$

to describe spline surfaces with control points $p_{j,k}$ and separable basis functions. This scheme can be extended to splines

$$\mathrm{s}_i(\boldsymbol{x}) = \langle \mathbf{b}_i(\boldsymbol{x}), \boldsymbol{p} \rangle \tag{2.34}$$

of arbitrary dimension along all segments denoted by the inner product of a vectorial basis functions $\mathbf{b}_i \colon \mathbb{R}^{N_\mathrm{I}} \to \mathbb{R}^{N_C}$ which determine N_C control point weights depending on N_I-dimensional inputs. Any vector valued spline $\mathbf{s} \colon \mathbb{R}^{N_\mathrm{I}} \to \mathbb{R}^{N_\mathrm{O}}$ can then be assembled by N_O scalar valued splines.

2.4 Machine & Deep Learning

As a field of artifical intelligence, machine learning (ML) can be defined as "the set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision-making under uncertainty" ([Mur12]).

ML is closely related to the field of statistics. Common tasks are classification, regression, density estimation and clustering. Usually, one builds a parametric *model* which aims to solve the task and uses optimization methods to determine model parameters.

Deep learning (DL), as a subfield of Machine Learning, uses artificial neural networks (ANNs) as models. Here, the adjective *deep* emphasizes that usually ANNs with many interacting parameters are used, e.g. feedforward neural networks (FFNNs) with many layers (see Section 2.4.1).

2.4.1 Artificial Neural Networks

ANNs are inspired by information processing and distributed communication nodes in biological systems. Information processing and flow is modeled by a directed graph where nodes represent operations and edges denote the flow of these computation results.

In the following, we represent features by vectors (rank 1 tensors) for notational simplicity. However, we note that tensors of any rank can always be reshaped into a vector representation by stacking.

The smallest operational unit

$$\mathbf{f} \colon \mathbb{R}^{N_\mathrm{I}} \to \mathbb{R}^{N_\mathrm{O}}, \quad \mathbf{f}(\boldsymbol{x}) = \mathbf{a}(\tilde{\boldsymbol{W}}\tilde{\boldsymbol{x}} + \boldsymbol{b}) = \mathbf{a}(\boldsymbol{W}\boldsymbol{x}) \tag{2.35}$$

of an ANN, often called *layer*, applies a non-linear activation function $\mathbf{a}\colon \mathbb{R}^{N_\mathrm{O}} \to \mathbb{R}^{N_\mathrm{O}}$ to linearly mapped inputs $\boldsymbol{W}\boldsymbol{x}$. The matrix $\tilde{\boldsymbol{W}} \in \mathbb{R}^{N_\mathrm{O} \times N_\mathrm{I}}$ and the vector $\boldsymbol{b} \in \mathbb{R}^{N_\mathrm{O}}$ are called weight matrix and bias vector, respectively. For notational simplicity, we define the augmented feature vector $\boldsymbol{x} = [\tilde{\boldsymbol{x}}\ 1]^\top$ and weight matrix $\boldsymbol{W} = \left[\tilde{\boldsymbol{W}}\ \boldsymbol{b}\right]$ and implicitly include the bias parameters when referring to *weights*. Usually, the same scalar activation function is applied separately to each output dimension. In this case, popular activation functions are ReLU, swish or tanh, summarized in Appendix A.4. Another popular activation function is the softmax function (cf. Eq. (2.53)).

Feedforward Neural Networks FFNNs, also called multi-layer perceptrons, are represented by a directed acyclic graph prohibiting feedbacks. Assuming N_L different layers $\mathbf{f}^{(l)}$, the mapping

$$\mathbf{f}(\boldsymbol{x}, \boldsymbol{w}) = \mathbf{f}^{(N_\mathrm{L})} \circ \mathbf{f}^{(N_\mathrm{L}-1)} \circ \ldots \circ \mathbf{f}^{(2)} \circ \mathbf{f}^{(1)}(\boldsymbol{x}, \boldsymbol{w}) \tag{2.36}$$

$$= \mathbf{f}^{(N_\mathrm{L})}\left(\mathbf{f}^{(N_\mathrm{L}-1)}\left(\ldots \mathbf{f}^{(2)}\left(\mathbf{f}^{(1)}(\boldsymbol{x}, \boldsymbol{w}), \boldsymbol{w}\right)\ldots\right)\right) \tag{2.37}$$

resembles a composition of these layers.

Convolutional Neural Networks Convolutional neural networks (CNNs) are a specialization of ANNs, which use discrete convolutions[1] with kernels of limited support operating only on a limited amount of input features, yielding a sparse weight matrix $\boldsymbol{W}$ in Eq. (2.35).

In general, if the input is represented by a multidimensional signal $\mathbf{x}\colon \mathbb{Z}^{N_\mathrm{I}} \to \mathbb{R}^{N_\mathrm{O}}$, the discrete N_I-dimensional convolution $\mathbf{x} * \overset{N_\mathrm{I}}{\cdots} * \mathbf{g}\colon \mathbb{Z}^{N_\mathrm{I}} \to \mathbb{R}$ between $\mathbf{x}$ and the kernel function $\mathbf{g}\colon \mathbb{Z}^{N_\mathrm{I}} \to \mathbb{R}^{N_\mathrm{O}}$ may be defined by

$$\left(\mathbf{x} * \overset{N_\mathrm{I}}{\cdots} * \mathbf{g}\right)\left(m_1, \ldots, m_{N_\mathrm{I}}\right) \tag{2.38}$$

$$= \sum_{n_1=-\infty}^{\infty} \ldots \sum_{n_{N_\mathrm{I}}=-\infty}^{\infty} \left\langle \mathbf{x}\left(m_1 - n_1, \ldots, m_{N_\mathrm{I}} - n_{N_\mathrm{I}}\right), \mathbf{g}\left(n_1, \ldots, n_{N_\mathrm{I}}\right)\right\rangle.$$

[1] In practice, discrete convolutions are implemented as discrete cross-correlations.

The convolution kernel support is called *perceptive field* and is usually kept symmetric and small to increase computation efficiency. In addition, as the amount of operations grows exponentially in N_I, in practice mostly 1D, 2D or 3D convolutions are used to reduce computational cost. Vector-valued convolutions $\mathbf{x} * \overset{N_\mathrm{I}}{\cdots} * \mathbf{g} \colon \mathbb{Z}^{N_\mathrm{I}} \to \mathbb{R}^{N_\mathrm{C}}$ may be defined by N_C scalar-valued convolutions as in Eq. (2.38) in each output dimension.

Example 2.5: Discrete 2D 3×3 Convolution

Given the kernel function $\mathbf{g} \colon [-1, 0, 1]^2 \to \mathbb{R}^{N_I}$, the function

$$(\mathbf{x} \ast\ast \mathbf{g})(m_1, m_2) = \sum_{n_1=-1}^{1} \sum_{n_2=-1}^{1} \langle \mathbf{x}(m_1 - n_1, m_2 - n_2), \mathbf{g}(n_1, n_2) \rangle \tag{2.39}$$

defines a 2D convolution with a 3×3 kernel and $9N_\mathrm{I}$ weights in total.

Separable convolution kernels

$$\mathbf{g}\big(n_1, \ldots, n_{N_\mathrm{I}}\big) = \mathbf{g}_1(n_1) \odot \ldots \odot \mathbf{g}_{N_\mathrm{I}}\big(n_{N_\mathrm{I}}\big) \tag{2.40}$$

$$\mathrm{g}_k\big(n_1, \ldots, n_{N_\mathrm{I}}\big) = \mathrm{g}_k(n_1) \cdot \ldots \cdot \mathrm{g}_k\big(n_{N_\mathrm{I}}\big) \tag{2.41}$$

can be factorized along each dimension k which reduces the number of parameters and further enables parallel evaluation of the convolution. Here, $\odot$ denotes the Hadamard product.

Example 2.6: Discrete Separable 2D Convolution

Given the separable kernel function

$$\mathbf{g} \colon [-1, 0, 1]^2 \to \mathbb{R}^{N_I}, \quad \mathbf{g}(m_1, m_2) = \mathbf{g}_1(m_1) \odot \mathbf{g}_2(m_2), \tag{2.42}$$

the function

$$\begin{aligned}(\mathbf{x} \ast\ast \mathbf{g})(m_1, m_2) &= \big(\mathbf{x} * \mathbf{g}_1\big)(m_1) \cdot \big(\mathbf{x} * \mathbf{g}_2\big)(m_2) \\ &= \sum_{n=-1}^{1} \langle \mathbf{x}(m_1 - n, m_2), \mathbf{g}_1(n) \rangle \cdot \sum_{n=-1}^{1} \langle \mathbf{x}(m_1, m_2 - n), \mathbf{g}_2(n) \rangle\end{aligned} \tag{2.43}$$

defines a separable 2D convolution with three-element kernels and $6N_\mathrm{I}$ weights in total.

A discrete convolution with kernel size $2k + 1$ within the valid range reduces the spatial output size by $2k$. To keep the output size constant, *padding* is

performed to extend the spatial input size on its borders. Often, zero padding is applied where the spatial input size is increased by adding zeros around the border.

Learning and Training In the DL community, *supervised learning* is used as a synonym for parameter optimization of ANNs in order to minimize a cost function c()

$$\mathrm{c}(\boldsymbol{w}) = \mathrm{l}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \mathrm{l}(\mathbf{f}(\boldsymbol{x}, \boldsymbol{w}), \boldsymbol{y}) \tag{2.44}$$

also called *loss*, given the predictions $\hat{\boldsymbol{y}} = \mathbf{f}(\boldsymbol{x}, \boldsymbol{w})$ of a model defined by the weights $\boldsymbol{w}$ and examples $(\boldsymbol{x}, \boldsymbol{y})$ of inputs and (ground truth) labels, respectively. With *training*, we describe the implementation of the *supervised learning* strategy.

Usually, SGD methods (cf. Section 2.2.2) are used to minimize Eq. (2.44). For instance, applying the total-derivative chain rule on an FFNN (cf. Eq. (2.36)), the gradients

$$\begin{aligned}
\left.\frac{\partial \mathrm{l}(\mathbf{f}(\boldsymbol{x}, \boldsymbol{w}), \boldsymbol{y})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}_0} &= \left.\frac{\partial \mathrm{l}(\tilde{\boldsymbol{x}}, \boldsymbol{y})}{\partial \tilde{\boldsymbol{x}}}\right|_{\tilde{\boldsymbol{x}}=\mathbf{f}(\boldsymbol{x}, \boldsymbol{w}_0)} \left.\frac{\partial \mathbf{f}^{(N_L)}(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}_0} \\
\left.\frac{\partial \mathbf{f}^{(N_L)}(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}_0} &= \left.\frac{\partial \mathbf{f}^{(N_L)}(\tilde{\boldsymbol{x}}, \boldsymbol{w}_0)}{\partial \tilde{\boldsymbol{x}}}\right|_{\tilde{\boldsymbol{x}}=\mathbf{f}^{(N_L-1)}(\boldsymbol{x}, \boldsymbol{w}_0)} \left.\frac{\partial \mathbf{f}^{(N_L-1)}(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}_0} \\
\left.\frac{\partial \mathbf{f}^{(N_L-1)}(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}_0} &= \dots
\end{aligned} \tag{2.45}$$

w.r.t. the weights $\boldsymbol{w}$ can be evaluated at the weights $\boldsymbol{w}_0$ for any input $\boldsymbol{x}$. An algorithm to determine the output gradient w.r.t. the model weights is called *Backpropagation* as it iterates from function outputs to inputs to obtain gradients.

2.4.2 Regularization

When learning models from examples, we expect them to *generalize* well to similar but previously unseen examples. However, depending on the number of parameters and learning examples, a model may *overfit* on the training examples, i.e. its error for new examples is much higher than for examples it was trained with. To mitigate this issue, regularization techniques are used which can be

categorized into data-, architecture-, cost function- and optimization-driven approaches [KGC17].

Batch Normalization A popular technique for data-driven regularization is batch normalization (BN). Given a set of B features $\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(B)}\right\}$ called a *batch*, BN transforms features for each dimension k to unbiased features

$$\hat{x}_k^{(b)} = \frac{x_k^{(b)} - \mu_k}{\sigma_k + \epsilon}, \quad \epsilon > 0 \tag{2.46}$$

with unit variance by estimating the empirical mean μ_k and standard deviation σ_k across the batch. Here, ϵ denotes a small constant to assure numerical stability. A feature

$$x_k^{(b)} = \gamma_k \hat{x}_k + \beta_k \tag{2.47}$$

in dimension k can be recovered by the parameters γ_k and β_k, which are learned from all training examples. Among other aspects, BN improves generalization by a smoother loss surface and yields better training stability and convergence due to smaller gradients.

Weight Regularization Weight regularization is a cost function-driven regularization technique where an additional weight-dependent loss $L_{\boldsymbol{w}}$ is added to the overall cost function.

Example 2.7: L2 Weight Decay

The loss

$$L_{\boldsymbol{w}} = \lambda \|\boldsymbol{w}\|^2 \tag{2.48}$$

penalizes large weights and is scaled by the constant hyperparameter λ.

2.4.3 Classification

A popular application in ML is the classification of samples into one of C classes. Given model inputs $\boldsymbol{x}_n$ with corresponding categorical labels $c_n \in C$ we aim to develop a model that approximates the true probability distribution $\mathrm{p}(c|\boldsymbol{x}_n)$ for every sample.

A common distance measure between two discrete distributions p and $\hat{\mathrm{p}}$ is the Kullback-Leibler divergence (KLD)

$$\mathrm{KLD}\{\mathrm{p}, \hat{\mathrm{p}}\} = \mathrm{CE}\{\mathrm{p}, \hat{\mathrm{p}}\} - \mathrm{E}\{\mathrm{p}\} \tag{2.49}$$

as the difference between the cross entropy (CE)

$$\mathrm{CE}\{\mathrm{p}, \hat{\mathrm{p}}\} = -\sum_{c \in C} \mathrm{p}(c) \log(\hat{\mathrm{p}}(c)) \tag{2.50}$$

and the entropy $\mathrm{E}\{\mathrm{p}\}$. KLD and CE both define asymmetric distance measures with global minima at $\mathrm{p} = \hat{\mathrm{p}}$. Minimizing the KLD with respect to $\hat{\mathrm{p}}$ is equivalent to minimizing the CE because $\mathrm{E}\{\mathrm{p}\}$ is independent of $\hat{\mathrm{p}}$.

Example 2.8: Binary Cross Entropy

If $|C| = 2$, Eq. (2.50) can be simplified to

$$\mathrm{CE}_2\{\mathrm{p}, \hat{\mathrm{p}}\} = -\mathrm{p}(c) \log(\hat{\mathrm{p}}(c)) - (1 - \mathrm{p}(c)) \log(1 - \hat{\mathrm{p}}(c)) \,, \tag{2.51}$$

which is called binary cross entropy (BCE).

The CE is often used as objective to estimate classification model parameters $\boldsymbol{p}$. Equation (2.50) can be efficiently evaluated because only $\mathrm{p}(c = c_n|\boldsymbol{x}_n) = 1$ and zero otherwise (one-hot encoding). For instance, given N samples one may define the classification loss

$$L_{\mathrm{cls}} = -\sum_{n=1}^{N} \log(\hat{\mathrm{p}}(c_n|\boldsymbol{x}_n)) \tag{2.52}$$

that evaluates the estimated likelihood only at the ground truth classes c_n.

Given an input vector $\boldsymbol{x}$, a popular distribution function is the softmax function

$$\hat{\mathrm{p}}(c_n|\boldsymbol{x}_n) = \sigma(\boldsymbol{x})_n = \frac{\exp(x_n)}{\sum_{c=1}^{|C|} \exp(x_c)} \,, \tag{2.53}$$

which is non-negative and fulfills $\sum_{c \in C} \hat{\mathrm{p}}(c|\boldsymbol{x}) = 1$ for every n so that it can be interpreted as a probability mass function.

3 Ground Surface Estimation

Top-view grid maps provide an orthographic view of the traffic scene along the ground surface. In the process of mapping LiDAR range measurements (cf. Chapter 4), we aim to distinguish between ground- and non-ground reflections, map other features such as the object height correctly and provide information on drivable areas. To accomplish this, we require geometric information on the ground surface.

The ground surface can be obtained from a map, estimated from range sensor measurements or combined from multiple sources. Using map information has the disadvantage that an accurate pose estimate (incl. roll and pitch) needs to be available which cannot be guaranteed at all times. For this reason, we follow the approach of estimating the ground surface from range sensor measurements to be independent to other sources of errors (e.g. from pose estimation).

In the following, we model the ground surface $g\colon \mathbb{R}^2 \to \mathbb{R}$ as a mapping from plane coordinates to distances from that plane.

Section 3.1 provides an overview on commonly used ground surface models and estimation techniques. We will then introduce a uniform B-spline model in Section 3.2 and present our algorithm to estimate ground surfaces in Section 3.3. In Section 3.4, we compare our model to other surface representations and present ablation studies on the optimization approach.

3.1 Related Work

Instead of estimating the ground surface shape, Moosmann et al. [MPS09] segment each point in range images using a local convexity criterion. The method provides accurate point classification results but it is only possible to apply this method to single range measurements in an image structure. Thus, it

is hard to resolve point classification conflicts when multiple measurements are available.

Zhang et al. [Zha+03] develop a progressive morphological filter to estimate the ground surface in airborne LiDAR measurements represented on an elevation grid. By gradually increasing the filter window size and using elevation thresholds, the authors remove non-ground measurements while preserving the ground surface elevation. Their method works well on accurate elevation maps with large ground areas and local elevations such as buildings or trees. However, Zhang et al. [Zha+03] do not consider measurement errors such as multi-path propagation which would lead to false elevation estimates. In addition, the approach is not real-time capable as the filtering has to be applied several times.

A popular representation for ground surfaces are polynomials

$$\mathrm{h}(\boldsymbol{x}) = \mathrm{h}(x_1, x_2) = \left\langle \begin{bmatrix} \tilde{w}_0 \\ \tilde{w}_1 \\ \tilde{w}_2 \\ \tilde{w}_3 \\ \tilde{w}_4 \\ \vdots \\ \tilde{w}_{D_1 D_2} \end{bmatrix}, \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ \vdots \\ x_1^{D_1} x_2^{D_2} \end{bmatrix} \right\rangle = \langle \tilde{\boldsymbol{w}}, \tilde{\boldsymbol{x}} \rangle \tag{3.1}$$

such as planes ($D = 1$), quadratics ($D = 2$) or cubics ($D = 3$) which can be expressed as a linear combination of weights $\tilde{\boldsymbol{w}}$ and a transformed input $\tilde{\boldsymbol{x}}$. This yields a system of linear equations which can be solved efficiently, e.g. by using a linear Least-Squares (LLS) method (cf. Section 2.2.1). For example, Saleem et al. [SRK17] use a polynomial representation to fit a ground surface on v-disparity estimates of stereo cameras.

Wedel et al. [Wed+09] model the ground surface along the driving direction by a univariate B-spline (cf. Section 2.3). They estimate and track its parameters in an LLS approach combined with a Kalman filter.

Beck [Bec20] uses uniform B-spline (UBS) surfaces with smoothness regularization to model viewing rays depending on camera image coordinates. He describes smoothness of the n-th derivative by the penalty

$$\rho = \int_0^1 \left\| \mathrm{s}^{(n)}(x) \right\|^2 \mathrm{d}x = \| \mathbf{f}_s(\boldsymbol{p}) \|_F^2 \tag{3.2}$$

and shows that its closed-form solution

$$\mathbf{f}_s(\boldsymbol{p}) = \boldsymbol{B}_s\ \boldsymbol{p}, \quad \boldsymbol{B}_s = \boldsymbol{B}_d \left(\int_0^1 \mathbf{a}^{(n)}(x)\ \mathbf{a}^{\top(n)}(x) \mathrm{d}x \right)^{\frac{1}{2}} \tag{3.3}$$

is a matrix-vector product of the control points $\boldsymbol{p}$ and the term $\boldsymbol{B}_s$ which only depends on the spline degree and the smoothness derivative n so that it can be precomputed.

3.2 Problem Formulation

In this work, we use UBSs (cf. Section 2.3) to model the ground surface. Due to their local support, splines are robust towards varying measurement densities as it is often the case for range sensors. However, it is still possible to impose smoothness constraints on splines to reduce overfitting, especially in areas with few measurements.

Given N positions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, we can denote ground surface height estimates in the form

$$\hat{h}_n = \hat{\mathrm{h}}(\boldsymbol{x}_n, \boldsymbol{p}) = \langle \mathbf{b}(\boldsymbol{x}_n), \boldsymbol{p} \rangle\ , \quad \mathbf{b} \colon \mathbb{R}^2 \to \mathbb{R}^{N_C} \tag{3.4}$$

$$\hat{\boldsymbol{h}} = \hat{\mathbf{h}}(\boldsymbol{p}) = \begin{bmatrix} \mathbf{b}^\top(\boldsymbol{x}_1) \\ \mathbf{b}^\top(\boldsymbol{x}_2) \\ \vdots \\ \mathbf{b}^\top(\boldsymbol{x}_N) \end{bmatrix} \boldsymbol{p} = \boldsymbol{B}\ \boldsymbol{p} \tag{3.5}$$

with the spline control points $\boldsymbol{p} \in \mathbb{R}^{N_C}$ using a vectorial basis function $\mathbf{b}$ weighting N_C control points depending on the 2D positions $\boldsymbol{x}_n$.

Then, given N pairs $(\boldsymbol{x}_n, h_n)$ of positions and height measurements we aim to find parameters

$$\boldsymbol{p}^* = \arg\min_{\boldsymbol{p}} \sum_{n=1}^{N} \rho\left(\left(\hat{\mathrm{h}}(\boldsymbol{x}_n, \boldsymbol{p}) - h_n \right)^2 \right) \tag{3.6}$$

that minimize the sum of squared and robustified errors. Here, we use the non-linear robustifier $\rho\colon \mathbb{R} \to \mathbb{R}^+$ as the optimization needs to be robust against outliers, i.e. non-ground points contained in the measurements. Assuming that the conditions Eq. (A.9) on the robustifier ρ hold, we can formulate the equivalent dual weighted Least-Squares (WLS) problem

$$\boldsymbol{p}^* = \underset{\boldsymbol{p}, w_1, \ldots, w_N}{\arg\min} \sum_{n=1}^{N} w_n \left(\hat{\mathrm{h}}(\boldsymbol{x}_n, \boldsymbol{p}) - h_n \right)^2 + \Phi(w_n) \tag{3.7}$$

according to the Black-Rangarajan duality (Appendix A.3.2). The weight penalty $\Phi\colon \mathbb{R}^+ \to \mathbb{R}^+$ prevents the weights to become zero.

We regularize the spline towards constant slope by penalizing variations of the second spline derivative. In absence of measurements, this will lead to an extrapolation with constant incline. As presented by Beck [Bec20], the closed-form solution of the smoothness cost term is a linear combination of the control points with precomputed weights $\boldsymbol{B}_\mathrm{S}$. This yields the final optimization problem

$$\begin{aligned} \boldsymbol{p}^*, w_1^*, \ldots, w_N^* = \underset{\boldsymbol{p}, w_1, \ldots, w_N}{\arg\min} & \sum_{n=1}^{N} w_n (\langle \mathbf{b}(\boldsymbol{x}_n), \boldsymbol{p} \rangle - h_n)^2 + \Phi(w_n) \\ & + \lambda \| \boldsymbol{B}_\mathrm{S} \boldsymbol{p} \|^2, \end{aligned} \tag{3.8}$$

including a weighted spline cost with weight penalty Φ and the smoothness cost weighted by λ. By reordering indices, this problem can be formulated by the WLS problem

$$\boldsymbol{p}^*, \boldsymbol{w}^* = \underset{\boldsymbol{p}, \boldsymbol{w}}{\arg\min} \left\| \mathrm{diag}(\tilde{\boldsymbol{w}})^{\frac{1}{2}} \left(\tilde{\boldsymbol{B}} \boldsymbol{p} - \tilde{\boldsymbol{h}} \right) \right\|^2 + \sum_{n=1}^{N} \Phi(w_n) \tag{3.9}$$

with

$$\tilde{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{B}_\mathrm{S} \end{bmatrix}, \quad \tilde{\boldsymbol{h}} = \begin{bmatrix} \boldsymbol{h} \\ \boldsymbol{0} \end{bmatrix}, \quad \tilde{\boldsymbol{w}} = \begin{bmatrix} w_1 & \ldots & w_N & w_S & \ldots & w_S \end{bmatrix}^\top. \tag{3.10}$$

3.3 Parameter Estimation

To solve Eq. (3.9), we use the iterative Graduated Non-Convexity (GNC) method presented by Yang et al. [Yan+20]. The approach uses weight penalty functions Φ_μ with a free parameter μ which controls problem convexity. The algorithm repeats two steps in an alternating fashion while changing μ in order to decrease convexity. In step 1, we fix the weights $\boldsymbol{w}$ and determine the optimal control points

$$\boldsymbol{p}^* = \arg\min_{\boldsymbol{p}} \left\| \operatorname{diag}(\boldsymbol{w})^{\frac{1}{2}} \left(\tilde{\boldsymbol{B}}\boldsymbol{p} - \tilde{\boldsymbol{h}} \right) \right\|^2 \tag{3.11}$$

$$= \left(\tilde{\boldsymbol{B}}^\top \operatorname{diag}(\boldsymbol{w}) \tilde{\boldsymbol{B}} \right)^{-1} \tilde{\boldsymbol{B}}^\top \operatorname{diag}(\boldsymbol{w}) \tilde{\boldsymbol{h}} \tag{3.12}$$

of the resulting linear WLS problem. In step 2, we fix the parameters $\boldsymbol{p}$ and determine the weights

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} w_n (\overbrace{\langle \mathbf{b}(\boldsymbol{x}_n), \boldsymbol{p} \rangle - h_n}^{\Delta h_n})^2 + \Phi(w_n) \tag{3.13}$$

$$w_n^* = \arg\min_{w_n} w_n \Delta h_n^2 + \Phi(w_n) \tag{3.14}$$

independently of each other. Here, we investigate the Geman McClure (GMC) and truncated Least-Squares (TLS) penalties

$$\Phi_{\mu,\mathrm{GMC}}(w) = \mu c^2 \left(\sqrt{w} - 1 \right)^2 \tag{3.15}$$

$$\Phi_{\mu,\mathrm{TLS}}(w) = \frac{\mu(1-w)}{\mu + w} c^2 \tag{3.16}$$

for which we can compute the optimal weights

$$w^*_{n,\mathrm{GMC}} = \left(\frac{\mu c^2}{\mu c^2 + \Delta h_n^2}\right)^2 \tag{3.17}$$

$$w^*_{n,\mathrm{TLS}} = \begin{cases} 1 & \text{if } \Delta h_n^2 < \frac{\mu}{\mu+1} c^2 \\ \frac{c\sqrt{\mu(\mu+1)}}{|\Delta h_n|} - \mu & \text{if } \frac{\mu}{\mu+1} c^2 \leq \Delta h_n^2 \leq \frac{\mu+1}{\mu} c^2 \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

in closed form (cf. Appendix A.3.4).

We adapt the convexity parameter $\mu^{(k)}$ in every iteration k such that the next

$$\mu^{(k+1)} = \alpha \mu^{(k)} \tag{3.19}$$

is changed by a constant factor α. This factor is set to $\alpha = 1.6^{-1}$ for the GMC penalty and $\alpha = 1.6$ for the TLS penalty. The optimization is stopped after a fixed number of steps or if $\mu^{(k+1)} < 1$.

As the measurements contain outliers (non-ground points) with a biased distribution, the method overestimates the true ground surface in general. To mitigate this issue, we scale positive and negative errors Δh_n differently. If $\Delta h_n > 0$, i.e. the point lies above the current ground estimate, we scale it with an asymmetry ratio $r_{\mathrm{asymm}} > 1$ yielding the asymmetric error

$$\Delta \tilde{h}_n = \begin{cases} r_{\mathrm{asymm}}\, \Delta h_n & \text{if } \Delta h_n > 0 \\ \Delta h_n & \text{otherwise} \end{cases}, \tag{3.20}$$

which we use to substitute Δh_n in Eqs. (3.17) and (3.18). Due to the error scaling, points above the current ground estimate may get a lower weight because of the higher distance. In other words, points below the current estimate are more likely to belong to the ground.

Figure 3.1 depicts the optimal weights as a function of the height difference Δh for different $\mu^{(k)}$, i.e. at different optimization steps. Due to the change of the convexity parameter μ, measurements with larger errors yield smaller weights, thus gain less influence on the total cost.

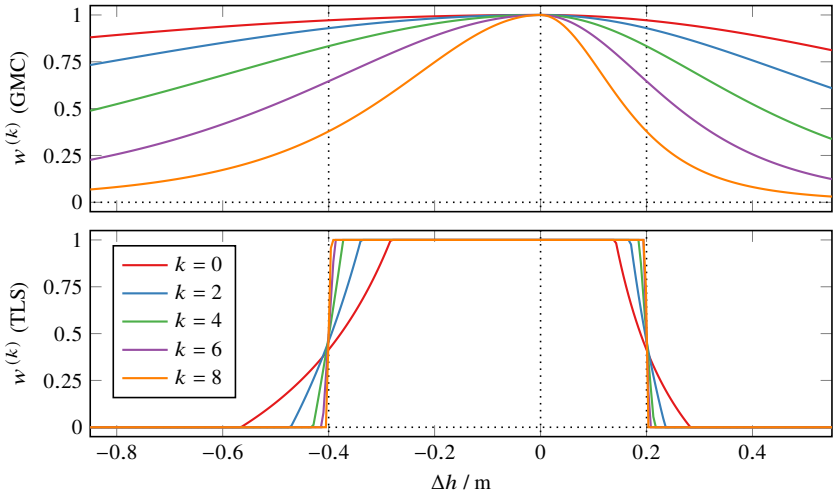


Figure 3.1: Weights of GMC and TLS penalty functions for $c = 0.4$ m and $r_{\text{asymm}} = 2$ at different convexity parameters $\mu^{(k)}$ changing during optimization.

3.4 Experiments

We evaluate on the training set of the SemanticKITTI data set ([Beh+19], cf. Appendix A.5) and split all range measurements into ground, non-ground and don't care classes, summarized in Table 3.1.

The parameters and default values of our method are summarized in Table 3.2.

3.4.1 Comparison of Different Ground Surface Models

We first compare the accuracy of different ground surface models when only ground points are used for estimation (outlier-free case). Here, we compare our UBS model to cubic polynomials, estimated and precalibrated ground planes. We randomly sample 10 % of all ground points for validation, i.e. these points are not used during optimization. We then compare the absolute height error between all validation points and the ground surface height estimated by the

Category	Classes	Abs. frequency in million	Rel. frequency in %
Ground	Lane marking, Other, Parking, Road, Sidewalk Terrain	10 196.462	43.949
Non-ground	Bicycle, Bicyclist, Building, Bus, Car, Fence, Motorcycle, Motorcyclist, Other, Person, Pole, Traffic sign, Truck, Trunk, Vegetation	12 559.459	54.133
Don't care	Unlabeled, Outlier	445.079	1.918
Total		23 201.000	100.000

Table 3.1: Distribution of ground, non-ground and don't care classes constructed from the SemanticKITTI training data set.

Parameter	Default Value	Related Exp.
Ground surface model	UBS	Section 3.4.1
Robustifier	TLS	Section 3.4.2
Error threshold c	0.4 m	Section 3.4.2
Initial convexity μ_0	1	
Number of iterations	10	
Asymmetry ratio r_{asymm}	2	Section 3.4.3
Spline degree	2	
Control point distance d_{C}	2 m	Section 3.4.4
Smoothness weight w_{S}	1	Section 3.4.4
Smoothness order	2	

Table 3.2: Parameters of our ground surface estimation method and their default values.

models. Figure 3.2 depicts the average absolute height errors and average errors depending on measurement distance.

In general, the UBS model maintains the lowest errors. Compared to the polynomial model the error only slightly increases with increasing measurement distances as the influence of measurements to the UBS model is restricted locally and thus is almost independent of locally varying measurement densities.

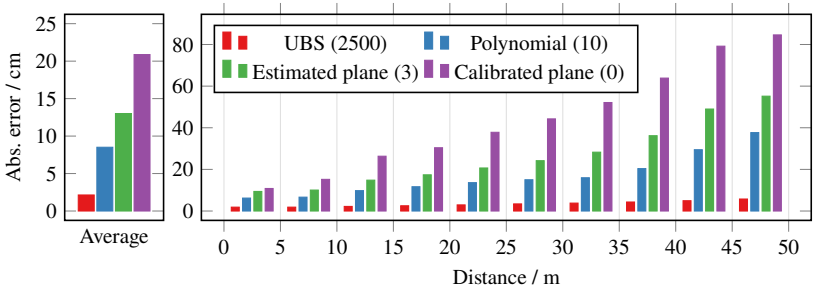


Figure 3.2: Abs. ground point error of different surface models (number of free parameters in brackets) when only ground points are used for optimization. Left: Average of all validation points. Right: Averaged within 5 m intervals of distance from the sensor.

3.4.2 Comparison of Robustifiers

Outlier Noise Estimation To justify the use of robust optimization methods, we first aim to estimate the ground distance distribution of non-ground points, i.e. the outlier noise. To estimate the outlier noise, we compute ground surfaces only based on the labeled ground points and compute a histogram of the errors between the estimated ground height and the non-ground points, which is depicted in Fig. 3.3.

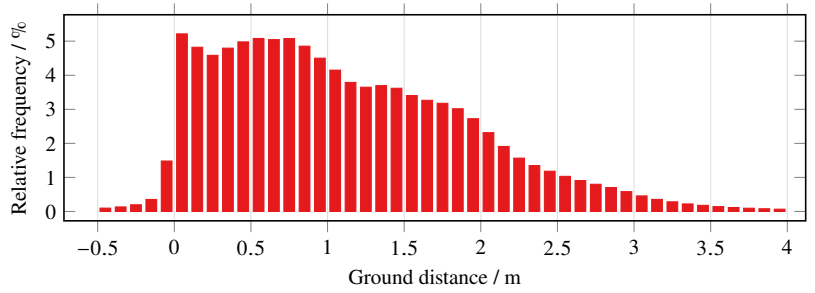


Figure 3.3: Ground distance histogram of non-ground points.

We observe from the histogram that the non-ground points are highly biased with a mean of 1.09 m and with a large height range from below the estimated surface to around 4 m. Note that the negative ground distances are likely due

to the control point distance of 2 m and the smoothness weight of 1 which on the one hand reduces overfitting on the inliers but on the other hand increases smoothing of abrupt changes of the ground surface.

Results Figure 3.4 compares the influence of different robustifiers on the optimization in two settings. On the one hand, we show the influence when only ground points are used, on the other hand when all points are used for optimization. For validation, we retain 10% of ground points in both experiments.

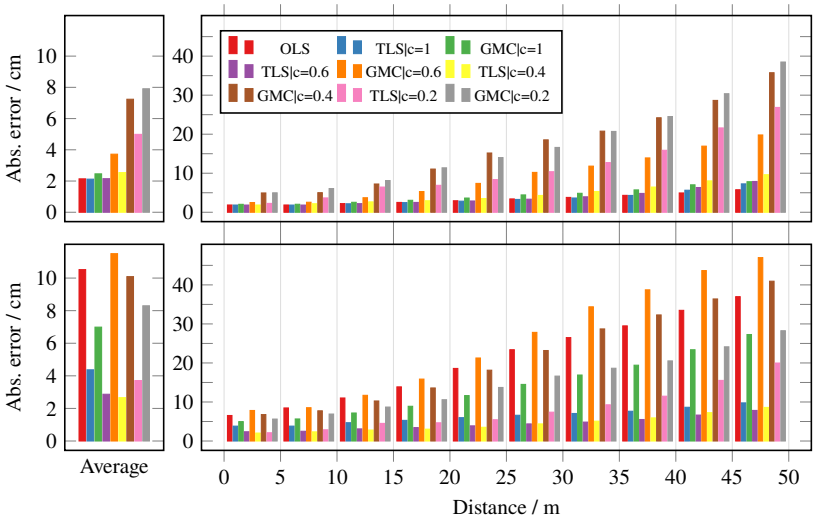


Figure 3.4: Abs. height error between estimated ground surface and ground validation points based on optimization using only labeled ground points (top) and including all points (bottom). Left: Average of all validation points. Right: Averaged within 5 m intervals of distance to the sensor.

We observe that the TLS method yields the best results in presence of outliers. An optimal error threshold for the TLS method seems to be in the range of 20 cm to 60 cm. The GMC approach does not always yield better results than the ordinary Least-Squares (OLS) baseline. This may be because the GMC method does not converge within 5 iterations.

3.4.3 Asymmetric Cost

Figure 3.5 summarizes the absolute errors between ground validation points and the estimated ground surface at different asymmetry ratios.

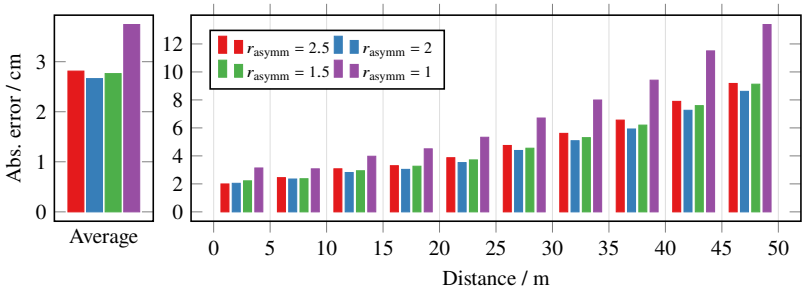


Figure 3.5: Abs. ground point error of TLS estimator for different asymmetry ratios. Left: Average of all validation points. Right: Averaged within 5 m intervals of distance to the sensor.

We observe that the TLS estimator yields the lowest ground surface errors for $1.5 < r_{\text{asymm}} < 2.5$.

3.4.4 Control Point Distance vs. Smoothness Weight

Figure 3.6 compares the absolute errors between estimated ground surface and ground validation points for different control point distances and smoothness weights.

In general, we observe that smaller control point distances and smoothness weights yield smaller errors. The influence of the smoothness weight increases with decreasing measurement density, often in areas far away from the sensor.

3.4.5 Validation on Experimental Vehicle

We implemented the ground surface estimation on our experimental vehicle (cf. Appendix A.6) equipped with four Velodyne VLP16 LiDARs on the roof corners and one Velodyne VLS128 on the roof center.

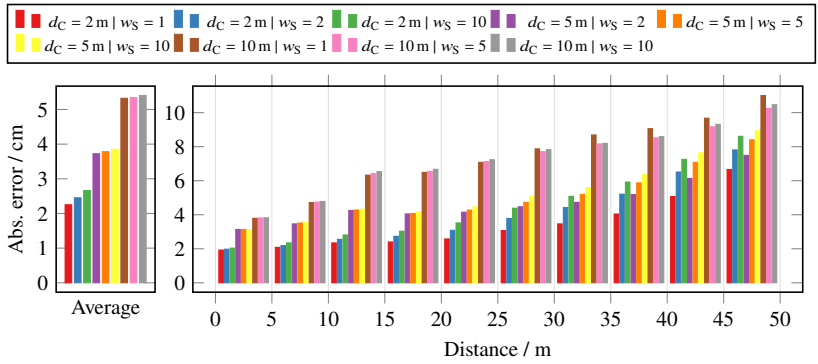


Figure 3.6: Abs. ground point error of TLS estimator for different control point distances d_C and smoothness weights w_S. Left: Average of all validation points. Right: Averaged within 5 m intervals of distance to the sensor.

The optimization time mainly depends on the number of iterations, point measurements and control points. The range sensor setup generates approximately 360 000 point measurements per scan at a scan rate of 10 Hz, which results in an approximate point measurement rate of 3.6 MHz. The optimization time is proportional to the number of iterations. However, as we receive sequential measurements, we initialize the ground surface with the last optimization result and perform only one iteration. Thus, we adapt the number of control points so that the optimization satisfies soft real-time constraints.

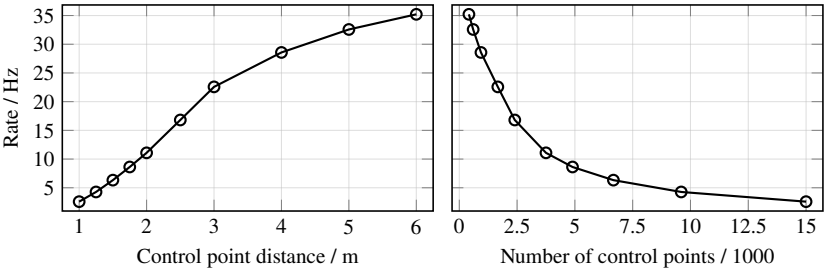


Figure 3.7: Ground surface estimation processing rate on the experimental vehicle depending on the control points within a 150 m × 100 m area.

Figure 3.7 summarizes the computation rates on the experimental vehicle depending on the control points within a 150 m × 100 m area. For example, if the ground surface estimation should process measurements at a rate of at least 10 Hz, the number of control points should be less than 3750 or in other words, the control point distance in this area should be at least 2 m.

Figure 3.8 shows the point set from full 360° scans of all LiDARs mounted on the experimental vehicle together with the estimated ground surface on a drive through Karlsruhe, Germany. We observe that the ground surface can be accurately estimated. Based on the resulting ground surface, we are able to distinguish between ground / non-ground points by applying a simple distance-based classifier.

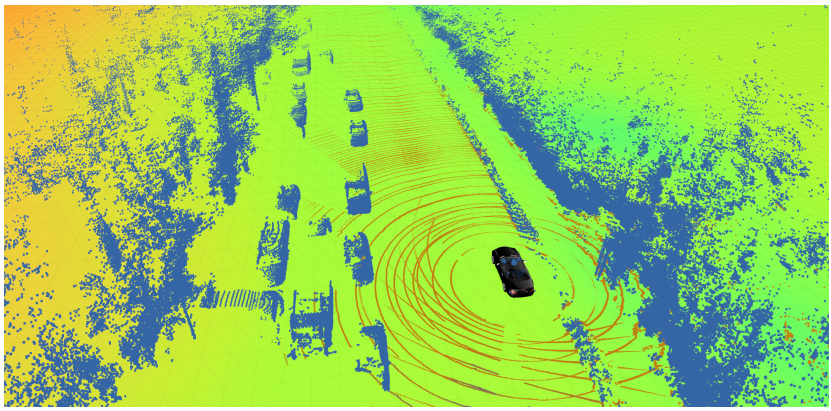


Figure 3.8: Point set from full 360° scans of all LiDARs on the experimental vehicle and estimated ground surface. The point set is colorized by the distance of points to the ground surface with brown denoting a distance of less than 10 cm and blue above 10 cm. The ground surface is colorized by its height relative to the vehicle reference frame.

Figures 4.8 and 4.13 illustrate more real-world examples with estimated ground surfaces.

4 Evidential Occupancy & Elevation Grid Mapping

For subsequent object detection and tracking tasks, we aim to estimate sensor-agnostic occupancy and elevation maps from multiple noisy and possibly contradicting measurements. This process requires an accurate estimate of the ground surface which was presented in Chapter 3. Then, observability and drivability information may be computed based on these grid maps.

Here, we assume the ground surface to be known so that we can distinguish between ground/non-ground reflections and model obstacle height relative to the ground. Furthermore, we assume a fixed elevation boundary above ground (e.g. the vehicle height) where we consider non-ground measurements to be obstacles. This assumption implies that obstacles are connected to the ground which is not true in general. However, this circumstance as well as measurements on permeable objects can be resolved as conflict in evidential occupancy mapping (cf. Section 4.3.3).

We start by summarizing related work on occupancy and elevation mapping as well as a versatile sensor model in Section 4.1. After making considerations on the sensor noise in Section 4.2, we introduce our mapping approach (Section 4.3) and the extraction of observability and drivability information (Section 4.4). Finally, we present the mapping of static environment with known poses as an application in Section 4.5.

We validate all processing steps on an experimental vehicle (cf. Appendix A.6) using five attached LiDARs. To save the reader from skipping back and forth, we provide the validation results in each section.

4.1 Related Work

Starting from the origins of occupancy grid mapping (Section 4.1.1), we provide an overview on elevation mapping (Section 4.1.2) and range sensor models (Section 4.1.3).

4.1.1 Occupancy Grid Mapping

Occupancy grid mapping of static environments was initially developed by Elfes [Elf89] and Moravec [Mor89]. Elfes defines an occupancy grid as a multidimensional random field that maintains stochastic estimates of the cell occupancy states in a spatial lattice. The cell states are then recursively updated by new measurements for which Moravec presents a computationally efficient version of the binary Bayes filter to estimate the occupancy state. Elfes and Moravec also describe methods for sensor fusion using multiple sensors and correlating grid maps in order to estimate odometry.

In the Bayesian approach occupancy is modeled as a Bernoulli-distributed random variable X such that $\mathrm{P}(X = \mathrm{o}) + \mathrm{P}(X = \mathrm{f}) = 1$ where o and f denote the occupied and free hypothesis, respectively. This approach, however, does not consider the amount of evidence collected in order to make a guess about the cell states which e.g. means that the case of no observations at all and the case of conflicting measurements may lead to the same probabilities $\mathrm{P}(X = \mathrm{o}) = \mathrm{P}(X = \mathrm{f}) = \frac{1}{2}$. This problem can be resolved if cell states are explained by means of evidence theory (e.g. [Yi+00; YA06]). In evidence theory, the frame of discernment

$$\Omega = \{\mathrm{o}, \mathrm{f}\} \tag{4.1}$$

usually contains the hypotheses *occupied* (o) and *free* (f) and basic belief assignments (BBAs)

$$\mathrm{m}\colon 2^{\Omega} \to [0, 1], \quad \mathrm{m}(\emptyset) = 0 \tag{4.2}$$

$$\sum_{\forall \omega \in 2^{\Omega}} \mathrm{m}(\omega) = \mathrm{m}(\{\mathrm{o}\}) + \mathrm{m}(\{\mathrm{f}\}) + \mathrm{m}(\Omega) = 1 \tag{4.3}$$

are derived from single measurements that operate on the power set $2^{\Omega} = \{\emptyset, \mathrm{o}, \mathrm{f}, \Omega\}$. The cell state is then estimated by fusing multiple BBAs using combination rules (cf. Section 2.1).

Compared to the before mentioned static grid mapping approaches, current research is focused on dynamic environments. Usually, the dynamic state of a cell is approximated by a set of particles. Nuss et al. [Nus+18] model the state of multiple grid cells as a random finite set (RFS) for which they derive a filter in a top-down manner with the binary Bayes filter being a special case under certain assumptions. The authors also present a real-time capable filter approximation based on evidence theory. Steyer et al. [STW18] divide the environment into static, dynamic and free-space modeled by belief functions and use custom combination rules to resolve conflicts. However, both authors do not further elaborate on the design of belief assignments from range measurements.

4.1.2 Elevation Grid Mapping

Most popular occupancy mapping approaches in top-view grid maps do not consider height or ray geometry information. Instead, *elevation mapping* is often considered as a different research field. Fankhauser et al. [FBH18] present an elevation mapping approach that takes into account sensor noise and yields probabilistic height estimates on an elevation grid, including upper and lower confidence bounds. Given point measurements, the authors associate points to cells and update their elevation independently. In order to deal with dynamic environments, Fankhauser et al. [FBH18] extend their approach by casting rays from the sensor origin to the measurement end points in order to detect inconsistencies and thus increase update speed. The authors report that ray casting is performed at a frequency of 1 Hz due to its computational cost.

4.1.3 Range Measurement Models

Yang et al. [YA06] simulate 2D range sensors and model two BBAs

$$\mathrm{m}_{\mathrm{o}}(\omega) = \begin{cases} \lambda_1 & \text{if } \omega = \{\mathrm{o}\} \\ 1 - \lambda_1 & \text{if } \omega = \Omega \\ 0 & \text{otherwise} \end{cases}, \quad \mathrm{m}_{\mathrm{f}}(\omega) = \begin{cases} \lambda_2 & \text{if } \omega = \{\mathrm{f}\} \\ 1 - \lambda_2 & \text{if } \omega = \Omega \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

with $0 < \lambda_1, \lambda_2 < 1$ which are applied in case of reflections or ray traversals from the sensor origin to the reflection position. The authors combine these single measurement BBAs using Dempster's rule with the recursive BBA

$$\mathrm{m}(\omega)^{(t)} = \begin{cases} \dfrac{\mathrm{m}(\omega)\mathrm{m}(\Omega)^{(t-1)} + \mathrm{m}(\Omega)\mathrm{m}(\omega)^{(t-1)}}{1 - \mathrm{m}(\overline{\omega})\mathrm{m}(\Omega)^{(t-1)} - \mathrm{m}(\Omega)\mathrm{m}(\overline{\omega})^{(t-1)}} & \text{if } \omega \in \Omega \\ 1 - \mathrm{m}(\{\mathrm{o}\})^{(t)} - \mathrm{m}(\{\mathrm{f}\})^{(t)} & \text{if } \omega = \Omega \end{cases} \tag{4.5}$$

$$\mathrm{m}(\omega)^{(0)} = \begin{cases} 0 & \text{if } \omega \in \Omega \\ 1 & \text{if } \omega = \Omega \end{cases} \tag{4.6}$$

which is updated for every new point measurement using either m_o or m_f for m. Unfortunately, this makes the approach intractable in case of high point measurement rates. In addition, the approach by Yang et al. [YA06] does not consider the 3D ray geometry and partial occlusions along the height dimension.

Richter et al. [Ric+19] propose a sensor model that maps range measurements and semantic estimates into occupancy and semantic class beliefs using also information on the sensor ray geometry. Here, occupancy is modeled by the two classes *obstacle* (O) and *ground* (G) which are super sets of other semantic classes such as *car*, *pedestrian*, *street* or *terrain*. For each sensor, the authors

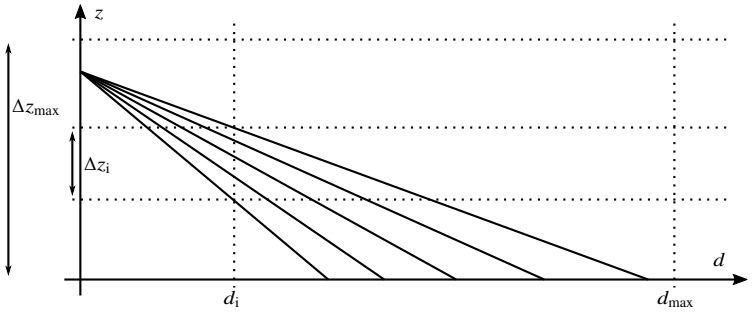


Figure 4.1: Illustration of the distance ratio $\frac{d_\mathrm{i}}{d_\mathrm{max}}$ and the height ratio $\frac{\Delta z_\mathrm{i}}{\Delta z_\mathrm{max}}$ used to determine the false negative probability mapping p_FN for arbitrary distances d_i [Ric+19]. Here, d_max denotes the sensor-dependent maximum observation distance and Δz_max the elevation boundary, depending on the vehicle height.

model the BBA

$$\mathrm{m}(\omega) = \begin{cases} p_{\mathrm{FN}}^{t}\left(1 - p_{\mathrm{FP}}^{r}\right) & \text{if } \omega = O \\ p_{\mathrm{FP}}^{r}\left(1 - p_{\mathrm{FN}}^{t}\right) & \text{if } \omega = \mathcal{G} \\ 1 - \sum_{\xi \neq \Omega} \mathrm{m}(\xi) & \text{if } \omega = \Omega \end{cases} \tag{4.7}$$

depending on the number of transmissions t, the number of reflections r, the false positive probability p_{FP} and the false negative probability p_{FN}, respectively. Whereas p_{FP} is set to a constant value, $p_{\mathrm{FN}} = \mathrm{p}_{\mathrm{FN}}(d, z)$ is modeled as a function of the ray geometry and the distance to the sensor origin. They compute the false negative probability

$$\mathrm{p}_{\mathrm{FN}}(d, z) = 1 - \left(1 - \frac{d}{d_{\max}}\right) \frac{\Delta z}{\Delta z_{\max}} \left(1 - p_{\mathrm{FN,\,max}}\right), \tag{4.8}$$

based on the distance ratio $\frac{d}{d_{\max}}$, the height ratio $\frac{\Delta z}{\Delta z_{\max}}$ and a maximum false negative rate $p_{\mathrm{FN,\,max}}$. Figure 4.1 illustrates the ratio on an exemplary range sensor with five vertical beams. The false negative probability p_{FN} increases when the ratio between observed height Δz and maximum height $\Delta z_{\max}$ decreases or when the ratio between observed distance d and maximum distance $d_{\max}$ increases. Fusion of two belief maps M_1 and M_2 is done by applying the conjunctive rule of combination

$$\mathrm{m}(\omega | M_1 \oplus M_2) = \sum_{\omega_1 \cap \omega_2 = \omega} \mathrm{m}_1(\omega_1)\mathrm{m}_2(\omega_2), \quad \omega \in 2^{\Omega} \tag{4.9}$$

to the non-conflicting hypotheses and assigning the conflicting masses

$$\zeta = \mathrm{m}_1(\{\mathrm{o}\})\mathrm{m}_2(\{\mathrm{f}\}) + \mathrm{m}_1(\{\mathrm{f}\})\mathrm{m}_2(\{\mathrm{o}\}) \tag{4.10}$$

conservatively to the *occupied* hypothesis. This yields the fused BBA

$$\mathrm{m}_{\mathrm{fused}}(\omega) = \begin{cases} \mathrm{m}(\omega | M_1 \oplus M_2) & \text{if } \omega \in \{\mathrm{f}, \Omega\} \\ \mathrm{m}(\omega | M_1 \oplus M_2) + \zeta & \text{if } \omega = \{\mathrm{o}\} \end{cases}. \tag{4.11}$$

4.2 Range Sensor Noise

Many available range sensors consist of a rotor on which multiple laser diodes at different azimuth angles are mounted and a receiver which measures the returned energy as a function of the time delay between pulse emission and return (or an equivalent reflection distance).

The absolute rotor angle is accurately measured by an encoder with usually negligible quantization errors[1]. However, LiDARs often suffer from calibration errors due to an imprecise initial intrinsic calibration or extrinsic calibration errors if multiple range sensor measurements are transformed to a common reference frame. In addition, heat deformation and mechanical wear applies so that the actual angle accuracy is rather in the range of $0.1°$ to $0.2°$.

Range accuracy is typically limited by close-distance time difference measurements and low material reflectivity. Usually, manufacturers provide the *typical accuracy* which refers to ambient wall test performance that excludes retro-reflection, averages across most channels and may vary based on factors including but not limited to range, temperature and target reflectivity [Vel19]. Thus, in reality, the true distance accuracy deviates from the data sheet values.

In order to find suitable grid map parameters, we identify the range noise of one sensor by measuring a static scene ($\approx 25 \times 10^6$ points) and estimating the standard deviation of corresponding points in range direction between two frames. Corresponding points are determined by nearest neighbor search. Figure 4.2 depicts a histogram of the standard deviation and its distance-dependence for the Velodyne VLS128 LiDAR. Around 50 % of all measurements have a standard deviation of less than 3 cm. However, the average standard deviation in a range of up to 50 m is 6.55 cm. As a trade-off between high range resolution and low computational cost during mapping and ray casting, we set the size of grid cells in range direction to 10 cm. When rotating at 10 Hz, there are around 2200 different rotor angles per full $360°$ scan. To reduce aliasing artifacts, we aim to map measurements from at least two rotor angles to one cell. Therefore, we set the cell size in rotational direction to $0.35°$.

[1] The azimuthal angle resolution of Velodyne LiDARs (used in this work) is around $0.01°$, yielding a lateral error of 0.3 mm at a distance of 100 m.

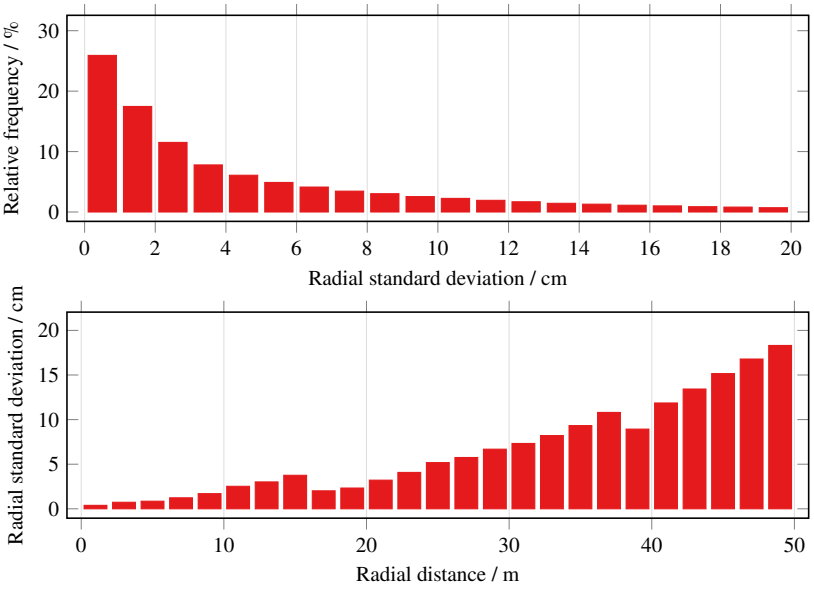


Figure 4.2: Radial error characteristics of the Velodyne VLS128 LiDAR.

4.3 Range Sensor Mapping

Given range measurements as a set of points with yaw angle, horizontal distance and height above ground, we first map these points into a polar grid which allows for efficient range traversal and parallelized ray casting and afterwards perform remapping into Cartesian coordinates.

4.3.1 Occupancy Mapping

Here, we incorporate deviations of range measurements (r_m, ϕ_m) into the grid mapping process by assuming that these deviations are equally distributed in

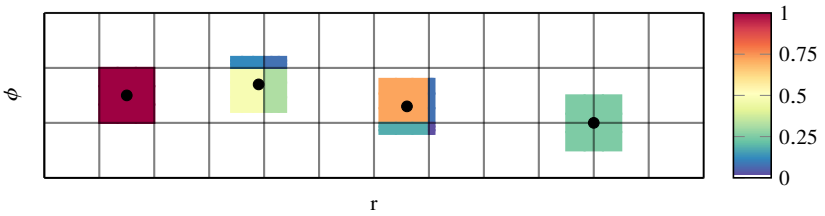


Figure 4.3: Occupancy probability assignment. Left: All probability mass is assigned to one cell. Right: Probability mass is distributed evenly along neighboring cells. Center: Probability mass is distributed differently along the cells.

the range $(\Delta r, \Delta\phi)$ of the grid map resolution (cf. Fig. 4.3), which we model by the probability density function (PDF)

$$\mathrm{p}(r, \phi | r_m, \phi_m) = \frac{1}{\Delta r}\mathrm{rect}\left(\frac{r - r_m}{\Delta r}\right)\frac{1}{\Delta\phi}\mathrm{rect}\left(\frac{\phi - \phi_m}{\Delta\phi}\right). \tag{4.12}$$

Then, up to four neighboring grid cells yield a non-zero emergence probability which we determine for each cell by evaluating the probability

$$\mathrm{P}_n = \int_{r_n - \frac{\Delta r}{2}}^{r_n + \frac{\Delta r}{2}} \int_{\phi_n - \frac{\Delta\phi}{2}}^{\phi_n + \frac{\Delta\phi}{2}} \mathrm{p}(r, \phi | r_m, \phi_m)\, \mathrm{d}r\mathrm{d}\phi \tag{4.13}$$

$$= \frac{1}{\Delta r}\int_{r_n - \frac{\Delta r}{2}}^{r_n + \frac{\Delta r}{2}} \mathrm{rect}\left(\frac{r - r_m}{\Delta r}\right)\mathrm{d}r \frac{1}{\Delta\phi}\int_{\phi_n - \frac{\Delta\phi}{2}}^{\phi_n + \frac{\Delta\phi}{2}} \mathrm{rect}\left(\frac{\phi - \phi_m}{\Delta\phi}\right)\mathrm{d}\phi \tag{4.14}$$

$$= \max\left(1 - \frac{|r_n - r_m|}{\Delta r}, 0\right)\max\left(1 - \frac{|\phi_n - \phi_m|}{\Delta\phi}, 0\right) \tag{4.15}$$

that the observation originated from cell n. Equation (4.15) can be evaluated efficiently by multiplying the overlap ratios in both directions.

In order to compute free space belief, we need to determine the number of transmissions in each grid cell by accumulating the probability masses along each azimuthal angle starting from the reflection at the largest distance. This step is independent for each azimuthal angle, thus can be parallelized and yields the transmissions layer after ray casting.

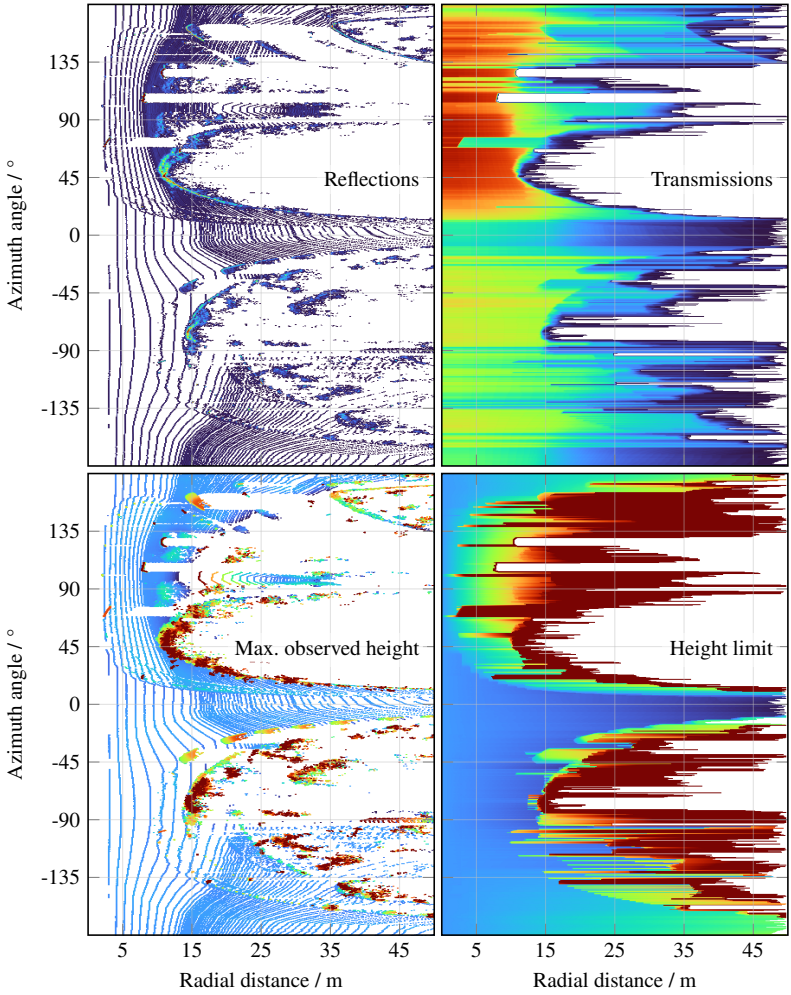


Figure 4.4: Reflections, transmissions, max. observed height and height limit layer after polar grid mapping. The reflections and transmissions layers count the number of reflections and ray cast transmissions, respectively. The maximum observed height layer denotes the maximum height above ground of all reflections in a grid cell. The height limit layer denotes a maximum height of objects above ground obtained from ray casting. Blue/red: Low/high values, white: No values.

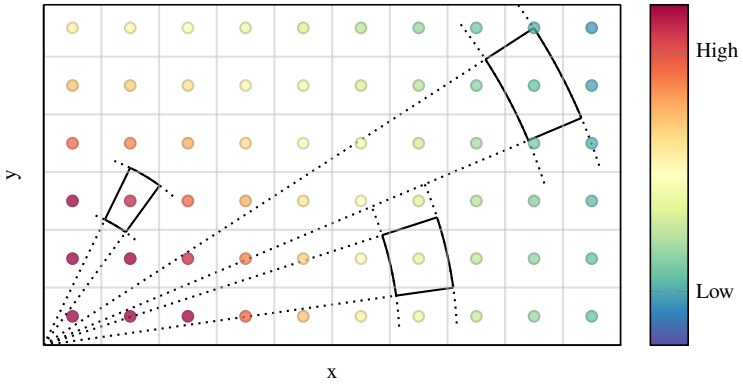


Figure 4.5: When mapping accumulated values (e.g. the number of reflections or transmissions) from polar grid cells (black) to corresponding Cartesian grid cells (gray) we compensate for the distance-dependent area ratio (color-coded) on the Cartesian cell center positions.

The first row in Fig. 4.4 depicts a full 360° VLS128 LiDAR scan mapped into the polar reflections and transmissions layer.

The final polar maps then need to be mapped into the Cartesian domain where we perform bilinear interpolation and store the interpolation weights in lookup tables. Here, it is important to conserve the energy due to the range-dependent area change when remapping from polar to Cartesian cells. Figure 4.5 indicates the large area differences for different distances between polar and Cartesian grids. Given a polar grid with cell sizes $[\Delta r, \Delta\phi]$, a Cartesian grid with cell sizes $[\Delta x, \Delta y]$ at position $[x_n, y_n]$ and the offset $[x_0, y_0]$ between the map origins, we compute the polar area

$$A_{\text{polar},n} = \frac{\Delta\phi}{2}\left((r_n + \Delta r)^2 - (r_n - \Delta r)^2\right) = \Delta\phi\ \Delta r\ r_n \tag{4.16}$$

at radius

$$r_n = \sqrt{(x_n - x_0)^2 + (y_n - y_0)^2} \tag{4.17}$$

and determine the ratio

$$\frac{A_{\text{cart},n}}{A_{\text{polar},n}} = \frac{\Delta x\ \Delta y}{\Delta\phi\ \Delta r\ r_n} \tag{4.18}$$

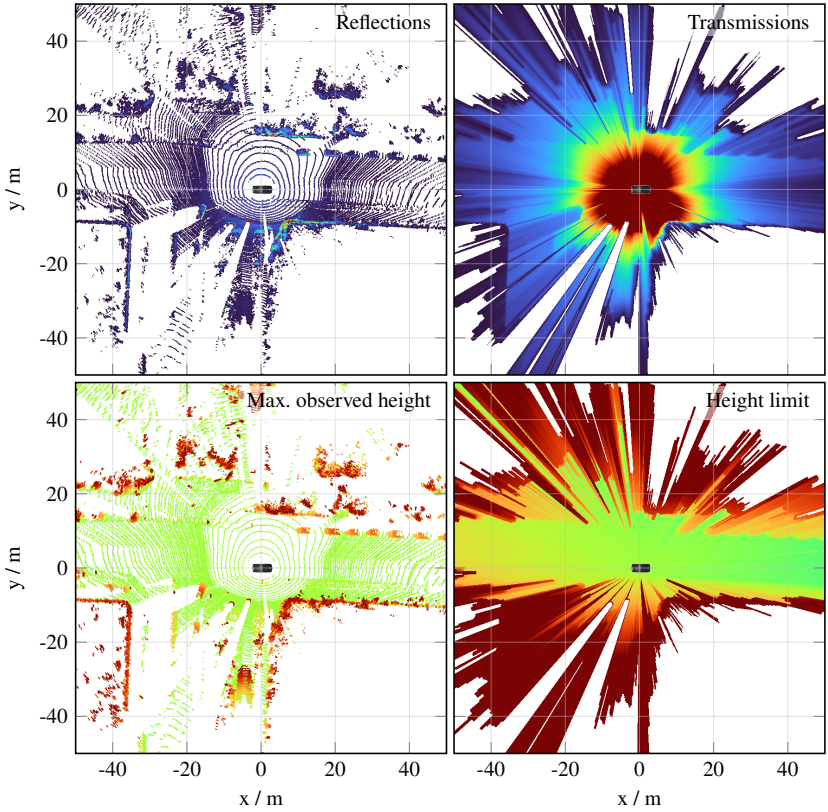


Figure 4.6: Reflections, transmissions, max. observed height and height limit layer after Cartesian grid mapping. The reflections and transmissions layers count the number of reflections and ray cast transmissions, respectively. The max. observed height layer denotes the max. height above ground of all reflections in a grid cell. The height denotes a maximum height of objects above ground obtained from ray casting. Blue/red: Low/high values, white: No values.

which is used to correct features that are accumulated in the polar grid such as the number of reflections and transmissions. This method can be efficiently implemented within the interpolation process by an additional cell-wise multiplication.

Figure 4.6 depicts the Cartesian sensor grid layers of a full 360° VLS128 LiDAR scan mapped from their polar representations. The Cartesian sensor grid maps of each sensor are then combined in the occupancy and elevation mapping steps.

4.3.2 Elevation Mapping

During the polar mapping process, we also assign elevation information to grid cells. For each grid cell, we store the smallest and highest measured height in two layers. As depicted in Fig. 4.7, we determine the maximum observation height and an upper height limit during ray casting. The upper height limit denotes the height of the lowest transmission above the max. observed height, which makes it an upper limit for the true obstacle height. Assuming the height to be equally distributed between max. observed height and the upper height limit, we may estimate the height as the mean of this distribution.

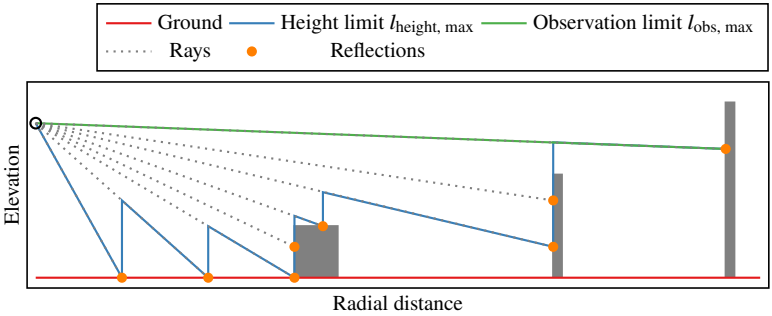


Figure 4.7: During ray casting we determine the height limit of obstacles (blue) and the upper observation limit (green) due to the vertical sensor field of view (FOV). We assume that the true obstacle height is always between highest reflection points (orange) and height limits.

The layers containing the max. observed height and the upper height limit for an exemplary scene are depicted in the bottom row of Fig. 4.4. Figure 4.8 depicts the estimated ground surface as presented in Chapter 3, the upper height limit, the max. observed height and the combined height map. The combined map is constructed from the estimated height of non-ground obstacles and the ground surface height sampled at cell centers if no objects are present.

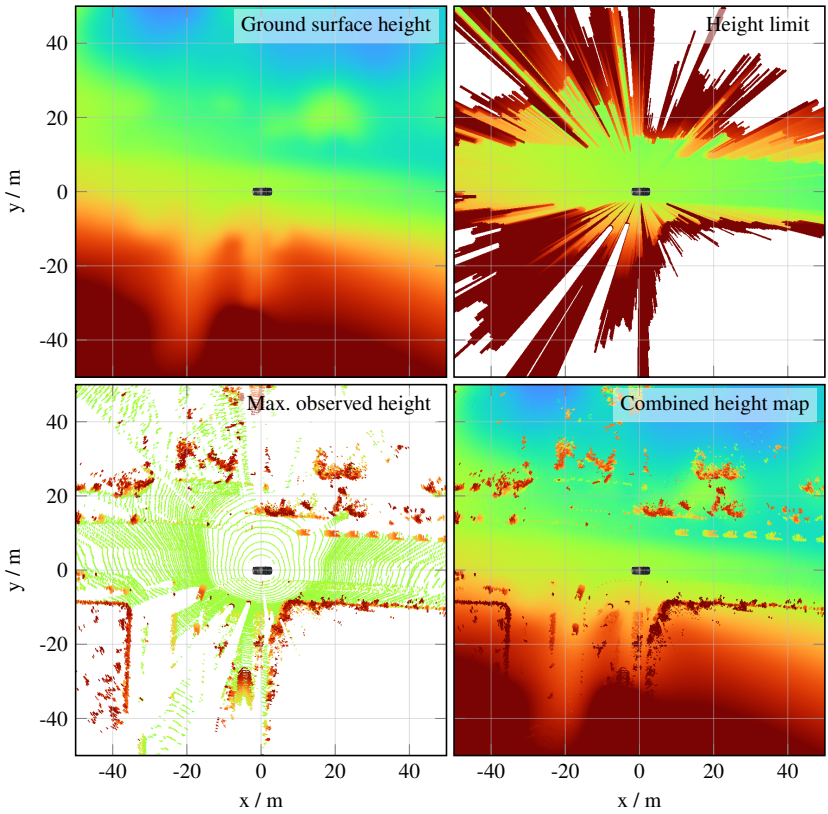


Figure 4.8: Estimated ground surface height, height limit, max. observed height and resulting combined height map. Blue/red: Low/high values, white: No values.

4.3.3 Occupancy Belief

We determine occupancy and free belief using the sensor model presented by Richter et al. [Ric+19] (cf. Section 4.1.3). The advantage of using an evidential sensor model is its ability to explicitly model conflicts such as erroneous measurements, occlusions and modeling errors. In addition, this sensor model can be efficiently evaluated as beliefs are computed using accumulated measurements instead of being updated for every new measurement (as in Eq. (4.5)). In this work, we substitute the height range Δz in Eq. (4.8) by

$$\Delta z = \Delta z_{\max} - \left(l_{\text{obs, max}} - l_{\text{height, max}}\right). \tag{4.19}$$

It denotes the difference between the maximum elevation range $\Delta z_{\max}$ and the minimum observable height range obtained from the observation limit $l_{\text{obs, max}}$ and the upper height limit $l_{\text{height, max}}$.

For example, permeable objects such as bushes yield an almost equal number of reflections and transmissions which are assigned to Ω. The same holds for objects not connected to the ground (e.g. birds, hanging traffic lights or signs) which are not regarded in our elevation model.

Figure 4.9 depicts the evidential layers after processing Cartesian occupancy and elevation grid maps of a single 360° VLS128 LiDAR scan. Here, we set $p_{FP} = 0.05$ and $p_{FN, \max} = 0.7$ which we determined heuristically for the LiDARs mounted on the vehicle. Along the occupied, free and uncertainty belief, we can determine the pignistic occupancy probability from the evidential layers by Eq. (2.7).

4.4 Observability & Drivability

The observability $\text{bel}(\{\text{o}\}) + \text{bel}(\{\text{f}\}) = 1 - \text{bel}(\Omega)$ describes observable and conflict-free areas in the map. We define drivable areas by the compound probability of all cells possibly occupied by the vehicle being free. As in evidence theory

$$\text{bel}(\{\text{f}\}) \leq \text{prob}(\{\text{f}\}) \leq \text{pl}(\{\text{f}\}) = 1 - \text{bel}(\{\text{o}\}), \tag{4.20}$$

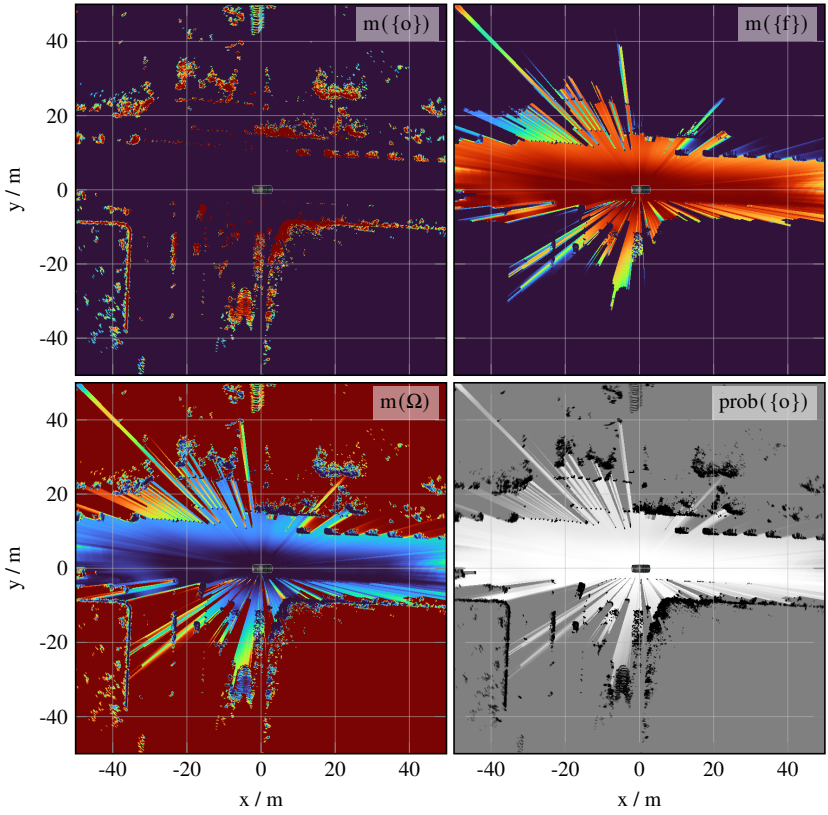


Figure 4.9: Evidential basic belief layers m({o}), m({f}), m(Ω) and pignistic occupancy probability prob({o}) after evidential occupancy mapping.
Note that m({o}) + m({f}) + m(Ω) = 1. Blue/white: Low values. Red/black: High values.

we use bel({f}) as a conservative estimate for the free space probability. Then, if grid cells are assumed independent, the compound probability

$$\text{prob}(\{\text{d}\}) = \prod_{\forall n \in \mathcal{N}} \text{bel}^{(n)}(\{\text{f}\}) \tag{4.21}$$

for drivable space is the product of free beliefs in all $|\mathcal{N}|$ cells the vehicle may occupy. If we rewrite drivability and belief as 2D signals

$$\text{prob}_\text{d} \colon \mathbb{Z}^2 \to [0, 1] \, , \text{bel}_\text{f} \colon \mathbb{Z}^2 \to [0, 1], \tag{4.22}$$

the drivability can be efficiently evaluated by computing the likelihoods

$$\log\big(\text{prob}_\text{d}(\boldsymbol{x})\big) = \log(\text{bel}_\text{f}(\boldsymbol{x})) \ * * \ \text{shape}(\boldsymbol{x}) \tag{4.23}$$

using a two-dimensional discrete convolution with an indicator function $\text{shape} \colon \mathbb{R}^2 \to \{0, 1\}$ describing the occupancy due to the vehicle shape. Here, we utilize the fact that

$$\log\left(\prod_{\forall n \in \mathcal{N}} \text{bel}^{(n)}(\{\text{f}\}) \right) = \sum_{\forall n \in \mathcal{N}} \log\Big(\text{bel}^{(n)}(\{\text{f}\})\Big). \tag{4.24}$$

In this work, we simplify the vehicle shape to a circle with a diameter equal to the vehicle's width. This simplification does neither consider the exact vehicle shape nor motion constraints. However, due to its rotational invariance we only need to compute one convolution. To compute the drivability more accurately one may perform multiple convolutions using differently oriented shape kernels with subsequent checking of motion constraints.

We suggest approximating observable or drivable regions by polygons, which reduces the amount of information and thus defines a simple interface to subsequent tasks such as motion planning. We do this by thresholding the layers extracting contours around the areas above this threshold. Figure 4.11 depicts observability, drivability and the extracted polygons at a threshold of 75 % in an exemplary scenario.

4.5 Static Environment Mapping with Known Poses

As an application, we consider the problem of mapping static environments with known poses. This includes mapping multiple sensors for short durations with known extrinsic calibration, mapping static environment for longer durations with single sensors and known transformations (e.g. from odometry), or both.

To perform mapping, we need to fuse multiple measurements. Here, we combine evidential occupancy and height information differently. Evidential occupancy is combined as in Eq. (4.11) where we account the conflicting masses to the more conservative *occupied* hypothesis.

To fuse the height information, we model the height $H \sim \mathcal{M}$ as a random variable following a mixture of N uniform distributions with equal weights described by the PDF

$$\mathrm{p}(h) = \frac{1}{N} \sum_{n=1}^{N} \mathrm{p}(h|n), \tag{4.25}$$

where the mixture PDFs

$$\mathrm{p}(h|n) = \frac{1}{\Delta h_n} \mathrm{rect}\left(\frac{h - \hat{h}_n}{\Delta h_n} \right) \tag{4.26}$$

$$\hat{h}_n = \frac{h_{n,\mathrm{ub}} + h_{n,\mathrm{lb}}}{2}, \quad \Delta h_n = h_{n,\mathrm{ub}} - h_{n,\mathrm{lb}} \tag{4.27}$$

are determined from the upper height limits $h_{n,\mathrm{ub}}$ obtained from ray casting and the maximum observed heights $h_{n,\mathrm{lb}}$ of all reflections in a grid cell (cf. Fig. 4.7 for an illustration).

We then determine the mean and variance

$$\mu = \frac{1}{N} \sum_{n=1}^{N} \mu_n \tag{4.28}$$

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^{N} \left(\sigma_n^2 + \mu_n^2 \right) - \mu^2 \tag{4.29}$$

of the mixture distribution (cf. Eqs. (A.35) and (A.36)) which we use to describe the fused height information.

Figure 4.10 illustrates this for an example and Fig. 4.12 in a mapping scenario where 30 range measurements are combined.

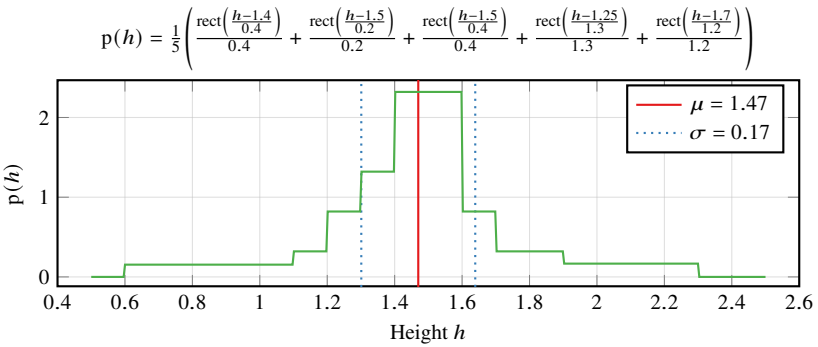


Figure 4.10: Exemplary height mixture PDF from five uniform components with intervals $[1.2, 1.6]$, $[1.4, 1.6]$, $[1.3, 1.7]$, $[0.6, 1.9]$ and $[1.1, 2.3]$, its mean μ and standard deviation σ.

For our experiments, observability and drivability are represented as polygons.

Figure 4.11 compares the evidential belief, observability and drivability layers when using a single VLP16 LiDAR to the layers obtained when using the full sensor setup (four VLP16 LiDARs and one VLS128 LiDAR).

On the one hand, we observe that using multiple sensors increases the observability and drivability by a large margin. On the other hand, we can see that our method provides meaningful estimates for different sensor setups and implicitly handles unobserved areas, e.g. due to sensor installation, occlusion or outage.

In contrast to the multi-sensor setup, Figs. 4.12 and 4.13 compare the different grid map layers estimated for a single scan or 30 subsequent scans. This validates our approach for mapping of static environment up to the grid map resolution. Although not implemented here, we believe that the mapping approach might even be able to handle semi-static objects such as parking vehicles when using a time-decay model to combine measurements at different

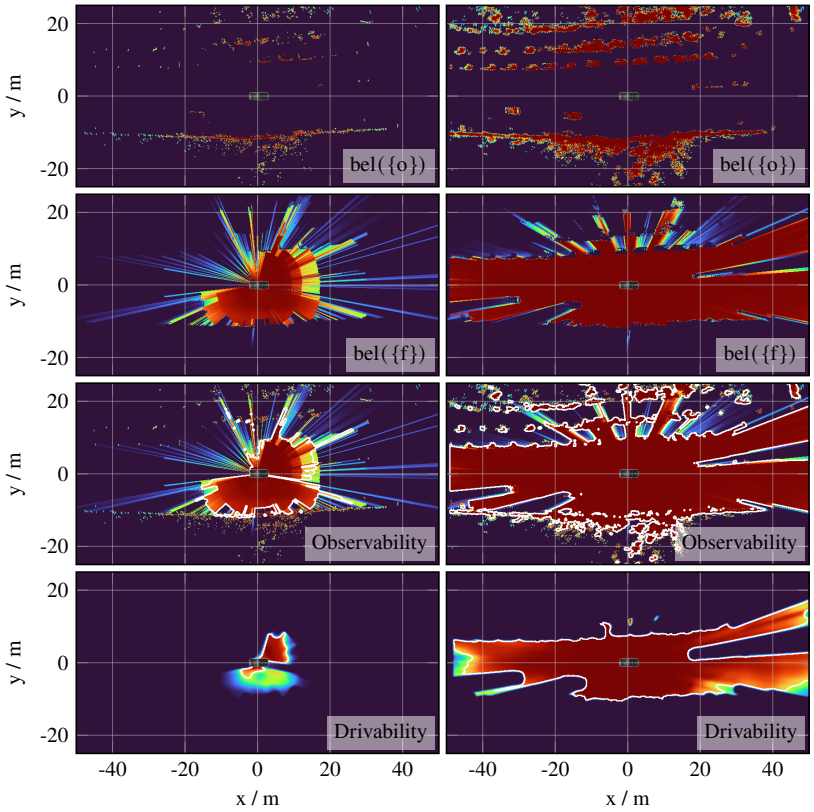


Figure 4.11: Occupied belief bel({o}), free belief bel({f}), observability bel({o}) + bel({f}) and drivability (cf. Eq. (4.21)) for a Velodyne VLP16 LiDAR mounted in the front right corner (left column) and the full sensor setup (right column). On top, we also draw white polygons defined by minimum observability and drivability, respectively. Blue/red: low/high values.

times. Note that we use the range measurements and odometry transformations provided in the KITTI odometry benchmark for this experiment.

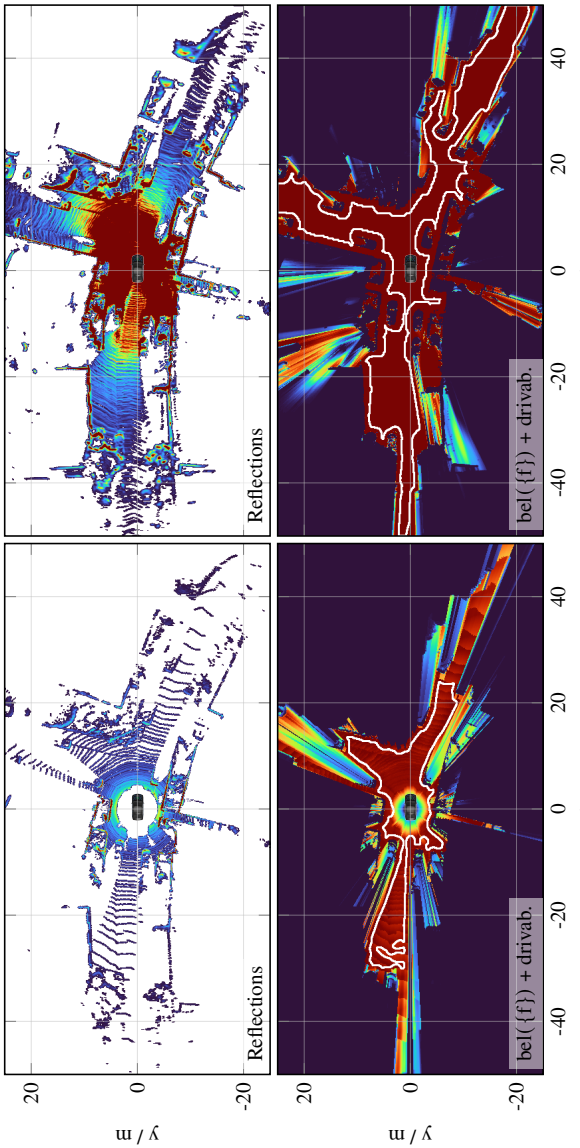


Figure 4.12: Number of reflections and free belief with drivability of a single 360° range measurement (left column) and 30 range measurements (right column) in scene 00 of the KITTI odometry benchmark. Blue/red: Low/high values, white: No values/boundary.

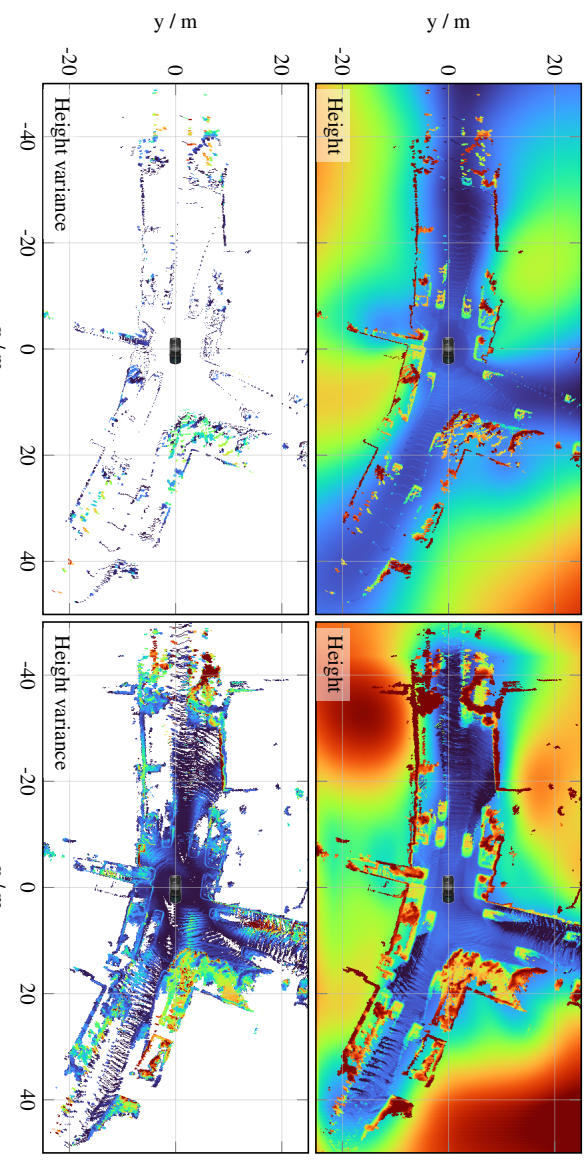


Figure 4.13: Estimated height and height variance of a single 360° range measurement (left column) and 30 range measurements (right column) in scene 00 of the KITTI odometry benchmark. Blue/red: Low/high values, white: No values.

5 Object Detection Considering Uncertainties

The task of object detection combines three closely related subtasks: The semantic classification of relevant objects (incl. their absence), their pose and shape estimation. In the traffic context, this includes the detection of traffic participants, e.g. cars, cyclists and pedestrians. In grid maps, small objects such as pedestrians may only cover a few grid cells, depending on the cell size[1] which makes their detection a particularly hard problem (cf. Fig. 5.1). For subsequent tracking tasks, also information about localization and shape estimation deviations may be taken into consideration. Therefore, it is beneficial to provide a meaningful and interpretable description of these deviations.

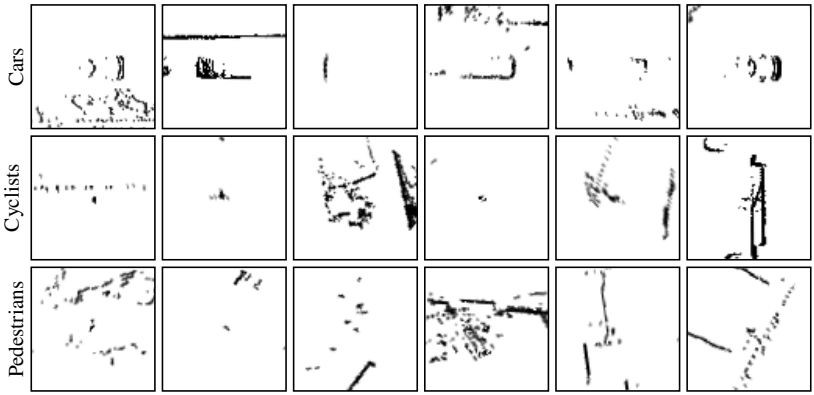


Figure 5.1: Occupied belief for different cars, cyclists and pedestrians. Each object is aligned with the image center. Patch size: 10 m×10 m. Grid cell size: 12.5 cm×12.5 cm.

[1] Given a grid cell size of 12.5 cm pedestrians usually cover less than 5×5 cells.

Here, we present two approaches to object detection based on the assumption that objects are always attached to the ground surface. Thus, an object pose can be represented by its 2D position and yaw angle. Furthermore, we model the object shape by a 2D polygon and a corresponding height.

Section 5.2 presents a fast and generic segmentation approach that produces obstacles represented by position and a generic hull from grid maps. In contrast to this method, a deep convolutional object detector, presented in Section 5.3 and learned on labeled training data, additionally estimates semantic classes and oriented bounding boxes. Finally, we evaluate our convolutional object detector on the nuScenes data set (Section 5.4).

5.1 Related Work

We present related work on image segmentation (Section 5.1.1) and convolutional object detectors (Section 5.1.2) before we introduce more recent work on object detection in top-view grid maps in Section 5.1.3.

5.1.1 (Range) Image Segmentation

Before the advent of machine learning (ML) models, hierarchical segmentation methods were developed to detect obstacles in images.

To segment objects in range images, Moosmann et al. [MPS09] estimate local surface convexity which holds if the point reconstructed from a range pixel is behind a neighboring pixel's plane approximation. Applying this convexity measure as a segmentation criterion, the authors are able to distinguish between obstacles and ground. However, the method only works on single range images and cannot be extended to multiple sensors easily which renders this method unsuitable for our application.

Beucher et al. [BM93] segment objects by applying the watershed algorithm to morphologically closed binary images. Their method is split into a marker selection and subsequent segmentation which provides closed contours in a computationally inexpensive way.

5.1.2 Convolutional Object Detectors

An object detector can be characterized by its meta-architecture and components, namely backbone, feature fusion and heads, which are described in the following.

Meta-Architectures Object detectors can be divided into two-stage and single-stage detectors. The Faster R-CNN meta-architecture presented by Ren et al. [Ren+15] is a two-stage approach divided into a region proposal network (RPN) and a box classification and regression network. The RPN extracts features used to predict class-agnostic box candidates in a set of prior boxes defined on the input image. Features in prior boxes with a high predicted *objectness* score are then cropped and fed into the box classification and regression network.

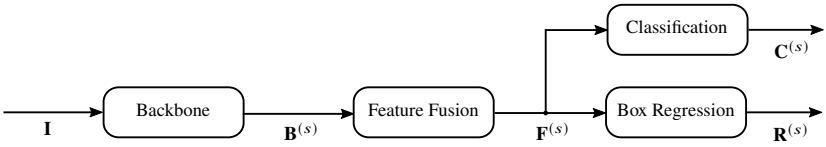


Figure 5.2: General structure of a single shot object detector. A backbone model takes an input image $\mathbf{I}(\boldsymbol{x})$ and computes S backbone features $\mathbf{B}^{(s)}(\boldsymbol{x})$ at different pyramid scales which are then fused into S features $\mathbf{F}^{(s)}(\boldsymbol{x})$. These features are then mapped by the heads into classifications $\mathbf{C}^{(s)}(\boldsymbol{x})$ and prior box-relative regressions $\mathbf{R}^{(s)}(\boldsymbol{x})$ where each location $\boldsymbol{x}^{(s)}$ is mapped to all prior boxes defined at that location.

In contrast, single-stage detectors (e.g. SSD proposed by Liu et al. [Liu+16]) predict bounding boxes and semantic classes in a single feedforward CNN without the additional RPN reducing inference time while, in many cases, still achieving sufficiently good performance. The general structure of a single-stage object detector is depicted in Fig. 5.2. A deep convolutional backbone maps input images (e.g. RGB images or grid maps) to backbone features $\mathbf{B}^{(s)}(\boldsymbol{x})$ at S different scales which may then be fused into features $\mathbf{F}^{(s)}(\boldsymbol{x})$. Here, each coordinate $\boldsymbol{x}_k^{(s)}$ defines a location where at least one prior box is defined.

The amount of prior boxes passed to the box classification and regression network is not only much higher than actual ground truth boxes in the image but also much higher compared to two-stage approaches that preselect object

candidates. Lin et al. [Lin+17b] mitigate this class imbalance by designing a loss function which focuses on more difficult examples, i.e. likelihoods close to the inverse number of classes.

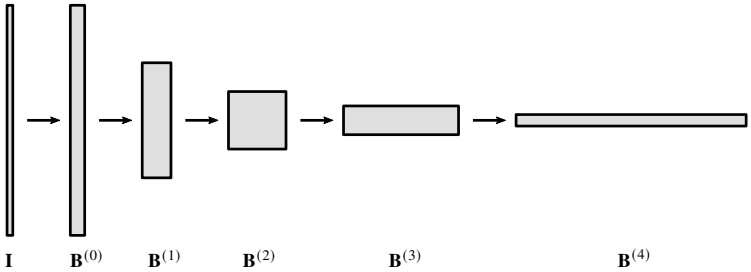


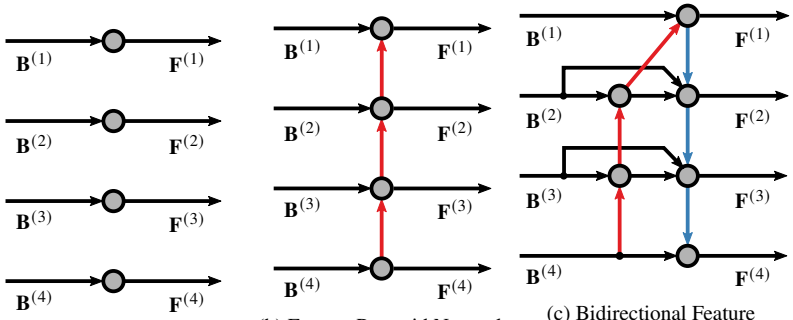
Figure 5.3: Backbone features $\mathbf{B}^{(0)} \dots \mathbf{B}^{(4)}$ during processing. The feature value dimension increases while spatial resolution is reduced. Here, convolutional transformations within the network are visualized by edges.

Backbones A detection stage input consists of semantic features computed by a deep backbone (or feature extractor) such as ResNet [He+16], MobileNet [San+18] or EfficientNet [TL19]. These models have in common that, during processing, intermediate features $\mathbf{B}(\boldsymbol{x}_0)$ decrease in spatial resolution at increasing value dimension, which can be interpreted as semantic information (cf. Fig. 5.3). In ResNets, computational units

$$\mathbf{X}_{k+1} = \mathbf{f}_k \circ \mathbf{X}_k + \mathbf{X}_k \Leftrightarrow \mathbf{f}_k \circ \mathbf{X}_k = \mathbf{X}_{k+1} - \mathbf{X}_k \tag{5.1}$$

are implemented as residual functions between output and input features $\mathbf{X}_{k+1}$ and $\mathbf{X}_k$, respectively. The authors show that networks with many concatenated units can be trained without vanishing gradients. Among other aspects, MobileNet and EfficientNet use spatially and depth-wise separable convolutions (cf. Example 2.6) to reduce the number of parameters and thus the amount of floating-point operations. The structure of EfficientNets is optimized by an extensive architecture search and their capacity can be scaled by a single hyperparameter which controls resolution, depth and width.

Feature Fusion The idea of fusing features at different scales is to combine accurate spatial with rich semantic information. Whereas SSD [Liu+16] does not



(a) No fusion

(b) Feature Pyramid Network (FPN) [Lin+17a]

(c) Bidirectional Feature Pyramid Network (BiFPN) [TPL20]

Figure 5.4: Feature fusion strategies. The gray nodes depict transformations while edges denote features. Additionally, blue / red edges involve up- and downsampling, respectively. We distinguish between no fusion (Fig. 5.4a), unidirectional fusion (Fig. 5.4b) as applied in FPN (cf. [Lin+17a]) and bidirectional fusion (Fig. 5.4c) as applied in BiFPN (cf. [TPL20]). The fused features $\mathbf{F}^{(1)}, \ldots, \mathbf{F}^{(4)}$ are then used as input for the object detection head.

apply any feature fusion (cf. Fig. 5.4a), FPNs [Lin+17a] perform feature fusion at different scales in a unidirectional, top-down manner (Fig. 5.4b). Compared to FPNs, BiFPNs add a bottom-up path and cross-scale connections [TPL20] for feature fusion (Fig. 5.4c).

Heads The features $\mathbf{F}^{(s)}$ are fed into separate classification and box regression heads. Each head performs additional convolutions to adapt the features to its specific task before mapping to a volume $[w_i, h_i, d_i]$ where w_i and h_i denote width and height and d_i the number of priors per position multiplied with the number of classes (classification head) or box regression parameters (box regression head). Thus, there is a classification and box regression for each prior box.

Most object detectors for camera images use the same heads across all scales (e.g. [Lin+17a; TPL20]). This accounts for the scale variance which is typical for camera images due to the projective transformation of objects. However, this is not the case for orthographic top-view grid maps.

5.1.3 Object Detection in Top-View Grid Maps

Steyer et al. [STW17] extract clusters of moving objects by applying DBSCAN (cf. [Est+96]) to dynamic grid cells considering Cartesian distance, velocity difference and evidence of free space between occupied cells. However, often static obstacles are also required by other automated driving (AD) components making it necessary to apply DBSCAN to all grid cells which is computationally expensive.

As 3D measurements are projected onto the ground surface, computationally efficient 2D convolutions can be applied when developing convolutional object detectors as in Complex-YOLO [Sim+18], BirdNet [Bel+18] or TopNet [Wir+18]. To mitigate the information loss during ground surface projection, Yang et al. [YLU18a; YLU18b] compute occupancy features at different heights, whereas Lang et al. [Lan+19] propose a low-level encoding of 3D point features before ground surface projection.

Choice of Input Features There is a large variety of grid cell features used for object detection across different publications. Often, the (normalized) number of reflections and reflection energy are considered [GKF09; BFG15; Che+17]. Hörmann et al. [Hör+18] use occupied and free belief, per-cell 2D velocity and its auto-covariance matrix estimated by a particle filter. To mitigate the information loss during projection, Golovinskiy et al. [GKF09] encode the height as a normal distribution and estimate standard deviations of points in the ground plane.

Box Encoding To predict axis-aligned bounding boxes in camera images, the prior box offset $\Delta \boldsymbol{x}$ and box extent offset $[\Delta w, \Delta h]$ are used. For oriented bounding boxes, Jiang et al. [Jia+18] additionally estimate an angle resulting in a minimal representation. However, during training the angle regression loss becomes non-differentiable due to the angle wrap. Beyer et al. [BHL15] represent the angle ϕ by a biternion representation $[\cos(\phi), \sin(\phi)]$. Modeling the orientation as a von Mises distribution, their model learns orientation regression using the loss $L_{\mathrm{VM}} = 1 - e^{\kappa(\langle \boldsymbol{y}, \boldsymbol{t} \rangle - 1)}$ where κ denotes the concentration and $\boldsymbol{y}$ and $\boldsymbol{t}$ the ground truth and regression biternion, respectively.

Uncertainty Estimation Uncertainty may be divided into epistemic (or systematic) and aleatoric (or statistical) uncertainty [Gal16]. Gal et al. [GG16] show that Bayesian inference in artificial neural networks (ANNs) to capture epistemic uncertainty can be approximated by Monte Carlo dropout applied to multiple forward passes during inference. This technique can be applied to object detection problems, however, at the cost of additional computation time due to multiple forward passes with random dropout [FRD18; Wir+19b]. Aleatoric uncertainty may be further divided into homoscedastic (or constant variance) and heteroscedastic (or variable variance) uncertainty. In object detection applications, Feng et al. [FRD18; Fen+19] show that estimating heteroscedastic aleatoric uncertainty of the box regression parameters improves accuracy.

5.2 Fast Segmentation Method

We present a fast object segmentation method for grid maps based on the free and occupied belief layers introduced in Section 4.3. Although this method is not able to estimate semantic classes or oriented bounding boxes, it is computationally inexpensive, robust and reliable as it also works for generic unseen and unlabeled objects/obstacles. Thus, it is aimed to complement our object detector presented in Section 5.3.

We first perform morphological closing on the occupied belief bel($\{o\}$) to increase occupied evidence in unobserved areas. Note that we apply grayscale instead of binary morphologies. We chose the kernel size to be in the range of 0.25 m to 0.75 m depending on the sampling density. Afterwards, we determine the difference between the morphologically closed occupied belief and the free belief bel($\{f\}$) reducing occupied mass in areas of high free space mass, e.g. when observing free space between parking cars or pedestrians walking next to each other. This result is binarized with a small threshold in the range of 0.05 to 0.15 before we apply connected-components labeling (CCL) to obtain object clusters. Then, for each cluster we compute a 2D concave hull approximation and assign its geometric center as object position. Finally, we determine the object height from the minimum and maximum observed height out of all grid cells associated to this cell.

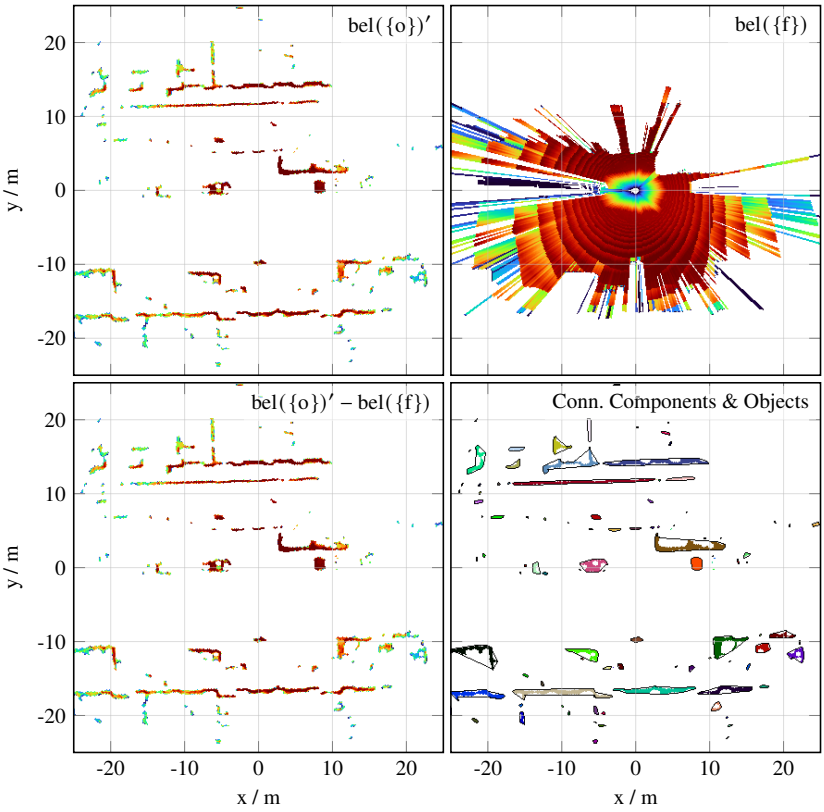


Figure 5.5: Processing steps of our fast segmentation method. The free belief bel({f}) is subtracted from the morphologically closed occupied belief bel({o})′ resulting in bel({o})′ – bel({f}) (lower left corner). We then apply connected-components labeling (CCL) and a concave hull approximation on the result yielding object boundaries.

Figure 5.5 illustrates the segmentation steps on a road-scenario recorded with our experimental vehicle (cf. Appendix A.6). Unfortunately, there is no data set available that allows for a detailed, quantitative evaluation.

5.3 Convolutional Object Detector

Our goal is to detect relevant traffic participants, estimate their position, shape and semantic class. Here, we represent a shape by oriented bounding boxes. As input, we use the fused occupancy belief and elevation grid map layers presented in Section 4.3.

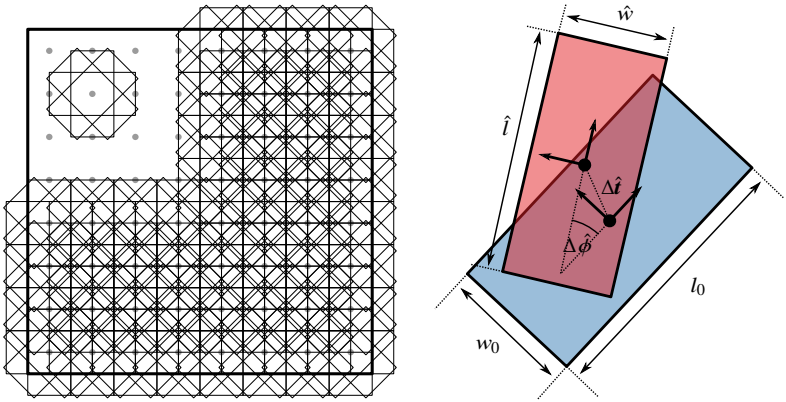
5.3.1 Overview

We follow the general structure of single-shot object detectors illustrated in Fig. 5.2. A deep, fully convolutional backbone takes grid maps as input and determines S intermediate backbone features $\mathbf{B}^{(s)}$ on different pyramid scales s. These backbone features are then fused into S features $\mathbf{F}^{(s)}$ and passed to the classification and box regression heads, respectively.

5.3.2 Prior Boxes & Box Matching

To achieve translational invariance, we create prior boxes at different locations on the grid map. Prior boxes are defined by their position, extent and orientation and should ideally cover the object of interest. We create prior boxes at different pyramid scales in order to capture objects of different sizes. Figure 5.6a illustrates the prior box creation for multiple orientations at a single scale and box extent (aspect ratio 2:1).

In order to train the object detector, we need to establish correspondences between prior boxes and ground truth labels. Therefore, we perform matching based on the rotated intersection over union (IoU) presented in Appendix A.8. We match each labeled ground truth box to the prior box with the highest IoU, resulting in N_{pos} *positive* matches. All N_{neg} prior boxes that are not matched to



(a) Prior boxes

(b) Box representation

Figure 5.6: In order to achieve translational and scale invariance we create multiple, oriented prior boxes at different locations (gray dots) and scales. Figure 5.6a illustrates the placement of prior boxes at a single scale with four different orientations. All but one prior box location in the top left part of the image were removed for better visualization of different prior box orientations. As illustrated in Fig. 5.6b, we represent a box by its translation $\Delta\hat{\boldsymbol{t}}$, extent $\Delta\hat{\boldsymbol{e}} = \left[\hat{l} - l_0, \hat{w} - w_0\right]$, angle $\Delta\hat{\phi}$ and height $\Delta\hat{h}$ relative to the prior box.

a ground truth label are considered *negatives* and assigned to the rejection class in the object classifier.

On each pyramid scale we use two prior box sizes at two aspect ratios (1:1 and 2:1) and six rotations (0°, 30°, 60°, 90°, 120°, 150°) yielding 24 prior boxes per location. We create prior boxes at pyramid scale level 2 to 4 at every cell location. With an original grid map size of 800×800 cells, this results in a total number of

$$N_{\text{prior}} = 24 \cdot \left(200^2 + 100^2 + 50^2 + 25^2\right) = 1275000 \tag{5.2}$$

prior boxes.

5.3.3 Box Representation & Regression

We represent a box by its relative translation $\Delta \boldsymbol{t}$, extent $\Delta \boldsymbol{e}$, angle $\Delta\hat{\phi}$ and height Δh relative to its corresponding prior box. Similar to Simon et al. [Sim+18], we model the angle as a complex number $e^{j\,\phi} = \cos(\phi) + j\,\sin(\phi)$ and estimate a 2D angle biternion. In a previous work, Wirges et al. [Wir+18] observed that this encoding is more stable during training due to its continuity and non-vanishing gradients and thus yields higher object detection performance. We also observed that estimation of the heading direction within $-180°$ to $180°$ is difficult in grid maps, whereas the box orientation can be estimated accurately in a range of $-90°$ to $90°$. For this reason we split the orientation estimation into two problems, a π-periodic angle regression and a binary heading classification.

In the following, we define the normalized box regression estimate

$$\hat{\boldsymbol{b}} = \left[\frac{\Delta\hat{t}_\mathrm{x}}{l_0}, \frac{\Delta\hat{t}_\mathrm{y}}{w_0}, \frac{\hat{l} - l_0}{l_0}, \frac{\hat{w} - w_0}{w_0}, \sin\big(2(\hat{\phi} - \phi_0)\big), \cos\big(2(\hat{\phi} - \phi_0)\big) \right] \tag{5.3}$$

and analogously the normalized regression target $\boldsymbol{b}$ which are independent of prior box size and orientation. Then, the box regression loss

$$L_\mathrm{reg} = \sum_{n=1}^{N_\mathrm{pos}} \mathcal{H}_1\Big(\|\hat{\boldsymbol{b}}_n - \boldsymbol{b}_n\| \Big) \tag{5.4}$$

denotes the sum of all Huber losses (cf. Eq. (A.3)) of all N_pos matched pairs of corresponding regression targets $\boldsymbol{b}_n$ and regression estimates $\hat{\boldsymbol{b}}_n$.

5.3.4 Classification

We assign each sample to one of C classes or an additional rejection class which means the absence of objects for this prior box resulting in $C + 1$ classes in total. Based on the focal loss presented by Lin et al. [Lin+17b], we define the classification loss

$$L_\mathrm{cls} = \lambda_1 L_\mathrm{dir} + \lambda_2 L_\mathrm{sem} \tag{5.5}$$

as the weighted sum of the direction loss L_{dir} for estimated direction likelihoods $\hat{\text{p}}(d_n)$ and semantic class loss L_{sem} for estimated semantic class likelihoods $\hat{\text{p}}(c_n)$.

We define the direction loss

$$L_{\text{dir}} = -\sum_{n=1}^{N_{\text{pos}}} \text{p}(d_n) \log(\hat{\text{p}}(d_n)) + (1 - \text{p}(d_n)) \log(1 - \hat{\text{p}}(d_n)) \tag{5.6}$$

as the binary cross entropy (BCE) (cf. Section 2.4.3) between the true probability distribution p of directions d_n and its approximating function $\hat{\text{p}}$. Then, the total classification loss

$$\begin{aligned} L_{\text{sem}} = \alpha \sum_{n=1}^{N_{\text{pos}}} (1 - \hat{\text{p}}(c_n))^{\gamma} \log(\hat{\text{p}}(c_n)) \\ + (1 - \alpha) \sum_{n=1}^{N_{\text{neg}}} (1 - \hat{\text{p}}(c_n))^{\gamma} \log(\hat{\text{p}}(c_n)) \end{aligned} \tag{5.7}$$

is based on the focal loss presented by Lin et al. [Lin+17b] with the modulation term $\left(1 - \hat{c}_{m,n}\right)^{\gamma}$ decreasing the weight of easy examples, i.e. examples with an estimated high semantic class likelihood $\hat{c}_m$. In addition, it provides balancing by the α parameter due to the large difference of positive and negative prior box matches.

5.3.5 Uncertainty Estimation

In another setting, we model the object position and orientation as normally and von-Mises distributed random variables. This approach differs from Wirges et al. [Wir+19b] who model the box position and extent as mutually independent normally and log-normally distributed random variables. In subsequent experiments we observed that the variances of box position and extent are highly correlated which leads to ambiguous position and box extent uncertainty estimates, e.g. increasing the position instead of the box extent variance. We resolve this ambiguity by only estimating the positional variance which leads to better convergence during training. The orientation is modeled as von Mises distributed random variable (cf. Appendix A.9) centered around

the predicted orientation for which we additionally estimate the concentration parameter κ. We define the normalized regression estimate

$$\hat{\boldsymbol{b}}_{\text{prob}} = \left[\hat{\boldsymbol{b}}, \hat{\sigma}_{\Delta x}, \hat{\sigma}_{\Delta y}, \hat{\kappa}\right] \tag{5.8}$$

which extends the regression estimate introduced in Eq. (5.3) by the estimated standard deviations $\hat{\sigma}_{\Delta x}$, $\hat{\sigma}_{\Delta x}$ in x and y directions and the estimated von Mises concentration $\hat{\kappa}$.

To estimate $\hat{\boldsymbol{b}}_{\text{prob}}$ we define the probabilistic regression loss

$$L_{\text{reg, prob}} = L_{\text{box, prob}} + L_{\text{dir, prob}} \tag{5.9}$$

that consists of the probabilistic box parameter regression

$$\begin{aligned} L_{\text{box, prob}} = \hat{\sigma}_{\Delta x}^{-2}\left(\hat{b}_1 - b_1\right)^2 + \hat{\sigma}_{\Delta y}^{-2}\left(\hat{b}_2 - b_2\right)^2 \\ + \log\left(\hat{\sigma}_{\Delta x}^2\right) + \log\left(\hat{\sigma}_{\Delta y}^2\right) \\ + \sum_{m=3}^{6}\left(\hat{b}_m - b_m\right)^2 \end{aligned} \tag{5.10}$$

and the direction loss

$$L_{\text{dir, prob}} = \log(2\pi \mathrm{I}_0(\hat{\kappa})) - \hat{\kappa}\cos\left(\phi - \hat{\phi}\right), \tag{5.11}$$

where we approximate I_0 by Eq. (A.42).

5.3.6 Optimization Objective

The total loss

$$L = \lambda_1 L_{\text{reg}} + \lambda_2 L_{\text{cls}} + \lambda_3 L_{\text{weight}} \tag{5.12}$$

denotes the weighted sum of regression loss (Eq. (5.4)), classification loss (Eq. (5.5)) and a weight penalty L_{weight} to reduce overfitting. We use L2 weight decay (cf. Example 2.7) which is computed on all convolution kernel weights $\boldsymbol{w}_{\text{kernel}}$.

5.3.7 Post-Processing

At inference time, we perform greedy IoU-based non-maximum suppression (NMS) to reduce the amount of object proposals. The NMS algorithm first selects the proposal with the highest class score and then discards all other proposals above a certain IoU threshold to that proposal. These steps are repeated in an alternating fashion until no more proposals are left to compare. Finally, we transform the coordinates of the resulting detections from grid map to world coordinates.

5.4 Experiments

If not specified otherwise, we train all models with the parameters summarized in Table 5.1. We choose the basic RetinaNet structure [Lin+17b] with a ResNet-38 backbone and an FPN as default architecture due to its fast training convergence and low memory consumption. For the same reason, we only perform random horizontal flipping around the y-axis.

Due to memory restrictions all models are trained with a batch size of 4 on one *Nvidia GeForce RTX 2080 Ti* GPU using the ADAM optimizer (cf. Example 2.3, $\beta_1 = 0.9$, $\beta_2 = 0.999$). After each nonlinearity, batch normalization (BN) (cf. Section 2.4.2) is performed. We use a linear learning rate warm-up from 1×10^{-5} to 1×10^{-4} within 3000 iterations and terminate the optimization at iteration 300 000 at a learning rate of 1×10^{-5} using a cosine learning rate decay.

5.4.1 Quantitative Evaluation

Table 5.2 summarizes the evaluation results of different model configurations. We use the metrics of the nuScenes object detection benchmark [Cae+20] for evaluation (cf. Appendix A.10.1). This includes the mean average precision (mAP) and the true positive metrics mean average translation error (mATE), mean average scale error (mASE) and mean average orientation error (mAOE).

Parameter	Value
Backbone	ResNet-38
Feature fusion	FPN (128 filters)
Convolutions	Separable 3×3 (cf. Example 2.6)
Weight decay	1×10^{-4} (L2, cf. Example 2.7)
Scales	2 to 5
Box-, class & association head	Scale-dependent, N=2, F=128
IoU method	Aligned, threshold 0.1
Data augmentation	Random horizontal flip
NMS threshold	0.5

Table 5.1: Default object detection model parameters used in evaluation.

Data Augmentation In addition to randomly horizontal flipping, we evaluate the effect of randomly rotated inputs on the model performance. Experiments Obj1, Obj2 and Obj3 in Table 5.2 depict the evaluation results for no rotation, rotations of $\pm 15°$ and $\pm 180°$, respectively.

We observe that the model performance slightly increases when the model is trained on grid maps randomly rotated in a range of $\pm 15°$. However, rotations in a range of $360°$ decrease the model performance. On the one hand, we suspect that large rotations lead to slower convergence so that training needs more iterations to converge. On the other hand, large rotations may change the object parameter training distribution which then varies from the evaluation set distribution.

Box Encodings We compare three different box encodings in experiments Obj1, Obj4 and Obj5, summarized in Table 5.2. To investigate the influence of the *Biternion* encoding introduced in Section 5.3.3, we model the orientation as single variable within $-180°$ to $180°$ in the *Angle* encoding. Lastly, we evaluate the uncertainty encoding presented in Section 5.3.5 (abbreviated as *Uncertainty*).

Compared to Obj1 that implements the Biternion encoding, we observe that encoding the orientation as a single variable (Obj4) decreases the performance.

We also observe that additionally estimating the positional and rotational variance slightly increases the mATE and mASE but decreases the mAOE by

ID	Rand. Rot. in °	Encoding	IoU	Backbone	Fusion	mAP▲ in %	mATE▲ in cm	mASE▲ in %	mAOE▲ in °
Obj1	0	Biternion	Aligned	ResNet-38	FPN-128	26.0	44.9	31.0	36.1
Obj2	±15	Biternion	Aligned	ResNet-38	FPN-128	26.8	44.5	30.6	34.4
Obj3	±180	Biternion	Aligned	ResNet-38	FPN-128	24.1	45.7	**30.5**	33.2
Obj4	0	Angle	Aligned	ResNet-38	FPN-128	24.2	47.1	31.2	39.0
Obj5	0	Uncertainty	Aligned	ResNet-38	FPN-128	25.6	45.0	35.6	**29.2**
Obj6	0	Biternion	Rotated	ResNet-38	FPN-128	**29.5**	**41.3**	30.9	33.1
Obj7	0	Biternion	Aligned	EfficientNet-B2	BiFPN-96	20.3	53.4	37.4	33.6

Table 5.2: Overview on all evaluated object detector configurations. The best results for each metric are highlighted in bold.

a large margin. The results of Obj5 indicate that especially the orientation estimation benefits from incorporating uncertainty measures.

Prior Boxes and IoU We compare the usage of rotated and axis-aligned prior boxes and IoU in experiments Obj1 and Obj6 in Table 5.2.

Using rotated prior boxes and IoU computation we observe an increased mAP, lower mATE and mAOE during evaluation. This may be explained by prior boxes that are in general more similar to the ground truth boxes.

Backbone and Fusion Experiments Obj1 and Obj7 in Table 5.2 compare the RetinaNet and EfficientDet architecture.

We could not observe increased performance for object detection in grid maps when using the EfficientDet model in contrast to object detection in camera images presented in related work [TPL20]. This, however, may be due to differing hyperparameters when dealing with camera images and grid maps.

5.4.2 Qualitative Results

Figure 5.7 visualizes the prior box matching during training. As we create rotated prior boxes, we are able to better fit differently rotated objects in the scene at the cost of more prior boxes in general (compared to axis-aligned prior boxes).

Figures 5.8 and 5.9 depict object detection results on scenes of the nuScenes validation set. We observe that larger objects such as cars, trucks and buses are detected at larger distances and a high recall, whereas small objects such as pedestrians or traffic cones need to be closer to the vehicle to be detected reliably.

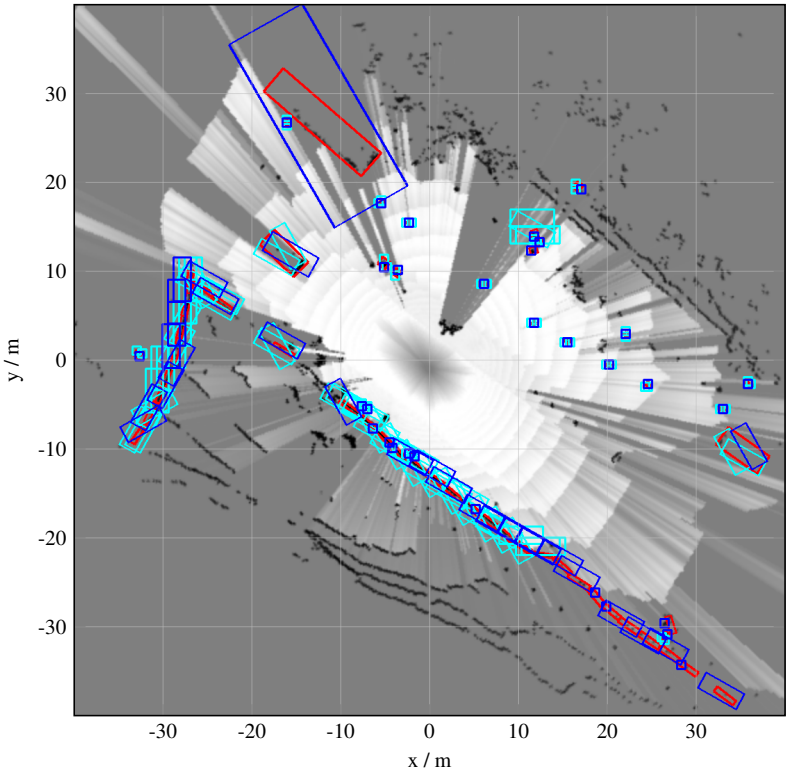


Figure 5.7: Prior box matching during training. For each ground truth box (red), we consider the prior box with the highest IoU (blue) as match. Prior boxes are created at different pyramid scales to account for different object sizes. The next best prior box candidates are depicted in light blue.

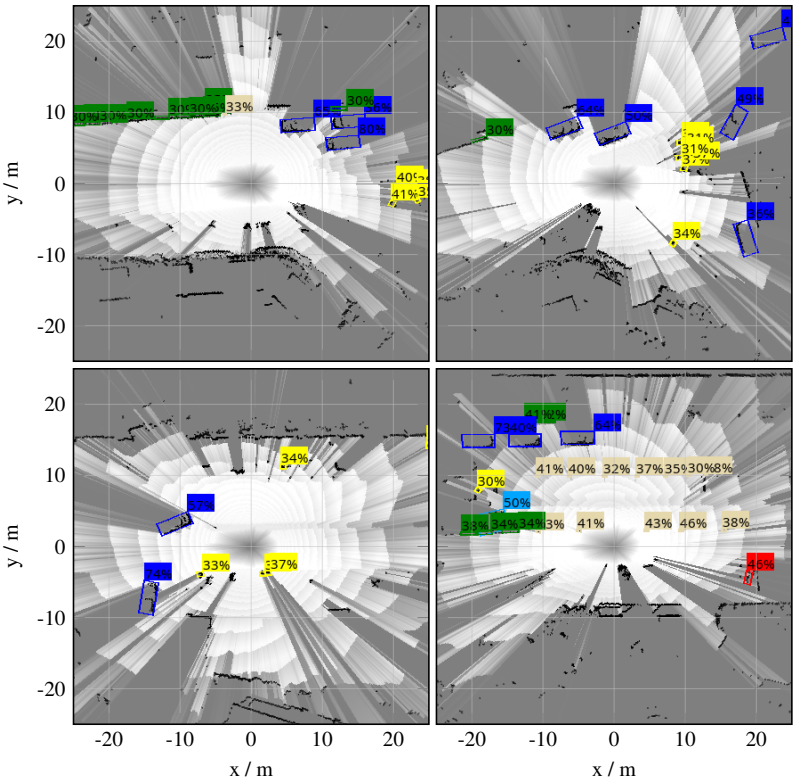


Figure 5.8: Detected objects with class likelihoods on top of the pignistic probability grid map layer in four exemplary scenes of the nuScenes validation set. The objects are colored by the most likely class. Blue: car, green: barrier, yellow: pedestrian, red: cyclist, beige: traffic cone.

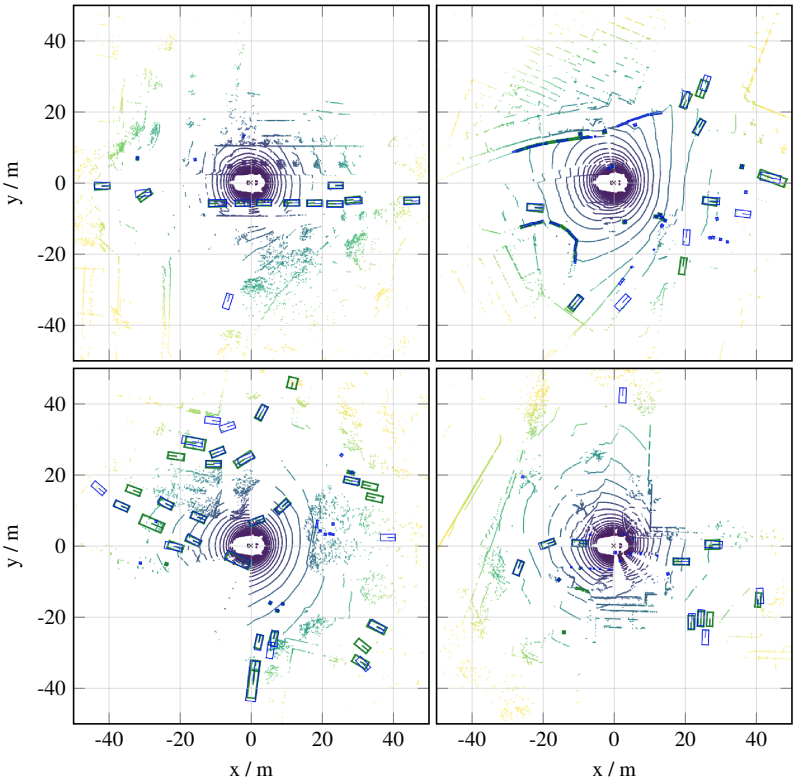


Figure 5.9: Detected objects (blue) and ground truth labels (green) on top of the sensor-relative height of range sensor reflections (blue: low, yellow: high) in four exemplary scenarios of the nuScenes validation set.

6 Self-Supervised Scene Flow Estimation

Scene flow may be defined as the 3D motion of object surfaces [Ved+05]. Then, optical flow can be interpreted as the projection of scene flow into the image plane. In top-view grid maps optical flow is the planar motion as a result of orthographically projected 3D motion of objects.

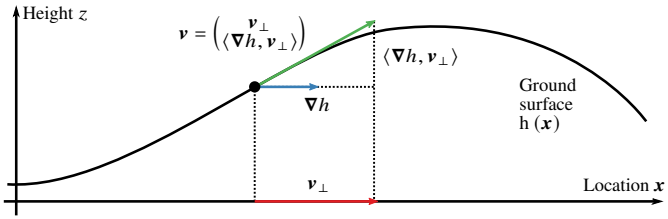


Figure 6.1: The 3D velocity $\boldsymbol{v}$ along a surface can be determined by the surface gradient ∇h and the planar velocity $\boldsymbol{v}_\perp$.

We assume that all traffic participants move on a common, known ground surface. Knowing this ground surface and the optical flow as a result of the orthographic plane projection, we are able to reconstruct the scene flow (cf. Fig. 6.1). Therefore, we divide the problem of scene flow estimation into two independent problems, namely ground surface estimation (presented in Chapter 3) and optical flow estimation, which we discuss in the following.

Scene flow and odometry estimation are closely related. Given scene flow estimates between two view points in a static environment, one can estimate the motion between these view points. Here, optical flow estimation methods establish feature correspondences on which odometry estimation methods operate on.

After a review of related work on scene flow estimation (Section 6.1), we present our optical flow estimation model on grid maps in Section 6.2, our approach for odometry estimation (Section 6.3) and conduct experiments in Section 6.4. The presented optical flow estimation method is a refined version of [Wir+19a].

6.1 Related Work

We provide an overview about related work on optical flow and odometry estimation (Sections 6.1.1 and 6.1.2). Finally, we present recent approaches for dynamic occupancy grid map estimation.

6.1.1 Optical Flow Estimation

Optical flow estimation is fundamental for camera-based motion estimation algorithms and has been studied extensively since the beginning of computer vision (cf. [HS81]).

In contrast to sparse optical flow estimation, which establishes correspondences between a few, well-recognizable key points, dense optical flow estimation aims to establish correspondences between all available measurements. Traditionally, these approaches suffer from low estimation accuracy in weakly-structured image regions. However, increased accuracy can be achieved by learning deep convolutional models in a supervised manner.

Dosovitskiy et al. [Dos+15] and Ilg et al. [Ilg+17] develop fully-convolutional encoder-decoder models which are applied sequentially to estimate optical flow at different magnitudes. They start with a model to estimate a flow field for large displacements, warp one image into the frame of the other and use the result as input for refined flow estimation. Depending on the application, this step is repeated several times in order to produce an accurate flow estimate. Although their models yield accurate results, the overall training strategy is difficult. For example, due to high memory consumption the different modules are trained separately.

Sun et al. [Sun+18] present a pyramidal architecture that produces competitive results while drastically reducing memory consumption. Their *PWCNet* uses

feature pyramids that are subsequently warped in order to compute a cost volume for a particular scale which is then used as input for the flow estimation model (cf. Fig. 6.2). Therefore, the estimated flow from the next (lower resolution) pyramid scale is upsampled by convolutional units and then used to warp a feature volume from one frame into the other using bilinear interpolation. The cost volume layer implements a normalized cross-correlation

$$\mathrm{c}(\boldsymbol{x}, \Delta\boldsymbol{x}) = \frac{1}{N_\mathrm{F}} \langle \mathbf{I}_1(\boldsymbol{x}), \mathbf{I}_2(\boldsymbol{x} + \Delta\boldsymbol{x}) \rangle, \tag{6.1}$$

between the two image features $\mathbf{I}_1, \mathbf{I}_2 \colon \mathcal{I} \to \mathbb{R}^{N_\mathrm{F}}$. As the correlation is computed for every scale, the maximum displacement $d = \|\Delta\boldsymbol{x}\|_1$ can be chosen small because a displacement of $d^{(s)} = 1$ at feature scale s corresponds to a displacement of $d^{(0)} = 2^s$ at feature scale 0, i.e. the input resolution.

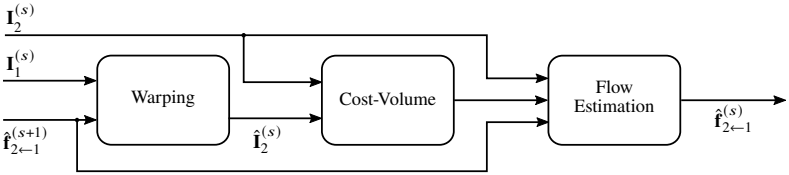


Figure 6.2: Overview of a PWCNet forward flow estimation unit at pyramid scale s (likewise for backward flow). The inputs are the image features $\mathbf{I}_1^{(s)}$ and $\mathbf{I}_2^{(s)}$ for two different frames and the estimated flow $\hat{\mathbf{f}}_{2\leftarrow1}^{(s+1)}$ from the next (lower resolution) pyramid scale. Then a warped feature volume $\hat{\mathbf{I}}_2^{(s)}$ is determined and used to compute the cross correlation between original and warped features. Finally, a refined flow $\hat{\mathbf{f}}_{2\leftarrow1}^{(s)}$ is determined by applying convolutional layers on the feature volume $\mathbf{I}_2^{(s)}$, the cost-volume and the input flow estimate $\hat{\mathbf{f}}_{2\leftarrow1}^{(s+1)}$.

By assuming photo-consistency

$$\mathbf{I}_2(\boldsymbol{x}) \overset{!}{=} \mathbf{I}_1(\mathbf{f}_{2\leftarrow1}(\boldsymbol{x})) \tag{6.2}$$

$$\mathbf{I}_1(\boldsymbol{x}) \overset{!}{=} \mathbf{I}_2(\mathbf{f}_{1\leftarrow2}(\boldsymbol{x})) \tag{6.3}$$

between corresponding pixels in subsequent frames, Meister et al. [MHR18] train the architecture proposed by Ilg et al. [Ilg+17] in a self-supervised fashion, i.e. without using manually annotated data.

6.1.2 Odometry Estimation

Optical flow estimation may be used to estimate odometry, i.e. the pose change over time w.r.t a fixed coordinate frame. Usually, a static scene is assumed so that every feature only changes according to ego-motion. However, due to other moving traffic participants in the scene and noisy flow estimates robust optimization methods are required that suppress the influence of outliers, i.e. features that do not change according to ego-motion. Additional constraints, such as motion on a manifold may further increase the estimation robustness and accuracy.

Given a set of 3D point correspondences, Horn [Hor87] introduces a closed-form solution to estimate transformations $\mathbf{T} \in \mathrm{SE}_3$ which minimize the sum of squared distances. First, he determines the translational component of $\mathbf{T}$ by estimating the average translation of all correspondences. To estimate the rotational component, Horn suggests an orthonormal matrix decomposition that effectively maximizes the similarity between transformed correspondences. Weighted Least-Squares (WLS) approaches to estimate rigid-body transformations exist (cf. Appendix A.11); However, these approaches expect a priori correspondence weights.

A robust iterative approach for parameter optimization was presented by Yang et al. [Yan+20] (cf. Section 2.2.3). Based on the Black-Rangarajan duality and graduated non-convexity the authors develop an alternating optimization scheme with an inner and an outer loop. In the inner loop, model parameters and measurement weights are estimated in an alternating fashion. Starting from a fully convex cost function, non-convexity is gradually increased in the outer loop until a stopping criterion is met.

6.1.3 State Estimation in Occupancy Grid Maps

In presence of dynamically moving objects, the occupancy grid cell state is often represented by occupancy and velocity information which is estimated by recursive filters. Usually, the cell state is approximated by particles which are created in areas of observed reflections and propagated by a constant velocity motion model.

Nuss et al. [Nus+18] represent traffic participants by several point objects (the grid cell centers) and model the relationship between point objects to real objects by random finite sets (RFSs). The point objects are predicted independently and then jointly combined in order to determine a posterior distribution. The authors approximate the distributions by particles and present a real-time approximation based on evidence theory.

Assuming known poses, Steyer et al. [STW18] model hypotheses for static, dynamic occupancy, free-space and their combined hypotheses cell-wise using evidence theory. The authors then apply a particle filter in areas of high evidence for dynamically moving objects, reducing the amount of particles needed. In contrast, corresponding static cells at different time steps are accumulated directly in the grid map based on past cell measurements.

Although the above-mentioned recursive approaches make assumptions similar to photo-consistency (Eqs. (6.2) and (6.3)) for generation and resampling of particles, the approaches expose low-pass behavior due to slower convergence as the underlying distribution is not known a priori. Thus, these methods need time to converge across multiple time steps. In addition, a state estimation of every cell requires a large amount of particles which results in large computational cost.

6.2 Optical Flow Estimation in Grid Maps

The recursive state estimation approaches discussed in Section 6.1.3 expose drawbacks that we aim to address with our approach. Our idea is that patterns across multiple grid cells yield strong features for matching across subsequent frames which are not considered in recursive, particle-based approaches. In addition, we drop the assumption of a constant-velocity motion model and estimate motion between two subsequent frames as a dense flow field, which is usually not done in recursive state estimation approaches due to the large computational cost of particle storage, update and resampling.

The estimated grid cell velocities may then be used to determine the average velocity of detected extended objects (cf. Chapter 5).

6.2.1 Model Structure

Our model for optical flow estimation in grid maps is based on the concepts presented in Unflow [MHR18] and PWCNet [Sun+18] and an extension of the model presented by Wirges et al. [Wir+19a].

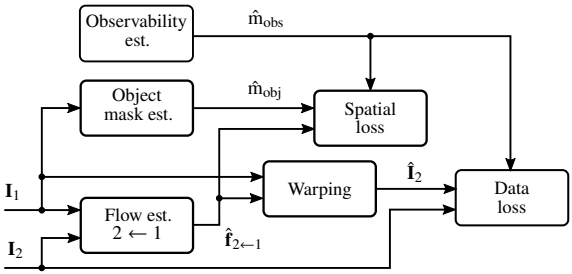


Figure 6.3: Overview of our training strategy for self-supervised flow estimation. Our goal is to estimate the flow $\hat{\mathbf{f}}_{2\leftarrow 1}$ from two subsequent measurements $\mathbf{I}_1$ and $\mathbf{I}_2$ that transforms coordinates from frame 1 into frame 2. During training, we determine the data loss between the warped image $\hat{\mathbf{I}}_2$ and the image $\mathbf{I}_2$, scaled by the observability $\hat{\mathrm{m}}_{\mathrm{obs}}$. In addition, we compute a smoothness loss to ensure smooth scene flow except at object boundaries $\hat{\mathrm{m}}_{\mathrm{obj}}$. Note that the temporal backward direction $1 \leftarrow 2$ and the cycle loss are omitted here for simplicity.

Figure 6.3 depicts the signal flow in our model during training for the forward direction, i.e. the flow $\hat{\mathbf{f}}_{2\leftarrow 1}$ from frame 1 to frame 2. The backward flow direction is computed likewise and has an equal share of the training objective.

Before flow estimation, image features $\mathbf{I}_1^{(s)}$ and $\mathbf{I}_2^{(s)}$ at different scales are computed by the backbone and the feature fusion presented in Section 5.3 for each input grid map at frame 1 and 2, respectively. These features are then fed into a PWCNet [Sun+18] yielding the forward flow estimate $\hat{\mathbf{f}}_{2\leftarrow 1}$ at the input scale which is used to warp grid map 1 into frame 2 resulting in $\hat{\mathbf{I}}_2$. The system output consists of the estimated forward flow $\hat{\mathbf{f}}_{2\leftarrow 1}$ and a rigid body transformation $\hat{\mathbf{T}}_{2\leftarrow 1}$ between the two frames that is estimated from the forward flow.

6.2.2 Objectives

Our self-supervised training strategy relies on the assumption of temporal feature persistence, i.e. we assume that features do not change drastically between two subsequent frames. As an extension of the work by Wirges et al. [Wir+19a], we incorporate observability information directly into the training process which is obtained by the grid mapping process presented in Chapter 4.

Given two subsequent frames 1 and 2, the estimated image coordinate

$$\hat{\boldsymbol{x}}_2 = \hat{\mathbf{f}}_{2\leftarrow 1}(\boldsymbol{x}_1) \tag{6.4}$$

in frame 2 corresponds to the image coordinate in frame 1 transformed by the estimated forward flow $\hat{\mathbf{f}}_{2\leftarrow 1}$. Here, we create coordinate-wise loss terms

$$\mathrm{L}_{2\leftarrow 1}(\boldsymbol{x}_1) = \mathrm{L}_{\mathrm{data},2\leftarrow 1}(\boldsymbol{x}_1) + \mathrm{L}_{\mathrm{spatial},2\leftarrow 1}(\boldsymbol{x}_1) + \mathrm{L}_{\mathrm{cycle},2\leftarrow 1}(\boldsymbol{x}_1) \tag{6.5}$$

to penalize data, spatial and cycle inconsistencies, which we introduce in the following. Likewise, we add these loss terms in temporal backward direction (frame 2 to frame 1) such that the final loss

$$\mathrm{L} = \sum_{\boldsymbol{x}_1 \in \mathcal{I}_1} \mathrm{L}_{2\leftarrow 1}(\boldsymbol{x}_1) + \sum_{\boldsymbol{x}_2 \in \mathcal{I}_2} \mathrm{L}_{1\leftarrow 2}(\boldsymbol{x}_2) \tag{6.6}$$

summarizes all losses in temporal forward and backward direction on valid image coordinates $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively.

Data Consistency By claiming data consistency between two corresponding cells, we assume that the residual

$$\mathbf{r}_{\mathrm{data},2\leftarrow 1}(\boldsymbol{x}_1) = \mathbf{I}_1(\boldsymbol{x}_1) - \mathbf{I}_2(\hat{\boldsymbol{x}}_2), \tag{6.7}$$

which models the difference between the cell value in frame 1 and the corresponding value at the transformed cell coordinate in frame 2, is small.

This assumption, however, is invalid if obstacles enter or leave the field of view (FOV) or in areas unobservable in any of the frames. Therefore, we scale the data loss by the likelihood

$$\begin{aligned}\hat{\mathrm{m}}_{\mathrm{obs}}(\boldsymbol{x}_1) &= \big(1 - \mathrm{bel}_{\Omega,1}(\boldsymbol{x}_1)\big)\big(1 - \mathrm{bel}_{\Omega,2}(\hat{\boldsymbol{x}}_2)\big) \\ &= \big(\mathrm{bel}_{\{\mathrm{o}\},1}(\boldsymbol{x}_1) + \mathrm{bel}_{\{\mathrm{f}\},1}(\boldsymbol{x}_1)\big)\big(\mathrm{bel}_{\{\mathrm{o}\},2}(\hat{\boldsymbol{x}}_2) + \mathrm{bel}_{\{\mathrm{f}\},2}(\hat{\boldsymbol{x}}_2)\big)\end{aligned} \tag{6.8}$$

that a cell is observable in both frames. Here, we write $\mathrm{bel}_{\omega,t}$ for notational simplicity as a 2D signal for proposition ω at frame t. Compared to related work on self-supervised flow estimation (e.g. [Wir+19a; MHR18]) we do not transform these observabilities which removes the feedback of the estimated flow on its loss scaling, leading to an improved training convergence at the cost of a lower total number of samples.

Then, we define the coordinate-wise loss function

$$\mathrm{L}_{\mathrm{data},2\leftarrow1}(\boldsymbol{x}_1) = \hat{\mathrm{m}}_{\mathrm{obs}}(\boldsymbol{x}_1)\ \rho\big(\|\mathbf{r}_{\mathrm{data},2\leftarrow1}(\boldsymbol{x}_1)\|\big) \tag{6.9}$$

where we chose the robustifier ρ as the generalized Charbonnier loss $C_{0.45,0.1}$ (cf. Appendix A.2) as Sun et al. [SRB14] found it to be optimal for flow estimation problems. Note that we do not need any further regularizer to avoid trivial solutions here because the estimated flow has no impact on the observability mask.

Spatial Consistency We assume that the environment is composed of rigid objects. Furthermore, we assume that the optical flow variance on a moving rigid object is small. This assumption is, however, violated in the case of instant centers of rotation being close to an object, especially for pedestrians which might turn in place. On the one hand, we are not able to identify trackable semantic features on small objects such as pedestrians due to the grid cell resolution (typically between 9 to 16 cells) in any case. On the other hand, cyclist and cars usually turn with a center of rotation far away so that the violation does not have a significant impact. With this in mind, we assume the spatial flow gradient

$$\mathbf{r}_{\mathrm{spatial},2\leftarrow1}(\boldsymbol{x}_1) = \boldsymbol{\nabla}\hat{\mathbf{f}}_{2\leftarrow1}(\boldsymbol{x}_1) \tag{6.10}$$

to be small for rigid objects. However, at object boundaries the flow might change drastically. To model this effect, we compute a spatial mask

$$\mathrm{m}_{\mathrm{obj},1}(\boldsymbol{x}_1) = 1 - \lambda \|\nabla \mathrm{bel}_{\mathrm{o}}(\boldsymbol{x}_1)\|^2 \|\nabla \mathrm{bel}_{\mathrm{f}}(\boldsymbol{x}_1)\|^2 \tag{6.11}$$

that maps to low values at rapid changes of either the occupied or the free belief, which are modeled as 2D signals $\mathrm{bel}_{\mathrm{o}}, \mathrm{bel}_{\mathrm{f}} \colon \mathbb{Z}^2 \to [0, 1]$. Here, λ denotes a normalization constant so that the spatial mask is non-negative at all times and $\nabla \mathrm{bel}(\cdot)$ denote the normalized Sobel derivatives of the belief maps.

Thus, the loss

$$\mathrm{L}_{\mathrm{spatial},2\leftarrow 1}(\boldsymbol{x}_1) = \mathrm{m}_{\mathrm{obj},1}(\boldsymbol{x}_1)\ \rho\big(\|\mathbf{r}_{\mathrm{spatial},2\leftarrow 1}(\boldsymbol{x}_1)\|\big) \tag{6.12}$$

assures spatial flow consistency w.r.t. the assumptions made.

Cycle Consistency Forward and backward flow should compensate at corresponding positions. Therefore, we model the residual

$$\mathbf{r}_{\mathrm{cycle},2\leftarrow 1}(\boldsymbol{x}_1) = \hat{\mathbf{f}}_{2\leftarrow 1}(\boldsymbol{x}_1) + \hat{\mathbf{f}}_{1\leftarrow 2}(\hat{\boldsymbol{x}}_2) \tag{6.13}$$

and define the loss

$$\mathrm{L}_{\mathrm{cycle},2\leftarrow 1}(\boldsymbol{x}_1) = \hat{\mathrm{m}}_{\mathrm{obs}}(\boldsymbol{x}_1)\ \rho\big(\|\mathbf{r}_{\mathrm{cycle},2\leftarrow 1}(\boldsymbol{x}_1)\|\big)\ , \tag{6.14}$$

that penalizes cycle inconsistencies. Note that cycle inconsistencies are only penalized in areas observable in both frames (cf. Eq. (6.8)).

6.2.3 Receptive Field

In the nuScenes data set all translations and rotations between two subsequent frames are below 2.5 m and 5°, respectively. With a grid map size of 60 m×60 m and a cell size of 12.5 cm we then determine the receptive field size as 135 cells in width and height to fully cover these transformations. Knowing the required receptive field size, we remove the last two layers of the PWCNet model and thus reduce the model size by 40 % compared to the original architecture. We call this model *PWCNet-small*.

6.3 Odometry Estimation

As the sensor platform moves, the scene flow of static environment is consistent to this motion. Here, we estimate the sensor motion $\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}$ between two frames from the estimated observable forward flow $\hat{\mathbf{f}}_{2\leftarrow 1}$.

To estimate the sensor motion in presence of other moving traffic participants, we use the robust optimization method presented by Yang et al. [Yan+20] (cf. Appendix A.3) to the point registration problem (cf. Appendix A.11). The set of weighted point correspondences $\left(\boldsymbol{x}_{1,n}, \hat{\boldsymbol{x}}_{2,n}, w_n\right)$ consists of original grid cell positions $\boldsymbol{x}_{1,n} \in \mathcal{X}_1$, new positions $\hat{\boldsymbol{x}}_{2,n} = \hat{\mathbf{f}}_{2\leftarrow 1}\left(\boldsymbol{x}_{1,n}\right)$ and weights w_n which are not known a priori and initialized with $\frac{1}{|O_1|}$. Here, O_1 denotes the set of grid cells with non-zero observability that are also observable in the other frame.

In the inner loop, a WLS point registration (cf. Appendix A.11) to estimate the optimal rotation and translation

$$\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}} = \underset{\boldsymbol{R}\in \mathrm{SO}_2, \boldsymbol{t}}{\arg\min} \sum_{n=1}^{N} \hat{w}_n \| \underbrace{\boldsymbol{R}\boldsymbol{x}_{1,n} + \boldsymbol{t} - \hat{\boldsymbol{x}}_{2,n}}_{\mathbf{r}_n(\boldsymbol{R},\boldsymbol{t})} \|^2 \tag{6.15}$$

w.r.t. fixed weights w_n and a weight estimation is executed in an alternating fashion. Using the Geman McClure (GMC) loss

$$\rho(x) = \frac{\alpha^2 x^2}{\alpha^2 + x^2} \tag{6.16}$$

with the free parameter α, the weights

$$\hat{w}_n = \left(\frac{\mu\alpha^2}{\|\mathbf{r}_n\|^2 + \mu\alpha^2} \right)^2 \tag{6.17}$$

can be determined based on the squared residual norm $\|\mathbf{r}_n\|^2$ and the parameter μ which controls the degree of non-convexity. After convergence of the inner loop, μ is updated towards a more non-convex overall loss function in the outer loop and then the inner loop is repeated until a convergence criterion is met. Here, we repeat these steps for a constant number of iterations.

Parameter	Value
Backbone	Image pyramid
Feature fusion	FPN (64 filters)
Convolutions	Separable 3×3 (cf. Example 2.6)
Weight decay	1×10^{-4} (L2) (FPN only)
Scales	2 to 5
Flow estimation head	PWCNet-small
Data augmentation	Rotation (±5°), translation (±1 m)
GNC optimizer iterations	5

Table 6.1: Default scene flow estimation model parameters used in evaluation.

Given an estimate $\hat{\boldsymbol{R}}_{2\leftarrow1}, \hat{\boldsymbol{t}}_{2\leftarrow1}$ of the sensor motion we can then generate the corresponding motion flow field

$$\hat{\mathbf{f}}_{\text{motion},2\leftarrow1}(\boldsymbol{x}_1) = \hat{\boldsymbol{R}}_{2\leftarrow1}\boldsymbol{x}_1 + \hat{\boldsymbol{t}}_{2\leftarrow1} \tag{6.18}$$

assuming static environment. Finally, we obtain motion-compensated object flow

$$\hat{\mathbf{f}}_{\text{object},2\leftarrow1}(\boldsymbol{x}_1) = \hat{\mathbf{f}}_{2\leftarrow1}(\boldsymbol{x}_1) - \hat{\mathbf{f}}_{\text{motion},2\leftarrow1}(\boldsymbol{x}_1) \tag{6.19}$$

in the fixed odometry frame by subtracting the motion from the scene flow.

6.4 Experiments

If not specified otherwise, we train all models with the parameters summarized in Table 6.1. Here, we replace the backbone by a simple image pyramid and use a Feature Pyramid Network (FPN) with 64 filters on each scale for feature fusion, reducing the total number of trainable parameters to around 916 000. This step enables fast training convergence and low GPU memory consumption. We randomly rotate one of the two grid map inputs within ±5° and add a random translation within ±1 m. To reduce overfitting, we use L2 weight decay (cf. Example 2.7), except in the flow estimation head. We choose the PWCNet-small configuration (cf. Section 6.2.3) as model head which is reduced in size.

Due to memory restrictions, all models are trained with a batch size of 4 on two *Nvidia GeForce RTX 2080 Ti* GPUs using the ADAM optimizer

(cf. Example 2.3) with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. After each nonlinearity, batch normalization (BN) (cf. Section 2.4.2) is performed. We use a linear learning rate warm-up from 1×10^{-5} to 1×10^{-4} within 3000 iterations and terminate the optimization after 300 000 iterations at a learning rate of 1×10^{-5} using a cosine learning rate schedule.

6.4.1 Quantitative Evaluation

Table 6.2 summarizes the evaluation results for different scene flow estimator configurations based on the metrics introduced in the following.

Metrics As there is no densely labeled scene flow available in any data sets, we evaluate the mean average velocity error (mAVE) of the nuScenes object detection benchmark ([Cae+20], cf. Appendix A.10.1) and introduce custom metrics to compare the estimated odometry to the odometry provided in nuScenes. Note that for velocity computation, we average the object flow along the area of ground-truth boxes as there are no object detections available at this point.

The average odometry translation error

$$\text{AOTE} = \frac{1}{N} \sum_{n=1}^{N} \frac{\|\hat{\boldsymbol{t}}_n - \boldsymbol{t}_n\|}{\|\boldsymbol{t}_n\|} \tag{6.20}$$

in % denotes the average relative translation error between all frames of a sequence. The average odometry rotation error

$$\text{AORE} = \frac{1}{N} \sum_{n=1}^{N} \frac{|\hat{\phi}_n - \phi_n|}{\|\boldsymbol{t}_n\|} \tag{6.21}$$

in °/m denotes the average rotational error normalized by the translation between all frames of a sequence. We then determine mean average odometry translation error (mAOTE) and mean average odometry rotation error (mAORE) by averaging AOTE and AORE across all sequences.

ID	HighRes	Observability	Deep	mAVE▲ in m/s	mAOTE▲ in %	mAORE▲ in °/m
Flow1		✓		0.71	16.91	0.16
Flow2	✓	✓		**0.68**	**16.29**	0.16
Flow3				0.95	83.36	0.41
Flow4		✓	✓	0.69	16.58	**0.14**

Table 6.2: Overview on all evaluated scene flow estimator configurations. The best results for each metric are highlighted in bold.

High Resolution Model We compare the influence of the data loss computation at different pyramid scales on the scene flow estimation performance. In our baseline experiment Flow1, we compute the losses on all pyramid scales (2 to 5), whereas in experiment Flow2, we additionally estimate the flow on scale 1. We perform bilinear interpolation (cf. [Sun+18]) for upsampling to the original resolution in both experiments. The evaluation results for Flow1 and Flow2 are summarized in Table 6.2. We observe that additionally computing the data loss on pyramid scale 1 (experiment Flow2) decreases the mAVE and mAOTE compared to experiment Flow1. However, as more convolutions are involved, the parameter size and inference time increases.

Observability Scaling Experiments Flow1 and Flow3 (Table 6.2) investigate the effect on the scene flow estimation performance of the observability scaling. In experiment Flow3, we scale all cell-wise losses equally which results in high errors. In contrast, in experiment Flow1 we only compute losses if the observability is above the threshold of 5 %. We observe that this step improves the results by a large margin and believe that this is due to the suppression of weakly observable, possibly noisy areas.

Deep Backbone Experiments Flow1 and Flow4 compare the effect of adding a deeper ResNet-38 backbone on the scene flow estimation performance. Although the depth (and with it the number of parameters) drastically increases (by a factor of approximately 13), we observe only a minor performance improvement (in contrast to related work on flow estimation on camera images [Ilg+17; Sun+18]). Compared to optical flow estimation in camera images this may be due to the scale invariance and simpler geometric primitives in grid maps.

Thus, we believe that fewer parameters are needed to describe grid map features and that we appropriately modeled the problem.

6.4.2 Qualitative Results

Figure 6.4 illustrates the scene flow estimation process during training. The occupied beliefs of two frames in the top left corner illustrate the data augmentation in form of random rigid-body transformations applied between the two frames. We observe that the scene flow field (top right) accurately aligns the two frames, illustrated by the transformed occupied and free beliefs (center). Here, the observability mask (bottom left) assures that the data consistency loss is only evaluated in observable areas and that the flow is nearly constant within rigid objects, which is controlled by the spatial mask (bottom right).

The scene flow estimation process during inference is depicted in Fig. 6.5. Starting from subsequent grid maps (visualized by occupied beliefs in the top left corner), we estimate the observable scene flow (top right, visualized only on occupied space). We then determine the rigid-body transformation between the two frames due to motion from the observable scene flow. Given this transform, we can then compute the *motion flow*, i.e. the expected flow of static environment due to ego motion (center left). Here, we observe an instant center of rotation which is equivalent to vanishing motion flow at a distance of around 40 m from the sensor. Finally, we obtain *object flow*, i.e. the ego motion-compensated flow field of moving objects (center right) by subtracting motion flow from scene flow. Note that moving vehicles in this scene expose a non-zero object flow, highlighted by the gray boxes. As a byproduct, the sensor odometry can be obtained by concatenating the subsequent inter-frame transformations (bottom). The point in the odometry plot denotes the pose at which the grid maps above were extracted.

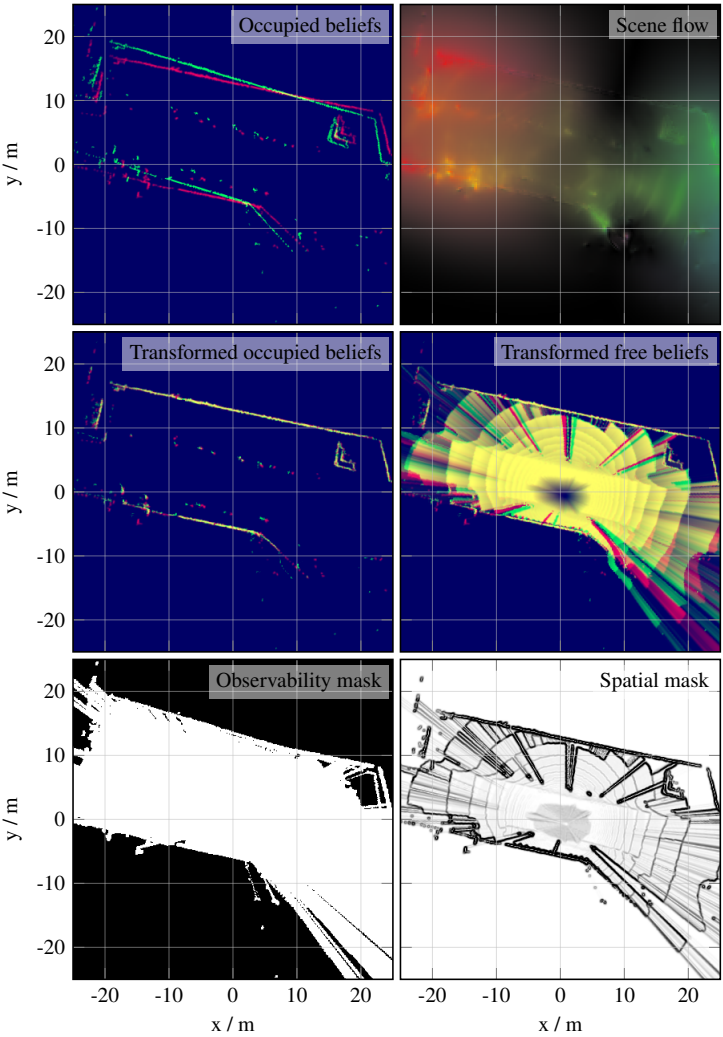


Figure 6.4: During training, we aim to estimate a scene flow field (top right, HSV color encoding) that transforms features between two subsequent frames, e.g. the occupied beliefs (top left), resulting in aligned features, e.g. the occupied or free beliefs (center). This goal is expressed by a data-consistency loss, weighted by an observability mask (bottom left). To impose constant flow on rigid objects, we penalize flow changes weighted by a spatial mask (bottom right) which describes object- or free-space boundary likelihoods.

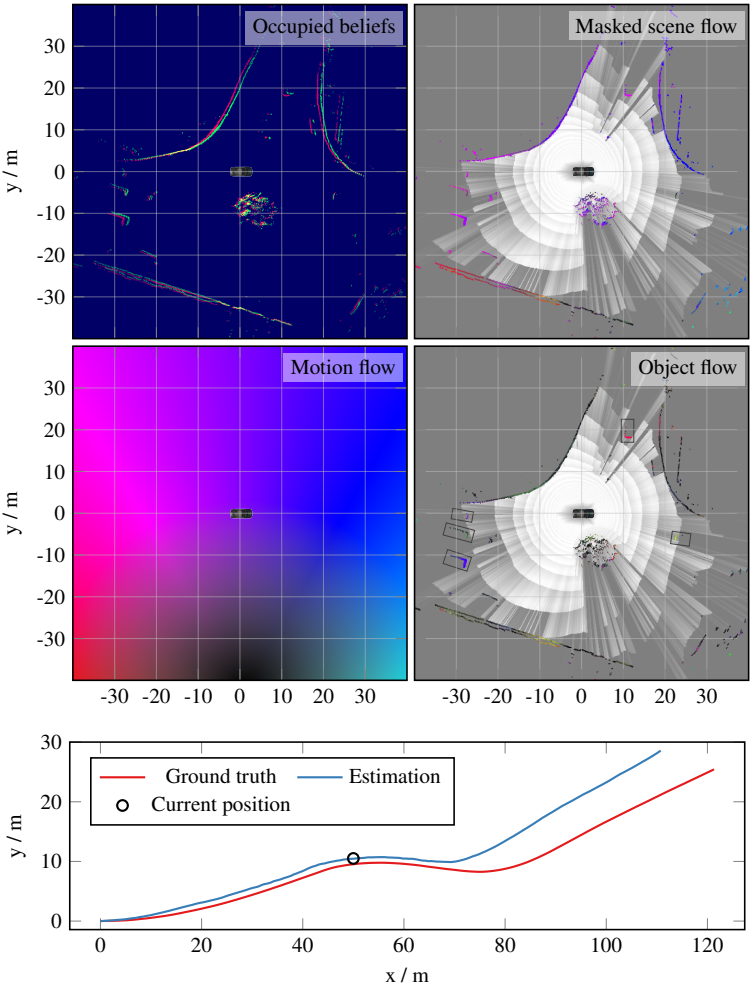


Figure 6.5: Scene flow estimation in an exemplary right-hand bend scenario. We estimate scene flow based on grid map features from two subsequent frames (top left). To estimate the transform due to motion we mask the scene flow field using the occupied belief layer (top right). Given this transform we can determine dense motion flow (middle left). Note that the instant center of rotation is visualized as vanishing motion flow at around $y = -40\,\mathrm{m}$. Finally, we obtain object flow by subtracting motion flow from scene flow (middle right). One can observe the non-zero flow of moving vehicles (black rectangles). Estimated transforms between subsequent frames resemble the sensor odometry which we compare to the nuScenes ground-truth trajectory (bottom).

7 Joint Object Detection, Scene Flow Estimation & Tracking

Multi-task learning leverages the idea of improving in solving a task by training a model to additionally solve another task [Zam+18]. Here, we aim to improve the object detection task by also training on the scene flow estimation task in a self-supervised fashion. This can be accomplished without the need of manually annotated labels, thus at no additional labeling cost. By enforcing model parameter sharing, image features can be computed once in a common model and then reused for different tasks which reduces computational effort and memory consumption.

In the following, we combine the approaches presented in Chapters 5 and 6 and extend our model by the estimation of features to track objects across subsequent frames. Here, we define tracking as the reidentification of the same object across subsequent frames[1] in presence of occlusions, noise, falsely detected or missed objects. Although there are many object tracking approaches specialized on occupancy grid maps (e.g. [Nus+18; Ste+20]), we focus on the tracking of arbitrary features here, which makes the approach more flexible.

After summarizing the basic concepts of feature aggregation and embedding learning in Section 7.1 we introduce our system for joint estimation in Section 7.2. Finally, we perform a quantitative evaluation in Section 7.3 based on the nuScenes data set.

[1] Contrary to research groups who also consider motion estimation to be part of tracking.

7.1 Related Work

Section 7.1.1 presents an approach to dynamically adjust the influence of different tasks during the training of multi-task models. A method for smoothing of corresponding features across two frames is introduced in Section 7.1.2.

7.1.1 Multi-Task Learning

Usually, the overall loss function

$$L = \sum_{n=1}^{N} w_n L_n \tag{7.1}$$

is modeled as a linear combination of all task-specific losses. However, different task-specific loss functions may map to different value ranges and thus have different impact on the optimization. Individually tuning the task weights w_n can be very expensive if there are many task objectives. To resolve this issue, Kendall et al. [KGC18] present an automatic task weighting based on the estimated homoscedastic aleatoric uncertainty of each task which can be interpreted as the task-dependent uncertainty.

In the regression task the authors assume the model output $\hat{\boldsymbol{y}} \sim \mathcal{N}\big(\mathbf{f}(\boldsymbol{x}), \sigma_{\text{reg}}\big)$ to follow a normal distribution. Maximizing the log likelihood results in the optimal regression loss

$$L_{\text{reg}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\|\boldsymbol{y} - \mathbf{f}(\boldsymbol{x})\|^2}{2\sigma_{\text{reg}}^2} + \log\big(\sigma_{\text{reg}}\big). \tag{7.2}$$

Classification outputs $\hat{\boldsymbol{y}} \sim \text{Softmax}\big(\sigma_{\text{class}}^{-2}\mathbf{f}(\boldsymbol{x})\big)$ are assumed to follow a Gibbs distribution. This results in the optimal classification loss

$$L_{\text{class}}(\boldsymbol{x}, \boldsymbol{y}) = -\frac{\log(\text{Softmax}(\boldsymbol{y}, \mathbf{f}(\boldsymbol{x})))}{\sigma_{\text{class}}^2} + \log(\sigma_{\text{class}}). \tag{7.3}$$

During training, task-uncertainties σ_n are optimized together with the model parameters in order to minimize the overall loss. Here, the additional $\log(\sigma_n)$

terms keep the uncertainties from becoming too large. In practice, the model is designed to predict the log variance $s = \log(\sigma^2)$ to improve numeric stability.

Kendall et al. [KGC18] assume different model outputs to be independent of each other. Although this is not true in general, their approach increases the performance for all tasks without the need for manually tuning the weights w_n which are usually set to 1.

7.1.2 Feature Aggregation

Given a flow field, Zhu et al. [Zhu+17b; Zhu+17a] determine aggregated features

$$\boldsymbol{f}^*_{t_0} = \sum_{t \in \mathcal{T}} \mathrm{w}\big(\boldsymbol{f}_{t_0}, \boldsymbol{f}_{t_0 \leftarrow t}\big) \boldsymbol{f}_{t_0 \leftarrow t} \tag{7.4}$$

as a superposition of the current feature $\boldsymbol{f}_{t_0}$ with corresponding features $\boldsymbol{f}_{t_0 \leftarrow t}$ acquired at different frames $t \in \mathcal{T}$. The terms are weighted by the similarity function

$$\mathrm{w}(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(\frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|} \right) \tag{7.5}$$

based on the cosine similarity metric (cf. [Luo+18]) and normalized such that

$$\sum_{t \in \mathcal{T}} \mathrm{w}\big(\boldsymbol{f}_{t_0}, \boldsymbol{f}_{t_0 \leftarrow t}\big) = 1 \tag{7.6}$$

holds. Note that the evaluation of Eq. (7.4) may be computationally expensive as it requires features at all time steps. To reduce computational cost, Zhu et al. [Zhu+17b; Zhu+17a] propose to aggregate features

$$\bar{\boldsymbol{f}}_{t_k} = \underbrace{\mathrm{w}\big(\boldsymbol{f}_{t_k}, \boldsymbol{f}_{t_k \leftarrow t_{k-1}}\big)}_{w} \boldsymbol{f}_{t_k} + \underbrace{\big(1 - \mathrm{w}\big(\boldsymbol{f}_{t_k}, \boldsymbol{f}_{t_k \leftarrow t_{k-1}}\big)\big)}_{1-w} \bar{\boldsymbol{f}}_{t_k \leftarrow t_{k-1}} \tag{7.7}$$

during subsequent time steps t_k and t_{k-1} via exponential smoothing. This has the effect that features with high similarity between two frames are quickly accepted, whereas differing features are only slowly accepted.

7.2 System Overview

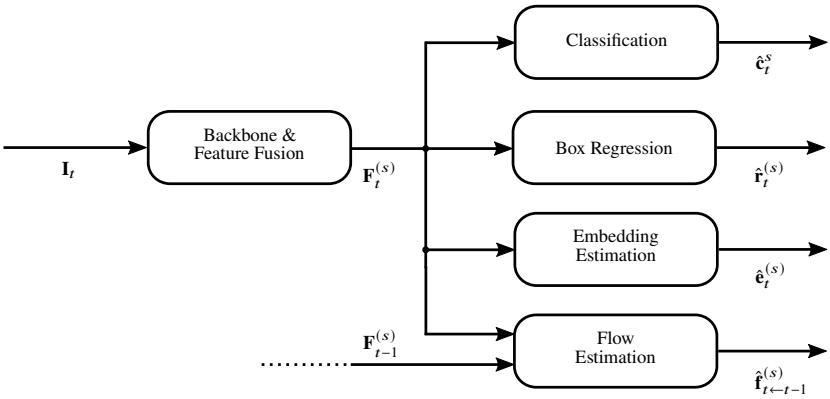


Figure 7.1: General structure of our multi-task model. The backbone and feature fusion (cf. Section 5.3) takes an input image $\mathbf{I}_t$ at time t and computes S common features $\mathbf{F}_t^{(s)}$ at different pyramid scales which are passed to classification, box regression, embedding and flow estimation. We add an estimation of object-specific feature embeddings $\mathbf{e}_t^{(s)}$ to track objects across subsequent frames. The flow estimation head additionally takes features $\mathbf{F}_{t-1}^{(s)}$ computed in the previous time step as input. Thus, the computationally expensive backbone and feature fusion have to be evaluated only once for each measurement.

Figure 7.1 provides an overview of our model which is based on the model presented by Fischer [Fis20]. For each input image $\mathbf{I}_t$ at time step t, we compute common image features $\mathbf{F}_t^{(s)}$ on S pyramid scales which are subsequently used as input for classification, box regression, association embedding and flow estimation. Classification and box regression are conducted as presented in Section 5.3.4 and Section 5.3.3, respectively. The embedding estimation head (presented in Section 7.2.1) aims to determine feature embeddings that can be used to reidentify objects across frames. The flow estimation head additionally takes the image features $\mathbf{F}_{t-1}^{(s)}$ computed in the previous time step as input in order to estimate a flow $\hat{\mathbf{f}}_{t\leftarrow t-1}^{(s)}$ transforming spatial features from time step $t-1$ to t.

7.2.1 Association Embedding

To track objects over time, we estimate an association embedding for every prior object. The embeddings of two different objects should be different, whereas two embeddings of the same object at different times should be similar. For two subsequent frames, this idea can be formalized by the batch-hard triplet loss

$$L_{\text{emb}} = \frac{1}{|\mathcal{P}^+|} \sum_{p \in \mathcal{P}^+} \max\left(\left\| \boldsymbol{e}_{2,p} - \boldsymbol{e}^+_{2\leftarrow 1,p} \right\|^2 - \left\| \boldsymbol{e}_{2,p} - \boldsymbol{e}^-_{2\leftarrow 1,p} \right\|^2 + \alpha, 0 \right) \quad (7.8)$$

as presented by Hermans et al. [HBL17]. For all positive priors $p \in \mathcal{P}^+_2$ in frame 2 with the feature embedding $\boldsymbol{e}_{2,p}$ we determine the corresponding feature embedding $\boldsymbol{e}^+_{2\leftarrow 1,p}$ in frame 1 and the *hardest* embedding $\boldsymbol{e}^-_{2\leftarrow 1,p}$ of all other prior objects in frame 1, i.e. the prior object

$$p^- = \underset{\tilde{p} \in \mathcal{P}^+_1 \setminus p}{\arg\min} \| \boldsymbol{e}_{2,p} - \boldsymbol{e}_{2\leftarrow 1,\tilde{p}} \|^2 \quad (7.9)$$

with the smallest distance to the feature embedding of prior object p. The parameter $\alpha \geq 0$ controls the influence of these hard negatives and is heuristically determined as $\alpha = 0.7$ in this work (cf. [Fis20]). The feature embedding dimension is set to 32.

At inference time, objects then can be tracked across subsequent frames by comparing their mutual association embedding distances (e.g. by the L2 norm).

7.2.2 Feature Aggregation

We implement a simplified flow-guided feature aggregation (cf. Section 7.1.2) to filter features and to increase the object detection performance. Due to the orthographic projection in grid maps, visual features of different objects do not overlap so that the main reason for rapid feature changes are occlusions, i.e. missing features. Therefore, we determine the weight

$$w = \text{bel}(\Omega) = 1 - \text{bel}(\{\text{o}\}) - \text{bel}(\{\text{f}\}) \quad (7.10)$$

based on the estimated observability (cf. Section 4.4) so that aggregated features

$$\bar{\boldsymbol{f}}_{t_k} = w\boldsymbol{f}_{t_k} + (1-w)\bar{\boldsymbol{f}}_{t_k \leftarrow t_{k-1}} \tag{7.11}$$

remain similar in areas of low observability and are updated quickly in case of high observability is high. Thus, compared to the approach presented by Zhu et al. [Zhu+17b; Zhu+17a] there is no additional feature transformation model required.

7.2.3 Loss Function

The total loss

$$L_{\text{MT}} = L^*_{\text{class}} + L^*_{\text{box}} + L^*_{\text{flow}} + \lambda_{\text{emb}} L_{\text{emb}} \tag{7.12}$$

denotes the sum of the object classification and box regression losses (cf. Eq. (5.12)), scene flow estimation (cf. Eq. (6.6)) and the embedding loss presented in Section 7.2.1. We employ uncertainty weighting as presented in Section 7.1.1 to each component (except embeddings) where the classification loss is adapted according to Eq. (7.3) and the box regression and flow losses are adapted as in Eq. (7.2).

7.2.4 Post-Processing

At inference time, we perform post-processing as in Section 5.3.7. Afterwards, we average the scene/object flow in the area an object occupies. This yields object velocities w.r.t. the vehicle/odometry frame.

7.3 Experiments

We conduct our experiments using the nuScenes data set (cf. Appendix A.10) with the training/validation split suggested by the authors. If not specified otherwise we train all models with the parameters summarized in Table 7.1.

Due to memory restrictions, all models are trained with a batch size of 4 on two *Nvidia GeForce RTX 2080 Ti* GPUs using the ADAM optimizer

Parameter	Value
Backbone	ResNet-38
Feature fusion	FPN (128 filters)
Convolutions	Separable 3×3 (cf. Example 2.6)
Weight decay	1×10^{-4} (L2, cf. Example 2.7)
Scales	2 to 5
Box, class & association head	Scale-dependent, $N = 2$, $F = 128$
Association embedding size	32
Embedding loss bias	0.7
Embedding loss weight	0.5
Flow head	PWCNet-small
intersection over union (IoU) method	Aligned, threshold 0.1
Data augmentation	Random horizontal flip
NMS threshold	0.5

Table 7.1: Default model parameters during evaluation.

($\beta_1 = 0.9$, $\beta_2 = 0.999$). After each nonlinearity, batch normalization (BN) (cf. Section 2.4.2) is performed. We use a linear learning rate warm up from 1×10^{-5} to 1×10^{-4} within the first 3000 iterations, a subsequent cosine learning rate schedule and terminate the optimization at iteration 300 000 at a learning rate of 1×10^{-5}.

7.3.1 Quantitative Evaluation

To evaluate scene flow estimation and object detection we use the nuScenes object detection benchmark metrics ([Cae+20], cf. Appendix A.10.1) and the odometry metrics presented in Section 6.4.1. The evaluation results are summarized in Table 7.2. We note that this evaluation investigates just two aspects of the joint approach and that a more in-depth evaluation should be conducted in future work.

Multi-Task Approach Table 7.3 compares the performance between our multi-task baseline model Joint1 and the single-task models Obj1/Flow1 presented in Table 5.2 and Table 6.2, respectively. We observe that the joint model improves in the object detection categories, whereas the flow estimation performance remains similar to the single-task flow estimation model.

ID	Feature aggreg.	mAP▲ in %	mATE▲ in cm	mASE▲ in %	mAOE▲ in °	mAVE▲ in m/s	AP▲ in %	AR▲ in %
Joint1	✓	**28.1**	42.8	30.1	**27.6**	0.72	**89.1**	**62.3**
Joint2		27.2	**42.7**	30.1	27.8	**0.71**	85.6	59.5

Table 7.2: Performance if model was trained with (Joint1) and without (Joint2) feature aggregation. Although we observe no significant improvement for true positive metrics (mATE, mASE, mAOE, mAVE), the mAP and tracking scores increase.

ID	mAP▲ in %	mATE▲ in cm	mASE▲ in %	mAOE▲ in °	mAVE▲ in m/s	mAOTE▲ in %	mAORE▲ in °/m
Obj1	26.0	44.9	31.0	36.1			
Flow1					**0.71**	**16.9**	0.16
Joint1	**28.1**	**42.8**	**30.1**	**27.6**	0.72	17.1	**0.15**

Table 7.3: Comparison between our multi-task baseline Joint1 and the single-task baselines Obj1/Flow1, presented in Table 5.2 and Table 6.2, respectively. The joint model improves in all object detection metrics and has a similar flow estimation performance. Note that the mAVE for Joint1 is computed based on the true positives, whereas for Flow1 it is computed on all labels.

Tracking To evaluate the object tracking based on association embeddings, we determine precision and recall of object associations between two labeled frames. In the nuScenes data set, measurements are annotated every 500 ms (key frames), however, there are LiDAR range measurements available every 100 ms. Therefore, we track an object for five frames and then check in the next key frame if the tracking was successful. Algorithm 7.1 describes the evaluation procedure. For our evaluation, we set $d = 0.3\,\text{m}$ and $\delta = 0.1$.

Table 7.2 compares models with and without feature aggregation. Although the positive metrics mATE, mASE, mAOE and mAVE remain similar, the mAP increases. Additionally, feature aggregation also improves the tracking metrics AP and AR. We therefore believe that feature aggregation leads to the better detection and tracking of partially occluded objects.

Algorithm 7.1: Association embedding evaluation

Initialize number of possible, correct and false tracks: $N_{\mathrm{P}} = N_{\mathrm{TP}} = N_{\mathrm{FP}} = 0$
for key frame i **do**
    Match detections to labels with smallest L2 distance $\leq d$ and assign track IDs
    Store embeddings and track IDs in buffer
    for n=1 **to** 5 **do**
        Match embeddings with smallest L2 distance $\leq \delta$ in buffer
        Store embeddings and track IDs in buffer
    Match detections to labels with smallest L2 distance $\leq d$ and assign track IDs
    Match embeddings with smallest L2 distance $\leq \delta$ in buffer
    Add number of possible ($N_{\mathrm{P},i}$), correct ($N_{\mathrm{TP},i}$) and false ($N_{\mathrm{FP},i}$) tracks based on matched IDs to N_{P}, N_{TP} and N_{FP}
Calculate average precision AP = $\frac{N_{\mathrm{TP}}}{N_{\mathrm{TP}}+N_{\mathrm{FP}}}$ and recall AR = $\frac{N_{\mathrm{TP}}}{N_{\mathrm{P}}}$

8 Conclusion & Future Directions

In this thesis, we presented and estimated an environment model for automated driving (AD) based on top-view grid maps. We argue that top-view grid maps are a suitable representation of the traffic environment as they enable data fusion from multiple sensor sources and explicitly model observability, which is a requirement for safe driving.

As a prerequisite for top-view grid mapping, we first proposed a method for ground surface estimation from noisy range measurements represented as point sets (Chapter 3). In our approach, we modeled the ground surface as uniform B-spline (UBS). UBSs implicitly impose smoothness and are insensitive to locally varying measurement densities. With robust optimization techniques and the UBS surface model, we were able to accurately estimate the ground surface in a wide distance range. Using this ground surface estimate, we are able to distinguish between ground and obstacle surface reflections so that we can model the traffic scene relative to the ground surface in top-view grid maps.

We then presented an approach for occupancy and elevation grid mapping that considers multiple noisy and possibly contradicting range sensor measurements (Chapter 4). Occupancy is estimated by means of evidence theory, which allows us to explicitly model occlusions and contradictions. We presented a computationally efficient LiDAR sensor model as basic belief assignment (BBA), which considers the number of grid cell reflections and transmissions within an elevation range. Given the resulting evidential occupancy information, we then presented a method to estimate the scene drivability based on the vehicle shape. The scene observability and drivability were represented as polygons, which is a compact representation for subsequent AD modules. Both ground surface estimation and grid mapping were implemented and validated on an experimental vehicle platform where we mapped static environment with known poses. In this sample application, we also showed how to combine elevation information from multiple sources in a probabilistic manner.

Based on the resulting top-view grid maps, we then developed object detection and scene flow estimation models using machine learning (ML) techniques (Chapters 5 and 6). These models take all grid map layers as input and output oriented bounding boxes with semantic classes and optional velocities, which constitutes condensed and meaningful information for subsequent AD modules. The scene flow estimation model is learned in a self-supervised manner, i.e. without the need of manually annotated data. By applying robust parameter optimization techniques, we were able to estimate sensor odometry and egomotion-compensated object flow, which describes velocities w.r.t. a static reference frame. Finally, we proposed a model for joint object detection, scene flow and tracking feature estimation (Chapter 7). This model outputs detected and classified objects with velocities and tracking features. All models were quantitatively evaluated on the publicly available nuScenes data set. Subsequently, we proved the general applicability of our models on exemplary scenarios.

Finally, the environment model contains top-view grid maps as sensor-agnostic, low-level information, observability and drivability polygons and traffic participants represented as oriented bounding boxes with semantic class likelihoods, velocities and tracking features. We believe that this information improves the performance of subsequent AD modules. For instance, our model enables probabilistic motion planning considering occlusions and collision probabilities.

Future Directions Depending on the computational power of the AD platform or in case of offline computation, it may also be possible to extend the top-view grid map to a full 3D volume grid. Here, we note that the same evidential modeling and ML models can also be applied to 3D grids, further increasing the accuracy of object detection and flow estimation (e.g. [Lan+19; LYB19]).

In the grid mapping process, an output feature selection (evidential and height layers) is made and accurate positional information is discarded. This reduces the amount of information and thus restricts the performance of subsequent ML models. To keep the spatial relationship between single points, low-level encoders may be developed (cf. [Qi+17]). However, most low-level encoders are still computationally more expensive compared to methods operating on grids.

The research field of ML is evolving at high pace with seminal publications being released every week. Recently, new anchor-free object detection approaches were presented that need less (hyper-)parameters and yield more accurate results than their anchor-based counterparts presented in this work. We refer to Tian et al. [Tia+20] for a promising anchor-free approach to object detection.

Recently, one of the most active fields of research is the modeling and development of multi-task approaches for object detection, flow estimation and tracking. Here, more experiments may be conducted, e.g. by feeding back motion prediction into the learning process.

Lastly, a large issue with supervised learning approaches remains: These methods usually only work well on the data set they were trained on and perform worse on other data sets. This *domain gap* may arise due to different sensor characteristics or traffic scenes. Techniques for domain adaptation may mitigate this loss in performance and are also subject to recent research (cf. [WDS20; And20]).

References

[AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein. “Least-Squares Fitting of Two 3-D Point Sets”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9.5 (1987), pp. 698–700. DOI: 10.1109/TPAMI.1987.4767965 (cit. on p. 139).

[And20] I. Andrussow. “Unsupervised Domain Adaptation for Object Classification On Grid Maps”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2020 (cit. on p. 105).

[Bec20] J. Beck. “Camera Calibration with Non-Central Local Camera Models”. Dissertation. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2020. DOI: 10.5445/IR/1000131090 (cit. on pp. 13, 22, 24).

[Beh+19] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. “SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences”. In: *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*. Seoul, Korea, 2019, pp. 9297–9307 (cit. on pp. 6, 27, 129).

[Bel+18] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García, and A. D. L. Escalera. “Birdnet: A 3d Object Detection Framework from Lidar Information”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI (Nov. 4, 2018). Maui, HI, USA: IEEE, 2018, pp. 3517–3523. DOI: 10.1109/ITSC.2018.8569311 (cit. on p. 62).

[BFG15] P. Babahajiani, L. Fan, and M. Gabbouj. “Object Recognition in 3D Point Cloud of Urban Street Scene”. In: *Computer Vision - ACCV 2014 Workshops*. Ed. by C. Jawahar and S. Shan. Cham: Springer International Publishing, 2015, pp. 177–190. DOI: 10.1007/978-3-319-16628-5_13 (cit. on p. 62).

[BHL15] L. Beyer, A. Hermans, and B. Leibe. “Biternion Nets: Continuous Head Pose Regression from Discrete Training Labels”. In: *German Conference on Pattern Recognition*. Ed. by J. Gall, P. Gehler, and B. Leibe. Aachen, Germany: Springer International Publishing, 2015, pp. 157–168. DOI: 10.1007/978-3-319-24947-6_13 (cit. on p. 62).

[BM93] S. Beucher and F. Meyer. “The Morphological Approach to Segmentation: The Watershed Transformation”. In: *Mathematical Morphology in Image Processing* 34 (1993), pp. 433–481 (cit. on p. 58).

[BR96] M. J. Black and A. Rangarajan. “On the Unification of Line Processes, Outlier Rejection, and Robust Statistics with Applications in Early Vision”. In: *International Journal of Computer Vision* 19.1 (1996), pp. 57–91. DOI: 10.1007/BF00131148 (cit. on p. 124).

[Cae+20] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. “nuScenes: A Multimodal Dataset for Autonomous Driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Virtual, 2020, pp. 11621–11631 (cit. on pp. 4, 70, 88, 99, 136, 137).

[Che+17] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. “Multi-View 3D Object Detection Network for Autonomous Driving”. In: *CVPR* (2017), pp. 1907–1915. DOI: 10.1109/CVPR.2017.691 (cit. on p. 62).

[De 78] C. De Boor. *A Practical Guide to Splines*. Vol. 27. New York, USA: Springer-Verlag, 1978. ISBN: 978-0-387-95366-3 (cit. on p. 14).

[Dos+15] A. Dosovitskiy, P. Fischery, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. V. D. Smagt, D. Cremers, and T. Brox. “FlowNet: Learning Optical Flow with Convolutional Networks”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Santiago (Dec. 2015). Santiago, Chile: IEEE, 2015, pp. 2758–2766. DOI: 10.1109/ICCV.2015.316 (cit. on p. 78).

[DSD04] J. Dezert, F. Smarandache, and M. Daniel. “The Generalized Pignistic Transformation”. In: *Proceedings of the 7th International Conference on Information Fusion*. Stockholm, Sweden, 2004, pp. 384–391 (cit. on p. 8).

[Elf89] A. Elfes. “Using Occupancy Grids for Mobile Robot Perception and Navigation”. In: *Computer* 22.6 (1989), pp. 46–57. DOI: 10.1109/2.30720 (cit. on p. 36).

[Est+96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. “A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Conference on Knowledge Discovery and Data Mining (KDD)*. Vol. 96. 34. Portland, OR, USA, 1996, pp. 226–231 (cit. on p. 62).

[FBH18] P. Fankhauser, M. Bloesch, and M. Hutter. “Probabilistic Terrain Mapping for Mobile Robots with Uncertain Localization”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3019–3026. DOI: 10.1109/LRA.2018.2849506 (cit. on p. 37).

[Fen+19] D. Feng, L. Rosenbaum, F. Timm, and K. Dietmayer. “Leveraging Heteroscedastic Aleatoric Uncertainties for Robust Real-time Lidar 3D Object Detection”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. Paris, France, 2019, pp. 1280–1287. DOI: 10.1109/IVS.2019.8814046 (cit. on p. 63).

[Fis20] T. Fischer. “Multi-Task Learning for Object Detection and Scene Flow Estimation using Multi-Layer Grid Maps”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2020 (cit. on pp. 96, 97).

[FRD18] D. Feng, L. Rosenbaum, and K. Dietmayer. “Towards Safe Autonomous Driving: Capture Uncertainty in the Deep Neural Network for Lidar 3d Vehicle Detection”. In: *Proceedings of the IEEE Conference on Intelligent Transportation Systems*. Vol. 2018-Novem. Maui, HI, USA, 2018, pp. 3266–3273. DOI: 10.1109/ITSC.2018.8569814 (cit. on p. 63).

[Gal16] Y. Gal. “Uncertainty in Deep Learning”. Dissertation. PhD thesis. Cambridge, UK: Department of Engineering, University of Cambridge, 2016, p. 3 (cit. on p. 63).

[GG16] Y. Gal and Z. Ghahramani. “Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. In: vol. 3. New York City, NY, USA, 2016, pp. 1651–1660 (cit. on p. 63).

[GKF09] A. Golovinskiy, V. G. Kim, and T. Funkhouser. “Shape-based Recognition of 3D Point Clouds in Urban Environments”. In: Kyoto, Japan, 2009, pp. 2154–2161. DOI: 10.1109/ICCV.2009.5459471 (cit. on p. 62).

[GLU12] A. Geiger, P. Lenz, and R. Urtasun. “Are We Ready for Autonomous Driving? the KITTI Vision Benchmark Suite”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Providence, USA, 2012, pp. 3354–3361 (cit. on pp. 4, 129).

[Gol18] P. Goldin. *10 Advantages of Autonomous Vehicles*. ITS, 2018. URL: https://www.itsdigest.com/10-advantages-autonomous-vehicles (visited on 12/14/2020) (cit. on p. 1).

[HBL17] A. Hermans, L. Beyer, and B. Leibe. “In Defense of the Triplet Loss for Person Re-Identification”. In: (2017). arXiv: 1703.07737 [cs.CV]. URL: https://arxiv.org/abs/1703.07737 (visited on 12/06/2020) (cit. on p. 97).

[He+16] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA, 2016, pp. 770–778 (cit. on p. 60).

[Hör+18] S. Hörmann, P. Henzler, M. Bach, and K. Dietmayer. “Object Detection on Dynamic Occupancy Grid Maps Using Deep Learning and Automatic Label Generation”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. Changshu, China, 2018, pp. 826–833. DOI: 10.1109/IVS.2018.8500677 (cit. on p. 62).

[Hor87] B. K. P. Horn. “Closed-form Solution of Absolute Orientation Using Unit Quaternions”. In: *Journal of the Optical Society of America A* 4.4 (1987), p. 629. DOI: 10.1364/josaa.4.000629 (cit. on p. 80).

[HS81] B. K. P. Horn and B. G. Schunck. “Determining Optical Flow”. In: *Artificial Intelligence* 17.1-3 (1981), pp. 185–203. DOI: 10.1016/0004-3702(81)90024-2 (cit. on p. 78).

[Ilg+17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. “FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA, 2017, pp. 2462–2470 (cit. on pp. 78, 79, 89).

[Jia+18] Y. Jiang, X. Zhu, X. Wang, S. Yang, W. Li, H. Wang, P. Fu, and Z. Luo. “R2CNN: Rotational Region CNN for Orientation Robust Scene Text Detection”. In: *2018 24th International Conference on Pattern Recognition (ICPR)*. Beijing, China, 2018, pp. 3610–3615. DOI: 10.1109/ICPR.2018.8545598 (cit. on p. 62).

[KB14] D. P. Kingma and J. Ba. “ADAM: A Method for Stochastic Optimization”. In: (2014). arXiv: 1412.6980. URL: https://arxiv.org/abs/1412.6980 (visited on 12/06/2020) (cit. on p. 12).

[KGC17] J. Kukačka, V. Golkov, and D. Cremers. “Regularization for Deep Learning: A Taxonomy”. In: (2017). arXiv: 1710.10686. URL: https://arxiv.org/abs/1710.10686 (visited on 12/06/2020) (cit. on p. 19).

[KGC18] A. Kendall, Y. Gal, and R. Cipolla. “Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics”. In: Salt Lake City, UT, USA, 2018, pp. 7482–7491. DOI: 10.1109/CVPR.2018.00781 (cit. on pp. 94, 95).

[Lan+19] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. “PointPillars: Fast Encoders for Object Detection from Point Clouds”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA, 2019, pp. 12689–12697. DOI: 10.1109/CVPR.2019.01298 (cit. on pp. 62, 104).

[Lin+17a] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. “Feature Pyramid Networks for Object Detection”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Honolulu, HI, USA, 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106 (cit. on p. 61).

[Lin+17b] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. “Focal Loss for Dense Object Detection”. In: *Proceedings of the IEEE International Conference on Computer Vision*. Venice, Italy, 2017, pp. 2999–3007. DOI: 10.1109/ICCV.2017.324 (cit. on pp. 60, 67, 68, 70).

[Liu+16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg. “SSD: Single Shot Multibox Detector”. In: *European Conference on Computer Vision*. Vol. 9905 LNCS. Springer. Amsterdam, Netherlands, 2016, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2 (cit. on pp. 59, 60).

[Luo+18] C. Luo, J. Zhan, X. Xue, L. Wang, R. Ren, and Q. Yang. “Cosine Normalization: Using Cosine Similarity Instead of Dot Product in Neural Networks”. In: *Artificial Neural Networks and Machine Learning – ICANN 2018*. Rhodes, Greece: Springer International Publishing, 2018, pp. 382–391 (cit. on p. 95).

[LYB19] X. Liu, M. Yan, and J. Bohg. “MeteorNet: Deep Learning on Dynamic 3D Point Cloud Sequences”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea, 2019 (cit. on p. 104).

[MHR18] S. Meister, J. Hur, and S. Roth. “UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. New Orleans, LA, USA, 2018, pp. 7251–7259 (cit. on pp. 79, 82, 84).

[ML14] M. Muja and D. G. Lowe. “Scalable Nearest Neighbor Algorithms for High Dimensional Data”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11 (2014), pp. 2227–2240. DOI: 10.1109/TPAMI.2014.2321376 (cit. on p. 3).

[Mor89] H. P. Moravec. “Sensor Fusion in Certainty Grids for Mobile Robots”. In: *Sensor Devices and Systems for Robotics* 9.2 (1989), pp. 253–276. DOI: 10.1007/978-3-642-74567-6_19 (cit. on p. 36).

[MPS09] F. Moosmann, O. Pink, and C. Stiller. “Segmentation of 3D Lidar Data in Non-flat Urban Environments Using a Local Convexity Criterion”. In: *2009 IEEE Intelligent Vehicles Symposium*. Xi’an, Shaanxi, China, 2009, pp. 215–220. DOI: 10.1109/IVS.2009.5164280 (cit. on pp. 21, 58).

[Mur12] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: The MIT Press, 2012. ISBN: 0262018020 (cit. on p. 15).

[Nus+18] D. Nuss, S. Reuter, M. Thom, T. Yuan, G. Krehl, M. Maile, A. Gern, and K. Dietmayer. "A Random Finite Set Approach for Dynamic Occupancy Grid Maps with Real-time Application". In: *The International Journal of Robotics Research* 37.8 (2018), pp. 841–866. DOI: 10.1177/0278364918775523 (cit. on pp. 37, 81, 93).

[OMV18] J. Olivares, P. Martin, and E. Valero. "A Simple Approximation for the Modified Bessel Function of Zero Order". In: *Journal of Physics: Conference Series* 1043.1 (2018), p. 012003. DOI: 10.1088/1742-6596/1043/1/012003 (cit. on p. 136).

[Qi+17] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Long Beach, CA, USA: Curran Associates, Inc., 2017, pp. 5099–5108 (cit. on p. 104).

[Ren+15] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems*. Montréal, Canada, 2015, pp. 91–99 (cit. on p. 59).

[Ric+19] S. Richter, S. Wirges, H. Königshof, and C. Stiller. "Fusion of Range Measurements and Semantic Estimates in an Evidential Framework". In: *tm - Technisches Messen* 86.s1 (2019), pp. 102–106. DOI: 10.1515/teme-2019-0052 (cit. on pp. 38, 48).

[RZL17] P. Ramachandran, B. Zoph, and Q. V. Le. "Searching for Activation Functions". In: (2017). arXiv: 1710.05941v2 [cs.NE]. URL: https://arxiv.org/abs/1710.05941 (visited on 12/06/2020) (cit. on p. 129).

[SAE18] SAE International. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Standard. SAE International, 2018. URL: https://saemobilus.

sae.org/content/J3016_201806/ (visited on 12/14/2020) (cit. on p. 1).

[San+18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: Salt Lake City, UT, USA, 2018, pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474 (cit. on p. 60).

[Sch18] M. Schreier. "Environment Representations for Automated On-Road Vehicles". In: *At-Automatisierungstechnik* 66.2 (2018), pp. 107–118. DOI: 10.1515/AUTO-2017-0104 (cit. on p. 2).

[Sim+18] M. Simon, S. Milz, K. Amende, and H.-M. Groß. "Complex-YOLO: An Euler-Region-Proposal for Real-time 3D Object Detection on Point Clouds". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. Munich, Germany, 2018, pp. 197–209. DOI: 10.1007/978-3-030-11009-3_11 (cit. on pp. 62, 67).

[SRB14] D. Sun, S. Roth, and M. J. Black. "A Quantitative Analysis of Current Practices in Optical Flow Estimation and the Principles behind Them". In: *International Journal of Computer Vision* 106.2 (2014), pp. 115–137 (cit. on p. 84).

[SRK17] N. H. Saleem, M. Rezaei, and R. Klette. "Extending the Stixel World Using Polynomial Ground Manifold Approximation". In: *24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. Auckland, New Zealand, 2017, pp. 1–6. DOI: 10.1109/M2VIP.2017.8211440 (cit. on p. 22).

[Ste+20] S. Steyer, C. Lenk, D. Kellner, G. Tanzmeister, and D. Wollherr. "Grid-based Object Tracking With Nonlinear Dynamic State and Shape Estimation". In: *IEEE Transactions on Intelligent Transportation Systems* 21.7 (2020), pp. 2874–2893. DOI: 10.1109/TITS.2019.2921248 (cit. on p. 93).

[STW17] S. Steyer, G. Tanzmeister, and D. Wollherr. "Object Tracking Based on Evidential Dynamic Occupancy Grids in Urban Environments". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. Redondo Beach, CA, USA: IEEE, 2017, pp. 1064–1070. DOI: 10.1109/IVS.2017.7995855 (cit. on p. 62).

[STW18] S. Steyer, G. Tanzmeister, and D. Wollherr. “Grid-based Environment Estimation Using Evidential Mapping and Particle Tracking”. In: *IEEE Transactions on Intelligent Vehicles* 3 (3 2018), pp. 384–396. DOI: 10.1109/TIV.2018.2843130 (cit. on pp. 37, 81).

[Sun+18] D. Sun, X. Yang, M. Y. Liu, and J. Kautz. “PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA, 2018, pp. 8934–8943. DOI: 10.1109/CVPR.2018.00931 (cit. on pp. 78, 82, 89).

[Tia+20] Z. Tian, C. Shen, H. Chen, and T. He. “FCOS: A Simple and Strong Anchor-free Object Detector”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–12. DOI: 10.1109/TPAMI.2020.3032166 (cit. on p. 105).

[TL19] M. Tan and Q. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, CA, USA: PMLR, 2019, pp. 6105–6114 (cit. on p. 60).

[TPL20] M. Tan, R. Pang, and Q. V. Le. “EfficientDet: Scalable and Efficient Object Detection”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Virtual, 2020, pp. 10778–10787. DOI: 10.1109/CVPR42600.2020.01079 (cit. on pp. 61, 73).

[Ved+05] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. “Three-Dimensional Scene Flow”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.3 (3 2005), pp. 475–480. DOI: 10.1109/TPAMI.2005.63 (cit. on p. 77).

[Vel19] Velodyne Lidar, Inc. *Velodyne Alpha Prime Data Sheet*. Ed. by Velodyne Lidar, Inc. VLS-128. Rev. 1. 2019. URL: https://velodynelidar.com/wp-content/uploads/2019/12/63-9679_Rev-1_DATASHEET_ALPHA-PRIME_Web.pdf (visited on 12/14/2020) (cit. on p. 40).

[WDS20] S. Wirges, S. Ding, and C. Stiller. “Single-Stage Object Detection from Top-View Grid Maps on Custom Sensor Setups”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA, 2020. DOI: 10.1109/IV47402.2020.9304759 (cit. on p. 105).

[Wed+09] A. Wedel, H. Badino, C. Rabe, H. Loose, U. Franke, and D. Cremers. “B-Spline Modeling of Road Surfaces with an Application to Free-Space Estimation”. In: *IEEE Transactions on Intelligent Transportation Systems* 10.4 (2009), pp. 572–583. DOI: 10.1109/TITS.2009.2027223 (cit. on p. 22).

[Wir+18] S. Wirges, T. Fischer, C. Stiller, and J. B. Frias. “Object Detection and Classification in Occupancy Grid Maps Using Deep Convolutional Networks”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI, USA, 2018, pp. 3530–3535. DOI: 10.1109/ITSC.2018.8569433 (cit. on pp. 62, 67).

[Wir+19a] S. Wirges, J. Gräter, Q. Zhang, and C. Stiller. “Self-Supervised Flow Estimation Using Geometric Regularization with Applications to Camera Image and Grid Map Sequences”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand, 2019, pp. 1782–1787. DOI: 10.1109/ITSC.2019.8916989 (cit. on pp. 78, 82–84).

[Wir+19b] S. Wirges, M. Reith-Braun, M. Lauer, and C. Stiller. “Capturing Object Detection Uncertainty in Multi-Layer Grid Maps”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. Paris, France, 2019, pp. 1520–1526. DOI: 10.1109/IVS.2019.8814073 (cit. on pp. 63, 68).

[YA06] T. Yang and V. Aitken. “Evidential Mapping for Mobile Robots with Range Sensors”. In: *IEEE Transactions on Instrumentation and Measurement* 55 (4 2006), pp. 1422–1429. DOI: 10.1109/TIM.2006.876399 (cit. on pp. 36–38).

[Yan+20] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone. “Graduated Non-Convexity for Robust Spatial Perception: From Non-Minimal Solvers to Global Outlier Rejection”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1127–1134. DOI: 10.1109/LRA.2020.2965893 (cit. on pp. 13, 25, 80, 86, 123, 125).

[Yi+00] Z. Yi, Y. K. Ho, C. S. Chua, and X. W. Zhou. “Multi-Ultrasonic Sensor Fusion for Autonomous Mobile Robots”. In: *Sensor Fusion: Architectures, Algorithms, and Applications IV*. Ed. by B. V. Dasarathy. Vol. 4051. IV. Orlando, FL, USA, 2000, pp. 314–321 (cit. on p. 36).

[YLU18a] B. Yang, M. Liang, and R. Urtasun. “HDNET: Exploiting HD Maps for 3D Object Detection”. In: *Conference on Robot Learning (CoRL)*. Ed. by A. Billard, A. Dragan, J. Peters, and J. Morimoto. Vol. 87. Proceedings of Machine Learning Research CoRL. Virtual: PMLR, 2018, pp. 146–155 (cit. on p. 62).

[YLU18b] B. Yang, W. Luo, and R. Urtasun. “PIXOR: Real-time 3D Object Detection from Point Clouds”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA, 2018, pp. 7652–7660. DOI: 10.1109/CVPR.2018.00798 (cit. on p. 62).

[Zam+18] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. “Taskonomy: Disentangling Task Transfer Learning”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Salt Lake City, UT, USA, 2018, pp. 3712–3722 (cit. on p. 93).

[Zha+03] K. Zhang, S.-C. Chen, D. Whitman, M.-L. Shyu, J. Yan, and C. Zhang. “A Progressive Morphological Filter for Removing Nonground Measurements from Airborne Lidar Data”. In: *IEEE Transactions on Geoscience and Remote Sensing* 41.4 (2003), pp. 872–882. DOI: 10.1109/TGRS.2003.810682 (cit. on p. 22).

[Zhu+17a] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. “Flow-guided Feature Aggregation for Video Object Detection”. In: *The IEEE International Conference on Computer Vision (ICCV)*. Venice, Italy, 2017, pp. 408–417 (cit. on pp. 95, 98).

[Zhu+17b] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. “Deep Feature Flow for Video Recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA, 2017, pp. 2349–2358 (cit. on pp. 95, 98).

A Appendix

A.1 Evidence Theory Example: A Crime Case

A crime was committed and the suspects can be narrowed down to Sven (S), Rebekka (R) and Hendrik (H). There are two witnesses. One witness is certain that the suspect was male. The other witness guesses that the suspect was less than 30 years old.

The frame of discernment $\Omega = \{S, R, H\}$ contains the three suspects with its power set

$$2^{\Omega} = \{\emptyset, S, R, H, \{S, R\}, \{S, H\}, \{R, H\}, \Omega\}.$$

As the first witness says the suspect was male, we may model the BBA

$$\mathrm{m}_1(X) = \begin{cases} c_1 & \text{if } X = \{S, H\} \\ 1 - c_1 & \text{if } X = \Omega \\ 0 & \text{otherwise} \end{cases}$$

with confidence $c_1 \in [0, 1]$ and the focal elements $\{S, H\}$ and Ω. As the second witness guesses that the suspect is less than 30 years old which only applies to Rebekka and Hendrik, the BBA may be modeled as

$$\mathrm{m}_1(X) = \begin{cases} c_2 & \text{if } X = \{R, H\} \\ 1 - c_2 & \text{if } X = \Omega \\ 0 & \text{otherwise} \end{cases}$$

with confidence $c_2 \in [0, 1]$ and the focal elements $\{R, H\}$ and Ω. For the two belief assignments above we can verify the properties of a BBA (cf. Eq. (2.1)).

These two witness statements can be combined to the BBA

$$(\mathrm{m}_1 \oplus \mathrm{m}_2)(X) = \begin{cases} c_1 c_2 & \text{if } X = \{H\} \\ c_1(1-c_2) & \text{if } X = \{S, H\} \\ (1-c_1)\, c_2 & \text{if } X = \{R, H\} \\ (1-c_1)(1-c_2) & \text{if } X = \Omega \\ 0 & \text{otherwise} \end{cases}$$

using Dempster's rule of combination (cf. Eq. (2.2)).

We can then determine the belief (cf. Eq. (2.4))

$$\mathrm{bel}(X) = \begin{cases} c_1 c_2 & \text{if } X = \{H\} \\ c_1 c_2 + c_1\,(1-c_2) & \text{if } X = \{S, H\} \\ c_1 c_2 + (1-c_1)\, c_2 & \text{if } X = \{R, H\} \\ 1 & \text{if } X = \Omega \\ 0 & \text{otherwise} \end{cases}$$

and the plausibility (cf. Eq. (2.5))

$$\mathrm{pl}(X) = \begin{cases} 1-c_2 & \text{if } X = \{S\} \\ 1-c_1 & \text{if } X = \{R\} \\ 1-c_1 c_2 & \text{if } X = \{S, R\} \\ 1 & \text{otherwise} \end{cases}$$

that the suspects committed the crime. Note that there is only non-zero belief for propositions including Hendrik as he is suspected by both sources. Furthermore, the plausibility for Sven or Rebekka having committed the crime is less than one because either one of them is excluded by one of the witnesses.

To determine the person that most likely committed the crime, we may determine the pignistic probability (cf. Eq. (2.7))

$$\text{prob}(\omega) = \frac{1}{3} + \begin{cases} \dfrac{c_1 - 2c_2 - c_1c_2}{6} & \text{if } \omega = \{S\} \\ \dfrac{c_2 - 2c_1 - c_1c_2}{6} & \text{if } \omega = \{R\} \\ \dfrac{c_1 + c_2 + 2c_1c_2}{6} & \text{if } \omega = \{H\} \end{cases}$$

for the three suspects. If we model the confidences of the two witnesses by $c_1 = 80\%$ and $c_2 = 50\%$, we get probabilities

$$\text{prob}(X) \approx \begin{cases} 23.3\% & \text{if } X = \{S\} \\ 8.3\% & \text{if } X = \{R\} \\ 68.3\% & \text{if } X = \{H\} \end{cases}.$$

This means that if we had to make a decision, Hendrik is the suspect that most likely committed the crime.

A.2 Generalized Charbonnier Loss

The generalized Charbonnier loss

$$C_{\alpha,\epsilon}(x) = \left(x^2 + \epsilon^2\right)^{\alpha} \tag{A.1}$$

with its derivative

$$C'_{\alpha,\epsilon}(x) = 2\alpha x\left(x^2 + \epsilon^2\right)^{\alpha-1} \tag{A.2}$$

has the free parameters α and ϵ. As depicted in Fig. A.1 it can be specialized to L1- ($\alpha = 0.5$) and L2 losses ($\alpha = 1$). The parameter ϵ controls the slope smoothness close to $x = 0$ and yields differentiable versions of the L1 loss for $\epsilon > 0$. For $\alpha < \frac{1}{2}$, $C_{\alpha,\epsilon}$ yields additional robustness towards outliers due to its sublinearity.

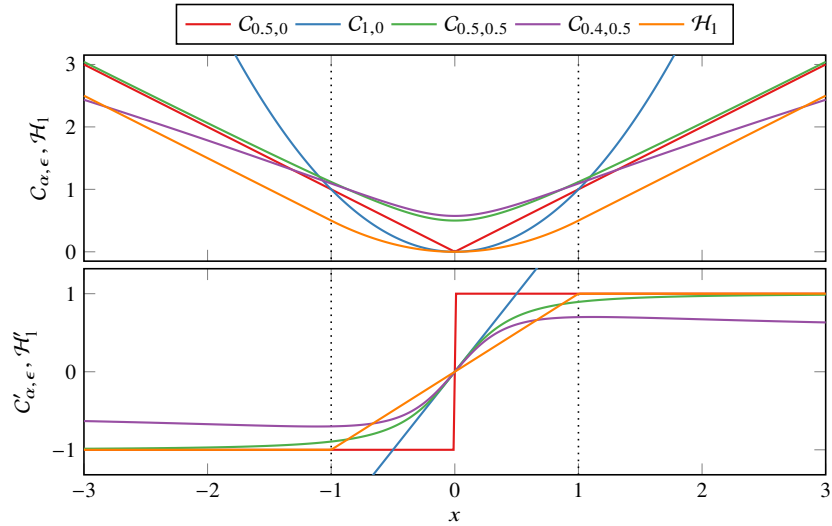


Figure A.1: Instances of the generalized Charbonnier loss $C_{\alpha,\epsilon}$ and the smooth L1 loss $\mathcal{H}_1$ with their derivatives $C'_{\alpha,\epsilon}$ and H'_1, respectively.

For computational efficiency, the smooth L1 loss $C_{0.5,\epsilon}$ is often approximated by the piecewise defined Huber loss

$$\mathcal{H}_1(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| \leq 1 \\ |x| - \frac{1}{2} & \text{otherwise.} \end{cases} \tag{A.3}$$

with the piecewise continuous derivative

$$\mathcal{H}'_1(x) = \begin{cases} -1 & \text{if } x \leq -1 \\ x & \text{if } -1 < x \leq \delta \\ 1 & \text{otherwise.} \end{cases} \tag{A.4}$$

A.3 Robust Estimation with Graduated Non-Convexity

Yang et al. [Yan+20] propose an iterative algorithm for robust estimation based on the concept of Graduated Non-Convexity (GNC) and the Black-Rangarajan duality which is summarized in the following.

A.3.1 Graduated Non-Convexity

To improve convergence for arbitrary initial values, surrogate functions ρ_μ with the free parameter μ may be used to replace the robustifiers ρ. By gradually changing μ during optimization the surrogate functions enable the transition from a convex to a non-convex problem.

For the Geman McClure (GMC) robustifier

$$\rho(x) = \frac{c^2 x^2}{c^2 + x^2} \tag{A.5}$$

we can verify that

$$\tilde{\rho}_\mu(x) = \frac{\mu c^2 x^2}{\mu c^2 + x^2}, \quad \lim_{\mu\to\infty} \tilde{\rho}_\mu(x) = x^2, \quad \tilde{\rho}_1(x) = \rho(x) \tag{A.6}$$

is a surrogate function as it yields a convex function for $\mu \to \infty$ and the initial GMC loss for $\mu = 1$.

For the truncation robustifier (cf. Eq. (2.26))

$$\tilde{\rho}_\mu(x) = \begin{cases} x^2 & \text{if } x^2 < \frac{\mu}{\mu+1} c^2 \\ 2c|x|\sqrt{\mu(\mu+1)} - \mu\left(x^2 + c^2\right) & \text{if } \frac{\mu}{\mu+1} c^2 \le x^2 < \frac{\mu+1}{\mu} c^2 \\ c^2 & \text{otherwise} \end{cases} \tag{A.7}$$

is a surrogate function because of

$$\lim_{\mu\to 0} \tilde{\rho}''_\mu(x) = \lim_{\mu\to 0} -2\mu = 0\,, \quad \lim_{\mu\to\infty} \tilde{\rho}_\mu(x) = \rho(x). \tag{A.8}$$

Figure A.2 depicts the GMC and truncation surrogate functions for different convexity parameters μ.

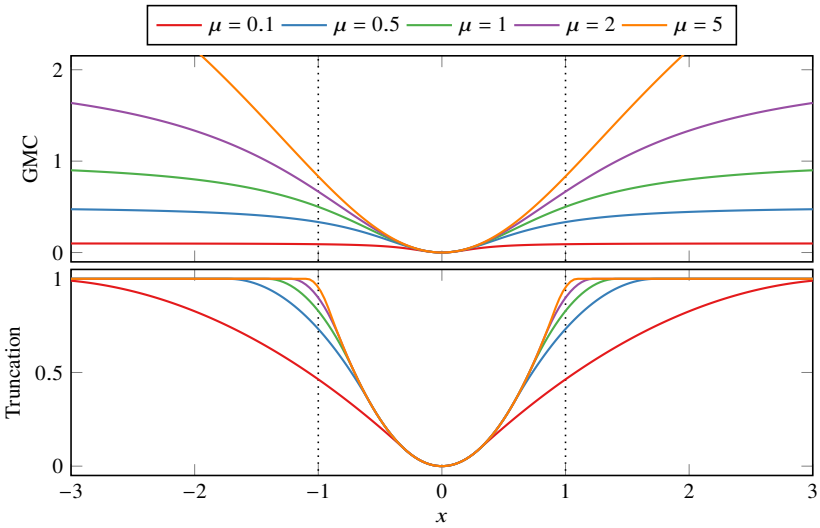


Figure A.2: GMC and truncation surrogate functions for $c = 1$ and different convexity parameters μ. The GMC surrogate becomes convex for $\mu \to \infty$, the truncation surrogate for $\mu \to 0$.

A.3.2 Black-Rangarajan Duality

Black et al. [BR96] show that an outlier process model can be recovered from certain robustifiers if the conditions

$$\lim_{x \to 0} \tilde{\rho}'(x) = 1 \,, \quad \lim_{x \to \infty} \tilde{\rho}'(x) = 0 \,, \quad \tilde{\rho}''(x) < 0 \tag{A.9}$$

on the robustifier $\tilde{\rho}(x) = \rho\left(\sqrt{x}\right)$ hold. Then, Eq. (2.25) is equivalent to the problem

$$\min_{\boldsymbol{p}, w_n \in [0,1]} \sum_{n=1}^{N_\mathrm{r}} w_n \|\mathbf{r}_n(\boldsymbol{p})\|^2 + \Phi_\rho(w_n) \,, \tag{A.10}$$

where $\Phi_\rho(w_k)$ acts as a weight penalty and depends on the robustifier selected. For instance, the GMC and truncation robustifiers yield the weight penalties

$$\Phi_{\rho,\mathrm{GMC}}(w) = \mu c^2 \left(\sqrt{w} - 1\right)^2, \tag{A.11}$$

$$\Phi_{\rho,\mathrm{TLS}}(w) = \frac{\mu(1-w)}{\mu + w} c^2. \tag{A.12}$$

Proofs can be found in [Yan+20].

A.3.3 The Algorithm

Yang et al. [Yan+20] model the optimization problem as

$$\min_{\boldsymbol{p}, w_n \in [0,1]} \sum_{n=1}^{N_\mathrm{r}} w_n \|\mathbf{r}_n(\boldsymbol{p})\|^2 + \Phi_{\tilde{\rho}}(w_n) \tag{A.13}$$

and propose an iterative algorithm with an outer and an inner loop. In the inner loop, the authors perform a parameter update

$$\boldsymbol{p}^{(t)} = \arg\min_{\boldsymbol{p}} \sum_{n=1}^{N_\mathrm{r}} w_n^{(t-1)} \|\mathbf{r}_n(\boldsymbol{p})\|^2 \tag{A.14}$$

with fixed weights $w_n^{(t-1)}$ and a weight update

$$w_1^{(t)}, \ldots, w_{N_\mathrm{r}}^{(t)} = \arg\min_{w_n \in [0,1]} \sum_{n=1}^{N_\mathrm{r}} w_n \left\| \mathbf{r}_n(\boldsymbol{p}^{(t)}) \right\|^2 + \Phi_{\tilde{\rho}}(w_n) \tag{A.15}$$

in an alternating fashion. In the outer loop, the non-convexity parameter μ is adapted to gradually increase non-convexity. The optimization may be stopped if a stopping criterion such as

$$\|\boldsymbol{p}^{(t)} - \boldsymbol{p}^{(t-1)}\|^2 < \epsilon \|\boldsymbol{p}^{(t)}\|^2, \quad \epsilon > 0 \tag{A.16}$$

is met or a maximum amount of outer loop iterations is reached.

A.3.4 Derivation of Weight Penalties

Equation (A.15) can be minimized independently for each weight in order to find an optimal weight

$$w = \underset{w \in [0,1]}{\arg\min}\, w\|\boldsymbol{r}\|^2 + \Phi_{\tilde{\rho}}(w)\,, \tag{A.17}$$

where we omitted the indices n and t for better readability.

GMC Penalty Inserting the GMC penalty $\Phi_{\rho,\mathrm{GMC}}$, we get

$$w = \underset{w \in [0,1]}{\arg\min}\, \underbrace{w\|\boldsymbol{r}\|^2 + \mu c^2\left(\sqrt{w} - 1\right)^2}_{\mathrm{c}(w)} \tag{A.18}$$

which can be derived and set to zero such that

$$\left.\frac{\partial \mathrm{c}}{\partial w}\right|_{w^*} \overset{!}{=} 0 = \|\boldsymbol{r}\|^2 + 2\mu c^2\left(1 - \frac{1}{\sqrt{w^*}}\right) \tag{A.19}$$

$$\Leftrightarrow w^* = \left(\frac{\mu c^2}{\mu c^2 + \|\boldsymbol{r}\|^2}\right)^2. \tag{A.20}$$

Because of

$$\frac{\partial^2 \mathrm{c}}{\partial w^2} = \mu c^2 w^{-\frac{3}{2}} > 0 \quad \forall w > 0\,, \tag{A.21}$$

w^* is a local minimum of Eq. (A.18).

TLS Penalty Inserting the truncated Least-Squares (TLS) penalty $\Phi_{\rho,\mathrm{TLS}}$ we get

$$w = \underset{w \in [0,1]}{\arg\min}\, \underbrace{w\|\boldsymbol{r}\|^2 + \frac{\mu(1-w)}{\mu + w} c^2}_{\mathrm{c}(w)} \tag{A.22}$$

which can be derived and set to zero such that

$$\left.\frac{\partial \mathrm{c}}{\partial w}\right|_{w^*} \stackrel{!}{=} 0 = \|\boldsymbol{r}\|^2 - \mu c^2 \frac{\mu+1}{(\mu+w^*)^2} \tag{A.23}$$

$$\Leftrightarrow 0 = w^{*2} + 2\mu w^* + \mu^2 - \frac{c^2}{\|\boldsymbol{r}\|^2}\mu(\mu+1) \tag{A.24}$$

$$\Leftrightarrow w^* = -\mu \pm \frac{c}{\|\boldsymbol{r}\|}\sqrt{\mu(\mu+1)}. \tag{A.25}$$

As

$$\frac{\partial^2 \mathrm{c}}{\partial w^2} = \frac{2\mu(\mu+1)}{(\mu+w)^3} > 0 \quad \forall w > 0 \tag{A.26}$$

we can compute weights that minimize Eq. (A.22). Because $w \in [0,1]$ but Eq. (A.25) is unbounded in general, only the positive solution

$$w^* = \frac{c}{\|\boldsymbol{r}\|}\sqrt{\mu(\mu+1)} - \mu \tag{A.27}$$

exists and we need to check the boundaries

$$w^* = 0 \quad \Leftrightarrow \quad \|\boldsymbol{r}\|^2 = \frac{\mu+1}{\mu}c^2\,, \tag{A.28}$$

$$w^* = 1 \quad \Leftrightarrow \quad \|\boldsymbol{r}\|^2 = \frac{\mu}{\mu+1}c^2 \tag{A.29}$$

explicitly which serve as optimal points on the borders. In conclusion, the optimal weights

$$w^* = \begin{cases} 1 & \text{if } \|\boldsymbol{r}\|^2 < \frac{\mu}{\mu+1}c^2 \\ \dfrac{c}{\|\boldsymbol{r}\|}\sqrt{\mu(\mu+1)} - \mu & \text{if } \frac{\mu}{\mu+1}c^2 \leq \|\boldsymbol{r}\|^2 \leq \frac{\mu+1}{\mu}c^2 \\ 0 & \text{otherwise} \end{cases} \tag{A.30}$$

can be computed depending on the sum of squared residuals.

Figure A.3 depicts the optimal penalty weights as a function of the residual norm $\|\boldsymbol{x}\|$ for different convexity parameters μ.

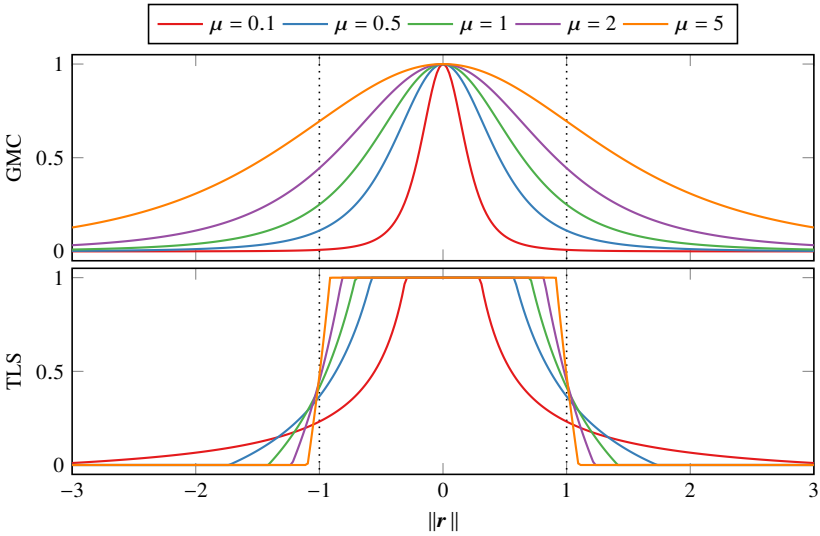


Figure A.3: Optimal weights of GMC and TLS penalties for $c = 1$ and different convexity parameters μ.

A.4 Non-linear Activation Functions

The rectified linear unit (ReLU), swish and tanh functions

$$\mathrm{a}_{\mathrm{ReLU}}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}, \qquad \mathrm{a}'_{\mathrm{ReLU}}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.31}$$

$$\mathrm{a}_{\mathrm{swish}}(x) = \frac{x}{1+e^{-x}}, \qquad \mathrm{a}'_{\mathrm{swish}}(x) = \frac{1+e^{-x}(1+x)}{(1+e^{-x})^2} \tag{A.32}$$

$$\mathrm{a}_{\tanh}(x) = \frac{e^{2x}-1}{e^{2x}+1}, \qquad \mathrm{a}'_{\tanh}(x) = \frac{2e^{2x}}{\left(e^{2x}+1\right)^2} \tag{A.33}$$

are commonly used as non-linear activation functions in artificial neural networks (ANNs). These functions are depicted with their derivatives in Fig. A.4. Whereas the gradient of the tanh function vanishes for large values,

the gradient $a'_{\text{ReLU}}(x)$ is discontinuous in $x = 0$. The swish function aims to resolve these disadvantages [RZL17].

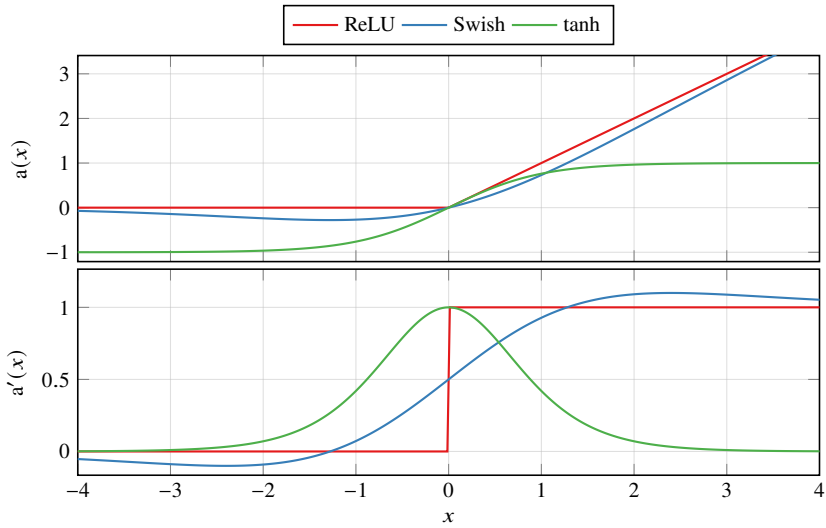


Figure A.4: ReLU, swish and tanh activation functions and their derivatives.

A.5 SemanticKITTI Data Set

The SemanticKITTI data set [Beh+19] is created from sequences of the KITTI odometry benchmark [GLU12] in which every range measurement within a radius of 50 m is annotated with a semantic class. The range sensor is a Velodyne HDL64E LiDAR and the measurements are provided as point set with a total number of 23 201 points in the training set. The semantic class distribution is summarized in Table A.1.

Category	**Class**	**Abs. frequency** in million	**Rel. frequency** in %
Ground	Road	4611.185	19.875
	Sidewalk	3339.157	14.392
	Parking	341.453	1.472
	Other	90.597	0.391
	Lane marking	1.092	0.005
Structure	Building	3078.452	13.269
	Other	55.569	0.240
Vehicle	Car	988.544	4.261
	Truck	142.375	0.217
	Other	38.645	0.167
	Motorcycle	9.243	0.040
	Bicycle	3.854	0.017
	Bus	3.281	0.014
Nature	Vegetation	6190.375	26.682
	Terrain	1812.978	7.814
	Trunk	140.018	0.604
Human	Person	4.106	0.034
	Bicyclist	2.949	0.013
	Motorcyclist	0.869	0.004
Object	Fence	1678.806	7.236
	Other	230.226	0.992
	Pole	66.25	0.286
	Traffic sign	14.282	0.062
Unlabeled / Outlier		445.079	1.918
Total		23 201	100.000

Table A.1: Class distribution in the SemanticKITTI training data set.

Sensor	Lasers	Range in m	Range Acc. in cm	Horiz. FOV in °	Vert. FOV in °
Velodyne VLP16	16	100	3	360	30
Velodyne VLS128	128	200	3	360	40
Ibeo LUX4L	4	50	10	110	3.2

Table A.2: Experimental vehicle range sensor specifics.

A.6 Experimental Vehicle

Our experimental vehicle is an automated Mercedes-Benz E class limousine. As depicted in Fig. A.5 it has mounted four Velodyne VLP16 LiDARs on the roof corners, one Velodyne VLS128 LiDAR on the roof center and an Ibeo LUX4L LiDAR below the front sign plate. Table A.2 summarizes key specifics.

Computations are made on a general-purpose PC with an AMD EPYC 7702P 64-Core processor and 256 GB RAM. The PC has two GPUs, an NVidia TITAN X and a Titan V.

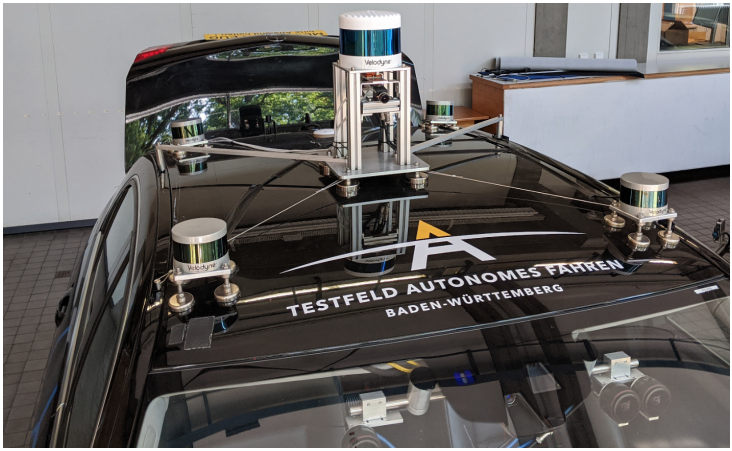
A.7 Mixture Distributions

A random variable $\boldsymbol{X} \sim \mathcal{M}$ may follow a mixture distribution $\mathcal{M}$ defined by the probability density function (PDF)

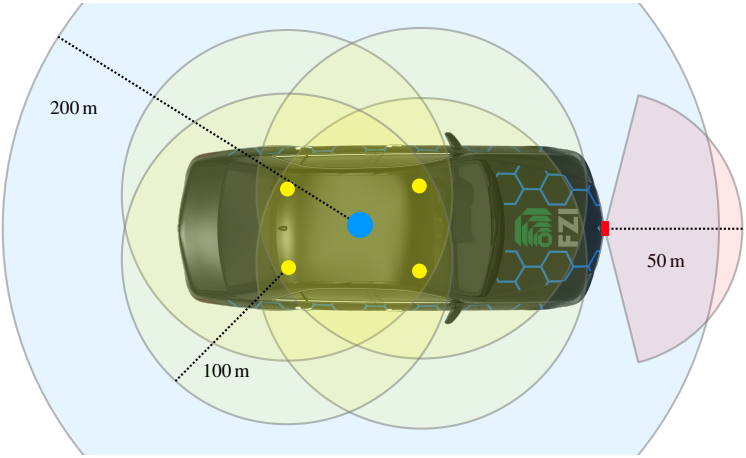
$$\mathrm{p}(\boldsymbol{x}) = \sum_{n=1}^{N} w_n \mathrm{p}(\boldsymbol{x}|n) \;, \quad \sum_{n=1}^{N} w_n = 1 \;, \tag{A.34}$$

where each mixture component $\boldsymbol{X}|n \sim \mathcal{D}$ of the N mixtures follows a certain distribution $\mathcal{D}$. A mixture distribution can be approximated by its mean

$$\begin{aligned} \boldsymbol{\mu} &= \mathrm{E}_{\mathrm{p}(\boldsymbol{x})}\{\boldsymbol{x}\} = \int_{-\infty}^{\infty} \boldsymbol{x}\mathrm{p}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \int_{-\infty}^{\infty} \boldsymbol{x} \sum_{n=1}^{N} w_n \mathrm{p}(\boldsymbol{x}|n)\mathrm{d}\boldsymbol{x} \\ &= \sum_{n=1}^{N} w_n \int_{-\infty}^{\infty} \boldsymbol{x}\mathrm{p}(\boldsymbol{x}|n)\mathrm{d}\boldsymbol{x} = \sum_{n=1}^{N} w_n \mathrm{E}_{\mathrm{p}(\boldsymbol{x}|n)}\{\boldsymbol{x}\} = \sum_{n=1}^{N} w_n \boldsymbol{\mu}_n \end{aligned} \tag{A.35}$$



(a) Roof-mounted LiDARs (not shown: LUX4L).



(b) Sensor setup top-view. Not to scale.
Yellow: VLP16, blue: VLS128, red: LUX4L.

Figure A.5: LiDARs mounted on the experimental vehicle.

and covariance

$$\begin{aligned}\boldsymbol{\Sigma} &= \mathrm{E}_{\mathrm{p}(\boldsymbol{x})}\left\{(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^\top\right\} = \mathrm{E}_{\mathrm{p}(\boldsymbol{x})}\left\{\boldsymbol{x}\boldsymbol{x}^\top\right\} - \boldsymbol{\mu}\boldsymbol{\mu}^\top \\ &= \left(\sum_{n=1}^{N} w_n \mathrm{E}_{\mathrm{p}(\boldsymbol{x}|n)}\left\{\boldsymbol{x}\boldsymbol{x}^\top\right\}\right) - \boldsymbol{\mu}\boldsymbol{\mu}^\top \\ &= \left(\sum_{n=1}^{N} w_n\left(\boldsymbol{\Sigma}_n + \boldsymbol{\mu}_n\boldsymbol{\mu}_n^\top\right)\right) - \boldsymbol{\mu}\boldsymbol{\mu}^\top\,, \end{aligned} \tag{A.36}$$

where $\boldsymbol{\mu}_n$ and $\boldsymbol{\Sigma}_n$ denote the mean and covariance of the n-th mixture component.

A.8 Intersection over Union of Rotated Rectangles

The intersection over union (IoU)

$$\mathrm{IoU}(B_1, B_2) = \frac{|B_1 \cap B_2|}{|B_1 \cup B_2|} = \frac{|B_1 \cap B_2|}{|B_1| + |B_2| - |B_1 \cap B_2|} \tag{A.37}$$

(also Jaccard index) denotes the similarity of two sets B_1 and B_2. This task can be split into the intersection area computation $|B_1 \cap B_2|$ and the single box area computations $|B_1|$ and $|B_2|$.

Here, we only deal with rotated rectangles which are convex polygons. The non-empty intersection between two rotated rectangles is always a convex polygon.

Algorithm A.1 shows how the intersection area is computed. To compute the intersection area, we interpret polygon vertices $\boldsymbol{v}_i = \left[v_{x,i}, v_{y,i}, 0\right]^\top$ as 3D vectors on the xy-plane. If the polygon is convex and its vertices are stored in counter-clockwise order w.r.t. a reference point within the polygon, the area

$$A = \frac{1}{2}\sum_{i=1}^{V} \langle \boldsymbol{e}_z, \boldsymbol{v}_i \times \boldsymbol{v}_{i-1}\rangle, \quad \boldsymbol{v}_0 = \boldsymbol{v}_V \tag{A.38}$$

Algorithm A.1: Intersection area of two rectangles

Input: Rectangles A, B
Output: Intersection area

```
1  hull_vertices = VertexList(∅);
2  foreach vertex v in A do
3      if v inside B then
4          hull_vertices.add(v);
5  foreach vertex v in B do
6      if v inside A then
7          hull_vertices.add(v);
8  foreach edge e_A in A do
9      foreach edge e_B in B do
10         if e_A intersects e_B then
11             hull_vertices.add(intersection(e_A, e_B));
12 hull_vertices = sortCounterClockwise(hull_vertices);
13 hull_vertices = convexHull(hull_vertices);
14 num_vertices = len(hull_vertices);
15 if num_vertices < 3 then
16     return 0
17 reference_vertex = hull_vertices[1]; area = 0;
18 for n = 2 to num_vertices − 1 do
19     triangle = (reference_vertex, hull_vertices[n], hull_vertices[n + 1]);
20     area += area(triangle);
21     n += 1;
22 return area
```

can be computed by the sum of pairwise vertex cross products iterated in counter-clockwise direction and projected onto the xy-plane. A visual explanation on two examples is depicted in Fig. A.6.

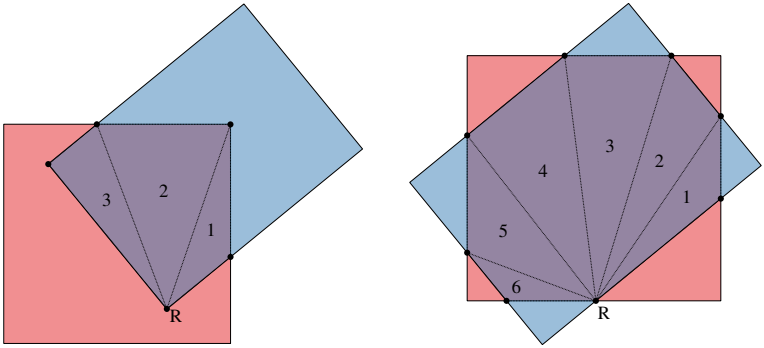


Figure A.6: Intersection area computation on two examples. First, we determine the hull vertices that either lie inside the boxes or on their boundaries. Second, the area of the resulting convex polygon is determined by summing over the triangle areas. The reference vertex is marked by R, the triangles over which is summed are numbered.

A.9 Von Mises Distribution

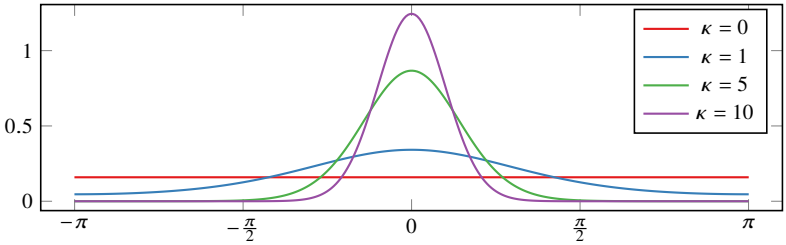


Figure A.7: Centered von Mises distribution for different concentration parameters κ.

The von Mises distribution is a continuous probability distribution on the circle. Its PDF

$$p(x|\mu,\kappa) = \frac{e^{\kappa\cos(x-\mu)}}{2\pi I_0(\kappa)}, \tag{A.39}$$

is centered around the mean μ and parameterized by the concentration κ. Here, the modified Bessel function

$$\mathrm{I}_0(\kappa) = \sum_{m=0}^{\infty} \frac{1}{(m!)^2} \left(\frac{\kappa}{2}\right)^{2m} = \sum_{m=0}^{\infty} \left[\frac{1}{m!} \left(\frac{\kappa}{2}\right)^{m}\right]^2 \tag{A.40}$$

$$= 1 + \frac{\kappa^2}{4} + \frac{\kappa^4}{64} + \frac{\kappa^6}{2304} + \frac{\kappa^8}{147456} + \dots \tag{A.41}$$

$$\approx \frac{\cosh(x)}{\left(1 + 0.25x^2\right)^{0.25}} \frac{1 + 0.24273x^2}{1 + 0.43023x^2} \tag{A.42}$$

may be approximated by a linear combination of polynomials or a rational function combined with a hyperbolic function [OMV18].

A.10 nuScenes Data Set & Object Detection Benchmark

The nuScenes data set [Cae+20] contains 1000 sequences of 20 s length recorded by moving sensor platforms in Boston and Singapore traffic. As depicted in Fig. A.8, the sensor platforms carry six cameras, one LiDAR, five RADARs, a GPS and an IMU unit.

The 32 beam spinning LiDAR operates at 20 Hz, and has a vertical field of view (FOV) from $-30°$ to $10°$. It has a range of 70 m with an accuracy of ± 2 cm and captures 1.39×10^6 points per second, provided as point sets.

At a frequency of 2 Hz, 23 object classes are annotated by 3D bounding boxes. Extrinsic sensor transformations are provided for every sequence. The labeled object categories and their absolute and relative frequencies are listed in Table A.3.

A.10.1 Object Detection Benchmark

Classes The object categories considered in the nuScenes object detection benchmark are summarized in Table A.4 with their absolute and relative

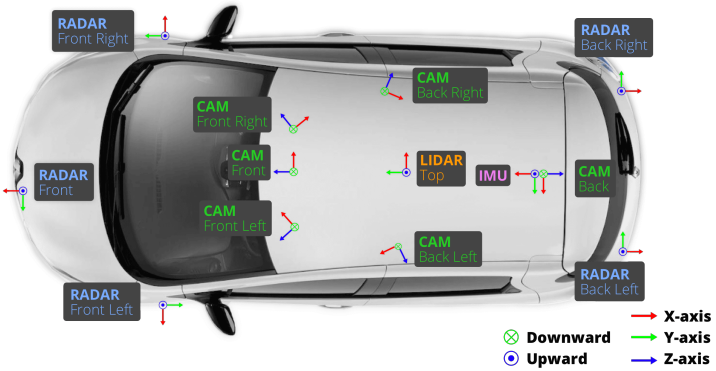


Figure A.8: A sensor platform used to record the nuScenes data set [Cae+20]. The spinning LiDAR is mounted on the vehicle top at a height of around 1.8 m.

frequencies. Aside from traffic participants such as cars, pedestrians or cyclists, also static elements are considered (barriers and traffic cones).

Evaluation Metrics The mean average precision (mAP) is determined by matching detections (bounding boxes and class scores) to the closest ground truth object within a defined distance defined by the object center positions. Here, different matching thresholds of 0.5 m, 1 m, 2 m and 4 m are used. For each matching threshold and a minimum accepted class score threshold, the number of true positives N_{TP}, false positives N_{FP} and false negatives N_{FN} is determined. Based on these values, the precision-recall curve is computed by varying the class score threshold and computing the average precision (AP) as the area under the curve of precisions

$$\text{P} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FP}}} \tag{A.43}$$

for recalls

$$\text{R} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FN}}} \geq 0.1. \tag{A.44}$$

Finally, to obtain the mAP, the AP is averaged over all matching thresholds and classes.

Category	Class	Abs. frequency	Rel. frequency in %
Animal		787	0.068
Pedestrian	Adult	208 240	17.857
	Child	2066	0.177
	Construction worker	9161	0.786
	Stroller	1072	0.092
	Police officer	727	0.062
	Wheelchair	503	0.043
	Personal mobility	395	0.034
Movable object	Barrier	152 087	13.041
	Traffic cone	97 959	8.400
	Pushable / pullable	24 605	2.110
	Debris	3016	0.259
Vehicle	Car	493 322	42.302
	Truck	88 519	7.591
	Trailer	24 860	2.132
	Construction	14 671	1.258
	Bus (rigid)	14 501	1.244
	Motorcycle	12 617	1.082
	Bicycle	11 859	1.017
	Bus (bendy)	1820	0.156
	Police	638	0.055
	Ambulance	49	0.004
Static object	Bicycle rack	2713	0.233
Total		1 166 187	100.000

Table A.3: Class distribution in the nuScenes data set.

Based on all true positives, the average orientation error (AOE), average scale error (ASE) and average translation error (ATE) are computed for each class. The ATE denotes the distance between the predicted and labeled object centers in m. The AOE denotes the smallest orientation angle difference in ° between predicted and labeled orientation within 360° except for barriers where it is evaluated within 180° and traffic cones where no AOE is computed. The ASE is computed by 1 − IoU after compensating translation and orientation errors. Additionally, the average velocity error is computed by averaging the velocity error

$$\mathrm{VE} = \frac{\|\Delta\hat{\boldsymbol{t}} - \Delta\boldsymbol{t}\|}{\tau} \tag{A.45}$$

Category	Classes	Abs. frequency	Rel. frequency in %
Car		493 322	43.562
Pedestrian	Adult, child, construction worker, police officer	220 194	19.444
Barrier		152 087	13.430
Traffic cone		97 959	8.650
Truck		88 519	7.817
Trailer		24 860	2.195
Bus	Bendy, rigid	16 321	1.441
Construction vehicle		14 671	1.296
Motorcycle		12 617	1.114
Bicycle		11 859	1.047
Total		1 132 463	100.000

Table A.4: Class construction and distribution in the nuScenes object detection benchmark.

across all class instances, where $\Delta \boldsymbol{t}$ denotes the translation difference and τ the time difference between two frames. Based on these metrics, the mean average translation error (mATE), mean average scale error (mASE), mean average orientation error (mAOE) and mean average velocity error (mAVE) are calculated by averaging across all classes.

A.11 Point Registration using Weighted Least-Squares

We introduce a WLS solution to the point registration problem as an extension of the Least-Squares (LS) approaches e.g. as proposed in [AHB87].

Given a set $\left(\boldsymbol{p}_n, \boldsymbol{q}_n, w_n\right) \in C$ of N weighted point correspondences, we aim to find a translation vector and rotation matrix

$$\boldsymbol{R}^*, \boldsymbol{t}^* = \underset{\boldsymbol{R},\boldsymbol{t}}{\arg\min} \sum_{n=1}^{N} w_n \|\boldsymbol{R}\boldsymbol{p}_n + \boldsymbol{t} - \boldsymbol{q}_n\|^2 = \underset{\boldsymbol{R}\in \mathrm{SO}_d,\boldsymbol{t}}{\arg\min} L(\boldsymbol{R}, \boldsymbol{t}) \tag{A.46}$$

that minimize the sum of weighted squared distances between the corresponding points after applying a rigid-body transformation to $\boldsymbol{p}_n$.

Determining the roots of the derivative $\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{t}}$ yields

$$\boldsymbol{t}^* = \hat{\boldsymbol{q}} - \boldsymbol{R}\hat{\boldsymbol{p}} \tag{A.47}$$

with the weighted centroids

$$\hat{\boldsymbol{p}} = \frac{\sum_{n=1}^{N} w_n \boldsymbol{p}_n}{\sum_{n=1}^{N} w_n}, \quad \hat{\boldsymbol{q}} = \frac{\sum_{n=1}^{N} w_n \boldsymbol{q}_n}{\sum_{n=1}^{N} w_n}. \tag{A.48}$$

Substituting $\boldsymbol{t}$ in Eq. (A.46) by $\boldsymbol{t}^*$ (Eq. (A.47)) and defining the centered points $\tilde{\boldsymbol{p}}_n = \boldsymbol{p}_n - \hat{\boldsymbol{p}}$, $\tilde{\boldsymbol{q}}_n = \boldsymbol{q}_n - \hat{\boldsymbol{q}}$ yields the optimal rotation

$$\begin{aligned}\boldsymbol{R}^* &= \operatorname*{arg\,min}_{\boldsymbol{R}\in\mathrm{SO}_d} \sum_{n=1}^{N} w_n \|\boldsymbol{R}\tilde{\boldsymbol{p}}_n - \tilde{\boldsymbol{q}}_n\|^2 = \operatorname*{arg\,max}_{\boldsymbol{R}\in\mathrm{SO}_d} \sum_{n=1}^{N} w_n \langle \tilde{\boldsymbol{q}}_n, \boldsymbol{R}\tilde{\boldsymbol{p}}_n \rangle \\ &= \operatorname*{arg\,max}_{\boldsymbol{R}\in\mathrm{SO}_d} \operatorname{tr}\big(\boldsymbol{W}\tilde{\boldsymbol{Q}}^\top \boldsymbol{R}\tilde{\boldsymbol{P}}\big) = \operatorname*{arg\,max}_{\boldsymbol{R}\in\mathrm{SO}_d} \operatorname{tr}\big(\boldsymbol{R}\tilde{\boldsymbol{P}}\boldsymbol{W}\tilde{\boldsymbol{Q}}^\top\big)\end{aligned} \tag{A.49}$$

which maximizes the trace of the diagonal weight matrix $\boldsymbol{W}$ including the weights $w_1, \ldots, w_n$, and the matrices $\tilde{\boldsymbol{P}}$ and $\tilde{\boldsymbol{Q}}$ containing the points in column-wise order. The matrix

$$\boldsymbol{S} \coloneqq \tilde{\boldsymbol{P}}\boldsymbol{W}\tilde{\boldsymbol{Q}}^\top \overset{\text{(SVD)}}{=} \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \tag{A.50}$$

can be decomposed into its singular values $\boldsymbol{\Sigma}$ (sorted from the largest to the smallest) and the orthonormal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ which yields the optimal rotation matrix

$$\boldsymbol{R}^* = \operatorname*{arg\,max}_{\boldsymbol{R}\in\mathrm{SO}_d} \operatorname{tr}(\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{R}\boldsymbol{U}) = \operatorname*{arg\,max}_{\boldsymbol{R}\in\mathrm{SO}_d} \operatorname{tr}(\boldsymbol{\Sigma}\boldsymbol{I}) = \boldsymbol{V}\boldsymbol{U}^\top. \tag{A.51}$$

Finally, to exclude the special case of a perfect reflection if $\det(\boldsymbol{V}\boldsymbol{U}^\top) = -1$, the second best rotation

$$\boldsymbol{R}^* = \boldsymbol{V}\begin{bmatrix} 1 & & \\ & \ddots & \\ & & \det(\boldsymbol{V}\boldsymbol{U}^\top) \end{bmatrix}\boldsymbol{U}^\top \tag{A.52}$$

is obtained by flipping the component corresponding to the smallest singular value.

To summarize, the optimal transformation parameters

$$\boldsymbol{R}^*, \boldsymbol{t}^* = \arg\min_{\boldsymbol{R},\boldsymbol{t}} \sum_{n=1}^{N} w_n \|\boldsymbol{R}\boldsymbol{p}_n + \boldsymbol{t} - \boldsymbol{q}_n\|^2 \tag{A.53}$$

can be estimated by the following steps:

1. Compute the weighted centroids $\hat{\boldsymbol{p}} = \frac{\sum_{n=1}^{N} w_n \boldsymbol{p}_n}{\sum_{n=1}^{N} w_n}$, $\hat{\boldsymbol{q}} = \frac{\sum_{n=1}^{N} w_n \boldsymbol{q}_n}{\sum_{n=1}^{N} w_n}$ and the centered points $\tilde{\boldsymbol{p}}_n = \boldsymbol{p}_n - \hat{\boldsymbol{p}}$, $\tilde{\boldsymbol{q}}_n = \boldsymbol{q}_n - \hat{\boldsymbol{q}}$.
2. Determine the covariance $\boldsymbol{S} = \tilde{\boldsymbol{P}}\boldsymbol{W}\tilde{\boldsymbol{Q}}^\intercal$ and its SVD $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal$.
3. Compute the optimal rotation $\boldsymbol{R}^* = \boldsymbol{V}\begin{bmatrix} 1 & & \\ & \ddots & \\ & & \det(\boldsymbol{V}\boldsymbol{U}^\intercal) \end{bmatrix}\boldsymbol{U}^\intercal$.
4. Determine the optimal translation $\boldsymbol{t}^* = \hat{\boldsymbol{q}} - \boldsymbol{R}\hat{\boldsymbol{p}}$.

A.12 Publications by the Author

S. A. Baur, F. Moosmann, S. Wirges, and C. B. Rist. “Real-time 3D LiDAR Flow for Autonomous Vehicles”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. Paris, France, 2019, pp. 1288–1295. DOI: 10.1109/IVS.2019.8814094.

F. Bieder, S. Wirges, J. Janosovits, S. Richter, Z. Wang, and C. Stiller. “Exploiting Multi-Layer Grid Maps for Surround-View Semantic Segmentation of Sparse LiDAR Data”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA, 2020. DOI: 10.1109/IV47402.2020.9304848.

J. Fei, W. Chen, P. Heidenreich, S. Wirges, and C. Stiller. “SemanticVoxels: Sequential Fusion for 3d Pedestrian Detection Using LiDAR Point Cloud and Semantic Segmentation”. In: *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. Karlsruhe, Germany, 2020, pp. 185–190. DOI: 10.1109/MFI49285.2020.9235240.

M. Harr, J. Janosovits, S. Wirges, and C. Stiller. “Fast and Robust Vehicle Pose Estimation by Optimizing Multiple Pose Graphs”. In: *2018 21st International Conference on Information Fusion (FUSION)*. Cambridge, UK, 2018, pp. 1707–1714. DOI: 10.23919/ICIF.2018.8455309.

A. Hellmund, S. Wirges, Ö. Ş. Taş, C. Bandera, and N. O. Salscheider. “Robot Operating System: A Modular Software Framework for Automated Driving”. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. Rio de Janeiro, Brazil, 2016, pp. 1564–1570. DOI: 10.1109/ITSC.2016.7795766.

H. Hu, J. Zhu, S. Wirges, and M. Lauer. “Localization in Aerial Imagery with Grid Maps Using LocGAN”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand, 2019, pp. 2860–2865. DOI: 10.1109/ITSC.2019.8917236.

T. Kühner, S. Wirges, and M. Lauer. “Automatic Generation of Training Data for Image Classification of Road Scenes”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand, 2019, pp. 1097–1103. DOI: 10.1109/ITSC.2019.8917089.

J. Quehl, H. Hu, S. Wirges, and M. Lauer. “An Approach to Vehicle Trajectory Prediction Using Automatically Generated Traffic Maps”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. Changshu, China, 2018, pp. 544–549. DOI: 10.1109/IVS.2018.8500535.

J. Quehl, S. Yan, S. Wirges, J.-H. Pauls, and M. Lauer. “Estimating Object Shape and Movement Using Local Occupancy Grid Maps”. In: *IFAC-PapersOnLine* 52.8 (2019), pp. 87–92. DOI: 10.1016/j.ifacol.2019.08.053.

S. Richter, S. Wirges, H. Königshof, and C. Stiller. “Fusion of Range Measurements and Semantic Estimates in an Evidential Framework”. In: *tm - Technisches Messen* 86.s1 (2019), pp. 102–106. DOI: 10.1515/teme-2019-0052.

S. Richter, J. Beck, S. Wirges, and C. Stiller. “Semantic Evidential Grid Mapping Based on Stereo Vision”. In: *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. Karlsruhe, Germany, 2020, pp. 179–184. DOI: 10.1109/MFI49285.2020.9235217.

Ö. Ş. Taş, N. O. Salscheider, F. Poggenhans, S. Wirges, C. Bandera, M. R. Zofka, T. Strauss, J. M. Zöllner, and C. Stiller. "Making Bertha Cooperate–Team AnnieWAY's Entry to the 2016 Grand Cooperative Driving Challenge". In: *IEEE Transactions on Intelligent Transportation Systems* 19.4 (2018), pp. 1262–1276. DOI: 10.1109/TITS.2017.2749974.

S. Wirges, S. Ding, and C. Stiller. "Single-Stage Object Detection from Top-View Grid Maps on Custom Sensor Setups". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. Las Vegas, NV, USA, 2020. DOI: 10.1109/IV47402.2020.9304759.

S. Wirges, F. Hartenbach, and C. Stiller. "Evidential Occupancy Grid Map Augmentation Using Deep Learning". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. Changshu, China, 2018, pp. 668–673. DOI: 10.1109/IVS.2018.8500635.

S. Wirges, B. Roxin, E. Rehder, T. Kühner, and M. Lauer. "Guided Depth Upsampling for Precise Mapping of Urban Environments". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. Redondo Beach, CA, USA, 2017, pp. 1140–1145. DOI: 10.1109/IVS.2017.7995866.

S. Wirges, T. Fischer, C. Stiller, and J. B. Frias. "Object Detection and Classification in Occupancy Grid Maps Using Deep Convolutional Networks". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI, USA, 2018, pp. 3530–3535. DOI: 10.1109/ITSC.2018.8569433.

S. Wirges, J. Gräter, Q. Zhang, and C. Stiller. "Self-Supervised Flow Estimation Using Geometric Regularization with Applications to Camera Image and Grid Map Sequences". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand, 2019, pp. 1782–1787. DOI: 10.1109/ITSC.2019.8916989.

S. Wirges, M. Reith-Braun, M. Lauer, and C. Stiller. "Capturing Object Detection Uncertainty in Multi-Layer Grid Maps". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. Paris, France, 2019, pp. 1520–1526. DOI: 10.1109/IVS.2019.8814073.

S. Wirges, Y. Yang, H. Hu, and C. Stiller. "Learned Enrichment of Top-View Grid Maps Improves Object Detection". In: *2020 IEEE Intelligent Transportation Systems Conference (ITSC)*. Rhodes, Greece, 2020. DOI: 10.1109/ITSC45102.2020.9294330.

A.13 Supervised Theses

I. Andrussow. “Unsupervised Domain Adaptation for Object Classification On Grid Maps”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2020.

S. Baur. “Learning Scene Flow in Consecutive LiDAR Point Clouds using Fully Convolutional Neural Networks”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2018.

S. Becker. “Erweiterung eines auf Rasterkarten basierenden Trackingverfahrens durch Geschwindigkeitsschätzung von Lidardaten”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2016.

S. Ding. “Evaluierung von Verfahren zur Registrierung von Punktwolken”. Bachelor thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2015.

S. Ding. “Development of Single Stage Detectors on Multi-Layer Grid Maps”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2019.

E. Divrikli. “Entwicklung und Evaluierung von Objektsegmentierungsalgorithmen”. Bachelor thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2015.

E. Divrikli. “Low-Level Fusion von Kamera und LiDAR Daten zur Generierung einer hochaufgelösten dreidimensionalen Umgebungsrepräsentation mittels Deep Learning”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2019.

M. Engelhorn. “Fusion von Tiefen- und Kamerabildern zur 3D Objektdetektion mittels Deep Learning”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2018.

T. Fischer. “Evaluierung von globalen Registrierungsverfahren in Punktwolken”. Bachelor thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2017.

T. Fischer. “Multi-Task Learning for Object Detection and Scene Flow Estimation using Multi-Layer Grid Maps”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2020.

F. Hartenbach. “Erweiterung von Information in Rasterkarten mithilfe lernbasierter Ansätze”. Bachelor thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2017.

D. Keck. “Improving Orientation Estimation in Sparse LiDAR Grid Maps”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2020.

F. Klein. “Schätzung des Szenenflusses auf Belegtheitsrasterkarten und Tiefenbildern mittels Deep Learning”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2018.

N. Kuhn. “Sequence Learning on Point Cloud Features for 3D Object Detection in an Autonomous Driving Scenario”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2019.

K. Li. “Intrinsische Kalibrierung von Laserscannern anhand von Ebenenmodellen”. Bachelor thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2015.

M. Mayr. “Guided Depth Upsampling with Extrinsic Calibration of Camera and Range Sensor Setups”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2018.

J. Petasch. “Untersuchung zur Schätzung von Unsicherheiten bei der semantischen Segmentierung von LiDAR- und Kameramessungen”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2019.

M. Reith-Braun. “Modellierung von Unsicherheiten in künstlichen neuronalen Netzen zur Objekterkennung”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2018.

B. Roxin. “Generating Dense Surface Representations based on Lidar and Camera Data”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2017.

P. Schnattinger. “Entwicklung eines Multi-Sensor SLAM Frontends unter Zuhilfenahme von Freiraumschätzung”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2017.

S. Trick. “Dynamic Objects Detection and Filtering in LiDAR Measurements”. Bachelor thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2016.

C. Wecht. “Batch-Optimierung von Lidarmessungen zur kombinierten Schätzung von Umfeldmodell und Kalibrierung”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2018.

Y. Yang. “Improving Object Detection in Grid Maps with Augmentation Networks”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2019.

W. Zahoransky. “Charakterisierung von Fahrbahnoberflächen auf Basis von LiDAR-Daten”. Bachelor thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2016.

Q. Zhang. “Scene Flow Estimation in Occupancy Grid Maps using Self-Supervised Learning”. Master thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2019.