LIDAR-BASED SEMANTIC LABELING

AUTOMOTIVE 3D SCENE UNDERSTANDING

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (DR.-ING.)

von der KIT-Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)
genehmigte

DISSERTATION

von

M.SC. FLORIAN PIERRE JOSEPH PIEWAK

Tag der mündlichen Prüfung:	17.06.2020
Referent:	Prof. Dr. J. Marius Zöllner
Korreferent:	Prof. Dr.-Ing. Rainer Stiefelhagen

ABSTRACT

Mobile robots and autonomous vehicles rely on multi-modal sensor setups to perceive and understand their surroundings. Aside from cameras and RaDAR sensors, LiDAR sensors represent a key component of state-of-the-art perception systems. In addition to accurate spatial perception, a comprehensive semantic understanding of the environment is essential for efficient, safe operation.

In this thesis, the LiLaNet - a novel real-time capable neural network architecture for high-quality LiDAR-based point-wise classification known as semantic labeling - is developed. Considering the well-known domain of image-based processing, the 3D LiDAR point cloud is represented as a 2D cylindrical image. As a result, published state-of-the-art methods for LiDAR-based semantic labeling are outperformed on different benchmarks.

For developing deep learning approaches, large-scale datasets are of paramount importance. Therefore, datasets are created based on two state-of-the-art mobile LiDAR sensors. Reducing the cost and time consumption of cumbersome manual annotation work, an automated cross-modal training data generation process for large-scale datasets is introduced by combining state-of-the-art camera-based semantic labeling approaches with precise calibration of a multi-modal sensor system. The resulting semantic information can be transferred from the camera domain to the LiDAR domain.

In addition, an efficient multi-modal data compression technique is proposed by transferring a stereoscopic camera compression approach known as Stixel-World to the LiDAR domain. This leads to a reduction in the large quantity of LiDAR measurements by retaining the underlying semantic and geometric information, increasing the real-time capability of downstream algorithms of autonomous driving or mobile robotic platforms.

Furthermore, two different extensions of the proposed LiDAR-based semantic labeling approach are outlined. First, a reduction in sensor dependence is presented by introducing the PiLaNet, a novel 3D neural network architecture for point-wise semantic labeling leveraging the strength of 3D representation compared to the rather sensor specific cylindrical projection of the LiDAR point cloud. Second, the uncertainty of deep learning approaches is implicitly modeled by introducing a label hierarchy into the training process of semantic labeling approaches.

In conclusion, this thesis significantly advances the state-of-the-art in real-time 3D scene understanding for LiDAR sensors, yielding a gain in performance, robustness, and redundancy for future mobile robots and autonomous vehicles.

ZUSAMMENFASSUNG

Mobile Roboter und autonome Fahrzeuge verwenden verschiedene Sensormodalitäten zur Erkennung und Interpretation ihrer Umgebung. Neben Kameras und RaDAR Sensoren repräsentieren LiDAR Sensoren eine zentrale Komponente für moderne Methoden der Umgebungswahrnehmung. Zusätzlich zu einer präzisen Distanzmessung dieser Sensoren, ist ein umfangreiches semantisches Szeneverständnis notwendig, um ein effizientes und sicheres Agieren autonomer Systeme zu ermöglichen.

In dieser Arbeit wird das neu entwickelte LiLaNet, eine echtzeitfähige, neuronale Netzarchitektur zur semantischen, punktweisen Klassifikation von LiDAR Punktwolken, vorgestellt. Hierfür finden die Ansätze der 2D Bildverarbeitung Verwendung, indem die 3D LiDAR Punktwolke als 2D zylindrisches Bild dargestellt wird. Dadurch werden Ergebnisse moderner Ansätze zur LiDAR-basierten, punktweisen Klassifikation übertroffen, was an unterschiedlichen Datensätzen demonstriert wird.

Zur Entwicklung von Ansätzen des maschinellen Lernens, wie sie in dieser Arbeit verwendet werden, spielen umfangreiche Datensätze eine elementare Rolle. Aus diesem Grund werden zwei Datensätze auf Basis von modernen LiDAR Sensoren erzeugt. Durch das in dieser Arbeit entwickelte automatische Verfahren zur Datensatzgenerierung auf Basis von mehreren Sensormodalitäten, speziell der Kamera und des LiDAR Sensors, werden Kosten und Zeit der typischerweise manuellen Datensatzgenerierung reduziert.

Zusätzlich wird eine multimodale Datenkompression vorgestellt, welche ein Kompressionsverfahren der Stereokamera auf den LiDAR Sensor überträgt. Dies führt zu einer Reduktion der LiDAR Daten bei gleichzeitigem Erhalt der zugrundeliegenden semantischen und geometrischen Information. Daraus resultiert eine erhöhte Echtzeitfähigkeit nachgelagerter Algorithmen autonomer Systeme.

Außerdem werden zwei Erweiterungen zum vorgestellten Verfahren der semantischen Klassifikation umrissen. Zum einen wird die Sensorabhängigkeit durch Einführung des PiLaNets, einer neuen 3D Netzarchitektur, reduziert indem die LiDAR Punktwolke im 3D kartesischen Raum belassen wird, um die eher sensorabhängige 2D zylindrische Projektion zu ersetzen. Zum anderen wird die Unsicherheit neuronaler Netze implizit modelliert, indem eine Klassenhierarchie in den Trainingsprozess integriert wird.

Insgesamt stellt diese Arbeit neuartige, performante Ansätze des 3D LiDAR-basierten, semantischen Szeneverstehens vor, welche zu einer Verbesserung der Leistung, Zuverlässigkeit und Sicherheit zukünftiger mobile Roboter und autonomer Fahrzeuge beitragen.

ACKNOWLEDGMENTS

This PhD thesis would not have been possible without the help and support of many people throughout the last few years.

First and foremost, I would like to sincerely thank Prof. Dr. J. Marius Zöllner as well as Dr. Peter Pinggera for guiding me through this work, having interesting and inspiring discussions, as well as their continuous support for all topics related to this PhD thesis. They introduced and taught me how to contribute to scientific research and enabled the progress of this PhD thesis.

In addition, I would like to thank Prof. Dr.-Ing. Rainer Stiefelhagen, Prof. Dr. Andreas Oberweis, and Prof. Dr. Maxim Ulrich for their cooperativeness and their willingness to act as second reviewer, examiner, and chairman.

Furthermore, I am grateful to my friends and colleagues at the LiDAR team for lively discussions and helpful advise. They provided a comfortable surrounding and made the work fun, even on long working days.

Additionally, I would like to thank my colleagues at the Forschungszentrum Informatik (FZI) for deep discussions about PhD-related topics. This facilitates the development of further ideas within this PhD thesis.

Moreover, I sincerely acknowledge Mercedes-Benz R&D for giving me the opportunity to realize my PhD research within an enjoyable environment, including a great infrastructure to develop large-scale machine learning approaches as well as the opportunity to integrate and evaluate all of my research work within the real-time environment of research vehicles.

Some parts of this thesis have been supported by students under my supervision. Thanks to Moritz Schillinger, Chysa Bozatzidou and Niclas Wüstenbecker for their contributions and friendship.

A special thanks goes to Dr. Peter Pinggera, Dr. Markus Enzweiler, Simon Appel, and Maren Seiz, who did parts of the proofreading of this thesis and provided valuable comments to increase overall quality.

Last but not least, I am profoundly thankful to my family for the continuous support, encouragement, and opportunity to pursue my own goals. My deepest thanks goes to Maren and her understanding, encouragement, and patience while supporting me in all kind of situations.

ACRONYMS

1D	**1**-**D**imensional
2D	**2**-**D**imensional
2.5D	**2.5**-**D**imensional
3D	**3**-**D**imensional
ABS	**A**nti-lock **B**raking **S**ystem
AN	**A**dversarial **N**etwork
ANN	**A**rtifical **N**eural **N**etwork
BEV	**B**ird's **E**ye **V**iew
CNN	**C**onvolutional **N**eural **N**etwork
CRF	**C**onditional **R**andom **F**ield
DOGMa	**D**ynamic **O**ccupancy **G**rid **Ma**p
ESC	**E**lectronic **S**tability **C**ontrol
FCN	**F**ully **C**onvolutional Neural **N**etwork
FFN	**F**eed **F**orward **N**etwork
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
GAN	**G**enerative **A**dversarial **N**etwork
GPU	**G**raphics **P**rocessing **U**nit
HDL-64	Velodyne **HDL-64**
HOG	**H**istogram of **O**riented **G**radients
IoU	**I**ntersection **o**ver **U**nion
LASER	**L**ight **A**mplification by **S**timulated **E**mission of **R**adiation
LiDAR	**Li**ght **D**etection **a**nd **R**anging
LiLaBlock	**Li**DAR **La**beling **Block**
LiLaNet	**Li**DAR **La**beling **Net**work
mIoU	**m**ean **I**ntersection **o**ver **U**nion

PiLaNet	**Pi**llar **La**beling **Net**work
RaDAR	**Ra**dio **D**etection **a**nd **R**anging
ReLU	**Re**ctified **L**inear **U**nit
SIFT	**S**cale-**I**nvariant **F**eature **T**ransform
SVM	**S**upport **V**ector **M**achine
TN	**T**rue **N**egative
TP	**T**rue **P**ositive
VFE	**V**oxel **F**eature **E**ncoder (also called **V**oxel **F**eature **E**xtractor within the literature)
VLP-32C	Velodyne **VLP-32C**
VLS-128	Velodyne **VLS-128**
w/	**w**ith
w/o	**w**ith**o**ut

MATHEMATICAL NOTATION

Matrices, Vectors, Scalars, Constants, and Functions

$\mathbf{M}$	Matrix of arbitrary size (bold, uppercase, and non-italic)
$\mathbf{M}_{i,j}$	Matrix element at position (i, j) (bold, uppercase, and non-italic)
$\boldsymbol{v}$	Vector of arbitrary size (bold and italic)
x	Scalar (italic)
c	Constant scalar (non-italic)
$\mathrm{f}(x)$	Function $\mathrm{f}(\dots)$ with a scalar argument x
$\mathrm{f}'(x)$	Derivative of function $\mathrm{f}(\dots)$ with a scalar argument x
$\frac{\delta \mathrm{f}}{\delta x}$	Partial derivative of function $\mathrm{f}(\dots)$ with respect to x

Probability Theory and Statistics

$Pr(A)$	Probability of event A
$Pr(A\|B)$	Conditional probability of event A, conditioned on event B

General Definitions

i, j, k, n, o	Indices
c	speed of light
e	Euler's number
α, γ	Different angles
β, ϵ	Different Model parameters
c	Class for classification

LiDAR measurements

t	Timestamp of measurements
$\boldsymbol{p}$	Single point of a LiDAR point cloud containing the cartesian coordinates
p_x	X value of the cartesian coordinates of a point $\boldsymbol{p}$
p_y	Y value of the cartesian coordinates of a point $\boldsymbol{p}$
p_z	Z value of the cartesian coordinates of a point $\boldsymbol{p}$
ξ	Reflectivity of a point $\boldsymbol{p}$
r	Distance to the sensor
$\mathbf{T}$	Homogeneous transformation matrix for a transformation of a point $\boldsymbol{p}$ to a different coordinate system

Artificial Neural Networks

Ξ	Threshold of the nucleus of a neuron
ω	Single artificial neuron of an Artificial Neural Network
x	Single input of an artificial neuron ω (scalar case)
w	Single weight for the input of an artificial neuron ω (scalar case)
$\boldsymbol{x}_{nb}$	Input vector of an artificial neuron ω without bias encoding (1D case)
$\boldsymbol{w}_{nb}$	Weight vector of an artificial neuron ω without bias encoding (1D case)
$\boldsymbol{x}$	Input vector of an artificial neuron ω with bias encoding (1D case)
$\boldsymbol{w}$	Weight vector of an artificial neuron ω with bias encoding (1D case)
$\mathbf{X}$	Input tensor of an artificial neuron ω with bias encoding ($\geq$ 2D case)
$\mathbf{W}$	Weight tensor of an artificial neuron ω with bias encoding ($\geq$ 2D case)
b	Bias of an artificial neuron ω
z	Weighted sum of the inputs of an artificial neuron ω

$\mathrm{f}_{act}(\dots)$	Activation function of an artificial neuron ω
y	Single output of an artificial neuron ω (scalar case)
$\boldsymbol{y}$	Output vector of multiple artificial neuron ω (1D case)
$\mathbf{Y}$	Output Tensor of a filter of a convolution layer ($\geq$ 2D case)
ϕ	Index of the class c_ϕ returned from an argmax operator
$\Phi(\dots)$	Softmax function
M	Number of classes used for prediction
N	Number of classifications
L	Number of training samples
$\boldsymbol{\chi}$	Inputs of training samples
$\boldsymbol{\psi}$	Desired classification outputs for training samples
η	Learning rate
$\mathrm{E}(\dots)$	Error function
$\rho_{\psi,k}$	Weight for the cross-entropy function related to the desired output ψ and the index k of the class c_k
$\Lambda, d\Lambda$	Hierarchical loss weighting
H	Height of a feature map
W	Width of a feature map
C	Number of channel of a feature map
g	Padding width of a convolution layer
δ	Stride of a convolution layer
ζ	Dilation width of a dilated convolution layer

Multi-Modal Stixels

$\boldsymbol{S}$	Vector of Stixels
$\boldsymbol{s}$	Single Stixel
l	Label of a Stixel
u	Bottom index of a Stixel
a	Top index of a Stixel
$\boldsymbol{D}$	Vertically ordered depth measurements
$\boldsymbol{L}$	Vertically ordered label measurements

$\boldsymbol{M}$	Vertically ordered multi-modal measurements containing depth measurements and label measurements
$\boldsymbol{d}$	Depth measurement
$\boldsymbol{l}$	Label measurement
$\boldsymbol{m}$	Multi-modal measurement containing depth and label measurements
$\Pi(\dots), \Theta(\dots), \Omega(\dots)$	Energy functions of the Stixel computation
θ	Compression rate

CONTENTS

1 INTRODUCTION

CONTENTS

Within this chapter, a motivation as well as an overview of the thesis in terms of the contributions and the structure of the thesis are provided.

1.1 MOTIVATION

This section describes the motivation of the thesis and provides an introduction to related topics.

1.1.1 *Autonomous Driving*

For several decades, the mobility of humans is constantly increasing concerning local journeys of day-to-day life as well as global journeys (e. g. holidays). Already grandparents of today could not imagine covering several hundred kilometers per day to reach their daily business. This can also be seen in the number of vehicles within Germany, which increased from 1953 to 2017 by a factor of 12 [Statistisches Bundesamt (Destatis), 2018].

The higher demand of mobility increases the risk of accidents. For that reason, a standardization of mobility took place (e. g. speed limits, maximum blood alcohol, mandatory seat-belts), which reduced the number of people killed in road traffic accidents and increased the safety of mobility as shown in Figure 1.1. In parallel to the safety standardization, technical systems were developed which prevent accidents or reduce the severity of an accident by assisting the driver of a vehicle. For example, assistance systems such as the **A**nti-lock **B**raking **S**ystem (ABS), **E**lectronic **S**tability **C**ontrol (ESC), or active braking systems were introduced [Reif, 2010].

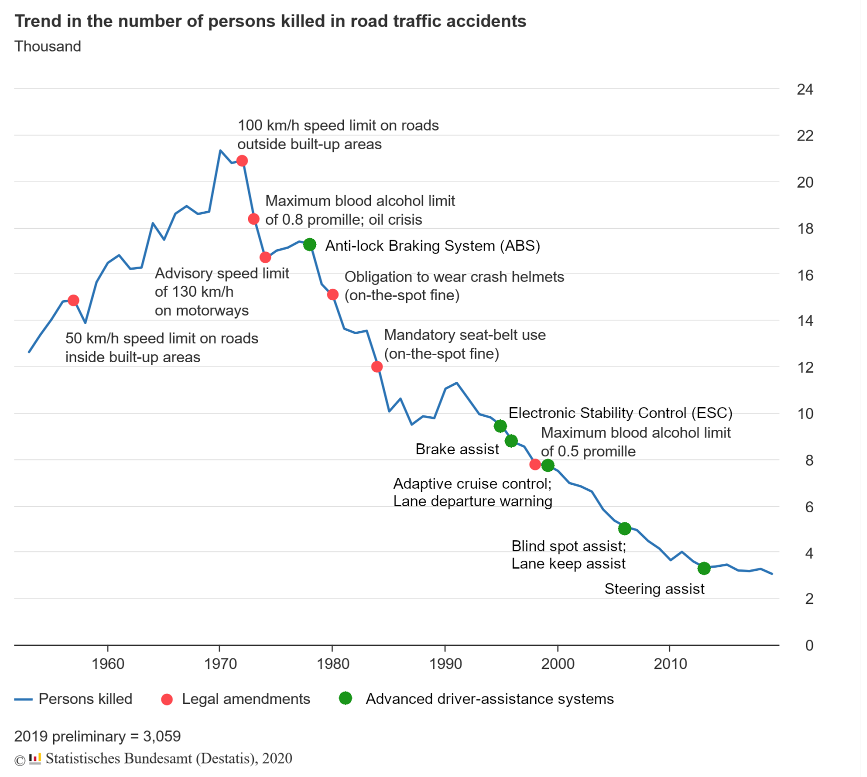


Figure 1.1: Trend in the number of persons killed in road traffic accidents from 1953 to 2019 in Germany, which is decreasing although the number of vehicles increased by a factor of 12 between 1953 and 2017. This figure is based on [Statistisches Bundesamt (Destatis), 2020] and [Verband der Automobilindustrie e. V. (VDA), 2015].

Although the number of people killed decreased over the last few decades, human mistakes are still responsible for 88.4% of the accidents in Germany in which people are injured [Statistisches Bundesamt (Destatis), 2019a]. For that reason, driver assistance systems and automated driver systems are developed further to increase safety in day-to-day mobility.

In general, driver assistance systems and automated driver systems are divided into six levels of automation as shown in Figure 1.2. The highest level of automation is full autonomous driving, which does not need a driver or driver responsibilities anymore. This level of autonomy should not only increase the safety of a vehicle, but also lead to the increased mobility of those who are not able to drive (e.g. impaired or young people). Furthermore, the comfort of mobility is increased by being a passenger instead of being stressed by traffic as a driver.

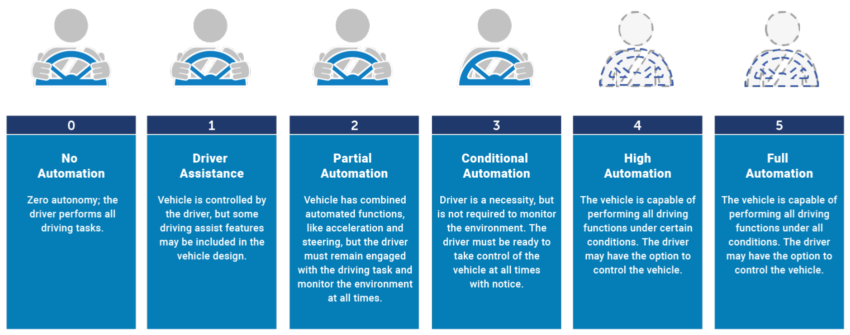

0
No Automation
Zero autonomy; the driver performs all driving tasks.
1
Driver Assistance
Vehicle is controlled by the driver, but some driving assist features may be included in the vehicle design.
2
Partial Automation
Vehicle has combined automated functions, like acceleration and steering, but the driver must remain engaged with the driving task and monitor the environment at all times.
3
Conditional Automation
Driver is a necessity, but is not required to monitor the environment. The driver must be ready to take control of the vehicle at all times with notice.
4
High Automation
The vehicle is capable of performing all driving functions under certain conditions. The driver may have the option to control the vehicle.
5
Full Automation
The vehicle is capable of performing all driving functions under all conditions. The driver may have the option to control the vehicle.

Figure 1.2: Level of automation for a vehicle defined by the Society of Automotive Engineers. The illustration is based on [National Highway Traffic Safety Administration (NHTSA), 2020].

1.1.2 *Modules of Autonomous Driving Platforms*

In realizing autonomous driving, systems are often divided into different modules [Thrun et al., 2006, Urmson et al., 2009, Ziegler et al., 2014], as shown in Figure 1.3. First of all, the autonomous vehicle has sensors to detect and extract information on the environment based on perception systems [Feng et al., 2020]. Afterwards, the sensor information can be combined by fusion approaches to create a holistic representation of the environment such as a **D**ynamic **O**ccupancy **G**rid **Ma**p (DOGMa) [Nuss et al., 2018]. This environmental representation provides the basis for high-level tasks such as localization and object tracking [Vu et al., 2011], which are important to create a well-founded scene understanding and situation analysis [Laugier et al., 2011], including the prediction of other traffic participants. By using these components, a maneuver can be planned through the analyzed environment [Bai et al., 2015]. This allows the cooperation of other traffic participants to be taken into account [Hobert et al., 2015]. Finally, the planned maneuver is executed based on the control system of the vehicle.

1.1.3 *Semantic Labeling*

For calculating a collision-free path through the environment, the autonomous vehicle has to understand the relevant parts of the scene. This includes the detection and classification of all measurements into e. g. movable or non-movable objects as well as even more detailed classes such as vehicles, pedestrians, and roads. This classification of each measurement is known as semantic labeling. It can be applied to a holistic environmental representation such as a DOGMa, as shown by Piewak et al. [2017]. Thereby, each grid cell of the DOGMa is classified as movable or non-movable to increase the performance of object tracking approaches. Other approaches use

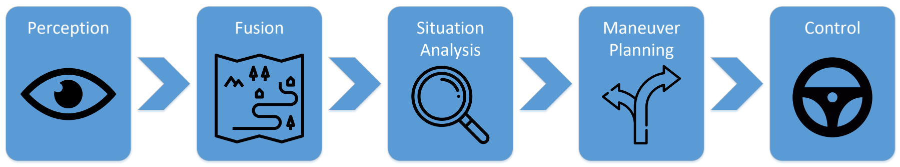


Figure 1.3: Modules of autonomous driving platforms: First the perception based on different sensor types, followed by the fusion of the perceived information to generate a holistic environmental representation. Afterwards, the situation analysis as a base for the maneuver planning. Finally, the execution of the maneuver over the vehicle control system. *Icons are designed by monkik and Freepik from Flaticon.*

semantic labeling as an additional task for detecting objects. Wirges et al. [2018], for example, uses semantic labeling of a DOGMa to detect vehicles, bicycles, and pedestrians. The results in both approaches showing that classification into a small number of classes is feasible on an abstract and compact environmental representation, but for an in-depth scene understanding and classification of the measurements as required for autonomous driving, classification has to be executed in an earlier module of autonomous driving platforms. For this reason, this thesis focuses on extracting multi-class semantic labeling on perception level.

1.1.4 *Perception in Autonomous Driving*

Within the field of autonomous driving, vehicles are typically equipped with different sensors of complementary modalities such as cameras, **Ra**dio **D**etection **a**nd **R**anging (RaDAR) sensors and **Li**ght **D**etection **a**nd **R**anging (LiDAR) sensors (see Figure 1.4) [Thrun et al., 2006, Urmson et al., 2009, Levinson et al., 2011, Wei et al., 2013, Ziegler et al., 2014]. Each sensor modality leverages its specific strengths in order to contribute to the overall geometric and semantic understanding of the scene.

Based on the history of the automotive industry, RaDAR sensors are already well known due to series production since the late 1990s [Meinel, 2014]. Related to the field of semantic labeling, camera images are already well known within the research domain, which aims to classify each pixel in a given camera image [Cordts et al., 2016, Garcia-Garcia et al., 2018, Feng et al., 2020]. By comparison, LiDAR sensors recently entered the market and the research domain [Mei et al., 2019]. However, each sensor modality has to extract as much information as possible from its environment independently of other modalities to maximize overall system performance, availability, and safety. This includes semantic information for generating an in-depth scene understanding. For that reason, this thesis focuses

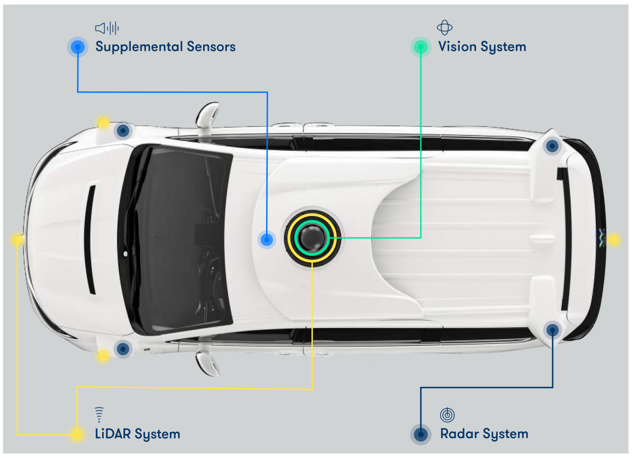


Figure 1.4: Example of a typical autonomous vehicle from Waymo LLC equipped with automotive sensors like cameras, RaDAR sensors, and LiDAR sensors. The illustration is based on [Waymo LLC, 2018]. © 2018 Waymo LLC

on the extraction of semantic information for each measurement based on LiDAR sensors.

1.1.5 *Deep Learning*

In recent years, the corresponding task of semantic labeling of camera images has experienced a significant boost due to the improvements in advanced deep learning techniques [Siam et al., 2017] as compared to classical approaches[1], which often use random forests [Shotton et al., 2008], **C**onditional **R**andom **F**ield (CRF) [Ladický et al., 2009], boosting methods [Shotton et al., 2006], or **S**upport **V**ector **M**achines (SVMs) [Yi Yang et al., 2012]. Thereby, the performance of most of the classical approaches was bound to the hand crafted features provided as an input for the classical approaches [Siam et al., 2017] (e. g. pixel colors [Szeliski, 2011], **H**istogram of **O**riented **G**radients (HOG) features [Dalal and Triggs, 2005], or **S**cale-**I**nvariant **F**eature **T**ransform (SIFT) features [Lowe, 2004]). In contrast, deep learning approaches are learning a complex feature representation leading to outperform classical approaches within the field of semantic labeling in terms of classification performance. This can be observed at several benchmarks such as the PASCAL Visual Object Classes (VOC) Challenges [Everingham et al., 2011, 2012], the NYU Depth Dataset V2 [Silberman et al., 2012], or the Cambridge Driving Labeled Video Database (CamVid) [Brostow et al., 2009]. Within newer benchmarks such as the Cityscapes Benchmark Suite [Cordts et al., 2016] or the semantic segmentation evaluation of the KITTI Vision Benchmark Suite [Abu Alhaija et al., 2018], classical

1 The reader is referred to [Thoma, 2016] and [Zhu et al., 2016] for details about classical approaches.

approaches either do not appear within the benchmark or reach only the last positions (e. g. [Kang and Nguyen, 2019][2]). This thesis therefore focuses on advanced deep learning approaches to be transferred to the field of LiDAR-based semantic labeling.

1.2 CONTRIBUTIONS

Well known within the domain of deep neural networks is the fact that large-scale datasets are of paramount importance for training competitive deep learning approaches [Sun et al., 2017]. Consequently, large-scale datasets such as ImageNet [Russakovsky et al., 2015] and Cityscapes [Cordts et al., 2016] have been made available for this purpose within the camera domain. In contrast, within the LiDAR domain, only a few datasets (most of them without relation to road scenarios or autonomous driving) have been made available in the context of semantic labeling [Silberman et al., 2012, Hackel et al., 2017, Behley et al., 2019]. Therefore, in this thesis, an approach is developed to generate large-scale datasets in an automatic fashion. This reduces not only the cumbersome and time-consuming manual annotation work, but also reduces the costs of generating such large-scale datasets at the same time.

In addition to automatic dataset generation, the semantic labeling approach is developed within the context of autonomous driving, which requires a real-time execution of proposed algorithms.

Overall, three different aspects are focused on in this thesis:

1. Robust, LiDAR-based semantic labeling with high classification performance
2. Automatic training data generation for large-scale datasets in the context of LiDAR-based semantic labeling
3. Real-time execution of the robust LiDAR-based semantic labeling approach

For approaching the three aspects of the thesis, different elements are developed and combined. The generation of a dataset as one aspect represents the basis for the development of semantic labeling approaches. Such a dataset is usually created manually by annotating each measurement. This manual annotation is a time and cost extensive task especially within the **3-D**imensional (3D) space of a LiDAR point cloud, where objects are seen from only one side. These types of point clouds are referred to as semi-dense point clouds. For generating such datasets in an efficient and automated fashion, multi-modal training data generation is proposed based on a camera and a LiDAR sensor. Thereby, the strength of a high-resolution

2 Although Kang and Nguyen [2019] inserted a representation learning component into their random forest approach, the ranking of shows that the classical methods are outperformed by deep learning approaches.

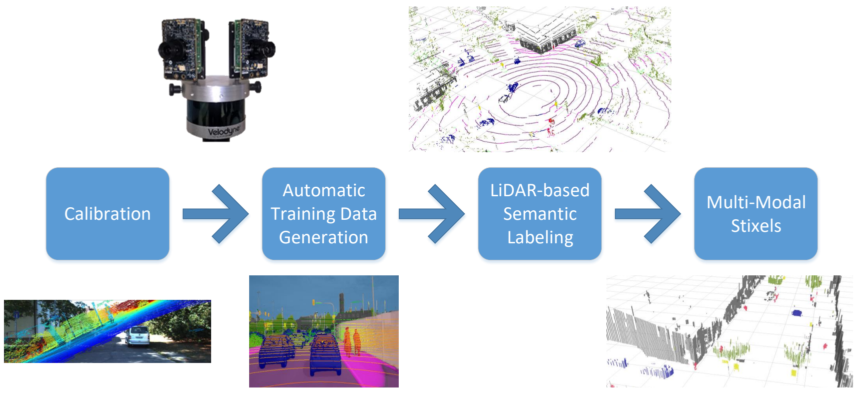

Calibration
Automatic Training Data Generation
LiDAR-based Semantic Labeling
Multi-Modal Stixels

Figure 1.5: Overview of the research pipeline starting with a precise calibration for an automatic multi-modal training data generation, followed by an approach for LiDAR-based semantic labeling and a multi-modal stixel approach as data compression technique at the end. (The images are extracted from the related publications [Schneider et al., 2017, Piewak et al., 2018a,b] © 2017-2018 IEEE)

camera is used to automatically extract labels and transfer them to the LiDAR measurements. To facilitate the transfer of the labels, a precise calibration of the two sensors has to be generated, which represents the first element of the research pipeline before starting the label transfer with a multi-modal training data generation approach (see Figure 1.5).

Afterwards, the automatically created dataset offers the opportunity to develop a robust and real-time capable approach for LiDAR-based semantic labeling. In this context, the well-known semantic labeling approaches of the camera domain [Garcia-Garcia et al., 2018, Feng et al., 2020] are used and adapted to develop a semantic labeling approach based on a LiDAR sensor.

In addition to the developed semantic labeling approach, two extensions are proposed. First, a reduction in sensor dependence based on an alternative deep learning approach is presented. Second, a hierarchical semantic labeling approach is outlined to encode a label hierarchy into the training process of deep neural networks for semantic labeling.

The resulting semantically labeled LiDAR measurements extracted from the developed LiDAR-based semantic labeling approach can then be used to generate a detailed environment model as the basis for high-level tasks of mobile robotics and autonomous driving. However, a measurement cycle of one LiDAR sensor consists of thousands of single measurements that are recorded in a fraction of a second. This decreases the real-time capability of downstream algorithms as well as of the entire system. At the same time, the LiDAR measurements are highly redundant in terms of geometric as well as semantic information. For this purpose, the Stixel-World of the camera domain [Badino et al., 2009, Pfeiffer, 2012, Schneider et al., 2016, Cordts et al., 2017, Hernandez-Juarez et al., 2017], which reduces the amount of data by retaining the underlying geometric and semantic information, is adapted and transferred to the LiDAR domain. Additionally, the approach is extended to directly combine different types of sensors into one Stixel representation. This increases the real-time capability of downstream algorithms while minimizing information loss.

The contributions can be summarized as follows:

- An efficient approach for high-quality semantic labeling of semi-dense LiDAR measurements as provided by state-of-the-art mobile LiDAR sensors. (Chapter 4)
- A large-scale automated cross-modal training data generation process for boosting LiDAR-based semantic labeling performance (Chapter 3)
- A novel deep neural network architecture reducing sensor dependence for high-quality semantic labeling of semi-dense point clouds (Chapter 5)
- An effective training technique encoding label hierarchy into high-quality semantic labeling approaches of semi-dense LiDAR measurements (Chapter 6)
- A compact and robust mid-level representation for semantic LiDAR measurements based on the Stixel-World (Chapter 7)
- A multi-modal fusion approach integrated into the proposed mid-level representation (Chapter 7)
- An in-depth analysis and evaluation of the proposed methods based on different datasets (Chapters 3 to 7)

Note that all contributions except the two extensions of semantic labeling (Chapters 5 and 6) are applied and evaluated in the context of a research vehicle for autonomous driving.

1.3 STRUCTURE OF THESIS

The thesis is structured similarly to the research pipeline of Figure 1.5. Following an introduction in Chapter 1, a technical background is provided in Chapter 2 including background knowledge of LiDAR sensors, artificial neural networks, and evaluation metrics as well as calibration approaches for different sensors as the first step of the research pipeline. Automatic training data generation, which represents the next step of the research pipeline, is covered in Chapter 3. The third part of the research pipeline is separated into three different chapters. Chapter 4 represents the main approach of LiDAR-based semantic labeling, and Chapters 5 and 6 represent the extensions of the LiDAR-based semantic labeling approach in terms of reducing sensor dependence as well as hierarchical semantic labeling. The last step of the research pipeline is mentioned in Chapter 7, where the multi-modal Stixel approach is described. Each main chapter containing steps of the research pipeline is divided into five sections representing an overview, related work, the method, the evaluation, and an outcome. At the end of the thesis, Chapter 8 summarizes the main findings and provides an outlook to future research in this area.

2 TECHNICAL BACKGROUND

CONTENTS

In this chapter, a basic technical background related to LiDAR-based semantic labeling is provided. This includes an introduction to LiDAR systems, **A**rtifical **N**eural **N**etworks (ANNs), evaluation metrics, and calibration of multiple sensor modalities. Parts of the section regarding the calibration of multiple sensor modalities have previously appeared in [Schneider et al., 2017] and [Piewak and Schneider, 2018].

2.1 LIDAR

This section describes LiDAR sensors, including their measurement principle, and presents the LiDAR sensors used within this thesis.

2.1.1 *Measurement Principle*

LiDAR sensors are based on **L**ight **A**mplification by **S**timulated **E**mission of **R**adiation (LASER) systems sending out a pulsed LASER light over an emitter. This LASER light is reflected on a target and returned to a receiver component of the LiDAR sensor

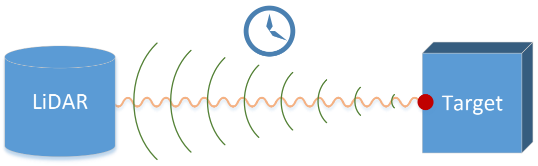


Figure 2.1: Measurement principle of a LiDAR sensor: The LiDAR sensor emits focused LASER light (orange) to a target point (red) and measures the reflection (green). Based on the elapsed time, a precise distance is calculated.

(see Figure 2.1). Based on the time of flight ${}_{\Delta}t_f$ of the emitted LASER light and the known speed of light c, a distance r can be measured with

$$r = \frac{{}_{\Delta}t_f \cdot c}{2} \quad . \tag{2.1}$$

Thereby, the LASER light is focused on a small area being able to measure one specific point as the target. It might be the case that the LASER light is not reflected e. g. by pointing the LASER system to the sky. This is recognized as well by the receiver by generating so-called invalid points without distance.

In a LiDAR sensor, multiple LASER systems or rotating LASER systems are integrated to measure points of the environment based on the precise pose of each LASER system. Each measurement cycle generates a 3D point cloud as shown in Figure 2.2. In addition to the distance of the targeted points, the receiver of modern LASER system measures the amount of received light to provide coarse reflectivity estimates. This represents color information of the used light spectrum similar to the color information extracted by the cones (light-sensitive cells) of the human eye[1]. Typically, the LiDAR sensors use light in the infra-red spectrum. This results in a higher reflectivity of traffic signs and road markings as compared to other obstacles, which increases the attractiveness of a LiDAR system for road scenario use cases.

For further details on LiDAR sensors, the reader is referred to [Rasshofer and Gresser, 2005] and [Wandinger, 2006].

2.1.2 *Sensor Types within this Thesis*

Within this thesis, two main LiDAR sensor types are used to validate the methods developed: A lower resolution Velodyne **VLP-32C** (VLP-32C) [Velodyne LiDAR Inc., 2019a] and a higher resolution Velodyne **VLS-128** (VLS-128) [Velodyne LiDAR Inc., 2019b]. Both sensors contain stacked LASER systems (32 LASER systems for the VLP-32C and 128 LASER systems for the VLS-128) which are

1 The human eye has a different light spectrum as a LASER system. The mentioned comparison is only used as a basic explanation.

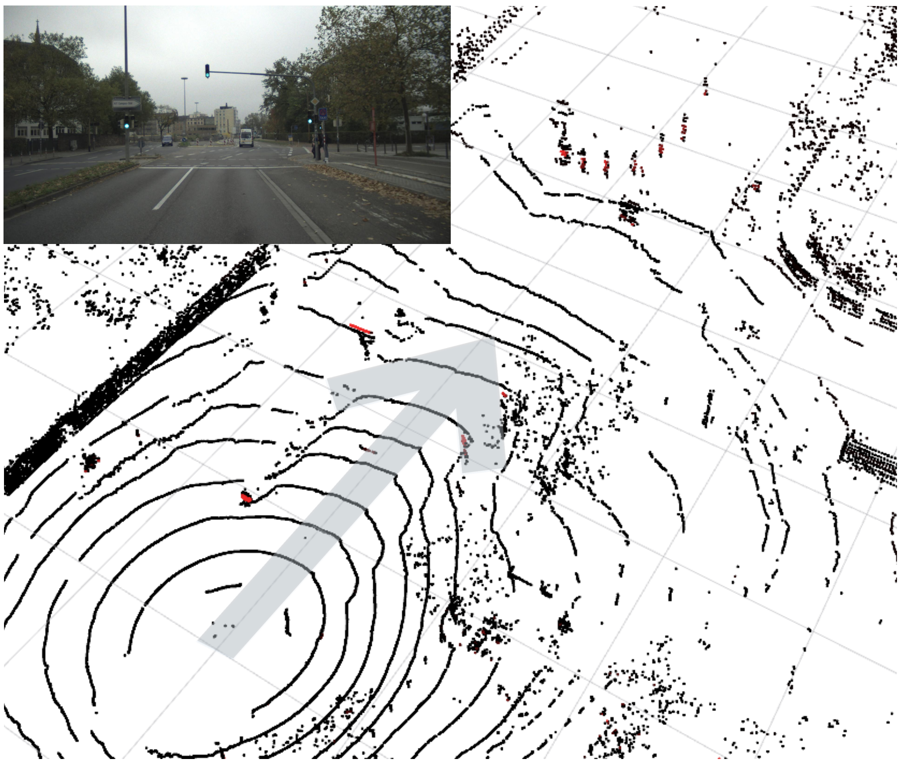


Figure 2.2: Example of a 3D point cloud of a VLP-32C. The color of the point cloud represents the reflectivity estimate obtained by the VLP-32C (black = low reflectivity, red = high reflectivity). The test vehicle is headed to the top right of the figure (arrow), the corresponding camera image is shown on the top left for clarity.

also called layers. These stacked LASER systems are rotating over a vertical axis to generate a 360-degree 3D point cloud as shown in Figure 2.2.

In addition to the two mentioned sensors, the developed method for semantic labeling is evaluated based on a recently published dataset for LiDAR point cloud semantics called SemanticKITTI [Behley et al., 2019], which is based on the Kitti odometry dataset [Geiger et al., 2012a]. This dataset uses a Velodyne **HDL-64** (HDL-64) [Velodyne LiDAR Inc., 2019c], representing a rotating system that contains 64 LASER systems.

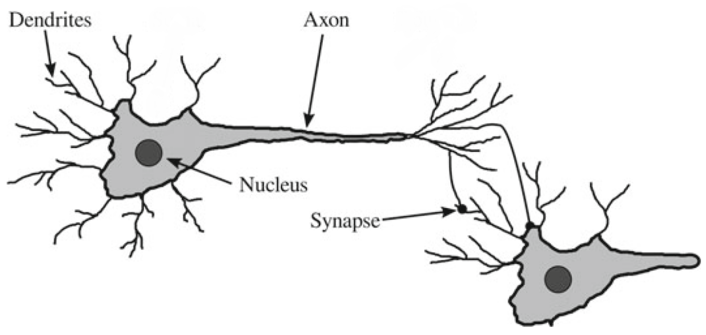


Figure 2.3: Illustration of a biological neuron as a motivation for artificial neurons of an ANN. Only parts that are relevant for this thesis are mentioned. The illustration is based on [Habibi Aghdam and Jahani Heravi, 2017]. *Reprinted/adapted by permission from Springer Nature Customer Service Centre GmbH: Springer, Guide to Convolutional Neural Networks by Habibi Aghdam and Jahani Heravi, © 2017.*

2.2 ARTIFICIAL NEURAL NETWORKS

Within this section a brief introduction to ANNs is provided as a basis for the next chapters. For an in-depth introduction to this topic, the reader is referred to [Goodfellow et al., 2016], [Habibi Aghdam and Jahani Heravi, 2017], or [Skilton and Hovsepian, 2018].

2.2.1 *Biological Motivation*

Since the invention of machines, there is an effort to increase their intelligence for solving problems adaptively without human interaction. The paradigm for this kind of machine are living beings, including the humans with their cerebral nervous system. The minimal module of such nervous systems is the biological neuron [Habibi Aghdam and Jahani Heravi, 2017] as shown in Figure 2.3.

The biological neuron is mainly divided into 4 parts [Kandel et al., 2013, Habibi Aghdam and Jahani Heravi, 2017]: Dendrites, nucleus, axon, and synapses. The dendrites represent the input of a neuron by being connected to other neurons over synapses or being connected directly to biological inputs such as an eye. These inputs are then collected within the nucleus by accumulation. The inputs are thereby represented over electrical signals and can be stimulating (positive influence) or inhibiting (negative influence), depending on the synaptic strengths of the input signal [Kandel et al., 2013]. In case the nucleus reaches a certain threshold, it sends out an electrical pulse of a certain intensity. The intensity also depends on the accumulated inputs. The electrical pulse is sent through the axon, ending at multiple synapses which are connected to other biological neurons.

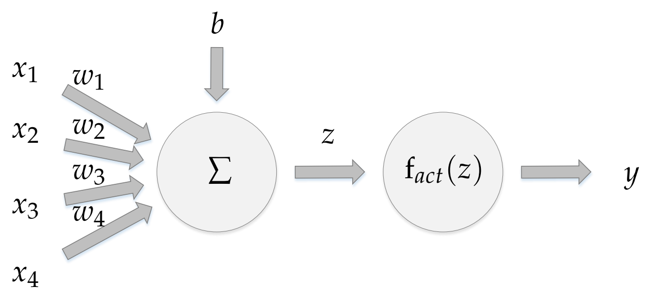


Figure 2.4: Example of an artificial neuron ω as part of an ANN.

2.2.2 *Artificial Neuron*

The aforementioned biological neuron is transferred to an artificial representation as shown in Figure 2.4. The dendrites are represented by multiple inputs $\boldsymbol{x}_{nb} = (x_1, x_2, \dots, x_n)$. The synaptic strength of the inputs is modeled over weights $\boldsymbol{w}_{nb} = (w_1, w_2, \dots, w_n)$ for each of the inputs. These weighted inputs are accumulated, which correspond to the operation of the nucleus. In addition to the accumulated inputs, a bias $b = -\Xi$ is added which replaces the threshold Ξ of the nucleus. At the end, an activation function $\mathrm{f}_{act}(z)$ is executed representing the signal which is sent through the axon to the synapses and other neurons [Habibi Aghdam and Jahani Heravi, 2017]. As a result, the artificial neuron ω can be formulated as

$$z = \boldsymbol{w}_{nb}\boldsymbol{x}_{nb} + b \tag{2.2}$$
$$y = \mathrm{f}_{act}(z) = \mathrm{f}_{act}(\boldsymbol{w}_{nb}\boldsymbol{x}_{nb} + b) \quad . \tag{2.3}$$

Simplifying further processing of the neuron, the bias is represented as separate weight $w_0 = b$, while the corresponding input is set to $x_0 = 1$ [Goodfellow et al., 2016, Tino et al., 2015]. As a result, the Equations (2.2) and (2.3) can be transformed to

$$z = \boldsymbol{w}\boldsymbol{x} \tag{2.4}$$
$$y = \mathrm{f}_{act}(z) = \mathrm{f}_{act}(\boldsymbol{w}\boldsymbol{x}) \quad , \tag{2.5}$$

while the inputs $\boldsymbol{x} = (x_0, x_1, \dots, x_n)$ as well as the weights $\boldsymbol{w} = (w_0, w_1, \dots, w_n)$ include the bias. Note that z represents the inner state of the artificial neuron before executing the activation function.

Similar to the biological archetype, the artificial neuron is flexible in terms of changing its behavior [Habibi Aghdam and Jahani Heravi, 2017]. This is realized by adapting the weights $\boldsymbol{w}$ including the bias b. This process is called learning.

The activation function can in general be an arbitrary function which fulfills the requirements of the learning rule[2], improving the

2 The activation function has to be e. g. partially differentiable to fulfill the requirements of the learning rule (see Section 2.2.4).

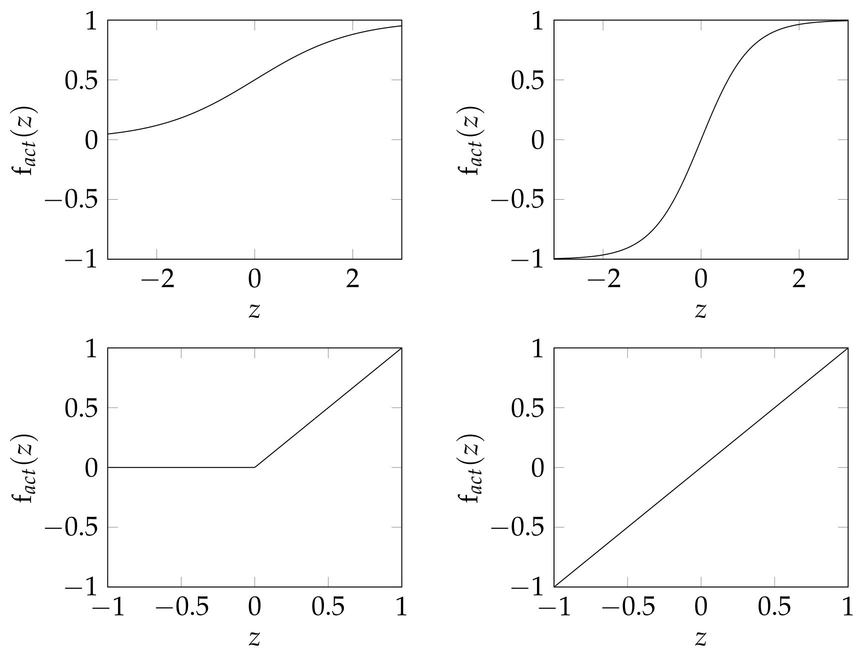


Figure 2.5: Example of activation functions of artificial neurons: Top left: Sigmoid function, top right: Hyperbolic tangent function, bottom left: ReLU function, bottom right: Identity function.

learning process or the execution speed of the neuron computation. Commonly used activation functions are e. g. the sigmoid, the hyperbolic tangent, the **Re**ctified **L**inear **U**nit (ReLU), or the identity function (see Figure 2.5) [Habibi Aghdam and Jahani Heravi, 2017, Goodfellow et al., 2016]. Over the last few decades, the ReLU functions became the activation function of choice caused by its simplicity which heavily accelerates training [Krizhevsky et al., 2012] and improves the training procedure [Maas et al., 2013] of large ANNs. Additionally, non-linear activation functions are required to learn arbitrary non-linear tasks. This property is fulfilled by the ReLU function.

2.2.3 *Feed Forward Networks*

With the definition of an artificial neuron, more complex neural networks can be defined by combining multiple artificial neurons similar to the biological archetype. Hence, two general types of ANNs exist which depend on the connection type of the different neurons [Habibi Aghdam and Jahani Heravi, 2017]. By connecting the neurons to a directed cyclic graph (e. g. see Figure 2.6), a recurrent neural network is created. This type of ANN is not related to this thesis and the reader is referred to [Goodfellow et al., 2016] for more details. The other neuron combination is represented by

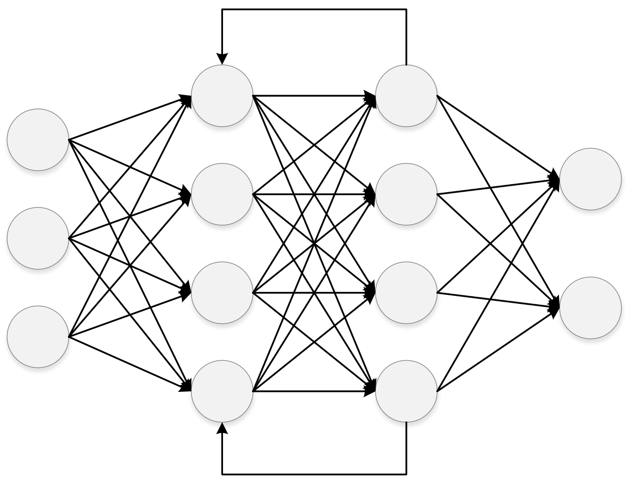


Figure 2.6: Example of a recurrent network represented by a directed cyclic graph. In comparison to Figure 2.7, this ANN has a connection from the third to the second layer, representing a cycle of a recurrent network.

a directed acyclic graph (e. g. see Figure 2.7) and is known as **F**eed **F**orward **N**etwork (FFN).

FFNs are typically structured in layers [Goodfellow et al., 2016]. Thereby, each layer L_i contains neurons which are only connected to the previous layer L_{i-1} (see Figure 2.7). The first layer which does not have any previous layer is called the input layer and represents the input of the ANN. The last layer which is not the input for another layer represents the output of the ANN and is called the output layer containing output neurons. The other layers are called hidden layers.

A specific type of layer is the fully connected layer [Skilton and Hovsepian, 2018], whereby each neuron within one layer is connected to all neurons of the previous layer. This type of layer is usually used within FFNs to homogenize the processing of the ANN.

In addition to the aforementioned definition of FFNs, neurons can be connected to previous layers by skipping the connections to the direct predecessor layer [Goodfellow et al., 2016]. This type of connection is called shortcut or skip connections as shown in Figure 2.8.

With these definitions, large ANNs can be created which are able to regress arbitrary non-linear functions per output neuron. Within this thesis, the focus is to generate a classification into multiple classes. For this reason, the FFN has to be extended for classification. This is usually realized by extracting a score y_i (also called a logit) based on an output neuron per class c_i. Afterwards, this score is

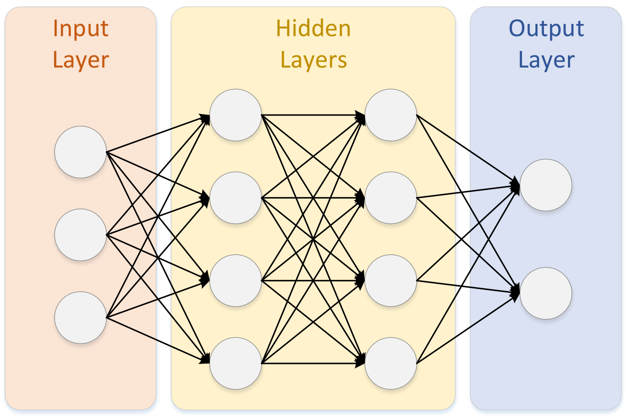


Figure 2.7: Example of an FFN represented by a directed acyclic graph. The network is composed of one input layer, two hidden layers, and one output layer.

normalized over the softmax function [Bishop, 2006] to generate a probability per class with

$$Pr(c_i|\boldsymbol{y}) = \frac{e^{y_i}}{\sum_{j=0}^{M} e^{y_j}} \quad , \tag{2.6}$$

while $\boldsymbol{y} = (y_0, y_1, \ldots, y_{M-1})$ represents the output of neurons within the output layer and the number of neurons M represents the number of classes. This normalization step is also called the softmax layer. Combining the softmax layer with the fully connected output layer results in the multinominal logistic function [Bishop, 2006, Habibi Aghdam and Jahani Heravi, 2017]

$$Pr(c_i|\boldsymbol{x}) = \frac{e^{w_i \boldsymbol{x}}}{\sum_{j=0}^{M} e^{w_j \boldsymbol{x}}} \quad , \tag{2.7}$$

which represents the multi-class classification within the probability theory. Consider that the activation function of the output layer is set to the identity function [Habibi Aghdam and Jahani Heravi, 2017] in contrast to the other layers, which typically use the ReLU function as an activation function. Finally, the classification can be solved by extracting the class with the highest probability based on an argmax layer, which applies an argmax function that selects the largest element.

2.2.4 *Learning Rule*

Given a defined network structure, the ANN has to learn the desired classification outputs $\boldsymbol{\psi} = (\psi_0, \psi_1, \ldots, \psi_{L-1})$ given corresponding input samples $\boldsymbol{\chi} = (\chi_0, \chi_1, \ldots, \chi_{L-1})$ for L training pairs by adaption of the weights $\boldsymbol{w}$ of the ANN. This type of learning technique

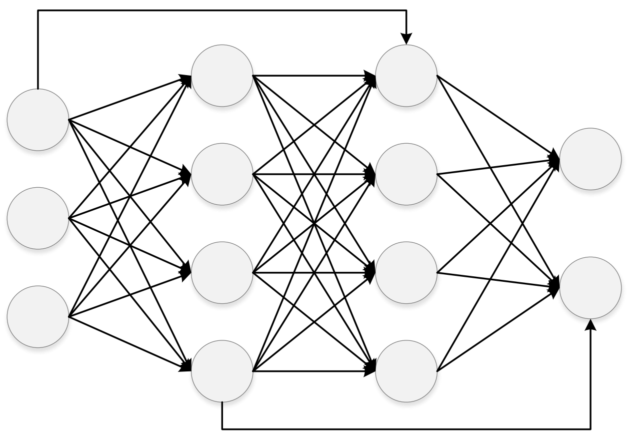


Figure 2.8: Example of an FFN with skip connections. In comparison to Figure 2.7, this ANN has a connection from the input layer to the second hidden layer and from the first hidden layer to the output layer, representing skip connections.

is called supervised learning, which is solved based on gradient descent approaches within the domain of ANNs [Goodfellow et al., 2016, Habibi Aghdam and Jahani Heravi, 2017].

The learning rule is separated into two steps. First, the outputs $\boldsymbol{y}_o$ of the ANN are processed based on the given input sample χ_o and the error $\mathrm{E}(\boldsymbol{y}_o, \psi_o)$ is calculated. Afterwards, the error is propagated back to each weight of the ANN to be adapted with a gradient descent-based approach.

The error $\mathrm{E}(\boldsymbol{y}_o, \psi_o)$ can be calculated with every function such as the mean squared error[3] [Goodfellow et al., 2016, Habibi Aghdam and Jahani Heravi, 2017] after processing the output of the ANN. Within the context of multi-class classification, the cross-entropy loss function is typically used as an error function which directly utilizes the output of the softmax layer for all classes (see Equation (2.7))

$$\mathrm{E}_{CES}(\boldsymbol{y}_o, \psi_o) = -\sum_{k}^{M} \rho_{\psi_o,k} \ln y_{o,k} \tag{2.8}$$

with

$$\begin{aligned} \boldsymbol{y}_o &= (y_{o,0}, y_{o,1}, \ldots, y_{o,M-1}) \\ &= (Pr(c_0|\chi_o), Pr(c_1|\chi_o), \ldots, Pr(c_{M-1}|\chi_o) \ . \end{aligned} \tag{2.9}$$

$\rho_{\psi_o,k}$ represents the elements of a binary vector with the length corresponding to the number of classes M and the values set to zero

3 For using the mean squared error, the desired classification output ψ_o has to be expanded to a binary vector with the length corresponding to the number of predicted classes and where all values are zero besides the index of the desired class, which is set to one.

except the index k of the desired class ψ_o, which is set to one as defined with

$$\rho_{\psi_o,k} = \begin{cases} 1, & \text{if } k = \psi_o \\ 0, & \text{otherwise} \end{cases} . \tag{2.10}$$

This definition indicates that only one class represents the true class. Expanding the error function from a single classification task to multiple classification tasks, the cross-entropy function is defined as [Bishop, 2006]

$$\mathrm{E}_{CE}(\boldsymbol{y}, \boldsymbol{\psi}) = -\sum_{n}^{N}\sum_{k}^{M} \rho_{\psi_n,k} \ln y_{n,k} \quad , \tag{2.11}$$

while $\rho_{\psi_n,k}$ represents the elements of a matrix which correspond to a binary vector similar to Equation (2.10) for each classification n.

The expansion to multiple classifications N can be carried out by executing multiple classifications for one sample (such as a classification for each measurement as performed in this thesis) by processing multiple samples at once or both in combination. The processing of multiple samples at once is a common regularization technique called batch normalization to increase the training speed as well as the performance.

The second step of the learning rule is to adapt the weights $\boldsymbol{w}$ of the ANN by iteratively applying a gradient-based method proposed by Rumelhart et al. [1986], who generalized the delta rule for an ANN with a single layer. Therefore, the weight w_{ij} connecting the neurons ω_i and ω_j is adapted with

$$\Delta w_{ij} = -\eta \frac{\delta \mathrm{E}}{\delta w_{ij}} = -\eta \frac{\delta \mathrm{E}}{\delta z_j} \frac{\delta z_j}{\delta w_{ij}} = -\eta \frac{\delta \mathrm{E}}{\delta z_j} y_i \tag{2.12}$$

based on the learning rate η, which represents the step size of the learning based on the gradient $\frac{\delta \mathrm{E}}{\delta w_{ij}}$. The learning rate controls the speed and the convergence of the learning method. Note that y_i represents the output of neuron ω_i. Furthermore, $\frac{\delta \mathrm{E}}{\delta z_j}$ is represented by

$$\frac{\delta \mathrm{E}}{\delta z_j} = \frac{\delta \mathrm{E}}{\delta y_j} \frac{\delta y_j}{\delta z_j} = \frac{\delta \mathrm{E}}{\delta y_j} \mathrm{f}'_{act}(z_j) \quad , \tag{2.13}$$

while $\mathrm{f}'_{act}(z_j)$ describes the derived activation function of neuron ω_j and $\frac{\delta \mathrm{E}}{\delta y_j}$ describes the partial derivative of the error function (e. g. cross-entropy function) with respect to the output y_j of neuron ω_j. Rumelhart et al. [1986] continued applying the chain rule of derivative to calculate the gradient for all weights within an ANN.

Based on the proposed backpropagation of error approach [Rumelhart et al., 1986], several adapted and improved methods have been published which decrease the training time and enable a better convergence of the gradient-based approach e. g. by dynamically adapting the learning rate. The current state-of-the-art technique is the Adam optimizer [Kingma and Ba, 2015], which is also used in this thesis.

2.2.5 *Convolutional Neural Networks*

The definition of the FFNs makes it possible to extract characteristics called features from an input for classification. Thereby, each feature extractor is represented with the weights corresponding to a neuron. These are related to the full previous layer of the neuron or the full input of the FFN. In contrast, within the cerebral nervous system of living beings (e. g. feature extraction of an eye), features are extracted on a local neighborhood which is also called the receptive field of the feature extractor. For example, the human can recognize a bird within the left part of an image as well as on the right part with the same feature extractor. This translation invariant behavior was first technically replicated by Fukushima [1980]. Afterwards, LeCun et al. [1998] extended this replication with the learning component of an ANN and created the **C**onvolutional **N**eural **N**etwork (CNN). In this context, the neurons of a layer are connected only to a part of the neurons of the previous layer to extract features of a local neighborhood. For enforcing the same feature extractor over the entire input, weight sharing between different neurons was introduced. As a result, this kind of feature extractor can be formulated as a convolution with

$$\mathbf{Z}_{i,j} = \mathbf{W} * \mathbf{X} = \int_k \int_o \mathbf{W}_{k,o} \mathbf{X}_{i-k,j-o} \tag{2.14}$$

for the **2-D**imensional (2D) case, where **Z** represents the 2D output for a specific feature extractor **W** (also called a filter within the context of CNNs). **X** represents the input of the feature extractor. In terms of a discrete input such as a 2D image, the equation becomes a sum with

$$\mathbf{Z}_{i,j} = \mathbf{W} * \mathbf{X} = \sum_k \sum_o \mathbf{W}_{k,o} \mathbf{X}_{i-k,j-o} \quad . \tag{2.15}$$

This formulation replaces Equation (2.4) to form the convolution layer of a CNN including the activation function. Consider that the input image which is also called a feature map can be of arbitrary dimensionality (e. g. an RGB image with two dimensions to determine its pixel position has three color channels: red, green, and blue), while Equation (2.15) convolves only 2 dimensions. The other dimensions are treated over the scalar product of the feature extractor **W** and the input **X**.

Compared to conventional computer vision techniques [Szeliski, 2011], the convolution layer represents similar image processing techniques as e. g. a Canny edge detector, except the fact that the filter is not defined manually, but implicitly by adapting the filter weights $\mathbf{W}$ during learning.

The output shape $H_{out} \times W_{out}$ of the convolution formulated in Equation (2.15) depends on the one hand on the size of the input feature map $H_{in} \times W_{in} \times C_{in}$ and the size of the filter $H_{\mathbf{W}} \times W_{\mathbf{W}} \times C_{in}$. On the other hand, it depends on two additional parameters: The stride δ, which defines the step size of the filter configuring the calculation of the convolution for every δ^{th} element (representing the increment of the sums of Equation (2.15)) and the padding g, which represents the extension at the border of the image. The extended elements can be e. g. zero for *zero padding*. Within this thesis, the *same padding* is used, which replicates the border elements of the input feature map. The dimension H of the output shape is defined with

$$H_{out} = \frac{H_{in} - H_{\mathbf{W}} + 2g}{\delta} + 1 \ , \tag{2.16}$$

correspondingly to the dimension W. The padding is typically used to ensure the same shape at the output of the convolution by setting it to

$$g = \frac{H_{\mathbf{W}} - 1}{2} \tag{2.17}$$

while using a stride of $\delta = 1$. This configuration is used within this thesis, unless explicitly stated otherwise.

By applying multiple filters per layer (similar to multiple neurons per layer within an FFN), the output feature map of the convolution layer becomes 3D with $H_{out} \times W_{out} \times C_{filter}$, while C_{filter} represents the number of filters as shown in Figure 2.9. Note that Figure 2.9 also shows the receptive field at the convolution layer, which represents the local neighborhood used as an input for the filter. Combining the receptive fields of each layer represents the receptive field of the entire CNN.

Based on the defined convolution layer, feature extractors can be learned with the learning rule of Section 2.2.4. These features are related to a specific size of the feature map. For learning feature extractors of different input features map sizes, LeCun et al. [1998] used pooling layers to reduce the size of the feature maps and increase the receptive field of the CNN. These pooling layers combine a number of elements within the feature map by applying an arbitrary operation. Commonly, the resulting element is calculated by choosing the mean, minimum, or maximum of the according region. Typically, a maximum operation is selected as for this thesis. As a result, the feature map size can be reduced as shown in Figure 2.10. The learned feature extractors of different input feature map sizes represent different abstractions of features [Zeiler and Fergus, 2014], similar to conventional computer vision techniques [Szeliski, 2011], which extract low-level features e. g. edges or blobs as a first step

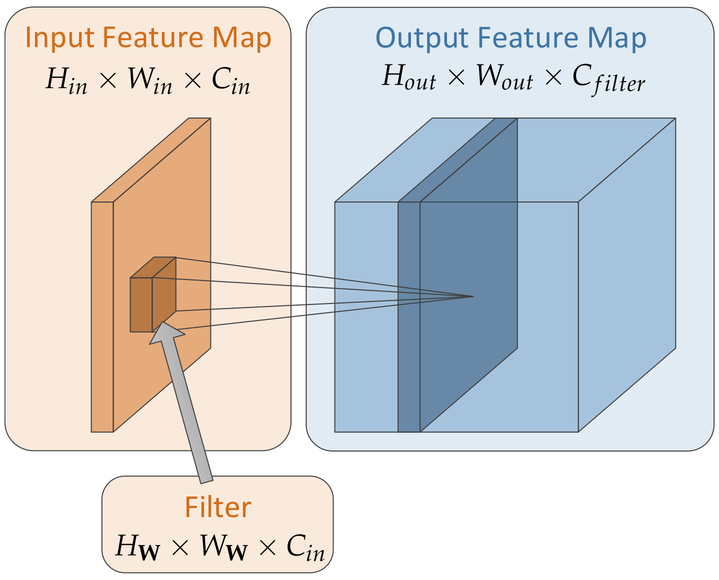

Input Feature Map
$H_{in} \times W_{in} \times C_{in}$
Output Feature Map
$H_{out} \times W_{out} \times C_{filter}$
Filter
$H_W \times W_W \times C_{in}$

Figure 2.9: Visualization of a convolution layer, whereby one filter, his input, and his output is highlighted.

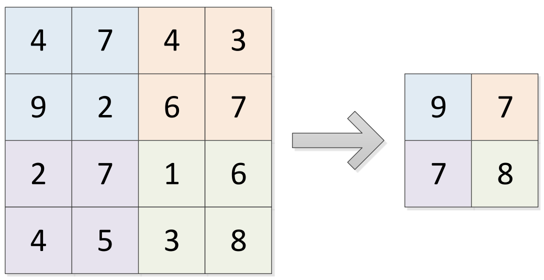


Figure 2.10: Visualization of a maximum pooling layer. The elements of the 2D feature map with the same color are combined by a maximum operation.

as well as high-level features e. g. human heads by combining the low-level features (see Figure 2.11).

Combining convolution and pooling layers with fully connected layers at the end of the network, LeCun et al. [1998] introduced the first trainable CNN for image classification (see Figure 2.12). Following his approach, several state-of-the-art CNN architectures were developed in the task of image classification such as AlexNet [Krizhevsky et al., 2012], VGG-net [Simonyan and Zisserman, 2015] or GoogLeNet [Szegedy et al., 2015].

2.2.6 *Fully Convolutional Neural Networks*

The CNNs presented use fully connected layers at the end of the network to reduce the size of the feature map to one output representing e. g. the classification of a complete image. In terms of e. g. image region classification, the output has to be represented

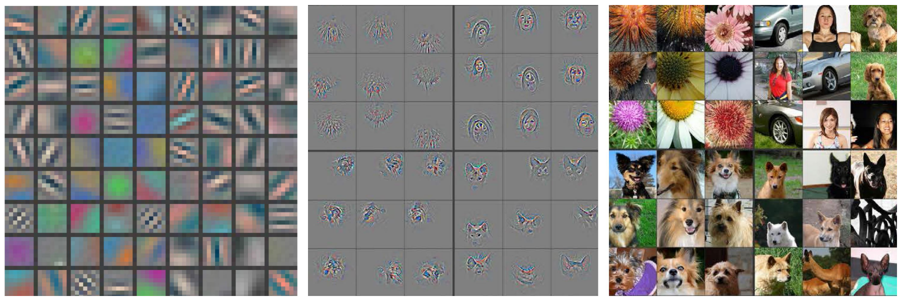


Figure 2.11: Visualization of CNN filters by Zeiler and Fergus [2014]. The CNN extracts low-level features within the first layers (left) and high-level features within the last layers (middle). For illustration, the input images which activates the high-level features (middle) are shown on the right. The images are based on [Zeiler and Fergus, 2014]. *Reprinted/adapted by permission from Springer Nature Customer Service Centre GmbH: Springer, Visualizing and Understanding Convolutional Networks by Zeiler and Fergus, © 2014.*

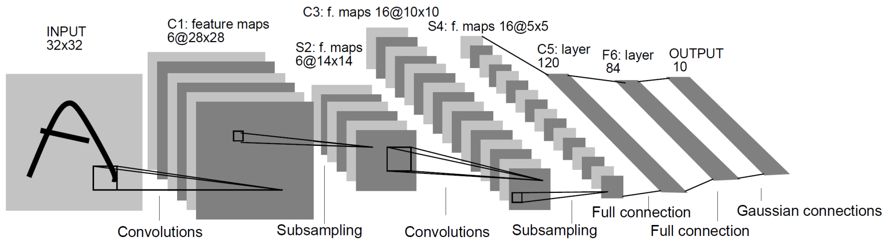


Figure 2.12: First trainable CNN architecture introduced by LeCun et al. [1998] containing convolution layers, pooling layers (called subsampling), and fully connected layers. The numbers at the top represent the sizes of the feature maps. The image is extracted from [LeCun et al., 1998]. © 1998 IEEE

by more than one classification, depending on the input shape. For that purpose, the fully connected layers are replaced by convolution layers, which was initially done by Matan et al. [1991] for the **1-D**imensional (1D) case and Wolf and Platt [1994] for the 2D case. As a result, the **F**ully **C**onvolutional Neural **N**etwork (FCN) can handle arbitrary sizes of inputs, in contrast to the CNNs containing fully connected layers, which require a fixed number of neurons as an input.

Note that the result of a fully connected layer is equivalent to a convolution layer with the filter size equal to the input feature map size, which reduces the outlay of converting a CNN to an FCN.

2.2.7 *Fully Convolutional Neural Network for Semantic Labeling*

The goal of semantic labeling is to classify each measurement of a certain input. In terms of an image, each pixel has to be classified.

With the presented FCNs, the output size is reduced by pooling layers or by the stride of the convolution layer. For this purpose, Long et al. [2015] used the transposed convolution layer[4] of Zeiler and Fergus [2014], which was initialized with a bilinear upsampling strategy to increase back the size of the feature maps to end up with the same output dimension as the input of the network. With this so-called encoder-decoder architecture of an FCN, it is possible to train semantic labeling approaches supervised in an end-to-end manner. As a result, the research within the field of image-based semantic labeling was accelerated [Siam et al., 2017, Garcia-Garcia et al., 2018]. Classical approaches for semantic labeling were outperformed by advanced deep learning techniques due to the ability of learning a complex feature representation compared to hand crafted features used within classical approaches as mentioned in Section 1.1.5. At the same time, the advanced deep learning approaches require a large amount of data to learn such a complex feature representation [Sun et al., 2017, Zhu et al., 2016] based on the learning rule described in Section 2.2.4. However, the performance benefits preponderate particularly within large-scale outdoor scenarios as within the context of autonomous driving and mobile robotics [Siam et al., 2017].

4 Sometimes the transposed convolution layer is also called the deconvolution layer, however a deconvolution represents a different mathematical operation.

2.3 EVALUATION METRICS

This section describes different evaluation metrics for semantic labeling related to this thesis. This includes accuracy, **I**ntersection **o**ver **U**nion (IoU), and confusion matrices. Other metrics used within this thesis are explained within the corresponding chapters. For further details, the reader is referred to [Rahman and Wang, 2016] and [Habibi Aghdam and Jahani Heravi, 2017].

2.3.1 *Terminology*

Evaluating semantic labeling approaches, different types of errors and successes can be extracted. For this reason, different terminologies are defined based on the difference between the predicted class c_p and the true class c_{gt} (also called ground-truth), while considering a specific class c_c (see Table 2.1):

- **T**rue **P**ositive (TP): A sample is classified correctly as considered class ($c_c = c_p = c_{gt}$)
- **T**rue **N**egative (TN): A sample is not classified as considered class, but is classified correctly ($c_c \neq c_p = c_{gt}$)
- **F**alse **P**ositive (FP): A sample is incorrectly classified as considered class ($c_c = c_p \neq c_{gt}$)
- **F**alse **N**egative (FN): A sample is incorrectly classified as another class instead of being classified as considered class ($c_c = c_{gt} \neq c_p$)

Based on this terminology, different performance measures are defined for classification of multiple classes as described within the next subsections.

2.3.2 *Confusion Matrix*

The confusion matrix visualizes the distribution of predicted classes per true classes as shown in Tables 2.2 and 2.3. This allows an in-depth analysis of incorrect classifications in terms of confusion between different classes. The ideal confusion matrix is an identity

		prediction	
		positive	negative
true label	positive	TP	FN
	negative	FP	TN

Table 2.1: Terminology of errors and successes for a binary classifier.

		prediction			
		vehicle	person	two-wheeler	road
true label	vehicle	✓	✗	✗	✗
	person	✗	✓	✗	✗
	two-wheeler	✗	✗	✓	✗
	road	✗	✗	✗	✓

Table 2.2: General description of a confusion matrix. Usually the confusion matrix is filled with probabilities. Here, the content describes the errors (✗) and successes (✓) independent of the considered class c_c.

		prediction			
		vehicle	person	two-wheeler	road
true label	vehicle	TP	FN	FN	FN
	person	FP	TN	TN	TN
	two-wheeler	FP	TN	TN	TN
	road	FP	TN	TN	TN

Table 2.3: Description of a confusion matrix considering the class vehicle. Usually the confusion matrix is filled with probabilities. Here, the content describes the errors and successes for the considered class c_c = vehicle while the predicted class c_p corresponds to the columns and the true class c_{gt} corresponds to the rows of the matrix. Note that all TNs are correctly classified related to the class vehicle but could contain errors related to other classes.

matrix representing that all samples are classified correctly as the true class (diagonal of the matrix) and no FP or FN exists. Typically, each row of the confusion matrix is normalized with the total number of samples per true class. This allows for a comparison without taking the dataset class distribution into account.

2.3.3 *Accuracy*

The confusion matrix represents an in-depth analysis of predicted and true classes. For directly comparing different approaches, a more general performance measure has to be defined. Therefore, the accuracy is represented by

$$\begin{aligned} \mathrm{acc}(c_i) &= \frac{\text{correctly classified samles}}{\text{all samples}} \\ &= \frac{TP_{c_i} + TN_{c_i}}{TP_{c_i} + TN_{c_i} + FP_{c_i} + FN_{c_i}} \end{aligned} \tag{2.18}$$

for a specific class c_i. This definition allows to directly compare different approaches based on a single performance measure per class, while the accuracy $\mathrm{acc}(c_i) = 1$ represents the highest and the accuracy $\mathrm{acc}(c_i) = 0$ the lowest performance concerning class c_i.

2.3.4 *Intersection over Union*

The disadvantage of the accuracy is the strong dependence on the dataset distribution. In other words, the accuracy of a class can be easily increased by adding samples of an easier class, increasing the number of TNs. For this reason, the IoU was defined to reduce the dependence on dataset distributions with

$$\begin{aligned} \mathrm{IoU}(c_i) &= \frac{\text{Intersection of predicted and true samples of } c_i}{\text{Union of predicted and true samples of } c_i} \\ &= \frac{TP_{c_i}}{TP_{c_i} + FP_{c_i} + FN_{c_i}} \quad , \end{aligned} \tag{2.19}$$

which became state-of-the-art for multi-class semantic labeling. In addition to the class-based IoU, a performance measure is used averaging all classes to a **m**ean **I**ntersection **o**ver **U**nion (mIoU) with

$$\mathrm{mIoU} = \frac{1}{M} \sum_{c_i=c_0}^{c_{M-1}} \mathrm{IoU}(c_i) \quad . \tag{2.20}$$

This performance metric is mainly used within this thesis for comparing different semantic labeling approaches.

2.4 CALIBRATION OF MULTIPLE SENSOR MODALITIES

In Section 1.1.4, the perception of autonomous driving is introduced with different sensors of complementary modalities. Fulfilling the needs of autonomous vehicles, these sensors have to be precisely calibrated in space and time to allow a common environmental representation. The calibration in time is often realized as a time synchronization to ensure a common time reference of the sensor measurements [Sivrikaya and Yener, 2004]. The spatial calibration is composed of two parts: The intrinsic and the extrinsic calibration of the sensor. The intrinsic calibration is sensor type-specific and well known in e. g. camera [Hemayed, 2003] or LiDAR domain context [Muhammad and Lacroix, 2010]. It represents internal parameters of the sensor, which can be initially defined by the manufacturer and adapted afterwards by an intrinsic calibration. The extrinsic calibration leads to precise positioning of the sensor related to another sensor or a reference coordinate system and is represented by an affine transformation matrix. Thereby, two types of extrinsic calibration are distinguished: Offline and online calibration. The offline calibration is performed before the sensor system starts to operate as an initial calibration of the system. Once the sensor system goes online, e. g. as a product or test fleet vehicle, external forces such as mechanical vibrations or temperature changes may decrease the calibration quality. In this case, the system has to detect and correct such decalibrations. This is referred to as online calibration.

The extrinsic calibration has been studied for a variety of sensor modalities and combinations. Most approaches can be divided into three steps:

1. Find distinct features in the sensor data, e. g. corners or artificial targets
2. Use those features to establish correspondences between the sensors
3. Given the correspondences, determine the affine transformation matrix by solving a system of equations or by minimizing an error function

The extraction of distinct features can be challenging as correspondences have to be made across different sensor modalities. Most offline calibration approaches therefore rely on special calibration targets which provide strong and distinct signals in all modalities, allowing for easy detection and localization [Geiger et al., 2012b, Mirzaei et al., 2012, Zhang and Pless, 2004]. However, those approaches are time-consuming as they need human interaction for feature selection or they have to be performed in a controlled environment. Therefore, several online calibration methods have been proposed [Bileschi, 2009, Levinson and Thrun, 2013, Pandey et al., 2015, Chien et al., 2016]. Challenging for online calibrations is to find matching patterns in an unstructured environment. Most of the state-of-the-art approaches address this by using handcrafted features such as image edges. An alternative was proposed by Schneider et al. [2017] and Piewak and Schneider [2018], who presented an online calibration based on CNNs, which learn features of an unknown but structured environment.

A precise calibration based on the proposed techniques represents the first step of the research pipeline (see Figure 1.5), allowing cross-modal training data generation for LiDAR-based semantic labeling approaches as described within the following chapter.

3 AUTOMATIC TRAINING DATA GENERATION

CONTENTS

Within this chapter, the second step of the research pipeline (see Figure 1.5) called *Autolabeling* is described. Parts of this chapter have previously appeared in [Piewak et al., 2018b] and [Piewak et al., 2019].

3.1 OVERVIEW

After generating a precise calibration of multi-sensor systems as described in Section 2.4, in the next step of the research pipeline (see Figure 1.5), automatic training data generation can be approached. Here, large-scale datasets are of paramount importance for training competitive deep neural networks [Sun et al., 2017]. Consequently, large-scale generic datasets such as ImageNet [Russakovsky et al., 2015] and COCO [Lin et al., 2014], as well as medium-scale datasets dedicated to road scenarios such as KITTI [Geiger et al., 2012a] and Cityscapes [Cordts et al., 2016] have been made available within the camera domain. In contrast, in the LiDAR domain, only indoor datasets [Silberman et al., 2012, Armeni et al., 2017, Dai et al., 2017] or outdoor datasets [Hackel et al., 2017] obtained from high-resolution stationary sensors have been published within the last years. For this reason, recently, a new dataset called SemanticKITTI [Behley et al., 2019] including semantic information for LiDAR point clouds has been published. This dataset is further discussed in Section 4.4.3.

Generating manually annotated point cloud data for LiDAR-based semantic labeling to scale presents a vast effort and involves even higher cost than manual image annotation in the computer vision domain. This is due to the additional spatial dimension as well as the sparsity of the data, which yields a representation that is non-intuitive and tedious for human annotators. Therefore, some authors have used existing datasets dedicated to other tasks such as 3D object detection [Geiger et al., 2012a] to extract point-wise LiDAR semantics [Dewan et al., 2017, Wu et al., 2018, 2019, Biasutti et al., 2019, Dewan and Burgard, 2019, Madawi et al., 2019]. However, only a small number of semantic classes can be extracted in this manner.

Hence, a so-called *Autolabeling* process is proposed - an effective approach for the automated generation of large amounts of semantically annotated mobile LiDAR data by the direct transfer of high-quality semantic information from a registered reference camera image.

3.2 RELATED WORK

Related to the topic of semantic labeling of semi-dense LiDAR point clouds, only a few publications directly operate in the field of road scenarios and autonomous driving. This is related to the lack of publicly available datasets (see Section 3.1).

Generating an additional dataset in an automatic manner represents a cost- and time-efficient opportunity for developing semantic labeling approaches within the field of road scenarios and autonomous driving. Related to the automatic annotation process, 3 different types can be found within the literature. These are related to the handling of different application fields, which are also called domains.

First, simulation strategies can be used to automatically generate training data and increase performance. Fang et al. [2020] used simulated LiDAR data to increase the performance of LiDAR-based instance segmentation and LiDAR-based object detection validated on the KITTI dataset [Geiger et al., 2012a]. Therefore, they introduced a simulation pipeline based on augmented real-world data. The first step of their pipeline is the acquisition of LiDAR data based on high-resolution LiDAR sensors. After extracting the static background of the recorded scenarios, dynamic objects are augmented into the scene serving as label injections. Finally, a specific sensor type is simulated within this high-resolution road scenario extracting a semi-dense LiDAR point cloud including the labels for LiDAR-based instance segmentation and LiDAR-based object detection. Wang et al. [2019b] as well as Wu et al. [2019] used fully simulated pipelines such as the CARLA simulator [Dosovitskiy et al., 2017] or game engines such as the GTA-V to extract point-wise labels. Additionally, they used real-world data for developing LiDAR-based semantic labeling approaches. All of the mentioned approaches emphasize the benefit of using simulated or augmented data in addition to real-world data in terms of performance gain and cost reduction for applying the methods to real-world data. At the same time, they show critical performance losses by using only simulated or augmented data due to the larger domain shift between the simulation and real-world [Fang et al., 2020, Wu et al., 2019, Wang et al., 2019b], which decreased the portability of the proposed methods for those domains. Tobin et al. [2017] and Andrychowicz et al. [2020] reduced the domain shift within the field of robotics by randomizing the simulated camera data. Hence, they changed randomly the color and the textures of the objects and the background to introduce more diversity into the simulated data. This approach is restricted to specific use cases, whereby the domain shift can be represented by randomization. Based on more complex scenarios with different object appearances, this approach is difficult to transfer. For this reason, Shrivastava et al. [2017] proposed a method based on **A**dversarial **N**etworks (ANs) to reduce the domain shift between simulated and real-world data applied to the gaze direction detection of humans. Although they presented valuable results, this approach still needs real-world data to generate the AN. Preventing the domain shift between simulation and real-world scenarios, within this chapter an automatic training data generation based on real-world data only is proposed.

The second type of automatic data generation is based on the combination of different sensor modalities. This can be applied to learning models that generate one output representing a specific sensor modality by using a different sensor modality as an input without any further processing. Bojarski et al. [2016] and Hubschneider et al. [2017] created CNN models to approach autonomous driving in an end-to-end fashion. Thereby, their CNN models use one or multiple camera images as an input and directly output vehicle control commands represented by the steering angle. Collecting data for this task can easily be performed by driving a prepared vehicle while recording camera images as well as the steering angle of the driver representing two different sensor modalities. Similar efforts are needed e. g. for generating depth information from a monocular camera [Kumar et al., 2018] by recording a LiDAR sensor, which generates depth information for training purposes. Within other application areas, this approach can also be applied as within bioinformatics, where e. g. Han [2017] used magnetic resonance images as inputs to generate computed tomography scans, which are recorded based on the same person. These approaches show a considerable benefit for data collection by generating large quantities of data without human labeling effort. However, they are restricted to a small set of tasks where the input and the output are recordable with different sensor modalities. In the case of LiDAR-based semantic labeling, no sensor modality exists to generate a point-wise classification without any further processing. As a result, this type of automatic data generation cannot be applied.

The last type of automatic training data generation is represented by data enrichment based on the same sensor modality. Here, typically a manual correction is needed to increase the quality of the labeled data. Pan et al. [2018] created a dataset for camera image-based lane marking based on the Hough transformation of a camera image. Thereby, they labeled straight lines as lane markings automatically. Other types of lane markings were annotated manually. Afterwards, they trained a CNN to detect the lane markings with the generated training data. A different approach is proposed by Das et al. [2019], which trained a CNN for camera image classification on a manual labeled dataset. Afterwards, they extracted the internal CNN features of the manually labeled data to create a feature distribution per class. This distribution is used to classify unknown images and extend the dataset automatically. Although these approaches represent a valid opportunity to automatically increase the amount of data, human interaction is still needed to initially generate data.

Combining the second and the third type of automatic training data generation, different sensor modalities are used for data enrichment. For example, Bhoi [2019] generated depth information from a monocular camera. In this context, they recorded a stereoscopic camera which generates depth information in a post-processing step for training purposes. This represents a valuable technique to automatically increase the amount of training data. For adaptation to the LiDAR sensor, Wang et al. [2019a] created a manual annotation tool which transfers e. g. bounding box proposals of the camera to the LiDAR sensor based on a precise calibration of the sensor system. This alleviates the effort of labeling 3D LiDAR data. Varga et al. [2017] proposed an alternative method to generate a semantically labeled point cloud at runtime based on a combined setup of fisheye cameras and LiDAR sensors. First, pixel-wise semantics are extracted from the camera images via a CNN model trained on Cityscapes [Cordts et al., 2016]. Subsequently, the LiDAR points are projected into the images to transfer the semantic information from pixels to 3D points. However, spatial and temporal registration of the sensor modalities remain a challenge. In this chapter, the idea of Varga et al. [2017] is taken one step further by utilizing a joint camera/LiDAR sensor setup to generate large amounts of 3D semantic training data while extending the temporal and spatial registration of both sensor modalities.

3.3 METHOD

This section describes an efficient automated cross-modal data generation process including a spatial optimization and a class mapping of the output which is referred to as *Autolabeling*.

3.3.1 *Autolabeling*

Generating manually annotated training data for LiDAR-based semantic labeling to scale presents a considerable effort and entails even higher cost compared to manual image annotations in the 2D domain. This is due to both the additional spatial dimension and the sparsity of the data, which results in a representation that is non-intuitive and cumbersome for human annotators. For these reasons, an efficient automated process for large-scale training data generation called *Autolabeling* is introduced.

As illustrated in Figure 3.1, the *Autolabeling* concept is based on the use of one or more reference cameras in conjunction with the LiDAR sensor capturing the point cloud data. The obtained reference camera images have to be registered to the LiDAR data in space and time as described in Section 2.4. Preferably, the spatial displacement between the sensor origins is minimized to avoid occlusion artifacts.

In the first step, a high-quality pixel-wise semantic labeling of the reference camera image is computed via state-of-the-art deep neural networks, as can be found on the leaderboard of the Cityscapes benchmark [Cordts et al., 2016]. Second, the captured point cloud is projected into the reference image plane to transfer the semantic information of the image pixels to the corresponding LiDAR points. While a single reference camera will in general only cover a fraction of the full point cloud, it is straightforward to extend the approach to multiple cameras for an increased coverage.

The described fully automated procedure yields semantically labeled point clouds, which can be directly used to train LiDAR-based semantic labeling networks as described in Chapter 4. In the following subsections, the various stages of the data generation process are explained in detail.

3.3.1.1 *Semantic Image Labeling*

For the experiments in this chapter, the efficient reference network described in [Cordts, 2017] to obtain the pixel-wise semantic labeling of the camera images is used[1]. The network is trained on the Cityscapes dataset and achieves a mIoU test score of 72.6% with re-

1 The proposed CNN architecture is based on the GoogLeNet architecture [Szegedy et al., 2015] adapted with a higher learning rate, data augmentation, context modules, and a usage of coarse labels for the training. For details, the reader is referred to [Cordts, 2017].

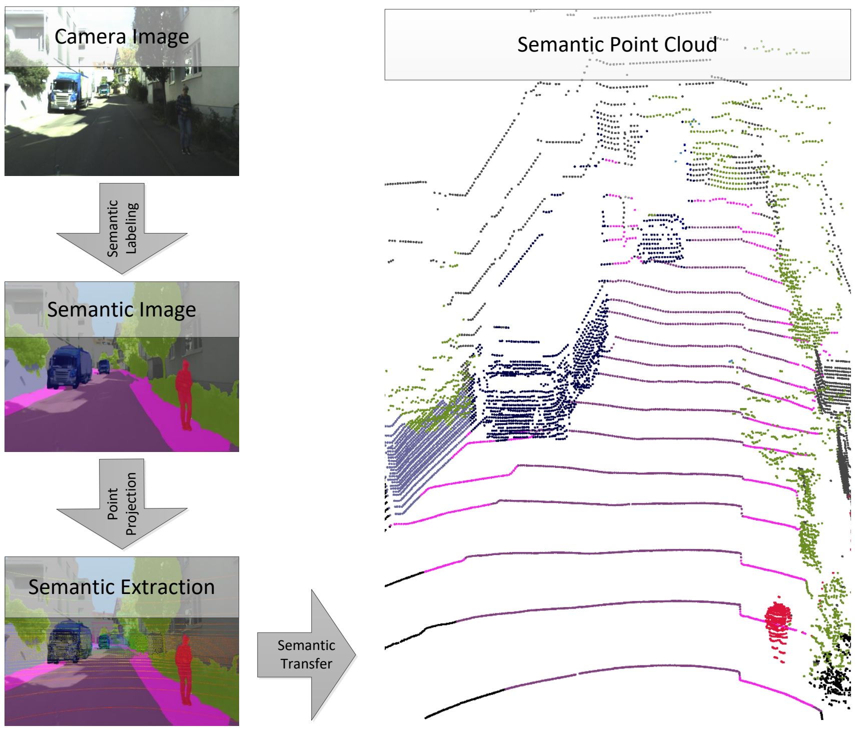


Figure 3.1: Overview of the *Autolabeling* process for large-scale automated training data generation. Each pixel of the reference camera image is classified via an image-based semantic labeling network. Subsequently, the point cloud is projected into the camera image and the reference labels are transferred to the corresponding LiDAR points. The following semantic classes are visualized: road, sidewalk, person, rider, small vehicle, large vehicle, two wheeler, construction, pole, traffic sign, vegetation, terrain. The figure is based on [Piewak et al., 2019] © 2019 IEEE.

gard to the original Cityscapes label set. Since the Cityscapes dataset was designed with vehicle-mounted front-facing cameras in mind, a single front-facing camera is used in this context to evaluate the proposed automated training data generation process.

Note that the *Autolabeling* process can be applied using any image-based reference network providing sufficient output quality. Moreover, the process will in general directly benefit from the ongoing research and improvements in image-based semantic labeling networks.

3.3.1.2 *Point Projection*

In order to project the 3D points captured by a scanning LiDAR sensor into the reference camera image plane, several aspects have to be taken into account. Since the LiDAR scanner rotates around its own axis in order to obtain a 360° point cloud, each measurement is taken at a different point in time. In contrast, the camera image is taken at a single point in time, or at least with a comparatively fast shutter speed.

To minimize potential adverse effects introduced by the scanning motion of the LiDAR sensor, a point-wise ego-motion correction using vehicle odometry is applied. First, the measured 3D points are transformed from the LiDAR coordinate system to the vehicle coordinate system via the extrinsic calibration parameters of the sensor (see Section 2.4). Given the points $\boldsymbol{p}_v = (p_{x_v}, p_{y_v}, p_{z_v})$ in the vehicle coordinate system, the wheel odometry data is used to compensate for the ego-motion of the vehicle. To this end, the time difference ${}_{\Delta}t_{pc}$ between the point measurement timestamp t_p and the image acquisition timestamp t_c of the camera is computed for each point. In case of a rolling shutter camera, half of the shutter interval t_r is added to move the reference timestamp to the image center:

$$ {}_{\Delta}t_{pc} = t_c - t_p + \frac{t_r}{2} \quad . \tag{3.1} $$

The time difference ${}_{\Delta}t_{pc}$ is used to extract the corresponding ego-motion data of the vehicle from the odometry sensor. This yields a transformation matrix $\mathbf{T}_{{}_{\Delta}t_{pc}}$ describing the motion that occurred between the two timestamps of interest. Using the transformation matrix $\mathbf{T}_{{}_{\Delta}t_{pc}}$, each point $\boldsymbol{p}_v$ is ego-motion corrected with

$$ \begin{bmatrix} \boldsymbol{p}_{t_c} \\ \\ 1 \end{bmatrix} = \begin{bmatrix} p_{x_{t_c}} \\ p_{y_{t_c}} \\ p_{z_{t_c}} \\ 1 \end{bmatrix} = \mathbf{T}^{-1}_{{}_{\Delta}t_{pc}} * \begin{bmatrix} p_{x_v} \\ p_{y_v} \\ p_{z_v} \\ 1 \end{bmatrix} \quad . \tag{3.2} $$

This effectively transforms each point $\boldsymbol{p}_v$ in the vehicle coordinate system to its corresponding position $\boldsymbol{p}_{t_c} = (p_{x_{t_c}}, p_{y_{t_c}}, p_{z_{t_c}})$ at camera timestamp t_c. Finally, the corrected points $\boldsymbol{p}_{t_c}$ are transformed to the camera coordinate system and projected to the image plane using a pinhole camera model [Hartley and Zisserman, 2005].

3.3.2 *Optimization in Space and Time*

An essential factor for accurate *Autolabeling* results is a small spatial distance between the LiDAR sensor and the reference camera, as occlusion artifacts tend to introduce inconsistencies to the datasets. This can be facilitated by both a small mounting distance of the two sensors and explicit time synchronization. In the used setup, the camera is not being triggered by the LiDAR sensor, which results in cases where the LiDAR sensor orientation is not well aligned with the front-facing camera during image acquisition time. This misalignment causes a significant timestamp difference between the projected point cloud and the image data. While this effect is com-

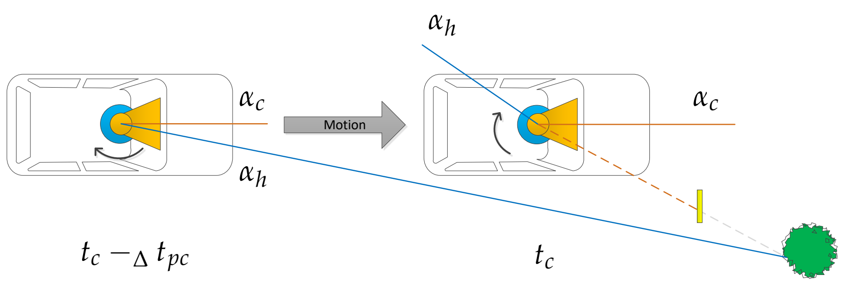


Figure 3.2: Illustration of the timing problem (top view), whereby the camera (orange) is mounted on top of the LiDAR sensor (blue). At a given timestamp, the rotating LiDAR sensor measures points only at the angle α_h (blue line). The tree (green) is detected by the LiDAR sensor at timestamp $t_c -_\Delta t_{pc}$, while the camera captures an image of the tree occluded by the traffic sign (yellow) at t_c. This time difference $_\Delta t_{pc}$ results in a spatial displacement of the sensor origins, which causes occlusion artifacts based resembling a large mounting distance between the two sensors. This illustration is based on [Piewak et al., 2018b].

pensated by the ego-motion correction described in Section 3.3.1.2, it usually results in a significant translatory motion between the two measurements of the sensors while driving. This in turns leads to considerable occlusion artifacts. See Figure 3.2 for an illustration of the problem. For this reason, a maximum heading deviation range α_m has to be defined which is allowed between the camera principal axis orientation α_c and the current LiDAR azimuth angle α_h at the camera timestamp t_c. All captured frames which lie outside of this maximum deviation range are discarded, yielding an optimized subset of the originally recorded data.

3.3.3 *Class Mapping*

Based on initial experiments with regard to the discrimination of different semantic classes in LiDAR data, a mapping of the original Cityscapes labelset [Cordts et al., 2016] to a reduced set of 13 semantic classes (see Table 3.1) is applied. This reduced label set is better tailored to the specific properties of the data provided by current LiDAR sensors, where limited spatial resolution and coarse reflectivity estimates prohibit truly fine-grained semantic differentiation. For example, the original 'truck' and 'bus' classes are merged into a common 'large vehicle' class. Similarly, the original 'motorcycle' and 'bicycle' classes are combined into a single 'two-wheeler' class. The 'fence' class is not retained in the mapping, as such thin and porous structures are hardly captured by LiDAR sensors additionally to a usually wrong classification of behind objects within the semantically labeled camera image. Note that the reduced label set

Cityscapes	LiDAR Semantics	Cityscapes	LiDAR Semantics
road	road	building	construction
sidewalk	sidewalk	wall	construction
person	person	fence	unlabeled
rider	rider	pole	pole
car	small vehicle	traffic sign	traffic sign
truck	large vehicle	traffic light	construction
bus	large vehicle	vegetation	vegetation
on rails	large vehicle	terrain	terrain
motorcycle	two-wheeler	sky	sky
bicycle	two-wheeler		

Table 3.1: Mapping of the Cityscapes label set [Cordts et al., 2016] to the reduced LiDAR label set. This table is based on [Piewak et al., 2018b].

still provides an abundance of valuable semantic information, which is highly beneficial for a large set of application scenarios.

3.4 EVALUATION

Within this section, the evaluation of the datasets generated based on the *Autolabeling* process is described. Therefore, a dataset overview, a dataset analysis, and an *Autolabeling* performance analysis is provided.

3.4.1 *Dataset Overview*

For evaluating methods within this thesis, two large datasets were created based on the *Autolabeling* process explained in Section 3.3. Both datasets are based on state-of-the-art LiDAR sensors: A VLP-32C and a VLS-128 (see Section 2.1.2). Approximately 550,000 frames with the VLP-32C and 300,000 with the VLS-128 of different road types (cities, rural roads, and highways) and different city areas (e.g. Karlsruhe or Stuttgart) were recorded and automatically labeled based on own recording drives of several days. The drives were realized with a Mercedes Benz E-Class equipped with LiDAR sensors and cameras as well as a vehicle interface to extract ego-motion information. Note that the variety within the recorded data is not complete in terms of all possible scenarios within the field of autonomous driving (e. g. adverse weather conditions or seasonal changes). However, it represents a valuable basis for further development and analysis of LiDAR-based semantic labeling approaches.

	Training	Validation	Testing
Autolabeled Frames VLP-32C	344,027	73,487	137,682
Autolabeled Frames VLS-128	179,042	31,139	84,839
Manually Annotated Frames VLP-32C	1,909	373	718
Manually Annotated Frames VLS-128	1,257	335	1,110

Table 3.2: Split of the two employed datasets into the subsets for training, validation, and testing. They are distinguished between automatically and manually annotated point clouds (frames). The table is based on [Piewak et al., 2019] © 2019 IEEE.

The recorded frames are split into subsets for training, validation and testing, with a split of approximately 60% - 10% - 30%. Details are listed in Table 3.2. The dataset split is performed on a sequence basis instead of via random selection in order to prevent correlations in between subsets. The training set is used to train the FCNs (see following chapters), while the validation set is used to optimize the hyper parameters of the network architectures. The testing set is used to evaluate the achieved results showed within this thesis. Note that the VLS-128 dataset is approximately half the size of the VLP-32C dataset (see Table 3.2).

Based on the optimized frames described in Section 3.3.2, 3,000 frames for the VLP-32C and approximately 2,700 frames for the VLS-128 called keyframes were selected for manual annotation[2] with the classes corresponding to the class mapping of Section 3.3.3. This manually annotated data forms the basis for the evaluation within this thesis. This includes the *Autolabeling* process (Chapter 3), the LiDAR-based semantic labeling (Chapter 4) including its extensions (Chapters 5 and 6), and the multi-modal stixel approach (Chapter 7). Note that invalid LiDAR points are not being annotated and, hence, the 'sky' class is not considered for evaluation within this thesis.

3.4.2 *Dataset Analysis*

In contrast to other classification tasks such as image classification, datasets for semantic labeling typically contain a bias within the label distribution as shown in Figures 3.3 and 3.4. This is related to the appearance of traffic scenarios, where e. g. the LiDAR measurements of 'road' are more probable than the measurements of

2 Preventing naming confusions within this thesis, the automatic generation of semantic data (e. g. based on a CNN) is called labeling, while the manual generation of semantic data is called annotation.

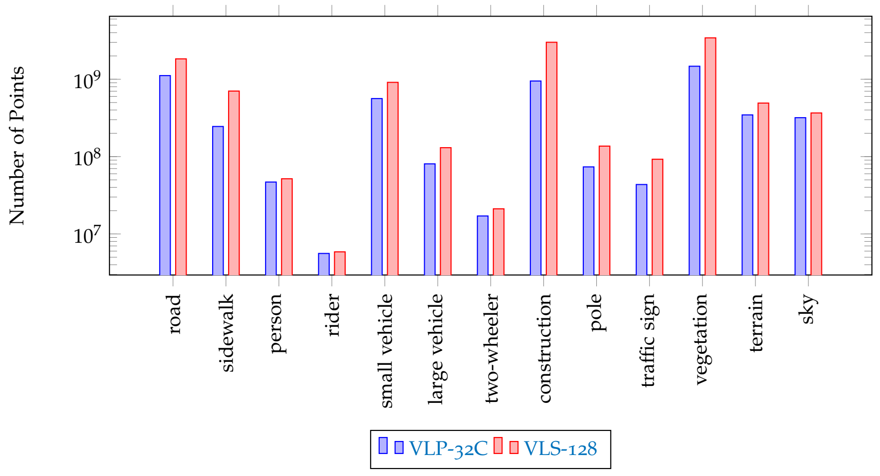


Figure 3.3: Label distribution of the VLP-32C and VLS-128 dataset based on the autolabeled frames.

'rider'. This appearance is captured in each frame, resulting in a certain label distribution per frame and hence in a label distribution of the dataset. Within the presented datasets, a large bias for the classes 'vegetation', 'construction', and 'road' can be recognized. This is due to the appearance of traffic scenarios within cities and rural roads where the dataset was captured. Similar distributions can be found at other datasets of traffic scenarios such as Cityscapes [Cordts et al., 2016] and SemanticKITTI [Behley et al., 2019]. Note that the keyframes used for the manual annotation were optimized to reach a similar label distribution of the manual annotated points and the autolabeled points. Thereby, the number of labeled points of the VLS-128 is higher even by recording a smaller number of frames due to the four times higher resolution of the LiDAR sensor.

Reducing the influence of the mentioned bias within the label distribution towards the comparison of LiDAR-based semantic labeling approaches, the IoU and the mIoU are chosen as evaluation metrics (see Section 2.3.4) similar to public benchmarks such as the Cityscapes Benchmark Suite [Cordts et al., 2016] or SemanticKITTI [Behley et al., 2019].

3.4.3 *Autolabeling Performance*

The *Autolabeling* process introduced in Section 3.3 was applied to automatically generate the large-scale datasets defined in Section 3.4.1, which in turn was used to develop LiDAR-based semantic labeling approaches described within the following chapters. Within this section, the performance of the *Autolabeling* process is evaluated concerning the optimized frames, the ego-motion correction as well as an overall performance analysis. Note that all evaluations are based on the testing set of the manually annotated frames for each dataset

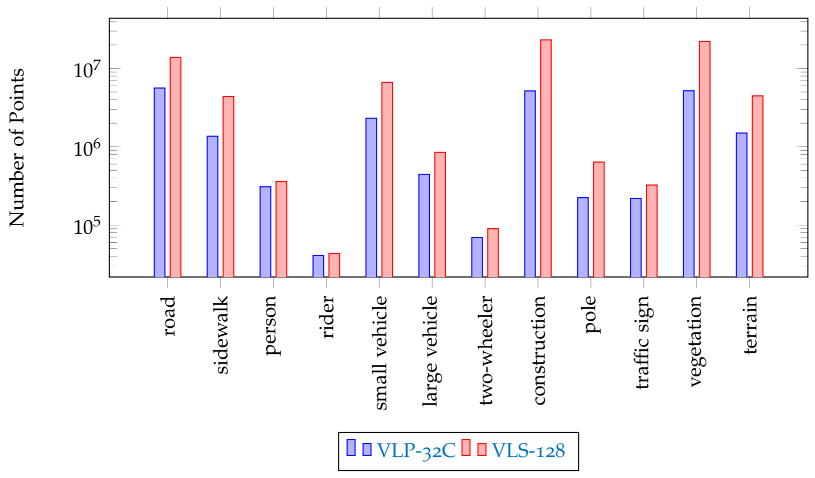


Figure 3.4: Label distribution of the VLP-32C and VLS-128 dataset based on the manually annotated frames. Note that invalid points are not labeled and, hence, the class 'sky' is not present within the manually annotated frames.

if not mentioned otherwise. According to the results of Section 3.4.2, the IoU metric for an overall performance evaluation is applied.

3.4.3.1 *Optimized Frames*

As described in Section 3.3.2, a maximum heading deviation range α_m has to be defined to create an optimized subset of the originally recorded data. Therefore, the performance of the *Autolabeling* process related to the maximum heading deviation range α_m is evaluated and visualized in Figure 3.5. Consider that this evaluation is performed based on the autolabeled validation set of the VLP-32C dataset and not based on the manually annotated testing set due to the impact of the selection of keyframes for the manually annotation. The results show that a higher heading deviation range reduces the performance of the *Autolabeling* process. This is related to the occlusion artifacts as shown in Figure 3.2. At the same time, a higher heading deviation range increases the number of frames. Due to the fact that the number of frames can easily be increased by additional recording drives, a lower maximum heading deviation range of $\alpha_m = 60°$ is selected for generating the optimized subset of the originally recorded data which is used to select keyframes for the manual annotation.

3.4.3.2 *Ego-Motion Compensation*

Projecting the 3D LiDAR points into the reference camera image plane requires, in addition to a spatial calibration of both sensors, an ego-motion compensation as described in Section 3.3.1.2. For evaluating the ego-motion compensation, Table 3.3 depicts the difference in

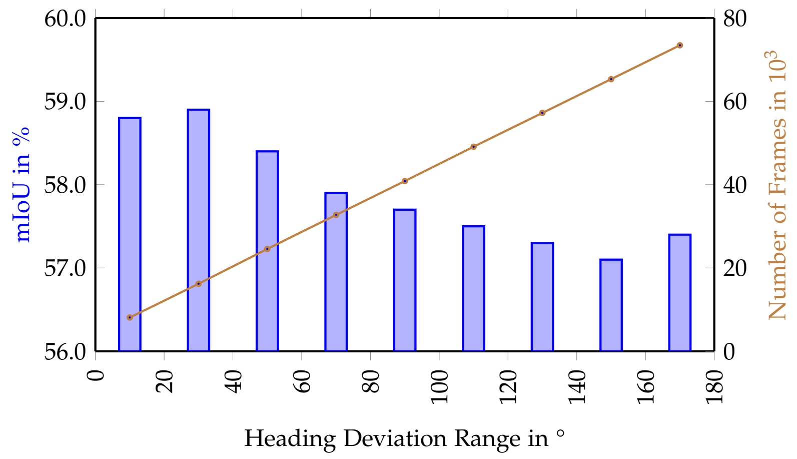


Figure 3.5: Impact of the heading deviation range α_m to the performance of the *Autolabeling* process. The evaluation is performed based on the autolabeled validation set for the VLP-32C dataset. The blue bars describe the mIoU score for each bin of the heading deviation range α_m while the brown line describes the number of frames up to the given heading deviation range α_m.

	road	sidewalk	person	rider	small vehicle	large vehicle	two-wheeler	construction	pole	traffic sign	vegetation	terrain	mIoU
Autolabeling w/o ego-motion compensation	80.5	42.9	67.0	36.8	72.8	35.6	26.1	43.8	24.1	22.2	42.6	5.0	41.6
Autolabeling w/ ego-motion compensation	89.5	61.2	77.0	51.3	80.7	58.5	45.4	74.7	29.8	40.4	80.5	54.7	62.0

Table 3.3: Impact of the ego-motion compensation of the *Autolabeling* process based on the class-wise and overall IoU scores in % for the VLP-32C dataset.

performance with and without ego-motion compensation. The ego-motion compensation clearly increases the IoU scores of all classes. This results in a relative improvement of the mIoU score of 49.0%. An example of the point projection with and without ego-motion correction is shown in Figure 3.6. Thereby, the recording vehicle takes a right turn at an intersection which causes a large angular ego-motion for this scene. This causes strong displacements of the projected points and the corresponding object (e. g. see the poles, the traffic lights, and the traffic sign). Applying the ego-motion correction, these displacements are compensated to align the projected points with the corresponding object. As a result, the ego-motion compensation is activated for both recorded datasets used within this thesis.

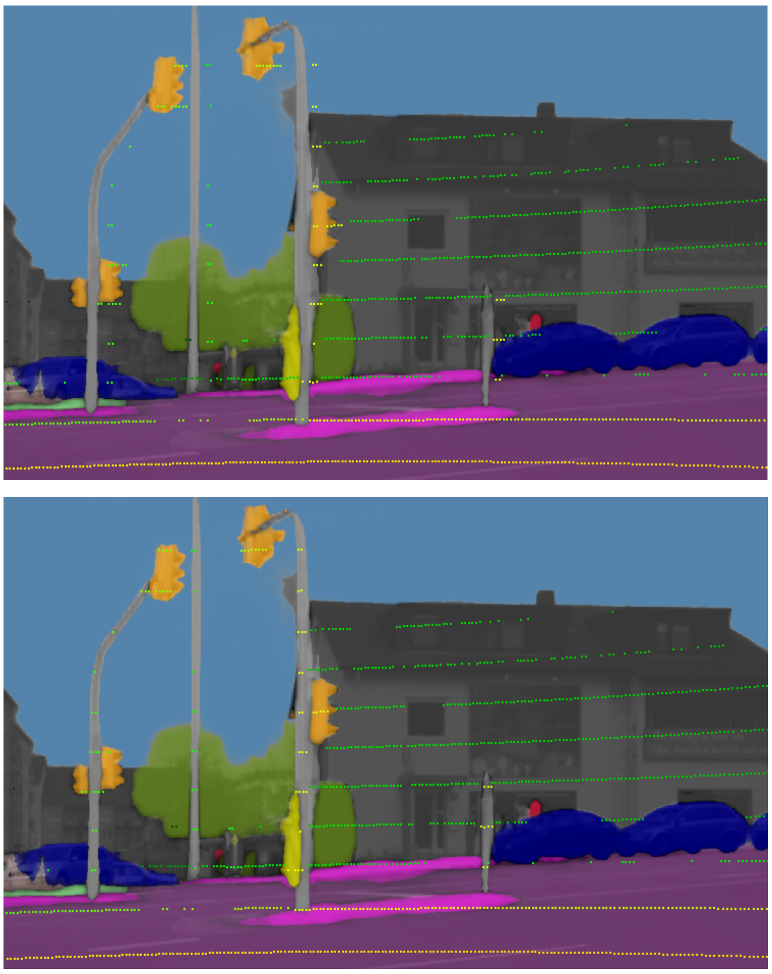


Figure 3.6: Example of the point projection without (top) and with (bottom) ego-motion correction. The colors of the semantically labeled images represent the Cityscapes label set [Cordts et al., 2016], while the projected points are color-coded according to distance (dark green = far away, yellow = close). Note that invalid LiDAR points are not projected into these images. Images are based on [Piewak et al., 2018b].

3.4.3.3 *Overall Performance*

The overall performance of the *Autolabeling* process is shown in Table 3.4. It also shows the evaluation of the used image-based semantic labeling network (see Section 3.3.1.1), which was previously analyzed in detail in [Cordts, 2017], achieving an IoU score of 72.6% on the Cityscapes Benchmark Suite. Here, the image-based approach is re-evaluated on the Cityscapes testing set by using the class mapping defined in Section 3.3.3. This yields an IoU score of 79.3% (see Table 3.4).

	road	sidewalk	person	rider	small vehicle	large vehicle	two-wheeler	construction	pole	traffic sign	vegetation	terrain	mIoU
Image Labeling[4] based on Cityscapes	*98.0*	*81.5*	*81.4*	*61.3*	*94.7*	*80.8*	*70.2*	*91.8*	*58.0*	*71.8*	*92.7*	*69.0*	*79.3*
Autolabeling VLP-32C dataset	89.5	61.2	77.0	51.3	80.7	58.5	45.4	74.7	29.8	40.4	80.5	54.7	62.0
Autolabeling VLS-128 dataset	87.7	57.6	63.0	44.0	81.4	54.0	39.3	82.5	26.5	35.3	82.8	55.0	59.1

Table 3.4: Class-wise and overall IoU scores in % of the *Autolabeling* process for the different datasets. Parts of this table are based on [Piewak et al., 2018b].

The quality of the point cloud semantics obtained by the *Autolabeling* process is summarized in the last two rows of Table 3.4[3]. It has to be noted that the results of the VLP-32C dataset are slightly better than the VLS-128 dataset. This is due to differences in the recording setup, including extrinsic calibration data of both sensors with respect to different calibration inaccuracies. Additionally, the lower resolution of the VLP-32C makes the process less susceptible to errors caused by both extrinsic and intrinsic calibration inaccuracies, which reduce the precision of the point-wise projection into the RGB camera image. Nevertheless, the results lie within a similar range and allow for an initial assessment of the quality of the datasets.

Compared to the pure image-based semantic labeling result, the *Autolabeling* output used to generate the large-scale datasets shows similar label distributions but worse performance scores. This may be attributed to various characteristics which are discussed within the following paragraphs.

CALIBRATION First, a multi-sensor setup is rarely free of small calibration offsets in practice, even if optimized manually. These imperfections cause projection misalignments of the LiDAR points within the camera images, which result in inaccurate label assignments, in particular for points at a large distance to the sensors. Also, despite the ego-motion correction efforts described in Section 3.3.1.2, the remaining inaccuracies still result in a certain number of occlusion artifacts, which slightly decrease the overall performance.

IMAGE CNN PERFORMANCE The second characteristic of the *Autolabeling* results relates to the used image-based semantic labeling approach [Cordts, 2017], which indeed results in a favorable perfor-

3 The *Autolabeling* results differ from the values reported in Piewak et al. [2018b] due to a restriction to the camera field of view of the *Autolabeling* process.

4 Image Labeling of Cordts [2017] based on the Cityscapes Benchmark Suite [Cordts et al., 2016] re-evaluated on the Cityscapes testing set based on the class mapping described in Section 3.3.3.

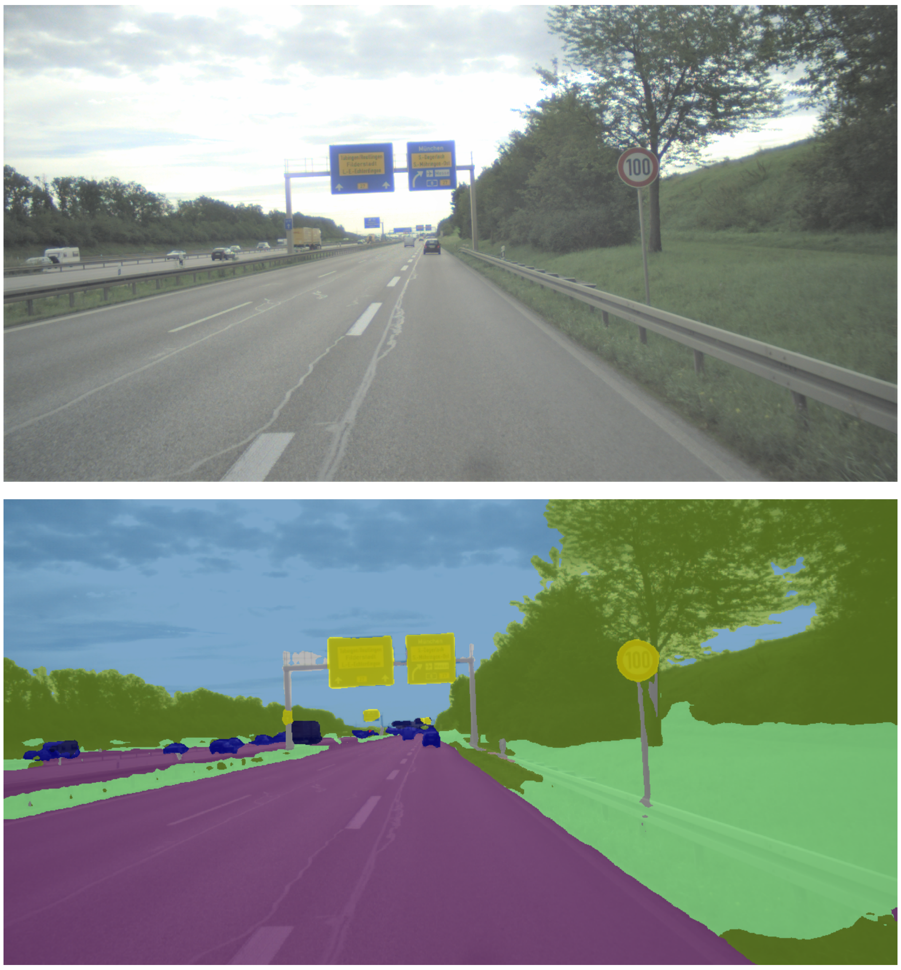


Figure 3.7: Example of plausible confusion of an image-based semantic labeling approach by presenting the orinal camera image (top) and the semantically labeled camera image (bottom). The classes 'vegetation' and 'terrain' are confused due to an unclear classification of the highway embankment. Label colors correspond to the Cityscapes semantic class color coding [Cordts et al., 2016]. The following semantic classes are visualized: road, sidewalk, person, rider, small vehicle, large vehicle, two-wheeler, construction, pole, traffic sign, vegetation, terrain, sky.

		Autolabeling											
		road	sidewalk	construction	pole	traffic sign	vegetation	small vehicle	large vehicle	person	rider	two-wheeler	terrain
True Label	road	99	0	0	0	0	0	0	0	0	0	0	0
	sidewalk	8	89	1	0	0	0	0	0	0	0	0	1
	construction	0	0	97	0	0	2	0	0	0	0	0	0
	pole	0	2	18	70	1	7	1	0	1	0	0	0
	traffic sign	0	0	14	2	79	3	0	0	0	0	0	0
	vegetation	0	0	2	0	0	96	0	0	0	0	0	0
	small vehicle	1	0	1	0	0	0	98	0	0	0	0	0
	large vehicle	1	0	6	0	0	1	6	85	0	0	0	0
	person	1	1	4	1	0	1	1	0	91	1	1	0
	rider	1	0	3	0	0	1	2	0	11	72	10	0
	two-wheeler	1	2	3	1	0	1	4	0	2	3	82	0
	terrain	4	11	1	0	0	6	0	0	0	0	0	78

Table 3.5: Confusion matrix of the image-based semantic labeling approach of Cordts [2017] based on the Cityscapes testing set [Cordts et al., 2016] with the reduced labelset defined in Section 3.3.3. The values are rounded to integer percentage.

mance score on the Cityscapes Benchmarks Suite [Cordts et al., 2016] but still produces inconsistencies for certain classes, as shown in Table 3.5. These inconsistencies directly influence the output of the *Autolabeling* process resulting in a similar confusion matrix shown in Table 3.6 and Table 3.7. Comparing these three confusion matrices, a similar layout, particularly concerning certain class confusions, can be recognized. For example, the confusion between 'rider' and 'two-wheeler', between 'road' and 'sidewalk', or between 'vegetation' and 'terrain' (see Figure 3.7) is apparent at all three confusion matrices since these classes are similar to each other. As a result, the labeling characteristics of the used image-based semantic labeling approach are directly transferred by the *Autolabeling* process.

DOMAIN TRANSFER Taking a closer look at the differences between the confusion matrices of the datatsets in Tables 3.5 to 3.7, it can be seen that the inconsistencies of the image-based semantic labeling approach are present and even larger within the LiDAR domain (e. g. the confusion between 'large vehicle' and 'small vehicle' or between 'terrain' and 'vegetation'). This is additionally related to the label transfer from the high-resolution image domain to the sparse LiDAR domain. The difference between the two domains is visualized in Figure 3.8, where the ratio between the number of measurements on the house surface to the house borders is larger within the image domain as in the LiDAR domain. Therefore, the used image-based semantic labeling approach is optimized towards the

Autolabeling

True Label	road	sidewalk	construction	pole	traffic sign	vegetation	small vehicle	large vehicle	person	rider	two-wheeler	terrain
road	95	1	0	0	0	0	0	0	0	0	0	0
sidewalk	15	75	2	0	0	1	0	0	0	0	0	3
construction	1	2	81	1	1	7	2	0	0	0	0	1
pole	1	2	15	53	2	15	2	0	0	0	0	4
traffic sign	4	0	10	4	55	16	3	0	0	0	0	3
vegetation	0	0	2	0	0	91	0	0	0	0	0	3
small vehicle	2	0	1	0	0	1	89	4	0	0	0	0
large vehicle	2	0	4	0	0	1	9	80	0	0	0	0
person	1	1	5	0	0	1	1	0	87	0	0	0
rider	0	0	2	0	0	5	2	0	8	65	13	0
two-wheeler	4	3	4	1	0	2	4	0	2	6	68	0
terrain	4	3	2	0	0	20	0	0	0	0	0	67

Table 3.6: Confusion matrix of the *Autolabeling* process with the VLP-32C dataset based on the manually annotated testing set. The values are rounded to integer percentage.

Autolabeling

True Label	road	sidewalk	construction	pole	traffic sign	vegetation	small vehicle	large vehicle	person	rider	two-wheeler	terrain
road	92	4	0	0	0	0	1	0	0	0	0	0
sidewalk	11	77	2	0	0	2	1	0	0	0	0	3
construction	0	2	86	1	1	7	0	0	0	0	0	0
pole	1	2	19	49	4	18	2	0	0	0	0	1
traffic sign	0	0	8	2	79	6	0	0	0	0	0	0
vegetation	0	0	2	0	0	94	0	0	0	0	0	1
small vehicle	0	0	2	0	0	1	90	2	0	0	0	0
large vehicle	0	0	7	0	0	2	7	80	0	0	0	0
person	0	1	6	0	0	2	1	0	84	0	0	0
rider	0	0	2	0	0	4	1	0	15	62	10	0
two-wheeler	1	1	3	0	0	1	5	0	2	10	72	0
terrain	3	8	1	0	0	20	0	0	0	0	0	64

Table 3.7: Confusion matrix of the *Autolabeling* process with the VLS-128 dataset based on the manually annotated testing set. The values are rounded to integer percentage.

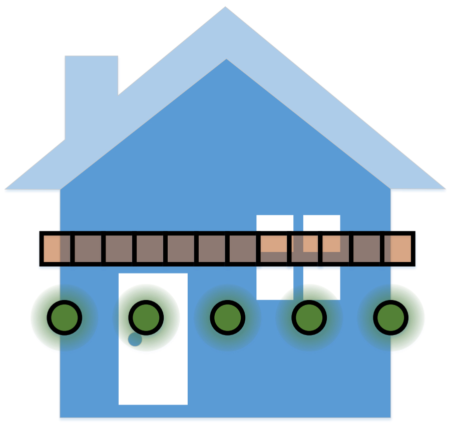


Figure 3.8: Illustration of the domain transfer problem of the *Autolabeling* process concerning a camera (orange pixel row) and a LiDAR sensor (green point row). Both sensor types measure the borders of the house with one measurement from each side. The surface of the house is detected with more measurements of the camera domain compared to the LiDAR domain. As a result, surfaces have a larger bias in the camera domain as in the LiDAR domain.

specific properties of high-resolution camera domain including the bias of the surfaces, which leads to a higher error rate at the object borders compared to surfaces (see Figure 3.7). Applying the label transfer from the camera domain to the LiDAR domain, the higher error rate at object borders of the camera domain and the larger bias towards object borders of the LiDAR domain are combined. This results in a slight reduction in performance when transferring the labels into the LiDAR domain. Note that this effect is not recognizable between the confusion matrices of the VLP-32C and the VLS-128 dataset due to the fact that the horizontal resolution of the LiDAR sensors is identical [Velodyne LiDAR Inc., 2019a,b].

3.5 OUTCOME

In this chapter, a fully automated process for large-scale cross-modal training data generation called *Autolabeling* was presented. The approach is based on the use of reference cameras in order to transfer high-quality image-based semantic labeling results to LiDAR point clouds. This approach was implemented into a research vehicle to be applied to two different datasets recorded with state-of-the art LiDAR sensors (VLP-32C and VLS-128). Both datasets show similar inconsistencies as the underlying image-based semantic labeling approach even with small calibration misalignments as well as the domain transfer which slightly decreases the performance. However, the *Autolabeling* process describes a cost- as well as time-efficient cross-modal approach for generating large-scale datasets. Further positive effects of the *Autolabeling* process will be discussed in the following chapter in combination with LiDAR based semantic labeling approaches.

Two different datasets for LiDAR-based semantic labeling are therefore generated including a small number of manually annotated LiDAR point clouds. This represents the second step of the research pipeline (see Figure 1.5) and forms the basis for the next step of LiDAR-based semantic labeling.

4 LIDAR-BASED SEMANTIC LABELING

CONTENTS

The focus of this chapter represents the third step of the research pipeline (see Figure 1.5). Parts of this chapter have previously appeared in [Piewak et al., 2018b] and [Schillinger, 2018].

4.1 OVERVIEW

After generating a large-scale dataset as described in Chapter 3, the next step of the research pipeline (see Figure 1.5) represented by the LiDAR-based semantic labeling can be approached. As mentioned in Sections 1.1.2 and 1.1.4, vehicles within the fields of mobile robotics and autonomous driving are typically equipped with multiple sensors of complementary modalities such as cameras, LiDAR sensors and RaDAR sensors in order to safely act within their environment.

In order to master the dynamics of road scenarios, it is essential for an autonomous vehicle to not only distinguish between generic obstacles and free-space, but to also obtain a deeper semantic understanding of its surroundings. Within the field of computer vision, the corresponding task of semantic image labeling has experienced a significant boost in recent years (see Section 1.1.5). However, detailed semantic information of similar quality has to be extracted independently from each of the sensor modalities to maximize system performance, availability, and safety. Therefore, in this chapter, the **Li**DAR **La**beling **Net**work (LiLaNet) - an efficient deep neural network architecture for point-wise, multi-class semantic labeling of semi-dense LiDAR point clouds - is introduced based on the VLP-32C dataset of Chapter 3.

4.2 RELATED WORK

LiDAR-based semantic labeling has gained increased attention in recent years due to the availability of improved mobile sensor technology, providing higher resolution, and longer range at reduced cost. The various proposed approaches of LiDAR-based semantic labeling can be discriminated by the way the point-wise 3D information is utilized.

First, the 3D information can be represented as RGB-D data, which complements RGB image data with an additional depth channel [Couprie et al., 2013, Gupta et al., 2014], allowing to recycle 2D semantic image labeling algorithms. Frequently, a stereo camera is used to create a dense depth image, which is then fused with the RGB image. Tosteberg [2017] developed a technique to use the depth information of a LiDAR sensor accumulated over time to project it into the camera space. The accumulation yields a depth image of increased density without requiring dedicated upsampling algorithms.

A different category of approaches considers the 3D LiDAR data as an unordered point cloud, including PointNet [Qi et al., 2017a], PointNet++ [Qi et al., 2017b], and PointCNN [Li et al., 2018]. The PointNet architecture [Qi et al., 2017a] combines local point features with globally extracted feature vectors, allowing for the inference of semantic information on a point-wise basis. Extending this idea, PointNet++ [Qi et al., 2017b] introduces a hierarchical PointNet architecture to generate an additional mid-level feature representation for improved handling of point neighborhood relations. Both approaches are evaluated successfully on indoor scenes but reach their limits in large-scale outdoor scenarios caused by the representation of the entire scene within one global feature vector (PointNet) or a restricted number of hierarchical feature vectors (PointNet++). The PointCNN [Li et al., 2018] approach is based on unordered point clouds as well, but introduces modified convolution layers extended by permutations and weighting of the input features. This allows for the transfer of the advantages of traditional CNNs (e. g. local neighborhoods based on convolution layers as described in Section 2.2.5) to unordered point cloud processing and prevents the restriction to a restricted number of feature vectors for the entire scene. However, the approach is only used for object detection and has not yet been applied to semantic labeling of point clouds.

Another way of representing LiDAR input data is within cartesian 3D space, which is used in the SEGCloud [Tchapmi et al., 2017] and OctNet [Riegler et al., 2017] methods. Here, a voxel (SEGCloud) or an octree (OctNet) representation is created, and the convolution layers are extended to 3D convolutions. These approaches retain the original 3D structure of the input points, making them more powerful in preserving spatial relations. However, the algorithms have to cope with the high sparsity of the data. Additionally, the inference time as well as memory requirements increase dramatically for large-scale outdoor scenes. Note that within recent years, this 3D representation was boosted due to different publications within the field of object detection [Zhou and Tuzel, 2018, Lang et al., 2019]. An introduction into these recent 3D representation approaches is provided in Section 5.2.

A possible solution to avoid the computational complexity of 3D convolutions is rendering 2D views of the input data. Based on such 2D views, state-of-the-art, image-based deep learning algorithms can be applied. Depending on the use case, different viewpoints or virtual cameras may be used. Caltagirone et al. [2017] used a top-view image of a LiDAR point cloud for labeling road points within a street environment. This top-view projection of the LiDAR points is a valid choice for road detection, but the resulting mutual point occlusions generate difficulties for more general semantic labeling tasks as in the case with this thesis. An alternative is to place the virtual camera origin directly within the sensor itself. The resulting 2D view is often visualized via a cylindrical projection of the LiDAR points (see Figures 4.1 and 4.2), which is particularly suitable for the regular measurement layout of common rotating LiDAR scanners (see Section 2.1) In this case, the sensor view provides a dense depth image, which is highly advantageous for subsequent processing steps. Dewan et al. [2017] used this type of input image for a CNN based on the Fast-Net architecture [Oliveira et al., 2016] to distinguish between movable and non-movable points. The bounding boxes of the KITTI object detection dataset [Geiger et al., 2012a] were used to transfer the point-wise semantic information required for training and evaluation. The approach of Wu et al. [2019] uses the cylindrical projection of the LiDAR data as an input for the Sqeeze-Seg architecture, which performs SqeezeNet-based [Iandola et al., 2017] semantic labeling to segment cars, pedestrians and cyclists. Similar to Dewan et al. [2017], the KITTI object detection dataset is used for transferring the ground-truth bounding box labels to the enclosed points. Similar approaches have been published within recent years [Mei et al., 2019, Wang et al., 2018] that exploit the real-time capacity and the performance of proposed CNN architectures based on the cylindrical projection of the LiDAR data for a small number of semantic classes. Consider that the mentioned approaches and CNN architectures are optimized towards the small number of classes extracted of the KITTI object detection dataset [Geiger et al., 2012a]. In contrast, within this thesis, a large number of diverse semantic classes is used for an in-depth semantic labeling of the semi-dense point cloud. Therefore, the mentioned state-of-the-art approaches cannot be applied directly and a novel CNN architecture is proposed.

4.3 METHOD

Within this chapter, a novel CNN architecture called LiLaNet for the point-wise, multi-class semantic labeling of LiDAR data is introduced. Obtaining high output quality and retaining efficiency at the same time, lessons learned from image-based semantic labeling methods are transferred to the LiDAR domain. The cylindrical projection of a 360° point cloud captured with a VLP-32C (see Section 2.1) is used as an input to the network, while the training of the LiLaNet is boosted by the *Autolabeling* process described in Chapter 3.

This section describes the cylindrical projection of the LiDAR point cloud, the base architecture of the LiLaNet, as well as optimization examples. For further optimization of the LiLaNet, the reader is referred to [Schillinger, 2018].

4.3.1 *LiDAR Images*

As mentioned in Section 2.1, the used VLP-32C is a rotating LiDAR sensor with 32 LASER systems. While rotating, each module periodically measures the distance and reflectivity at its current orientation, i.e. at the respective azimuth and elevation angles. In this chapter, the measurements of a full 360° scan are combined to create cylindrical depth and reflectivity images, as illustrated in Figures 4.1 and 4.2. This projection represents the view of a virtual 360° cylindrical camera placed at the sensor origin. At 10 revolutions per second, images of size 1,800 $\times$ 32 pixels are obtained.

The cylindrical point cloud projection provides dense depth and reflectivity images which are free from mutual point occlusions. This allows for the application of optimized 2D convolution layers, as used with great success in state-of-the-art image-based CNN architectures. In this manner, inference time is reduced dramatically compared to the use of full 3D input representations such as voxel grids or octrees due to the reduction of one dimension [Shen, 2019]. Furthermore, since measurement times and orientation angles are known with high accuracy, it is straightforward to transform the cylindrical image back to a full three-dimensional point cloud representation without any loss of information. Note that invalid points are transferred as well into the cylindrical projection, represented by the distance and reflectivity value zero (see Figure 4.2).

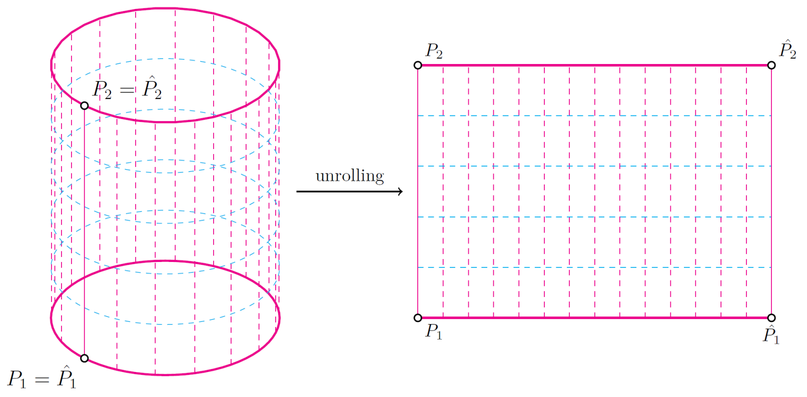


Figure 4.1: Illustration of the cylindrical image generation of a LiDAR point cloud. The illustration is based on [Schillinger, 2018].

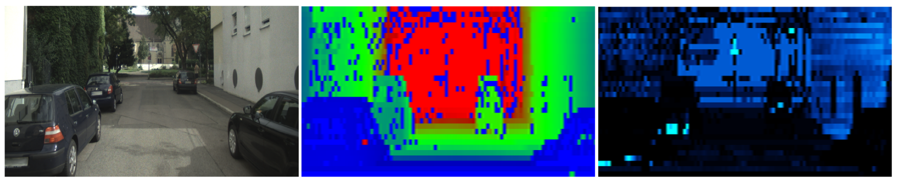


Figure 4.2: Example of a depth image (center, blue = close, red = far) and reflectivity image (right, black = non-reflective, cyan = highly reflective) resulting from the cylindrical point cloud projection. The corresponding camera image is shown on the left. Note that here the 360° LiDAR scan has been cropped to the camera field of view for illustration. The images are based on [Piewak et al., 2018b].

4.3.2 *LiLaNet Architecture*

Using the LiDAR images described in Section 4.3.1 as input, a dedicated CNN architecture for high-quality LiDAR-based semantic labeling is presented. To cope with the low resolution and extreme asymmetry in the aspect ratio of the used LiDAR images, a dedicated network block called **Li**DAR **La**beling **Block** (LiLaBlock) is proposed, which is inspired by the GoogLeNet inception modules of Szegedy et al. [2016]. The block structure is illustrated in Figure 4.3. In order to successfully handle relevant objects of various aspect ratios, the usual filter size of 3×3 is extended to apply filters of sizes 7×3, 3×7, and 3×3 in parallel[1]. The output is then concatenated and the dimension is decreased by a factor of three via a bottleneck. In this way, the dimensionality of the feature space is reduced, yielding a more compact representation. At the same time, the inference complexity of the LiLaNet is lowered. Note that each convolution is followed by a ReLU layer.

1 Note that the maximum height of the filters is restricted to 7 to prevent a larger receptive field of the entire LiLaNet compared to the LiDAR image height of a VLP-32C LiDAR scan.

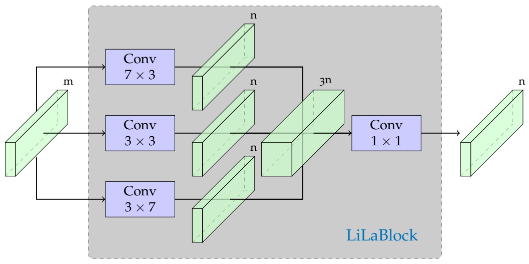


Figure 4.3: The LiLaBlock structure allows to cope with the extreme asymmetry in the aspect ratio of the input LiDAR images. The illustration is based on [Piewak et al., 2018b].

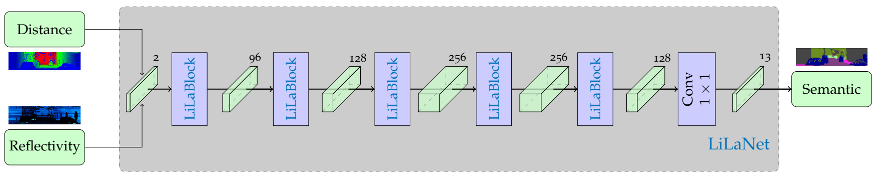


Figure 4.4: The LiLaNet consists of a sequence of five consecutive LiLaBlocks. The final 1×1 convolution reduces the dimensionality to the desired label set. The illustration is based on [Piewak et al., 2018b].

The full LiLaNet consists of a sequence of five LiLaBlocks with a varying number of filters as shown in Figure 4.4. The sequence of LiLaBlocks is followed by a 1×1 convolution layer to reduce the number of feature maps to the number of output classes according to the label set defined in Section 3.3.3. The two input channels represent the concatenated depth and reflectivity images, while the output provides the corresponding point-wise semantic image labeling based on an argmax layer as described in Section 2.2.

4.3.3 *Optimization of the LiLaNet Architecture*

The proposed LiLaNet represents the basis for further optimizations towards performance as well as inference time. For this reason, three optimization techniques are discussed in this subsection. For further optimization strategies of the LiLaNet, the reader is referred to [Schillinger, 2018].

4.3.3.1 *Filter Reduction*

Optimizing a CNN architecture requires a large number of trainings to differentiate the influence of certain parameters or architectural choices. Each training represents a time-consuming task until the CNN converges to its final stage. For this purpose, the first optimization step represents the reduction of filters per LiLaBlock. This reduces the total number of weights of the CNN, influencing both the required training time and the inference time, thereby leading

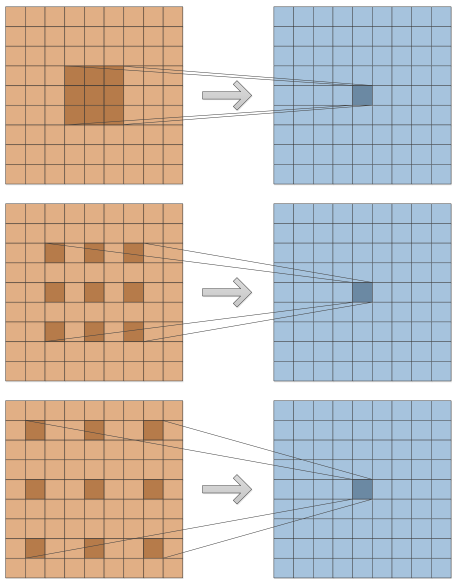


Figure 4.5: Example of the dilated convolution operation based on an input feature map (orange) and the resulting output features map (blue). Different dilation widths are shown for visualization: A dilation width $\zeta = 1$ corresponding to the common convolution layer, a dilation width $\zeta = 2$ (middle), and a dilation width $\zeta = 3$ (bottom).

to an increase in the real-time capability of the LiLaNet. At the same time, this reduction of filters reduces the number of parameters, which reduces the learning capacity of the CNN, resulting in potentially decreasing performance. Both consequences have to be analyzed against each other, which is realized within the first optimization step.

4.3.3.2 *Dilation*

As described in Section 2.2.5, pooling layers leads to an increase in the receptive field of the CNN to generate features of a higher abstraction level. Related to the small size of LiDAR images (see Section 4.3.1) as well as the high-resolution output required by semantic labeling, the application of pooling layers is restricted. For this reason, Yu and Koltun [2016] introduced the dilated convolution layer, which is an extension of the convolution layer by inserting the dilation width ζ into the equation of the convolution layer (see Equation (2.15)) with

$$\mathbf{Z}_{i,j} = \mathbf{W} * \mathbf{X} = \sum_k \sum_o \mathbf{W}_{k,o} \mathbf{X}_{i-\zeta\cdot k, j-\zeta\cdot o} \quad . \tag{4.1}$$

This increases the receptive field by retaining the resolution of the input feature map as well as the number of weights as shown in Figure 4.5. As a result, more complex features can be extracted,

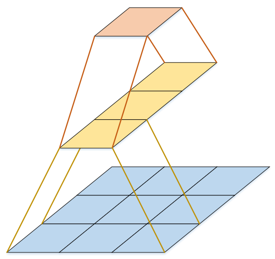


Figure 4.6: Factorization of a 3×3 filter by a 3×1 and 1×3 filter as proposed by Szegedy et al. [2016].

leading to a higher resolution for image-based semantic labeling without increasing the number of parameters [Yu and Koltun, 2016].

Transferring this knowledge to the architecture of the LiLaNet, parts of the feature extracting convolutions of the LiLaBlock (7×3, 3×7, and 3×3) are replaced by dilated convolution layers as a second optimization step.

4.3.3.3 *Factorization*

In terms of optimizing network architectures for shorter inference time, Szegedy et al. [2016] proposed to factorize a convolution layer into multiple convolution layers. This leads to a reduction of parameters at the same receptive field. They proposed e. g. the substitution of a convolution layer with a filter size $k \times o$ with two convolution layers of a filter size $k_i \times o_i$ and $k_j \times o_j$ with

$$k = k_i + k_j - 1 \text{ and} \tag{4.2}$$

$$o = o_i + o_j - 1 \text{ , while} \tag{4.3}$$

$$k_i, k_j, o_i, o_j > 0 \quad , \tag{4.4}$$

as shown in Figure 4.6. As explained in Section 4.3.3.1, the reduction of the parameters potentially decreases performance. Both the increase of the real-time capability as well as the potential performance decrease are analyzed in the last step of optimization.

For investigating this suggestion, three additional types of LiLaBlocks are defined as shown in Figures 4.7 to 4.9. There, different substitutions are applied to reduce the number of parameters and increase the real-time capability. Figure 4.7 shows a LiLaBlock that factorizes the large asymmetric filter sizes into a squared convolution layer and a smaller asymmetric convolution layer. Afterwards, the squared convolution layers are combined into one common squared convolution layer to reduce duplicated convolution layers (see Figure 4.8). Finally, the squared convolution layer is factorized into smaller asymmetric convolution layers as proposed by Szegedy et al. [2016] (see Figure 4.9).

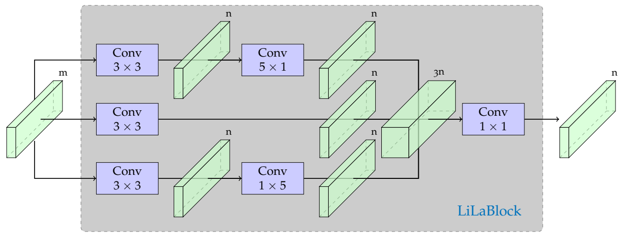


Figure 4.7: The optimized LiLaBlock structure by substitution of large convolution filters. The asymmetric filters are factorized into a smaller asymmetric filter and a 3×3 filter compared to the original LiLaBlock (see Figure 4.3).

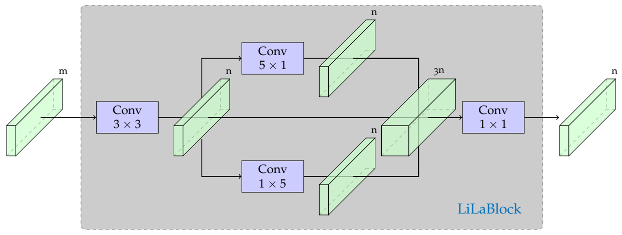


Figure 4.8: The optimized LiLaBlock structure by combining similar filters. The symmetric filters are combined to reduce duplicated filters compared to the first optimized LiLaBlock (see Figure 4.7).

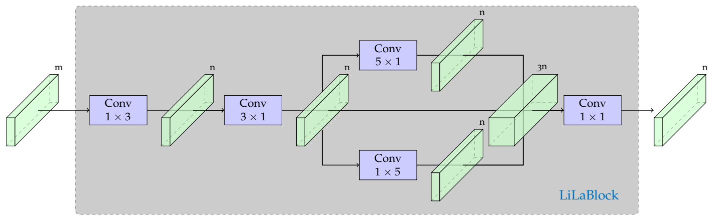


Figure 4.9: The optimized LiLaBlock structure by substitution of symmetric convolution filters. The symmetric 3×3 filters are factorized into a 1×3 and a 3×1 filter to reduce parameters compared to the second optimized LiLaBlock (see Figure 4.8).

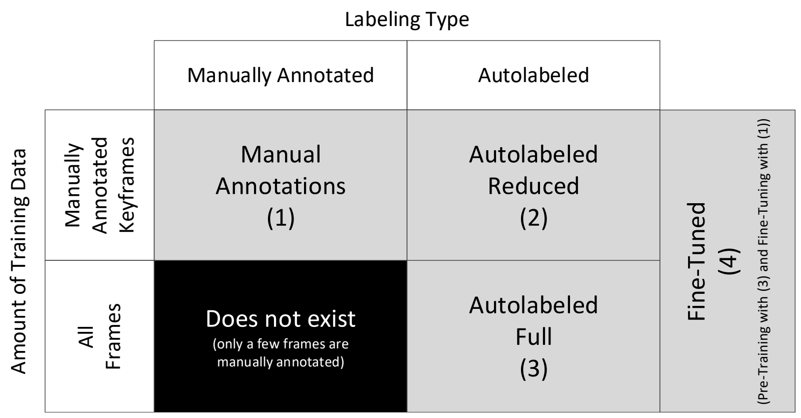


Figure 4.10: Overview of the various training strategies, which differ in the amount of training data (see Table 3.2) as well as the labeling type of the true labels. The figure is based on [Piewak et al., 2018b].

4.4 EVALUATION

The *Autolabeling* process introduced in Chapter 3 was applied to automatically generate the large-scale dataset described in Section 3.4.1, which in turn was used to train the proposed LiLaNet architecture for LiDAR-based semantic labeling. In this section, different training strategies as well as optimizations for LiLaNet corresponding to the proposed methods in Sections 4.3.2 and 4.3.3 are evaluated in detail.

The network training is performed via the Adam solver [Kingma and Ba, 2015]. The suggested default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ are used. The learning rate is fixed at $\eta = 10^{-3}$ ($\eta = 10^{-4}$ for fine-tuning) and the batch size is set to $\beta_{batch} = 5$ for the base LiLaNet architecture in Section 4.4.1 and to $\beta_{batch} = 10$ for a training speed-up of the optimized LiLaNet architectures in Section 4.4.2[2], while the weights are initialized with MSRA [He et al., 2015] for the base LiLaNet architecture in Section 4.4.1 and with Xavier [Glorot and Bengio, 2010] for the optimized LiLaNet architectures in Section 4.4.2. According to the results of Section 3.4.2, the IoU metric for performance evaluation is applied (see Section 2.3).

4.4.1 *LiLaNet Architecture*

Within this subsection, the impact of boosting the training via large-scale datasets obtained from the *Autolabeling* process (see Chapter 3) is analyzed. All evaluations are based on the test subset of the manually annotated frames of the VLP-32C dataset.

2 Note that the changes of the training parameters to increase training speed influence the learning process. For this reason, the base LiLaNet architecture is trained additionally with these parameters as a reference model to evaluate the optimizations proposed in Section 4.4.2.

The following evaluation schemes are applied, which are also visualized in Figure 4.10:

(1) **LiLaNet Manual Annotations:**
This evaluation assesses the performance of the LiDAR-based semantic labeling using a small set of cost-intensive manually annotated point clouds from the annotated keyframes for training (see Table 3.2).

(2) **LiLaNet Autolabeled Reduced:**
This evaluation measures the performance of the LiDAR-based semantic labeling using a small set of automatically generated training data based on the annotated keyframes (same keyframes that constitute the training set of the manually annotated dataset shown in Table 3.2).

(3) **LiLaNet Autolabeled Full:**
This evaluation assesses the performance of the LiDAR-based semantic labeling when using the *Autolabeling* process on the full training dataset of the original frames (see Table 3.2).

(4) **LiLaNet Fine-Tuned:**
This evaluation measures the performance of the LiDAR-based semantic labeling by fine-tuning the network using a small set of manually annotated data (1) with a pre-training based on the full *Autolabeling* process (3).

The detailed results of the various training strategies applied to LiLaNet are listed in Table 4.1 and illustrated in Figure 4.11. It can be seen that the training on manually annotated data (1) yields a higher performance than the training on autolabeled data (2), but only for as long as the same amount of data is being used. This is due to the imperfect results of the *Autolabeling* output itself. However, in practice, the amount of available manually annotated data is severely limited by the high cost of point-wise manual annotation. In contrast, the *Autolabeling* process allows to automatically generate training datasets of arbitrary size at low cost. When using the large amount of automatically generated data for training (3), LiLaNet in fact outperforms its respective variant trained on manual annotations (1) by 4.6 percentage points with regard to mIoU. Moreover, the network seems to generalize well within the LiDAR domain and suppresses some errors introduced by the *Autolabeling* process, which is indicated by the improved performance for some classes when compared to the pure *Autolabeling* output (e.g. 'pole' or 'traffic sign'). Note that the significant improvement of the class 'traffic sign' is related to the reflectivity information used as an input for the LiLaNet. Hence, the classification of that class is simplified due to the higher reflectivity of traffic signs within the LiDAR domain. Furthermore, Figure 4.11 shows that the training on the small amount of

	road	sidewalk	person	rider	small vehicle	large vehicle	two-wheeler	construction	pole	traffic sign	vegetation	terrain	mIoU
Autolabeling (see Table 3.4)	89.5	61.2	**77.0**	**51.3**	80.7	**58.5**	45.4	74.7	29.8	44.4	80.5	54.7	62.0
(1) LiLaNet Manual Annotations	90.8	61.6	48.8	15.2	79.7	37.4	22.4	71.1	35.9	69.4	75.1	59.9	55.6
(2) LiLaNet Autolabeled Reduced	86.8	51.3	44.9	13.2	72.6	32.7	19.0	60.2	20.9	45.7	66.4	44.2	46.5
(3) LiLaNet Autolabeled Full	89.7	61.7	72.2	46.6	79.6	49.6	38.3	75.0	31.5	50.2	78.0	49.8	60.2
(4) LiLaNet Fine-Tuned	**94.1**	**73.9**	73.8	48.9	**86.4**	52.2	**49.2**	**83.4**	**46.6**	**75.7**	**84.8**	**67.4**	**69.7**
(3) SqueezeSeg[3] Autolabeled Full	89.0	60.9	56.7	6.1	76.4	39.2	25.9	66.6	18.6	46.8	73.0	57.3	51.4
(4) SqueezeSeg[3] Fine-Tuned	92.2	68.2	56.8	12.9	80.1	38.5	33.1	72.0	26.1	67.1	75.7	63.0	57.1

Table 4.1: Class-wise and overall IoU scores in % of the different training strategies of the LiLaNet based on the VLP-32C dataset. The highest IoU scores of each column are marked in bold. The table is based on [Piewak et al., 2018b].

data in (1) and (2) saturates after several thousand iterations, while the training on the large-scale dataset (3) continues to increase output performance. Finally, using the manual annotations to fine-tune LiLaNet after pre-training on the automatically generated dataset (4) boosts performance by another 9.5 percentage points. This corresponds to a total gain of 14.1 percentage points over the training on manually annotated data only. Note that after fine-tuning, most classes achieve a significantly higher performance than obtained by the pure *Autolabeling* output itself. Hence, the LiDAR-based semantic labeling result provided by LiLaNet with a training supported by the *Autolabeling* process outperforms the image-based semantic labeling results projected into the LiDAR domain. It is worth noting that the network fine-tuning reaches its maximum performance only after a few thousand iterations and soon starts to overfit on the small manually annotated dataset, as can be seen in Figure 4.11. A qualitative result of the fine-tuned network is shown in Figure 4.12.

In order to compare the presented LiLaNet architecture to the state-of-the-art, the performance of the SqueezeSeg architecture [Wu et al., 2018] proposed for LiDAR-based semantic labeling on a smaller set of semantic classes ('car', 'pedestrian', 'cyclist') is also analyzed. Consider that the SqueezeSeg approach is evaluated without its CRF stage for a fair comparison. The results in Table 4.1 as well as Figure 4.11 illustrate that LiLaNet outperforms the SqueezeNet architecture in each class. This may be due to the following reasons: First, SqueezeNet uses five horizontal pooling layers which increase the learnable feature size dramatically. Consequently, the network

3 LiDAR-based semantic labeling approach of [Wu et al., 2018]

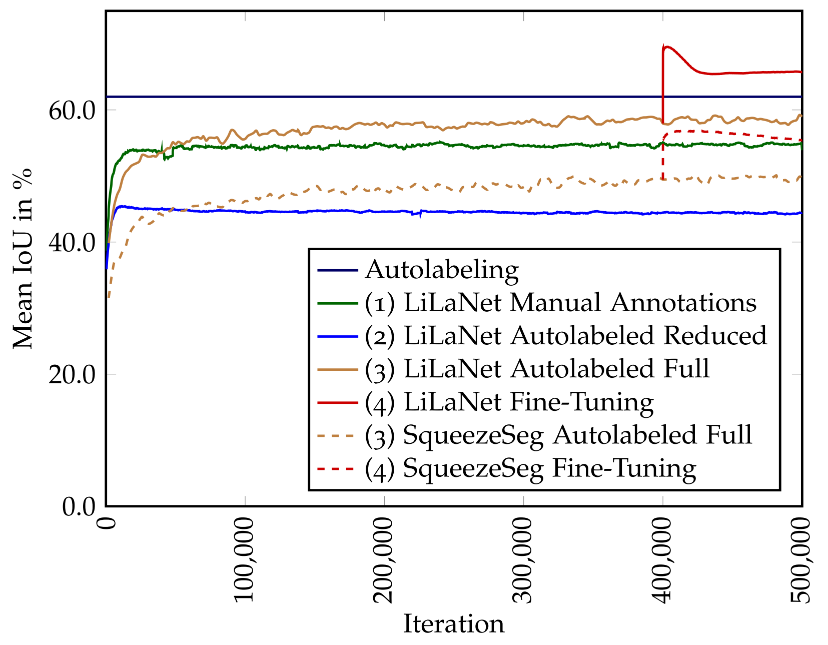


Figure 4.11: mIoU of the different training strategies and network architectures during training based on the VLP-32C dataset. The figure is based on [Piewak et al., 2018b].

may have difficulties in capturing small but relevant objects. Furthermore, the SqueezeSeg architecture does not distinguish between different object shapes and sizes in the design of the convolution filters, as is done in the LiLaBlock structure. However, the SqueezeNet does indeed also benefit from the combined process of fine-tuning after pre-training on an automatically generated dataset.

4.4.2 *Optimization*

Within this subsection, the proposed optimization strategies of Section 4.3.3 are evaluated in terms of inference time as well as classification performance. All strategies represent changes within the architecture of the LiLaBlock or hyper parameter tuning. As a result, the evaluation is performed on the optimized validation set of the autolabeled frames of the VLP-32C dataset.

4.4.2.1 *Filter Reduction*

The reduction of the filter size tends to reduce the overall number of parameters of the LiLaBlock. As a result, the training time as well as the inference time decrease. At the same time, the learning capacity of the LiLaNet decreases by a slight decrease of the mIoU. The results can be seen in Table 4.2. The evaluated filter sizes represent a decreased number related to the base LiLaNet. Thereby, in experiment (a) and (b), the typical increase to the middle and decrease to

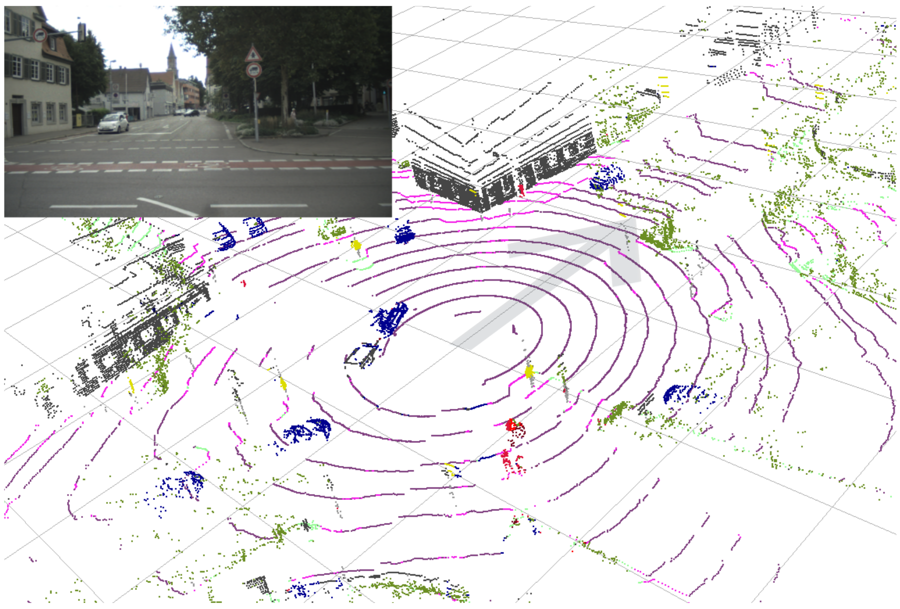


Figure 4.12: Example of a LiDAR point cloud labeled by LiLaNet. Point colors correspond to the Cityscapes semantic class color coding Cordts et al. [2016]. The following semantic classes are visualized: road, sidewalk, person, rider, small vehicle, large vehicle, two-wheeler, construction, pole, traffic sign, vegetation, terrain. The test vehicle is headed to the top right of the figure (arrow), the corresponding camera image is shown on the top left for clarity. The images are based on [Piewak et al., 2018b].

the end of the CNN as realized in encoder-decoder architectures (see Section 2.2.7) is retained. In experiments (c) and (d), this structure is replaced with a consistently growing filter size. The results show that the training time as well as the inference time can be decreased by more than 50% while reducing the mIoU score by 0.7 percentage points (see experiment (c) in Table 4.2). The highest mIoU score is realized with the base LiLaNet while using the highest inference time. As a compromise between performance and real-time capability, the filter reduction (d) is chosen for further experiments within this subsection, which decreases the inference time by 45.7% (second best inference time) while accepting a reduction of the mIoU score of 0.4 percentage points (second best mIoU score). Note that this configuration is called Slim LiLaNet within this thesis.

4.4.2.2 *Dilation*

The usage of dilation within the convolution layer (see Section 4.3.3.2) leads to an increase in the receptive field of the FCN without reducing the resolution of the input and without increasing the number of parameters. As a result, larger and more abstract features can

4 Runtime measured on a Nvidia GeForce RTX 2080 Ti.

	LiLaBlock Filters					Training Time in h	Parameters in 10^6	Inference Time[4] in ms	mIoU in %
LiLaNet	96	128	256	256	128	60.5	7.85	74.4	**60.2**
(a)	64	96	128	256	128	39.6	4.63	47.1	59.7
(b)	48	64	128	256	128	36.1	4.24	42.3	59.1
(c)	48	64	96	128	256	**29.6**	**3.07**	**33.8**	59.5
(d)	64	96	128	128	256	33.6	3.79	40.4	59.8

Table 4.2: Results of the first optimized LiLaBlock to evaluate the performance against the inference time related to the number of filters. The lowest value of the training time, the number of parameters, and the inference time as well as the highest mIoU score is marked in bold. The table is based on [Schillinger, 2018].

	Dilation Width			Share of Dilated Convolutions	mIoU
	7×3	3×3	3×7		in %
Slim LiLaNet	1	1	1	-	59.8
(a)	2	2	2	25%	62.1
(b)	2	2	2	50%	62.0
(c)	2	2	2	100%	59.0
(d)	3	3	3	25%	**62.9**
(e)	3	3	3	50%	**62.9**
(f)	3	3	3	100%	55.6

Table 4.3: Results of the second optimized LiLaBlock to evaluate the performance related to the different dilation strategies. The dilation is only applied to a share of the 7×3, 3×3, and 3×7 filters. The highest mIoU score is marked in bold. The table is based on [Schillinger, 2018].

be learned. The results can be seen in Table 4.3. The influence of the dilation is analyzed while applying the dilation to parts of the convolution layers within the LiLaBlock. For example, in experiment (a), the dilation is only applied to 25% of the filers of size 7×3, 3×3, and 3×7. Additionally, different dilation widths ($\zeta = 2$ and $\zeta = 3$) are analyzed to distinguish between the influence of a larger receptive field.

Due to the larger receptive field, dilated convolution layers focus on larger structures within the input (e. g. ‘construction’), while undilated convolutions focus on smaller structures (e. g. ‘rider’). Using both types of convolution layer increases the performance as shown in Table 4.3 (e. g. mIoU increases by 3.1 percentage points for (e)). In contrast, using only one type of convolution layer results in a lower performance (Slim LiLaNet, (c), or (f)).

The different dilation strategies clearly show a performance increase by using partly dilated convolution layers. Thereby, a larger dilation width is advantageous in combination with undilated convolutions.

	LiLaBlock Layout			Parameters in 10^6	Inference Time[5] in ms	mIoU
	tall	square	wide			
Slim LiLaNet	7×3	3×3	3×7	3.79	40.4	59.8
(a)	3×3 5×1	3×3	3×3 1×5	3.29	35.2	**60.4**
(b)	5×1	3×3 $\swarrow\downarrow\searrow$	1×5	2.07	**25.3**	59.8
(c)	5×1	1×3 3×1 $\swarrow\downarrow\searrow$	1×5	**2.00**	28.0	59.5

Table 4.4: Results of the third optimized LiLaBlock to evaluate the performance against the inference time related to different factorization strategies. The lowest value of the number of parameters and the inference time as well as the highest mIoU score is marked in bold. The table is based on [Schillinger, 2018].

4.4.2.3 *Factorization*

The factorization of convolution layers as proposed in [Szegedy et al., 2016] tends to increase the real-time capability of an FCN while maintaining performance. Therefore, different configurations of factorized LiLaBlocks are proposed in Section 4.3.3.3 (see Figures 4.7 to 4.9) and evaluated in Table 4.4.

The first step of factorization is the substitution of the large asymmetric filters with a 3×3 filter and a smaller asymmetric filter (see Figure 4.7). The result can be seen in experiment (a), where the inference time decreases by 12.9% and the performance slightly increases. This configuration confirms the potential of factorizing large filter sizes.

The second step is the combination of the same 3×3 filters to reduce duplicated filters and create common features extractors (see Figure 4.8). This configuration (see experiment (c) in Table 4.4) reduces the inference time by additional 28.1%, while loosing the performance gain of experiment (a). As a result, the combination of similar convolution layers shows a clear inference time improvement while slightly decreasing performance.

The last step of factorization replaces the remaining 3×3 filter with the corresponding factorized convolution layers (see Figure 4.9 and experiment (c)). Here, a decrease in the parameters as well as a further performance decrease can be recognized. However, the inference time increases. This is related to the hardware structure of a **G**raphics **P**rocessing **U**nit (GPU), which is able to process identical instructions in parallel. By factorizing the 3×3 filter, the execution is serialized, while the parallel GPU cores are not utilized optimally.

5 Runtime measured on a Nvidia GeForce RTX 2080 Ti.

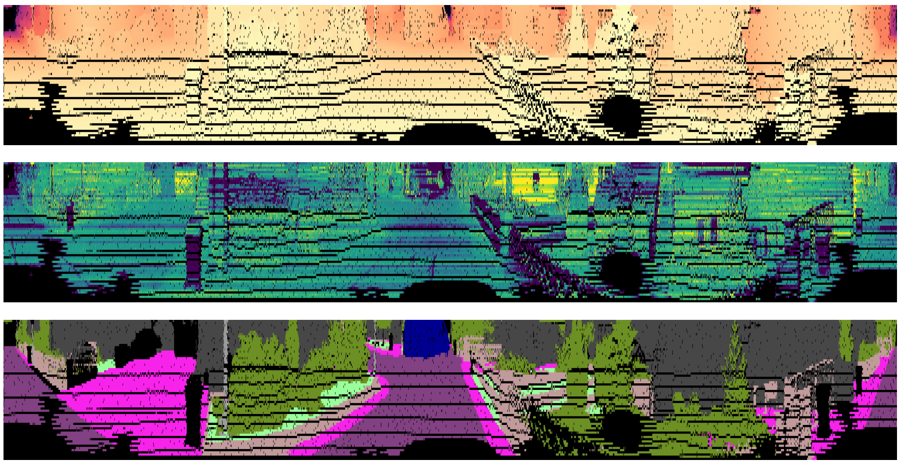


Figure 4.13: Sparse LiDAR images of the SemanticKITTI benchmark [Behley et al., 2019]. Due to the ego-motion correction of the point cloud, the 2D cylindrical LiDAR images contain a large amount of invalid points, which can be seen within the distance image at the top (yellow = near, red = far, black = invalid), within the reflectivity image in the middle (yellow = high reflectivity, blue = low reflectivity, black = invalid), and within the semantic label image at the bottom (color coding is based on the Cityscapes label colors [Cordts et al., 2016]).

As a result, this configuration shows that the factorization of small filter sizes can be disadvantageous.

Comparing the results to the Slim LiLaNet, an inference time reduction by 37.4% could be achieved by factorizing large filters and combining similar convolutions while maintaining the performance measured based on the mIoU score.

4.4.3 *SemanticKITTI Benchmark Evaluation*

Recently, a benchmark for LiDAR-based semantic labeling known as SemanticKITTI was published [Behley et al., 2019]. In it, Behley et al. [2019] used the LiDAR point clouds of the Kitti Odometry benchmark [Geiger et al., 2012a] and manually annotated them for a LiDAR-based semantic labeling benchmark. Although the point clouds are only available in ego-motion corrected form (see ego-motion compensation in Section 3.3.1.2), which results in a lossy projection of the 3D point cloud into a sparse 2D LiDAR image (see Figure 4.13), the optimized LiLaNet was applied to this benchmark. Note that the 2D projection of the ego-motion corrected LiDAR point cloud into an image of size 2,048 $\times$ 64 results in a mean loss of approximately 20% of the LiDAR points, depending on the speed of the ego-motion due to the projection of multiple points to the same pixel position. The optimized LiLaNet is represented by the Slim LiLaNet of Section 4.3.3.1 by applying a dilation with a dilation width $\zeta = 3$ on 50% of the filters (see Section 4.3.3.2) and using the optimized factorized LiLaBlock of Figure 4.8 (see experiment (b) in Section 4.3.3.3). Additionally, it has to be mentioned that the optimized LiLaNet architecture was not optimized towards the HDL-64, which is used within this benchmark. Reducing the information loss of the projection method, the optimized LiLaNet is executed twice per point cloud as a second experiment: Once for the closest points and once for the farthest points, which are projected into the same pixel positions of the 2D cylindrical LiDAR image. Afterwards, the results are merged.

The results of the benchmark are shown in Table 4.5, where each approach is trained on the training set (sequences 00 to 10, except sequence 08, which is used for validation) and evaluated on the testing set (sequences 11 to 21) of the SemanticKITTI benchmark [Behley et al., 2019]. The optimized LiLaNet returns similar results as state-of-the-art approaches with a factor of nearly 25 fewer parameters and an inference time of 73.8 ms[6]. Consider that the lossy projection introduces false classifications, reducing the overall mIoU score. Adapting the semantic labeling approach with a second execution of the LiLaNet concerning the farthest points (LiLaNet Twice in Table 4.5), all mentioned state-of-the-art approaches are outperformed by at least 1.5 percentage points based on the mIoU score. Note that all mentioned approaches use the 2D cylindrical LiDAR images except the PointNet [Qi et al., 2017a] and the PointNet++ [Qi et al., 2017b]. These two network architectures show difficulties of recognizing the large-scale outdoor scenes as mentioned in Section 4.2.

4.5 OUTCOME

In this chapter, the point-wise multi-class semantic labeling of 3D point clouds is considered by transferring the concept of pixel-wise image-based semantic labeling to the LiDAR domain. As a result, the LiLaNet, a novel CNN architecture for efficient LiDAR-based semantic labeling, is proposed including some optimization in terms of real-time capability as well as performance. This architecture significantly outperforms current state-of-the-art CNNs for multi-class LiDAR-based semantic labeling when evaluated on different manually annotated datasets.

Furthermore, a training technique is presented combining the automatically generated training dataset of Chapter 3 with a fine-tuning step based on small-scale manually annotated data. This yields a performance boost of up to 14 percentage points, while keeping manual annotation efforts low.

6 Runtime measured on a Nvidia GeForce RTX 2080 Ti for an HDL-64 scan.

	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign	mIoU
PointNet[1]	61.6	35.7	15.8	1.4	41.4	46.3	0.1	1.3	0.3	0.8	31.0	4.6	17.6	0.2	0.2	0.0	12.9	2.4	3.7	14.6
PointNet++[2]	72.0	41.8	18.7	5.6	62.3	53.7	0.9	1.9	0.2	0.2	46.5	13.8	30.0	0.9	1.0	0.0	16.9	6.0	8.9	20.1
SqueezeSeg[3]	85.4	54.3	26.9	4.5	57.4	68.8	3.3	16.0	4.1	3.6	60.0	24.3	53.7	12.9	13.1	0.9	29.0	17.5	24.5	29.5
SqueezeSegV2[4]	88.6	67.6	45.8	17.7	73.7	81.8	13.4	18.5	17.9	14.0	71.8	35.8	60.2	20.1	25.1	3.9	41.1	20.2	36.3	39.7
DarkNet21Seg[5]	91.4	74.0	57.0	26.4	81.9	85.4	18.6	26.2	26.5	15.6	77.6	48.4	63.6	31.8	33.6	4.0	52.3	36.0	50.0	47.4
LiLaNet Once[6]	89.9	71.6	49.5	21.8	81.5	86.1	20.8	32.6	31.3	18.3	80.1	55.0	64.5	44.3	30.0	8.5	51.2	41.5	50.7	48.9
DarkNet53Seg[5]	**91.8**	**74.6**	**64.8**	**27.9**	84.1	86.4	**25.5**	24.5	32.7	**22.6**	78.3	50.1	64.0	36.2	33.6	4.7	**55.0**	38.9	52.2	49.9
LiLaNet Twice[7]	90.1	72.7	50.1	21.6	**84.9**	**91.1**	21.2	**34.9**	**33.7**	19.1	**82.6**	**60.6**	**65.8**	**47.5**	**33.7**	**9.2**	54.7	**48.9**	**54.1**	**51.4**

Table 4.5: Class-wise and overall IoU scores in % of the different network architectures based on the SemanticKITTI benchmark (testing set) [Behley et al., 2019]. The highest IoU scores of each column are marked in bold. Parts of this table are based on [Behley et al., 2019] © 2019 IEEE.

1 [Qi et al., 2017a]
2 [Qi et al., 2017b]
3 [Wu et al., 2018]
4 [Wu et al., 2019]
5 [Behley et al., 2019]
6 Slim LiLaNet with the optimized LiLaBlock of Sections 4.3.3.2 and 4.3.3.3 executed once (for the closest points).
7 Slim LiLaNet with the optimized LiLaBlock of Sections 4.3.3.2 and 4.3.3.3 executed twice (for the closest and the farthest points).

This results in a high-quality real-time capable multi-class semantic labeling approach, which represents the third step of the research pipeline (see Figure 1.5) and forms the basis for the next step of multi-modal Stixels.

5 EXTENSION - REDUCTION OF SENSOR DEPENDENCE

CONTENTS

Within this chapter, an extension of the third step of the research pipeline (see Figure 1.5) is presented. This extension focuses on the cross-sensor portability of LiDAR-based semantic labeling approaches. Parts of this section have previously appeared in [Piewak et al., 2019].

5.1 OVERVIEW

The task of semantic labeling originated in the field of computer vision (see Section 1.1.3), with the aim to individually classify each pixel in a given image [Garcia-Garcia et al., 2018]. Within recent years, this task has been applied successfully to other sensor modalities such as LiDAR sensors or RaDAR sensors [Feng et al., 2020]. Across all modalities, state-of-the-art results are obtained by modern deep learning techniques. However, approaches based on the application of deep CNNs are often tailored to the specific characteristics of the respective sensor instance. Transferring a network architecture from e.g. one LiDAR sensor model to another represents a significant challenge, especially due to sensor specific design choices with regard to network architecture as well as data representation. This effect is intensified by the fact that LiDAR sensor technology keeps evolving at a fast pace, with numerous new sensor types being announced or released to the market every year, featuring novel scanning patterns, as well as ever increasing range and spatial resolution.

Given the sensor-specific tailoring of many current CNN architectures as well as the difficulty of generating annotated training data to scale (see Chapter 3), in this section, the cross-sensor portability of neural network architectures is considered for LiDAR-based semantic labeling. Portable network architectures could provide a solution to above challenges by enabling the reuse of annotated data and reducing the effort required for adapting CNN architectures to new sensor models.

5.2 RELATED WORK

In recent years, LiDAR point cloud processing has gained more and more attention due to decreasing hardware cost and an increasing number of sensor models available on the market. The main difference between the different algorithmic approaches is the representation of the 3D point cloud to be processed as discussed in Section 4.2. Three different types of point cloud representations are commonly found in the literature.

First, the point cloud can be represented as a projection of the 3D data to a 2D space e. g. as a **B**ird's **E**ye **V**iew (BEV) [Yang et al., 2018, Beltran et al., 2018] or a cylindrical projection as described in Section 4.3.1. The main advantage of these projection methods is the efficient processing based on 2D convolution layers. However, the generated CNNs are usually rather sensor specific and can only be transferred to different sensor models with significant effort. Such transfer learning techniques [Torrey and Shavlik, 2009] use either a pre-trained CNN on the same [Yosinski et al., 2014] as well as on different tasks [Garcia-Garcia et al., 2018] or multi-task approaches [Hong et al., 2016, Liu et al., 2019] to benefit from the combination of different datasets. Most existing approaches have been developed for the camera domain, where the input representation is very similar between different cameras, especially after a rectification of the obtained image. Within the LiDAR domain, the input representation can strongly differ between LiDAR sensors with regard to range, reflectivity representation[1], non-uniform resolution, and aspect ratio[2]. Particularly for 2D projections of the LiDAR data, this can represent a significant challenge due to the changes of feature representations for classification tasks.

An alternative method to represent point clouds is by way of an unordered point set [Qi et al., 2017a,b, Li et al., 2018]. While this approach is able to handle an arbitrary number of unordered points,

1 The characteristics of the reflectivity representation can already change between LiDAR sensors of the same type.

2 Some LiDAR sensors have e. g. a different vertical resolution around the horizon compared to the full field of view.

the resulting classification performance tends to be problematic in large-scale outdoor scenarios as are usually encountered in the field of autonomous driving (see Sections 4.2 and 4.4.3).

The third common type of point cloud representation performs a discretization of the 3D space into a voxel grid [Maturana and Scherer, 2015] or an octree [Riegler et al., 2017]. Riegler et al. [2017] evaluated a semantic segmentation task by predicting a single semantic class for all points within an octree cell. Occupancy information is used as a feature for each voxel. Zhou and Tuzel [2018] proposed VoxelNet for the task of object detection. Here, the idea of PointNet [Qi et al., 2017a] is adopted to extract features from an arbitrary number of points per voxel. While this representation increases cross-sensor portability due to the regularity of the voxel grid, the additional dimension and the employed 3D convolution layers dramatically increase the training and inference time for larger CNN architectures. Lang et al. [2019] alleviated this problem by introducing PointPillars. Here, the 3D space is compressed to a **2.5-D**imensional (2.5D) space by reducing the number of voxels along the vertical axis to one, leading to an intermediate structure resembling pillars. Similar to BEV-based approaches, this 2.5D pseudo-image can then be processed using 2D convolutions, thereby benefiting from the learned features per pillar. Both VoxelNet [Zhou and Tuzel, 2018] and PointPillars [Lang et al., 2019] are optimized for object detection tasks. In this section, a CNN architecture for point-wise semantic segmentation is built upon these approaches, which increase the portability across different LiDAR sensors compared to other point cloud representation strategies.

5.3 METHOD

In this section, previous work [Zhou and Tuzel, 2018, Lang et al., 2019] is extended and a new CNN architecture for the point-wise semantic labeling of LiDAR data is proposed. Increasing the cross-sensor portability, a voxel-based processing approach is chosen, similar to the data structures used by VoxelNet [Zhou and Tuzel, 2018] and PointPillars [Lang et al., 2019]. Hence, the resulting pillar-based labeling network is called **Pi**llar **La**beling **Net**work (PiLaNet).

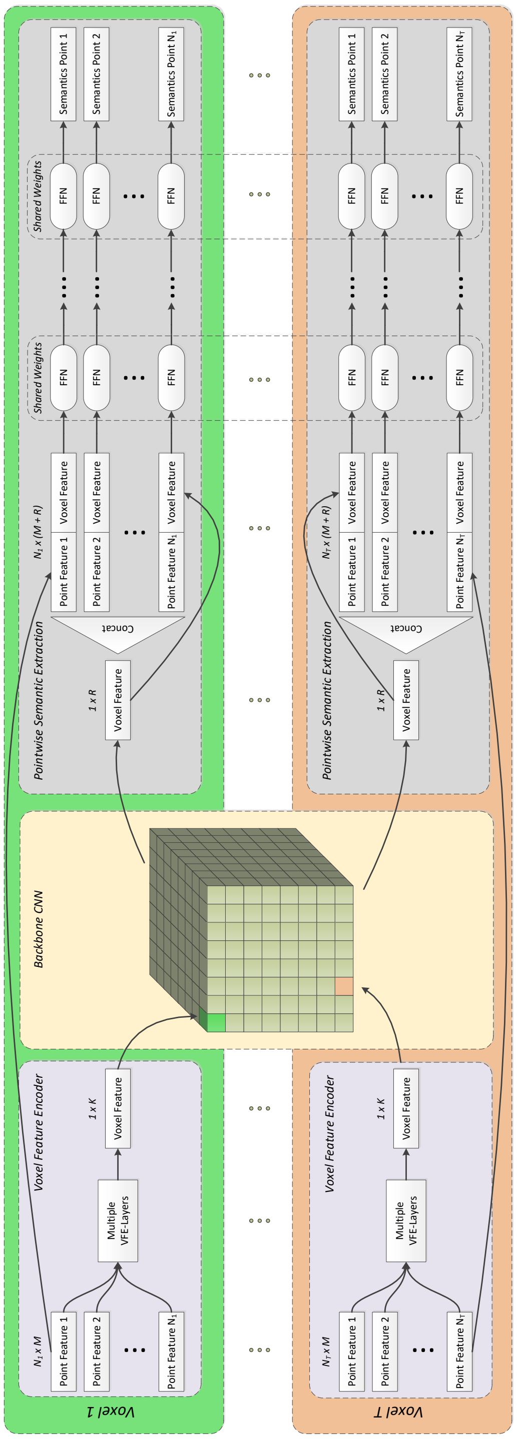


Figure 5.1: 3D voxel CNN architecture for point-wise semantic labeling. The voxel space as well the processing chain per voxel are represented in green and orange (different colors per voxel). The three main components (**V**oxel **F**eature **E**ncoder (also called **V**oxel **F**eature **E**xtractor within the literature) (VFE), backbone CNN, and point-wise semantic extraction head) are represented as violet, yellow, and gray boxes. This illustration is based on [Piewak et al., 2019] © 2019 IEEE.

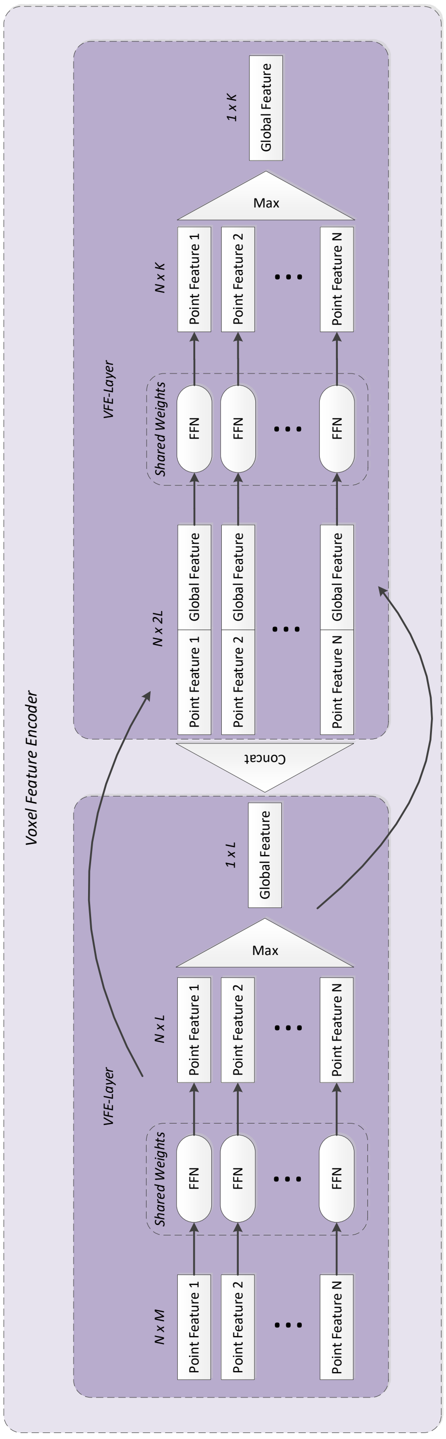


Figure 5.2: Example of a voxel feature encoder based on two consecutive VFE-Layers. The original point features serve as input to the first VFE-Layer, while the second VFE-Layer uses the combination of the point-wise features with the global features as input. This illustration is based on [Piewak et al., 2019] © 2019 IEEE.

Within the following subsections, the network architecture of the PiLaNet is described in detail. It consists of three main components (see Figure 5.1):

- The VFE, which generates a feature vector that encodes the properties of the voxel content
- The backbone CNN, which accumulates the generated voxel features in 3D space
- The point-wise semantic extraction head, which infers a semantic label for each point from the encoded voxel features[3]

5.3.1 *Voxel Feature Encoder*

The VFE represents a network component designed to condense the essential properties of all points contained within a voxel into one feature vector (see Figure 5.2). This idea was originally proposed in PointNet [Qi et al., 2017a] to encode a full point cloud into a single feature vector. It was later adapted and applied to individual voxels in the VoxelNet architecture [Zhou and Tuzel, 2018]. Each point is represented by its global cartesian coordinates p_{x_g}, p_{y_g}, and p_{z_g}, the measured reflectivity ξ as well as the relative cartesian coordinates with respect to the mean of all points within a voxel p_{x_v}, p_{y_v}, and p_{z_v}. To obtain an initial point-wise feature encoding, each point is processed individually via FFNs. Each FFN consists of one layer, representing a trainable feature combination of each point. These sub-networks use shared weights to enforce identical feature encoders for each point. Subsequently, a maximum pooling operator is applied to the point-wise features in order to generate a single feature vector per voxel. The combination of these processing steps is also known as the VFE-Layer in the literature [Zhou and Tuzel, 2018].

The VFE-Layer can be applied repeatedly by concatenating the encoded point-wise features with the voxel feature vector resulting from the previous step (see Figure 5.2). Eventually, the FFN as well as the maximum pooling operator are applied once more to obtain a final refined feature vector for each voxel.

Since the VFE can handle arbitrary numbers of input points, it is applicable to various voxel sizes and point cloud densities. It provides a parameterizable representation which is highly portable between different sensor types.

The features computed by the VFE-Layers are combined into a 3D voxel grid, which forms the input to the backbone CNN described in the next subsection.

3 This semantic extraction head replaces the region proposal network of related object detection approaches [Zhou and Tuzel, 2018, Lang et al., 2019].

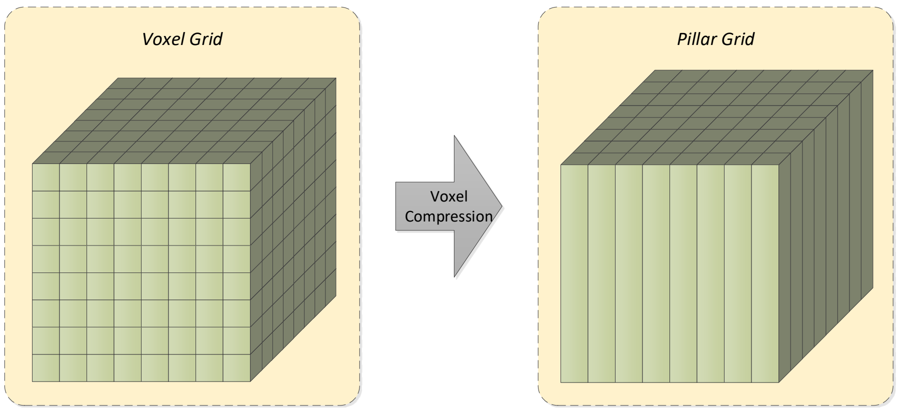


Figure 5.3: Compression of the 3D space to the 2.5D space by reducing the number of voxels along the vertical axis to one, leading to an intermediate structure resembling pillars as proposed in [Lang et al., 2019]. This illustration is based on [Piewak et al., 2019] © 2019 IEEE.

5.3.2 *Backbone CNN*

To fully exploit the spatial context within the encoded data, a CNN backbone architecture is applied to the voxel grid. Here, various architectural choices are possible. VoxelNet [Zhou and Tuzel, 2018] employs a full 3D backbone CNN to extract features from the voxel representation via 3D convolution layers. This architecture is well suited for the task of object detection, where the size of the 3D representation can be reduced by applying pooling or strided convolution layers without a significant loss in output accuracy. However, for the semantic labeling task considered in this thesis, a fine-grained point-wise prediction is required. In this case, a full 3D CNN entails high computational complexity, leading to excessive memory consumption as well as long training and inference times. To reduce the dimensionality of the voxel grid and, hence, the computational complexity of the backbone CNN, the concept of pillars as proposed in [Lang et al., 2019] (see Figure 5.3) is adopted. The dimensionality of the voxel grid is reduced by directly encoding the height information of each point within its associated pillar. The resulting representation resembles a 2D pseudo-image similar to a BEV. Consequently, a 2D CNN architecture can be applied to process the voxel representation in an efficient manner while retaining the full information of the encoded data.

5.3.3 *Point-Wise Semantic Extraction*

The backbone CNN yields voxel-wise output features similar to the probability score maps of object detection approaches [Zhou and Tuzel, 2018, Lang et al., 2019]. These features can be used to infer

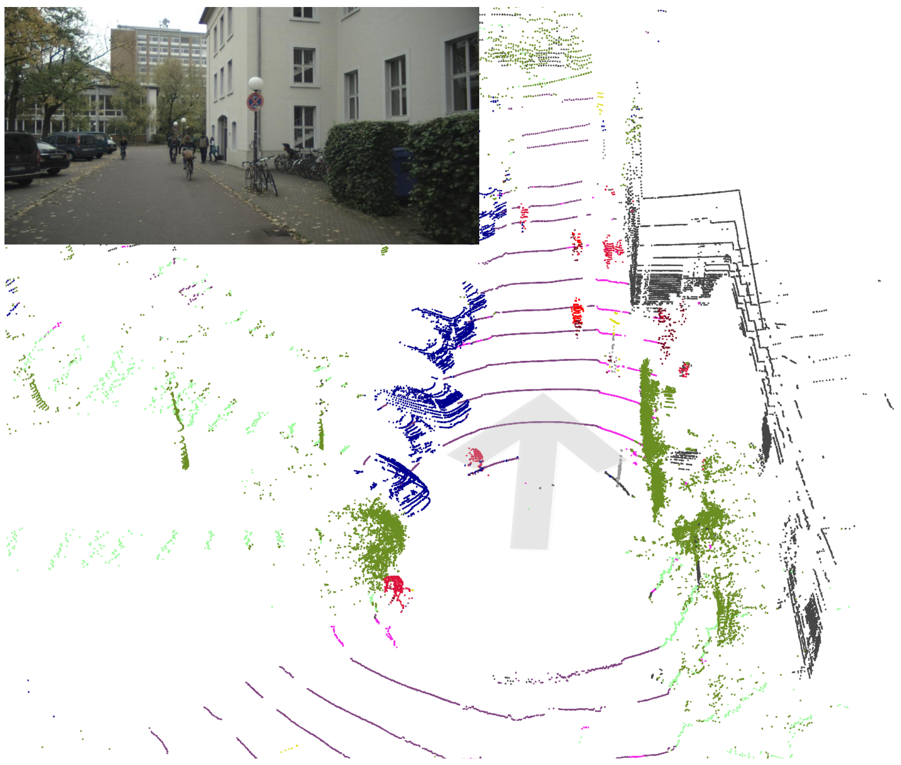


Figure 5.4: Exemplary semantic labeling result obtained via the proposed PiLaNet on a VLP-32C point cloud. The corresponding camera image on the top left is shown for clarity, with the camera's field of view covering the top center of the point cloud (arrow). The following semantic classes are visualized: road, sidewalk, person, rider, small vehicle, large vehicle, two-wheeler, construction, pole, traffic sign, vegetation, terrain. The images are based on [Piewak et al., 2019] © 2019 IEEE.

a semantic class for each voxel. However, to obtain a point-wise semantic labeling result, this approach is not sufficient, particularly for large voxels such as pillars, where all points within a voxel would be assigned the same semantic class. Therefore, a point-wise semantic extraction head is introduced as shown in Figure 5.1. For each point, the corresponding input vector $(p_{x_g}, p_{y_g}, p_{z_g}, \xi, p_{x_v}, p_{y_v}, p_{z_v})$ of the voxel feature encoder is concatenated with the voxel-wise features extracted by the backbone CNN. Subsequently, each point is processed independently by multiple FFNs to extract a point-wise classification result independent of its containing voxel. In this manner fine-grained semantic class predictions within a voxel are achieved.

5.4 EVALUATION

This section describes the evaluation of the proposed network architecture PiLaNet against a state-of-the-art reference method, with a

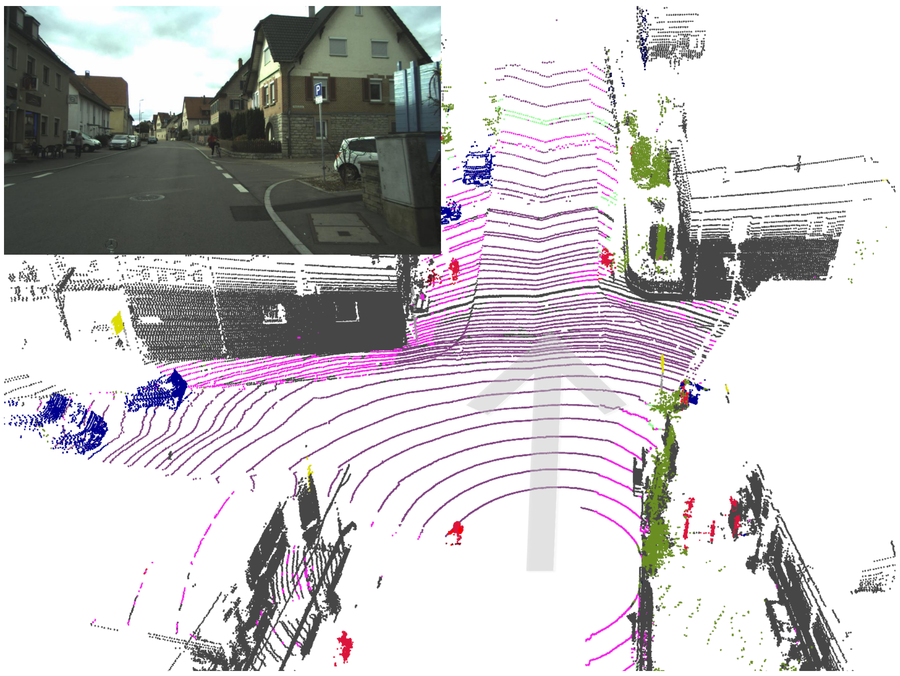


Figure 5.5: Exemplary semantic labeling result obtained via the proposed PiLaNet on a VLS-128 point cloud. The corresponding camera image on the top left is shown for clarity, with the camera's field of view covering the top center of the point cloud (arrow). The following semantic classes are visualized: road, sidewalk, person, rider, small vehicle, large vehicle, two-wheeler, construction, pole, traffic sign, vegetation, terrain. The images are based on [Piewak et al., 2019] © 2019 IEEE.

focus on cross-sensor portability as visualized in Figures 5.4 and 5.5. The evaluation is based on the manually annotated frames of the testing set of the VLP-32C and of the VLS-128 dataset (see Section 3.4.1). As a reference, the LiLaNet is used (see Section 4.3.2).

PiLaNet implements the voxel representation as described in Section 5.3, whereby the voxel space is limited to the range[4] $(0.0\text{ m}, -30.0\text{ m}, -2.0\text{ m}) \leq (p_{x_g}, p_{y_g}, p_{z_g}) \leq (60.0\text{ m}, 30.0\text{ m}, 9.2\text{ m})$. The number of voxels is set to $(num_{p_x}, num_{p_y}, num_{p_z}) = (300, 300, 1)$. These parameters were optimized based on available GPU memory, overall network performance, and training time. The used VFE is composed of two VFE-Layers as explained in Section 5.3.1, with a voxel feature vector size of 128. For the implementation of the VFE, the maximum number of points per voxel is restricted to 35 and a random sampling is applied in case the limit is exceeded. Since a single voxel is used along the Z-axis (pillars), the backbone is modeled as a 2D CNN. For a valid comparison, the LiLaNet is used as a reference method and as a backbone CNN, with the number

4 The axis of the cartesian coordinate system follows the ISO 8855 [Deutsches Institut für Normung, 2011].

of output features per voxel set to 24. The point-wise semantic extraction head includes three consecutive fully connected layers with 64, 64, and 12 features. The last layer provides the scores of the 12 semantic classes (see Section 3.3.3 without the class 'sky'). Aside from the classification score, after each layer, a ReLU is applied.

Note that no optimization of the considered network architectures for the specific sensor types is performed. In this context, a valid evaluation of the unmodified CNN architectures and corresponding point-cloud representations in terms of cross-sensor portability are achieved.

The training of both LiLaNet and PiLaNet is performed with a batch size[5] $\beta_{batch} = 8$ via the Adam solver [Kingma and Ba, 2015]. With regard to the training strategy, the fine-tuning strategy of Section 4.4.1 is applied. Thereby, the training on the autolabeled set is run for 300,000 iterations before starting the fine-tuning. As training parameters, the suggested default values for the Adam solver of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ are used. The learning rate is fixed at $\eta = 10^{-3}$ ($\eta = 10^{-4}$ for fine-tuning), while the weights are initialized with MSRA [He et al., 2015].

Evaluation is performed based on the testing set of the manually annotated frames, whereby for each class, the IoU metric as well as the overall mIoU are calculated. The evaluation is restricted to the defined voxel range to ensure comparability between the different point cloud representations, i.e. the voxel representation used by PiLaNet and the cylindrical 2D projection of LiLaNet. This results in slightly different IoU scores in Table 5.1 as compared to Table 3.4.

Several evaluation stages are performed on both datasets, which are discussed in more detail in the following subsections:

1. Performance of the networks (LiLaNet and PiLaNet) trained, fine-tuned, and evaluated on the same sensor
2. Performance of the networks (LiLaNet and PiLaNet) trained and fine-tuned on one sensor and evaluated on the other sensor
3. Performance of the networks (LiLaNet and PiLaNet) trained and fine-tuned on one sensor and additionally fine-tuned as well as evaluated on the other sensor

5.4.1 *Same-Sensor Evaluation*

First, the networks are trained as described in Section 5.4, whereby training, fine-tuning as well as evaluation are performed on the same dataset (but on different subsets). The second block of Table 5.1

5 In case the network does not fit into the GPU memory, the batch is distributed over multiple devices.

	road	sidewalk	person	rider	small vehicle	large vehicle	two-wheeler	construction	pole	traffic sign	vegetation	terrain	mIoU
Autolabeling evaluation VLP-32C	90.3	62.4	79.7	52.7	83.1	61.9	46.5	76.5	33.6	45.1	79.6	55.5	63.9
Autolabeling evaluation VLS-128	88.3	58.3	64.2	44.6	82.8	56.6	39.8	83.0	26.5	36.7	83.1	55.4	60.1
LiLaNet VLP-32C evaluation VLP-32C	**93.9**	**73.0**	72.2	45.3	86.3	49.3	47.2	**84.1**	48.3	**79.0**	**83.0**	**66.6**	69.0
PiLaNet VLP-32C evaluation VLP-32C	93.2	69.8	**81.8**	**50.2**	**88.9**	**67.3**	**47.5**	81.6	**48.8**	77.5	79.4	64.3	**70.9**
LiLaNet VLS-128 evaluation VLS-128	89.9	62.3	59.5	19.1	82.1	24.8	31.5	83.5	41.3	**48.3**	**85.0**	**65.7**	57.7
PiLaNet VLS-128 evaluation VLS-128	**91.1**	**63.9**	**69.1**	**49.7**	**88.5**	**41.9**	**40.0**	**85.3**	**47.4**	42.2	84.5	64.2	**64.0**
LiLaNet VLP-32C evaluation VLS-128	55.6	19.9	17.5	7.5	45.1	4.6	9.5	65.7	33.4	**36.5**	72.7	28.0	33.0
PiLaNet VLP-32C evaluation VLS-128	**46.5**	**20.0**	**48.5**	**35.4**	**81.2**	**34.5**	**20.3**	**73.7**	**41.3**	33.9	**77.1**	**30.7**	**45.2**
LiLaNet VLS-128 evaluation VLP-32C	83.0	30.8	40.2	4.8	68.3	22.7	19.5	63.8	29.6	**46.3**	68.1	**49.7**	43.9
PiLaNet VLS-128 evaluation VLP-32C	**86.2**	**50.4**	**75.9**	**36.5**	**80.8**	**34.0**	**28.4**	**69.5**	**35.4**	35.5	**70.5**	47.5	**54.2**
LiLaNet VLP-32C fine-tuned VLS-128 evaluation VLS-128	87.8	**58.0**	46.0	13.6	77.1	14.1	25.0	81.5	39.0	**46.0**	**84.4**	**65.2**	53.2
PiLaNet VLP-32C fine-tuned VLS-128 evaluation VLS-128	**88.5**	**58.0**	**64.9**	**42.5**	**87.3**	**49.5**	**36.9**	**84.4**	**45.6**	45.1	83.6	63.1	**62.5**
LiLaNet VLS-128 fine-tuned VLP-32C evaluation VLP-32C	92.2	66.3	63.8	29.1	83.1	45.6	36.5	**80.3**	41.1	75.8	**80.1**	**63.8**	63.1
PiLaNet VLS-128 fine-tuned VLP-32C evaluation VLP-32C	**92.6**	**66.6**	**77.6**	**53.1**	**86.8**	**58.8**	**39.0**	78.9	**43.6**	**76.3**	77.3	62.3	**67.7**
LiLaNet VLP-32C fine-tuned VLS-128 evaluation VLP-32C	89.4	56.8	67.1	27.4	81.6	34.0	35.2	78.2	44.0	**55.2**	**77.5**	59.4	58.8
PiLaNet VLP-32C fine-tuned VLS-128 evaluation VLP-32C	**90.8**	**61.8**	**81.2**	**51.0**	**86.8**	**56.1**	**42.4**	**79.0**	**45.7**	52.1	77.4	**60.5**	**65.4**
LiLaNet VLS-128 fine-tuned VLP-32C evaluation VLS-128	74.5	39.2	40.3	7.1	72.6	16.2	21.6	79.8	38.2	39.2	81.5	49.0	46.6
PiLaNet VLS-128 fine-tuned VLP-32C evaluation VLS-128	**86.0**	**53.9**	**67.3**	**51.9**	**87.7**	**48.6**	**37.5**	**84.4**	**45.6**	**42.6**	**83.1**	**55.6**	**62.0**

Table 5.1: Overview of the results obtained in the different evaluation stages discussed in Section 5.4. Each row represents a semantic labeling approach. The corresponding descriptions are given in the first column, where the first two rows describe the architecture and the training dataset as well the additional fine-tuning dataset (if used). The last row of the description denotes the dataset used for evaluation. The top results of the respective network architectures trained with the same strategy are marked in bold. Parts of this table are based on [Piewak et al., 2019] © 2019 IEEE.

shows the results of this same-sensor evaluation strategy. The proposed PiLaNet clearly outperforms LiLaNet on the VLS-128 dataset and reaches slighly better results on the VLP-32C dataset as well. This indicates that the voxel representation outperforms the cylindrical 2D representation in terms of output quality. This might be related to the feature representation within the voxel space, which is more location independent (e.g. height or distance to the sensor) as compared to the features within the cylindrical 2D representation.

Interestingly, both network architectures achieve better results on the VLP-32C dataset than on the VLS-128 dataset. This effect is mainly due to the smaller overall size of the VLS-128 dataset. Also, the decrease in performance is larger for LiLaNet than for PiLaNet. This can be attributed to the higher resolution of the VLS-128, which directly influences the object sizes within the cylindrical point cloud representation. This indicates that the PiLaNet architecture is more suitable for transfer between sensors than LiLaNet.

5.4.2 *Cross-Sensor Evaluation*

Using the already trained networks of Section 5.4.1, the second evaluation stage is performed on the data of the opposite sensor in order to evaluate cross-sensor portability. The corresponding results are listed in the third block of Table 5.1. PiLaNet clearly outperforms LiLaNet by more than 10 percentage points, confirming that the voxel representation results in a far more portable architecture than the cylindrical projection. At the same time, the mIoU of PiLaNet drops by more than 16 percentage points compared to the same-sensor evaluation results. This drop might in part be due to the backbone CNN, which has to handle strongly varying densities for the different sensor types and resolutions. While the cross-sensor results are very promising, it is worth mentioning that there is still ample room for tuning the voxel representation for portability.

5.4.3 *Cross-Sensor Fine-Tuning*

As seen in Section 5.4.2, the direct application of network models to different sensors leads to a significant drop in output performance. Therefore, a data-driven adaptation step is proposed, whereby the pre-trained model is fine-tuned on the target sensor using manually annotated data. Note that only a small amount of manually annotated data is required, while the full amount of autolabeled data of the target sensor is not used. This represents the use-case of changing a sensor model without reacquiring a large amount of data for the target sensor, which reduces the time as well as the cost of adapting CNNs towards a different sensor, especially for the fast-paced development of LiDAR hardware technology. The results

Network Architecture	Inference Time[6] in ms
LiLaNet	74.4
PiLaNet with $(num_{p_x}, num_{p_y}, num_{p_z}) = (300, 300, 1)$	138.1
PiLaNet with $(num_{p_x}, num_{p_y}, num_{p_z}) = (600, 600, 1)$	468.2

Table 5.2: Inference time of the LiLaNet and the PiLaNet evaluated on the VLP-32C dataset. Due to the restriction of the voxel space, two different voxel space ranges are evaluated for the PiLaNet.

of this strategy are shown in the fourth block of Table 5.1. After fine-tuning the network architecture on the target sensor, PiLaNet still outperforms LiLaNet, which once more confirms the superior portability of the PiLaNet architecture, allowing for an adaptation to the target sensor type with only small amounts of additional data. Note that the performance on the pre-trained sensor is decreased as shown in the fifth block of Table 5.1 (compared to the second block of Table 5.1). However, the performance drop of the PiLaNet is smaller with 5.5 percentage points on the VLP-32C dataset and with 2.0 percentage points on the VLS-128 dataset compared to the performance drop of the LiLaNet with 10.2 percentage points on the VLP-32C dataset and with 11.1 percentage points on the VLS-128 dataset. This confirms once more the superior portability of the PiLaNet architecture.

When compared to pure *Autolabeling*, the training strategy of fine-tuning the architecture on a target sensor increases the mIoU of PiLaNet by 2.4 percentage points on the VLS-128 dataset and by 3.8 percentage points on the VLP-32C dataset. This shows that the presented adaptation process can be used to successfully transfer network architectures across sensors by applying only a small manually annotated dataset for fine-tuning instead of using another sensor modality such as cameras to generate reference data. It is conceivable that a fine-tuned PiLaNet can be used to extend the *Autolabeling* concept, which originally relies on an additional sensor modality, in order to automatically generate large-scale datasets for new sensor types of the same modality (e.g. LiDAR to LiDAR *Autolabeling*).

5.4.4 *Inference Time*

As seen within the previous sections, the PiLaNet outperforms the LiLaNet in terms of mIoU performance and portability between different sensors. This advantage is caused by the 3D representation of the LiDAR point cloud. At the same time, this representation causes an increased inference time due to the additional dimension

6 Runtime measured on a Nvidia GeForce RTX 2080 Ti.

added to the network architecture[7], as shown in Table 5.2 where the inference time for the PiLaNet is by a factor of 1.9 longer than the LiLaNet based on the VLP-32C. Additionally, the voxel range of the PiLaNet is restricted compared to the LiLaNet, which predicts a semantic label for each point as mentioned in Section 5.4. Extending this range from 60.0 m to 120.0 m for the width and the length of the voxel space, the inference time increases further and reaches a factor of 6.2 compared to the LiLaNet based on the VLP-32C.

5.5 OUTCOME

This chapter focuses on CNN architecture for the fine-grained semantic labeling of LiDAR point clouds based on a pillar-like voxel representation. The proposed architecture is designed for portability across different LiDAR sensor types to successfully handle varying spatial resolution and scanning patterns. The network architecture is evaluated against the LiLaNet representing a state-of-the-art semantic labeling approach based on a cylindrical projection of LiDAR data. The evaluation on manually annotated data across different sensors shows that the proposed architecture is indeed highly portable between sensors, yielding an improvement of 10 percentage points in mIoU when compared to the LiLaNet. However, the employed voxel representation leads to an increase in computational complexity, resulting in significantly longer inference times.

Furthermore, the presented architecture can be fully transferred across different sensor types with minimal adaptation effort by fine-tuning the pre-trained network on a small target sensor dataset. This represents a significant advantage given the fast-paced development of LiDAR hardware technology. The results indicate that the proposed network architecture can provide an efficient way for the automated generation of large-scale training data for novel LiDAR sensor types without the need for a multi-modal sensor setup. Hence, it might complement or even replace the multi-modal *Autolabeling* method as shown in Figure 5.6 to train real-time capable LiDAR-based semantic labeling approaches such as the LiLaNet.

7 Even if only 2D convolution layers are executed, the PiLaNet operated within the 3D space.

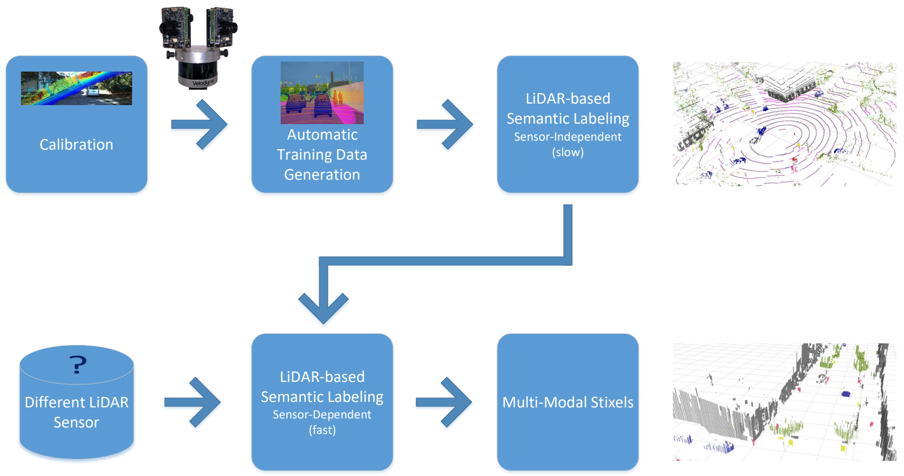

Calibration
Automatic Training Data Generation
LiDAR-based Semantic Labeling Sensor-Independent (slow)
?
Different LiDAR Sensor
LiDAR-based Semantic Labeling Sensor-Dependent (fast)
Multi-Modal Stixels

Figure 5.6: Extended research pipeline to minimize the effort for LiDAR-based semantic labeling of new LiDAR sensor models by using sensor independent network architectures and transferring the knowledge from one sensor model to another. (The images are extracted from the related publications [Schneider et al., 2017, Piewak et al., 2018a,b] © 2017 IEEE)

6 EXTENSION - HIERARCHICAL SEMANTIC LABELING

CONTENTS

Within this chapter, an extension of the third step of the research pipeline (see Figure 1.5) is presented. This extension focuses on a proof of concept for the combination of LiDAR-based semantic labeling with hierarchical labels. Parts of this section have previously appeared in [Bozatzidou, 2019] and [Piewak et al., 2020c].

6.1 OVERVIEW

As already mentioned in Section 1.1.3, semantic labeling represents the foundation for an in-depth scene understanding within the context of autonomous vehicles. High-performance semantic labeling is needed to maximize overall system performance, availability, and safety. For this reason, different datasets are available [Cordts et al., 2016, Behley et al., 2019] or created (see Chapter 3) in road traffic scenarios. The achieved results already reach a favorable classification performance e. g. within the camera domain [Cordts et al., 2016] as well as within the LiDAR domain (see Chapter 4). However, there is still room for improvement.

Taking a closer look at the different failure cases of the proposed semantic labeling approach of Chapter 4, the confusion matrix (see Table 6.1) presents two types of misclassification. First, misclassifications for the predicted classes 'construction' and 'vegetation' can be recognized (slightly red vertical line within the confusion matrix). These two classes have a larger occurrence in the dataset and therefore create a bias within the prediction (see Figures 3.3 and 3.4). This bias can be reduced in data-driven fashion by adapting the dataset label distribution. Most of the remaining misclassifications

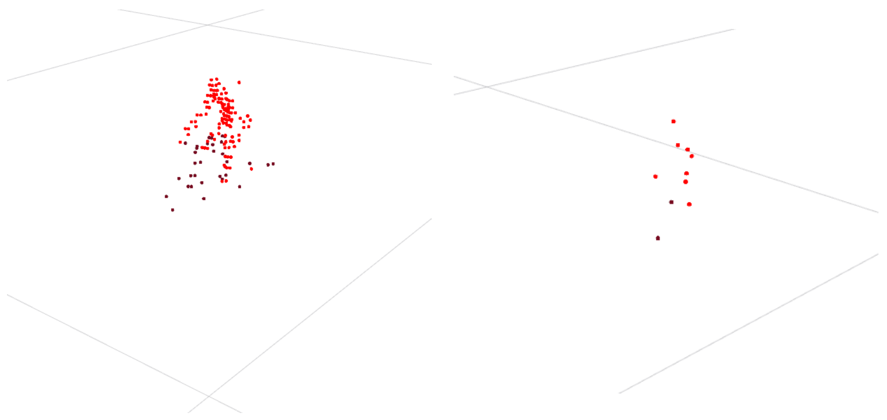


Figure 6.1: Example of point cloud of a 'bicycle' with a 'rider' in 13 meters (left) and 60 meters (right) distance to the LiDAR sensor. In 13 meters distance, the difference between the 'rider' and the 'bicycle' can clearly be seen based on the 3D shape of the point cloud. In 60 meters, the differentiation is challenging resulting in a higher confusion of both classes. The point cloud is captured based on a VLP-32C.

are plausible due to sensor-specific constraints. For example, the class 'two-wheeler' is confused with the class 'rider' due to the high uncertainty of the object borders within the sparse LiDAR point cloud, especially at large distances between the obstacle and the sensor (see Figure 6.1). Also, the class 'small vehicle' is confused with the class 'large vehicle', which is obvious due to unclear definitions of mid-sized vehicles.

The obvious and plausible confusions within the semantic labeling are caused by the equal weighting of the label classes based on the learning rule (see Section 2.2.4). Therefore, the CNN is neither able to distinguish more plausible confusions (like 'person' and 'rider') from implausible confusions (like 'person' and 'road'), nor predict a fallback class in case of high uncertainty (like 'vehicle' instead of 'small vehicle' or 'large vehicle').

To prevent these kinds of misclassification, a hierarchical semantic labeling approach is introduced within this section as a proof of concept by adapting the learning rule of CNNs based on a label class hierarchy.

6.2 RELATED WORK

Related to the topic of managing confusions within classification tasks of CNNs, two different types of approaches can be found within the literature.

The first type of approach models the uncertainty of the CNN prediction explicitly in a probability estimate which can be employed in higher-level modules of autonomous driving platforms

		Prediction											
		road	sidewalk	construction	pole	traffic sign	vegetation	small vehicle	large vehicle	person	rider	two-wheeler	terrain
True Label	road	96	2	0	0	0	0	0	0	0	0	0	0
	sidewalk	7	86	1	0	0	0	0	0	0	0	0	3
	construction	0	0	92	0	0	3	0	0	0	0	0	0
	pole	0	0	16	65	2	10	1	0	0	0	0	2
	traffic sign	0	0	7	2	87	1	0	0	0	0	0	0
	vegetation	0	0	3	0	0	89	0	0	0	0	0	5
	small vehicle	0	0	2	0	0	0	91	3	0	0	0	0
	large vehicle	0	0	9	0	0	0	22	63	2	0	0	0
	person	0	0	3	0	0	1	0	0	91	1	0	0
	rider	0	0	2	0	0	1	3	0	21	57	12	0
	two-wheeler	1	2	8	0	0	2	8	0	4	7	62	0
	terrain	2	2	1	0	0	10	0	0	0	0	0	82

Table 6.1: Confusion matrix of the LiLaNet trained and evaluated on the VLP-32C dataset. The values are rounded to integer percentage.

(see Figure 1.3). Guo et al. [2017] therefore analyzed different training techniques that influence the probability estimate extracted from the softmax layer of the CNN. They proposed a so-called temperature scaling to adjust the probability estimate. For this purpose, a logistic regression model is trained on the validation set of a dataset to return probabilities based on the non-probabilistic predictions of a CNN. Gal and Ghahramani [2016] introduced the dropout at inference time as a Bayesian approximation for probability estimates. Therefore, the CNN is executed multiple times by randomly deactivating neurons. Although these approaches increase the quality of the probability estimates, they require a completeness of the dataset or increase the inference time which is difficult to realize within real-time environments (e. g. autonomous driving).

The second type of approach introduces a label hierarchy (see Figure 6.2) into the training process to formulate the label uncertainty. Three different types of hierarchical classification exist as defined by Silla and Freitas [2011] (see Figure 6.3).

The common way of classification is the usage of a flat hierarchy which represents the classification of each leaf of the label hierarchy without introducing the knowledge of the hierarchy into the training process. This kind of classification is typically utilized for semantic labeling within the camera [Cordts et al., 2016] or the LiDAR domain (see Chapter 4). Nevertheless, a label hierarchy is provided by grouping the label classes (e. g. 'vegetation' and 'terrain' belongs to 'nature' [Cordts et al., 2016]).

The second type of hierarchical classification is represented by a local classification, whereby for each level of the label hierarchy or for each parent of the class hierarchy, a separate classifier is executed. Mo et al. [2019] hence introduced a top-down architecture which represents a classifier per parent label class of the hierarchy for a semantic segmentation of dense point clouds of indoor environments.

The third hierarchical classification is represented by a global classification which takes the full label hierarchy into account. Here, leaves of the label hierarchy are predicted at a low uncertainty and parent label classes at a high uncertainty. Redmon and Farhadi [2017] therefore introduced a combination of global classification at training time and local classification at inference time by replacing the overall softmax layer at the end of the CNN with multiple softmax layers per parent class extracting probability estimates per parent class. At inference time, the label hierarchy is descended until a threshold of the probability estimate is reached. Weber et al. [2018] adopted this approach for hierarchical traffic light detection in terms of classifying the state and the direction of a traffic light. Lippe [2018] extended the approach of Redmon and Farhadi [2017] to handle rare classes for image-based semantic labeling in combination with multiple datasets of different label hierarchies. Although these approaches reach favorable results by using label hierarchies, they lead to similar problems as the above-mentioned approaches by generating a meaningful probability estimate. Additionally, the chosen threshold for descending the label hierarchy is modeled by a manually adjusted parameter.

In contrast, Nourani-Vatani et al. [2015] introduced a fully global classification approach of label hierarchy based on a Support Vector Machine for seafloor image classification. They therefore adapted the true labels to represent the class hierarchy.

Within this section, the concept of Nourani-Vatani et al. [2015] is taken one step further to a fully global classification approach of label hierarchy by preventing an explicit formulation of the probability estimate and learning the inference of the label hierarchy implicitly with the knowledge of the CNN based on an adaption of the cross-entropy function. This has the advantage of an implicit uncertainty representation of a CNN within the inferred classification without the requirements of an explicit probability estimate. As a result, the ascending or descending of the label hierarchy is performed implicitly by the inference of the CNN.

6.3 METHOD

This section introduces the hierarchical semantic labeling. Hence, a label hierarchy as well as the adaption of the cross-entropy function is described.

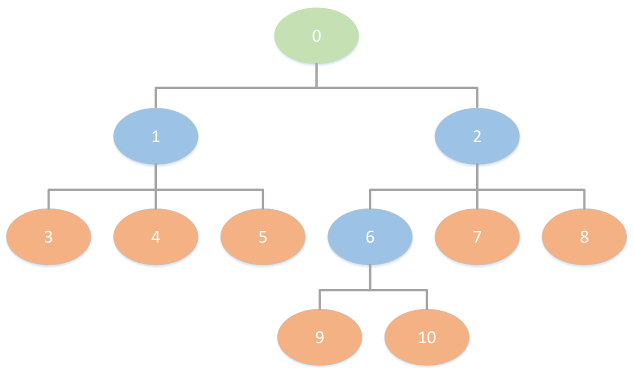


Figure 6.2: Example of a label hierarchy represented by a tree. Leaf nodes are visualized in orange while parent nodes are visualized in blue. The green node represents the root of the class hierarchy.

6.3.1 *Class Taxonomy*

Formulating the relations between label classes, the label hierarchy is defined as a tree as shown in Figure 6.2. The tree consists of nodes which represent the label classes (e. g. class '1' and class '3'). Each node is related at most to one parent class like

$$parent(4) = 1 \tag{6.1}$$

and can be related to child classes like

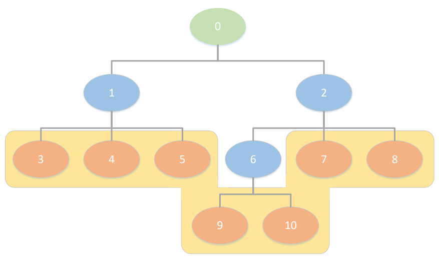
$$children(6) = \{9, 10\} \ . \tag{6.2}$$

Only one node exists within the tree that does not relate to a parent class which represents the root node (class '0'). Note that this class represents the overall fallback class within the context of semantic labeling. Nodes that do not relate to child classes are called leaves of the label hierarchy (e. g. class '4' or '7'). These classes represent the most specific classes for semantic labeling. Nodes between the root node and the leaf nodes (here, class '1', '2', and '6') represent abstract label classes or categories, which combine classes together (e. g. 'Golden Retriever' and 'Border Terrier' both correspond to the class 'Dog').

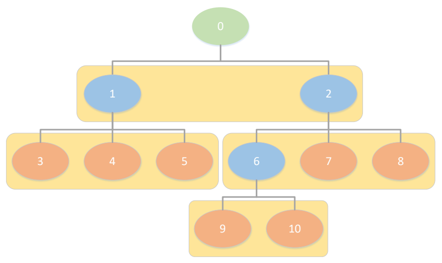
Additionally, nodes are related to their *ancestor*, representing all parent nodes until reaching the root node like

$$\begin{aligned} ancestor(6) &= \{0, 2\} \ , \\ ancestor(4) &= \{0, 1\} \ , \text{or} \\ ancestor(0) &= \{\} \ . \end{aligned} \tag{6.3}$$

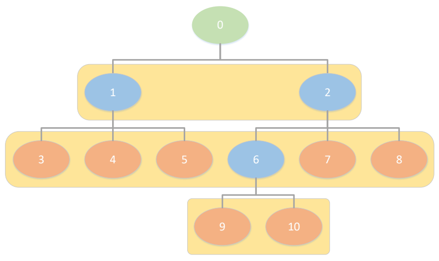
Based on these definitions, the *path* between two nodes is represented by all nodes lying on the shortest path between both nodes,



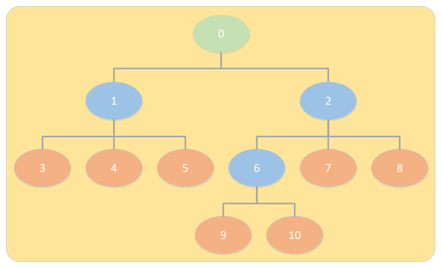
(a) Flat classification



(b) Local classification per parent



(c) Local classification per hierarchy level



(d) Global classification

Figure 6.3: Illustration of different hierarchical classification types as defined by Silla and Freitas [2011] based on the label hierarchy of Figure 6.2. Each yellow box represents a classifier. (a) has one single classifier for all leaf class of the hierarchy. Parent classes cannot be predicted. (b) has four classifiers, one per parent class. (c) has three classifiers, one per hierarchy level. (d) has one classifier to predict all classes of the label hierarchy.

including all parents until the first common *ancestor* of both nodes and the nodes itself like

$$\begin{aligned} path(5,7) &= path(7,5) = \{0,1,2,5,7\} \ , \\ path(9,7) &= path(7,9) = \{2,6,7,9\} \ , \text{or} \\ path(6,6) &= \{6\} \ . \end{aligned} \tag{6.4}$$

The number of elements within a *path* of two classes is known as the distance of these classes.

Furthermore, each mode corresponds to a hierarchy level as shown in Figure 6.4 representing the abstraction level of the label hierarchy.

6.3.2 *Hierarchical Learning Rule*

Assuming the definition of label hierarchy of Section 6.3.1, the hard constraint of the learning rule that only one class represents the true class (see Section 2.2.4) is softened, similar to [Nourani-Vatani et al., 2015]. Therefore, Equation (2.10) is adapted in different ways to adapt the weighting of the class hierarchy within the cross-entropy function. Four different weighting functions are presented within this chapter.

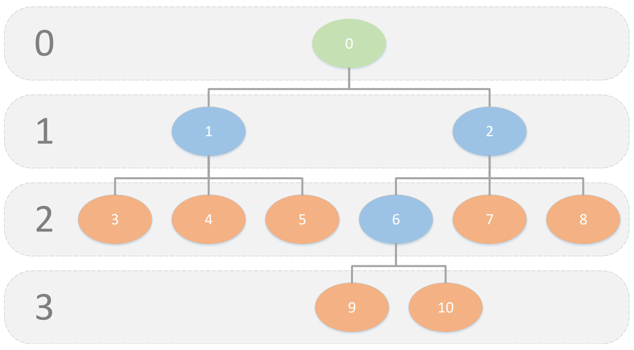


Figure 6.4: Illustration of hierarchy levels (gray) of a label hierarchy. Level 0 represents the highest and most abstract hierarchy level while level 3 represents the lowest and most specific hierarchy level containing only leaf classes.

First, the hierarchy can be inserted by allowing classes corresponding to the *ancestor* of the desired class ψ_o compared to Equation (2.10) with

$$\Lambda_{\psi_o,k} = \begin{cases} 1, & \text{if } k \in ancestor(\psi_o) \text{ or } k = \psi_o \\ 0, & \text{otherwise} \end{cases} . \tag{6.5}$$

This represents the base idea of a label hierarchy which allows fallback classes corresponding to the *ancestor*.

Similarly, this definition can be extended to the *path* between the desired class ψ_o and the predicted class ϕ_o to insert the full knowledge of the label hierarchy with

$$\Lambda_{\phi_o,\psi_o,k} = \begin{cases} 1, & \text{if } k \in path(\psi_o, \phi_o) \\ 0, & \text{otherwise} \end{cases} , \tag{6.6}$$

where ϕ_o is represented by the argmax of the outputs $\boldsymbol{y}_o$ with

$$\phi_o = \underset{i}{(\text{argmax})}\, y_{o,i} \ . \tag{6.7}$$

In addition, to the definition in Equations (6.5) and (6.6), the classes can be weighted to the distance to the desired class ψ_o similar to [Nourani-Vatani et al., 2015] to insert a weighting towards the leaf class with

$$d\Lambda_{\psi_o,k} = \begin{cases} \frac{1}{|path(\psi_o,k)|}, & \text{if } k \in ancestor(\psi_o) \text{ or } k = \psi_o \\ 0, & \text{otherwise} \end{cases} , \tag{6.8}$$

and

$$d\Lambda_{\phi_o,\psi_o,k} = \begin{cases} \frac{1}{|path(\psi_o,k)|}, & \text{if } k \in path(\psi_o, \phi_o) \\ 0, & \text{otherwise} \end{cases} . \tag{6.9}$$

For preventing the scaling of the loss based on the hierarchy, a softmax function Φ (see Equation (2.6)) is applied to the class weights. As a result, the Equation (2.10) is adapted e. g. with $\Lambda_{\psi_o,k}$ to

$$\rho_{\psi_o,k} = \Phi(\mathbf{\Lambda}_{\psi_o})_k = \frac{e^{\Lambda_{\psi_o,k}}}{\sum_{j=0}^{M} e^{\Lambda_{\psi_o,j}}} \quad , \tag{6.10}$$

while $\mathbf{\Lambda}_{\psi_o} = (\Lambda_{\psi_o,0}, \ldots, \Lambda_{\psi_o,M-1})$ represents the weight vector for all predicted classes. For Equations (6.6), (6.8), and (6.9), the Equation (2.10) is adapted analogously.

6.4 EVALUATION

Within this section, the proposed fully global hierarchical semantic labeling approach is evaluated. The evaluation is performed on the autolabeled frames of the optimized validation set of the VLP-32C dataset (see Section 3.4.1) instead of the testing set due to the parameter optimization of the learning strategy. Adapting the learning rule according to Section 6.3.2 with the four proposed class weightings, the Slim LiLaNet (see Section 4.3.3) is used as a reference CNN.

The network training is performed via the Adam solver [Kingma and Ba, 2015]. The suggested default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ are used. The learning rate is fixed at $\eta = 10^{-3}$ and the batch size is set to $\beta_{batch} = 10$, while the weights are initialized with Xavier [Glorot and Bengio, 2010]. The training is performed on the autolabeled frames of the training set of the VLP-32C dataset, and no fine-tuning on the manually annotated frames is performed.

Given the available classes for LiDAR-based semantic labeling (see Section 3.3.3) and inspired by Lippe [2018], a label hierarchy is defined as shown in Figure 6.5. It contains 21 classes, including the 13 leaf classes of the generated or manually annotated datasets (see Section 3.4.1). Note that the 8 parent classes are not present within the dataset.

The performance is evaluated with two steps. First, the performance of a hierarchical training is evaluated based on the mIoU metric (see Section 2.3.4) to be compared to non-hierarchical semantic labeling approaches. Afterwards, a qualitative evaluation of the different weighting approaches is depicted. Note that the uncertainty cannot be evaluated directly due to the implicit formulation within the CNN instead of an explicit formulation at inference time.

6.4.1 *Non-Hierarchical Comparison*

According to Section 3.4.2, the mIoU metric is evaluated for different hierarchy levels of the label hierarchy (see Figure 6.5) while project-

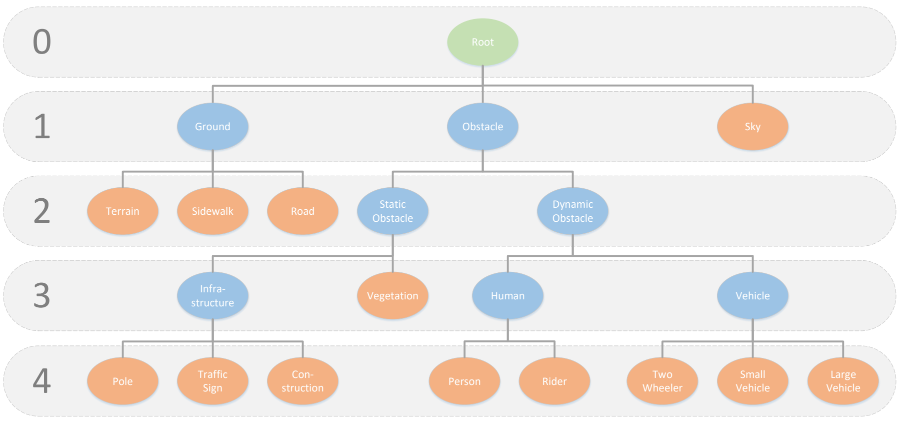


Figure 6.5: Label hierarchy applied to the Slim LiLaNet. The leaf classes represent the 13 classes of *Autolabeling* process (see Section 3.3.3). The different hierarchy levels are visualized in gray. The label hierarchy is based on [Bozatzidou, 2019].

ing leaf nodes of higher hierarchy levels to lower hierarchy levels[1]. The results can be seen in Table 6.2, where the flat classification represents a semantic labeling of the leaf classes, while the hierarchical classification takes the full label hierarchy into account. Note that the mIoU metric is not intended to handle a label hierarchy. However, the mIoU is used to compare the proposed hierarchical approach to non-hierarchical semantic labeling approaches. The result is the prediction of a parent class at a high uncertainty of the leaf classes representing a FN in terms of IoU of the leaf classes. For this reason, the mIoU of the hierarchical classification is lower than the flat classification for the hierarchy level 2 and 3. The hierarchy level 0 represents only one class (root class), which produces the mIoU score at 100%. Nevertheless, the hierarchy level 1 in Table 6.2 presents the potential of hierarchical classification by outperforming the flat classification due to an implicit formulation of the class hierarchy within the learning rule. As a result, a high uncertainty at the leaf classes can be reduced at a higher label class hierarchy level.

6.4.2 *Qualitative Results*

Due to a restricted evaluation capability of the IoU regarding a hierarchical semantic segmentation with a validation set containing only leaf classes of the label hierarchy, this section presents qualitative results of the four weighting strategies of Section 6.3. Therefore, a representative frame is visualized for all four strategies (see Figures 6.6 to 6.9).

First, the equal weighting of all *ancestor* classes related to the true class (strategy $\Lambda_{\psi_o,k}$) is visualized in Figure 6.6. There, it is

1 The class 'sky' is a leaf node at the hierarchy level 1. It is therefore used for the evaluation of hierarchy levels 1, 2, 3, and 4.

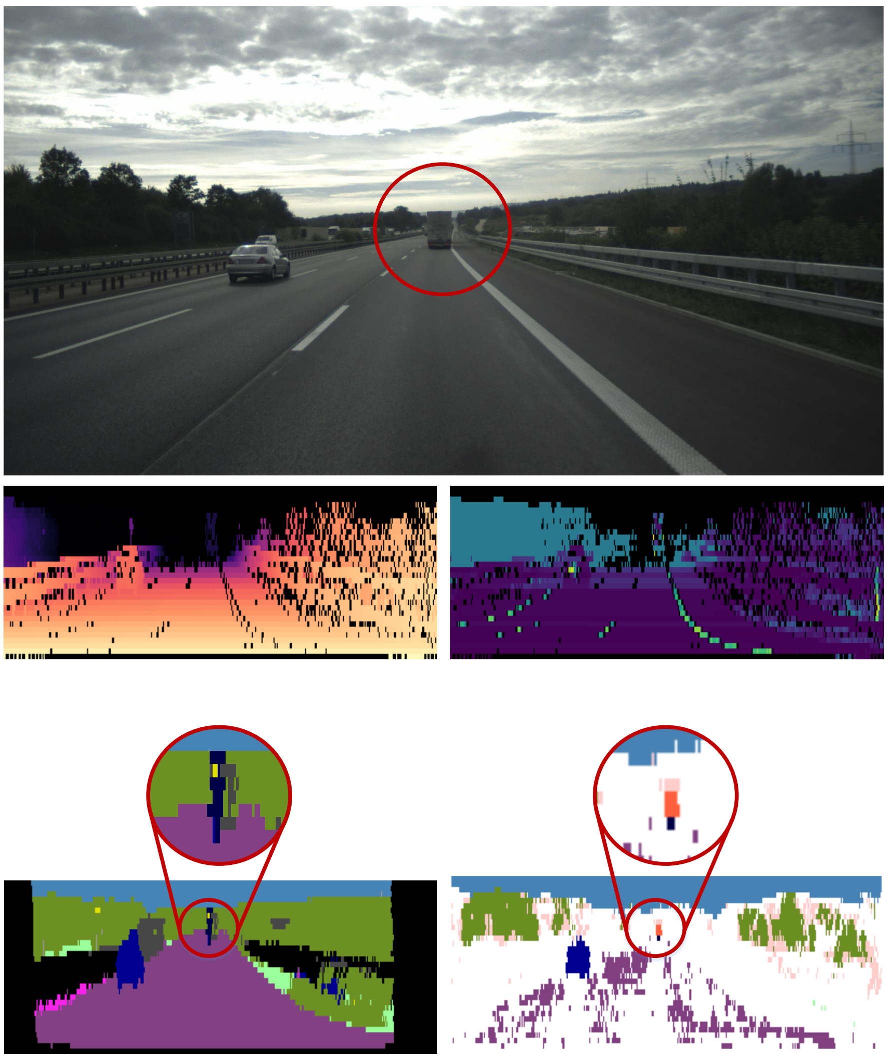


Figure 6.6: Visualization of an evaluation sample for the hierarchical weighting with $\Lambda_{\psi_o,k}$. On the top the camera image of the evaluation sample is visualized, middle left the distance LiDAR image is visualized (yellow = near, violet = far, black = invalid), middle right the reflectivity LiDAR images is shown (yellow = high reflectivity, blue = low reflectivity, black = invalid), bottom left represents the true labels generated by the *Autolabeling* technique, and bottom right shows the prediction of the hierarchical training with a weighting of $\Lambda_{\psi_o,k}$. The zoomed part within the LiDAR image visualizes the leading 'large vehicle' as shown within the camera image. The following semantic classes are visualized: road, sidewalk, small vehicle, large vehicle, construction, traffic sign, vegetation, terrain, sky, obstacle, dynamic obstacle, root, unlabeled. Note that the LiDAR images are cropped to the front facing part of the point cloud. The images are based on [Bozatzidou, 2019].

Hierarchy Level	mIoU Performance	
	Flat Classification	Hierarchical Classification
0	100.0%	100.0%
1	89.2%	91.5%
2	74.8%	52.5%
3	70.5%	44.0%

Table 6.2: Evaluation of the mIoU for hierarchical semantic labeling based on the upper four different hierarchy levels. The flat classification represents the Slim LiLaNet of Section 4.3.3, while the hierarchical classification is defined by the hierarchical training of the Slim LiLaNet. Note that leaf classes are projected to lower hierarchy levels for evaluation (e. g. class 'sky' is used for evaluation of all hierarchy levels besides level 0). The table is based on [Bozatzidou, 2019].

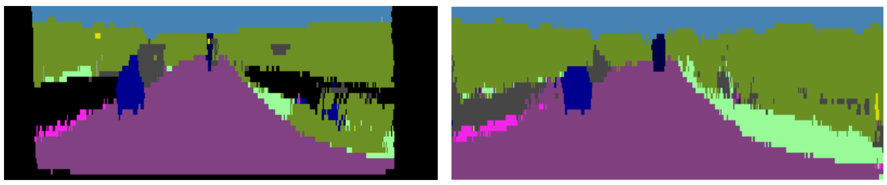


Figure 6.7: Visualization of an evaluation sample for the hierarchical weighting with $\Lambda_{\phi_o,\psi_o,k}$. The left image represents the true labels generated by the *Autolabeling* technique and the right image shows the prediction of the hierarchical training with a weighting of $\Lambda_{\phi_o,\psi_o,k}$. Note that the images are cropped to the front facing part of the point cloud. The following semantic classes are visualized: road, sidewalk, small vehicle, large vehicle, construction, traffic sign, vegetation, terrain, sky, unlabeled. The images are based on [Bozatzidou, 2019].

clearly visible that the Slim LiLaNet takes the label hierarchy into account to ascend the label hierarchy. The main parts of the predicted image are labeled as root class, which is obvious due to the same weighting of the leaf class and the root class, while the root class is a valid classification for all LiDAR points. Taking a closer look at the remaining other class predictions, it is apparent that the CNN implicitly learns the label hierarchy. For example, the leading 'large vehicle' is correctly classified as related classes like 'dynamic obstacle' and 'obstacle'. However, this kind of semantic labeling is unusable within the context of robotics and autonomous driving due to the high rate of root class predictions representing the overall fallback class.

Second, the equal weighting of all *path* classes between the true and the predicted class (strategy $\Lambda_{\phi_o,\psi_o,k}$) is visualized in Figure 6.7. There, the CNN does not learn the label hierarchy and predicts only leaf classes. This is related to the hierarchical learning rule, whereby the minimal value of the loss function is achieved by predicting the outputs $\boldsymbol{y}_o$ with a similar distribution as the weight vector $\boldsymbol{\Lambda}$ of the

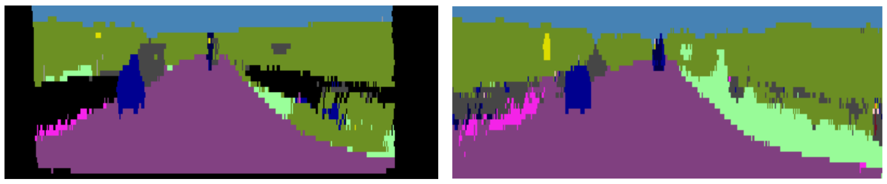


Figure 6.8: Visualization of an evaluation sample for the hierarchical weighting with $d\Lambda_{\phi_o,\psi_o,k}$. The images are arranged identical as Figure 6.7 including the class labels and are based on [Bozatzidou, 2019].

hierarchical learning rule[2]. At a prediction of an equal distribution, this initially leads to a selection of the argmax ϕ_o similar to the above-mentioned strategy. As a result, one of the *path* classes (e. g. $path$(‘terrain’, ‘root’) = ‘terrain’, ‘ground’, ‘root’) is inferred. In case of inferring the class with the largest hierarchical distance to the true class (e. g. ‘root’), the weight vector $\mathbf{\Lambda}$ stays unchanged. In case of inferring another *path* class (e. g. ‘ground’), the weight vector $\mathbf{\Lambda}$ is changed to an equal weighting of the *path* classes between the true class and the inferred class due to the dependence of the weighting $\Lambda_{\phi_o,\psi_o,k}$ to the argmax ϕ_o. This results in the predicted distribution being changed based on the learning process. Thereby, the classes with the originally longer hierarchical distance to the true class will not be predicted anymore. Caused by the fact that the dataset only contains leaf classes, this effect optimizes the weight vector $\mathbf{\Lambda}$ and corresponding the CNN towards leaf class predictions, which can be seen in Figure 6.7.

In the case of the distance-dependent weighting strategy of all *path* classes (strategy $d\Lambda_{\phi_o,\psi_o,k}$), this effect is even accelerated due to the fact that the maximal value of the weight vector $\mathbf{\Lambda}$ is equal to the true class, which corresponds to a leaf class within the given dataset. This is visualized in Figure 6.8.

Finally, the distance-dependent weighting of all *ancestor* classes related to the true class (strategy $d\Lambda_{\psi_o,k}$) is visualized in Figure 6.9. There, the weight vector $\mathbf{\Lambda}$ is not dependent on the argmax ϕ_o as the weighting related to the *path* classes (strategy $\Lambda_{\phi_o,\psi_o,k}$ and $d\Lambda_{\phi_o,\psi_o,k}$). As a result, the full hierarchical class tree can be predicted. Additionally, an improvement to the equal weighting (strategy $\Lambda_{\psi_o,k}$) is clearly visible at the reduced number of root class predictions in Figure 6.9. At the same time, the hierarchical knowledge is still present as visible at the leading ‘large vehicle’. Furthermore, it can be observed that the parent classes including the root class primarily appear at object borders where a higher uncertainty of the CNN is expected.

2 The weight vector $\mathbf{\Lambda}$ depends on the weighting strategy and is represented by $\mathbf{\Lambda}_{\phi_o,\psi_o}$ for the equal weighting of all *path* classes or by $\mathbf{\Lambda}_{\psi_o}$ for the equal weighting of all *ancestor* classes.

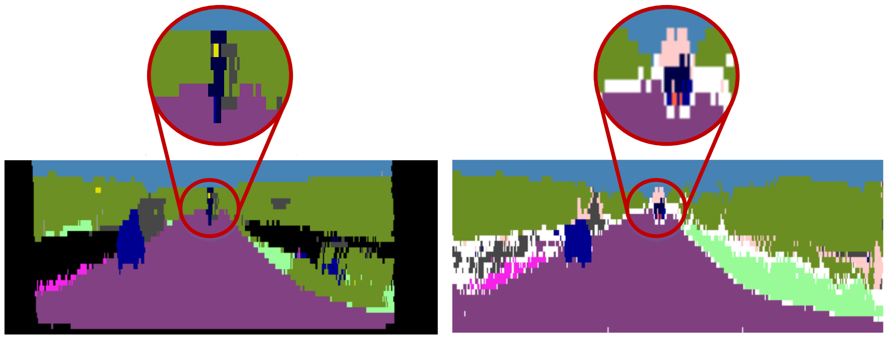


Figure 6.9: Visualization of an evaluation sample for the hierarchical weighting with $d\Lambda_{\psi_o,k}$. The images are arranged identical as Figure 6.7 including the class labels. The following additional classes that are visualized: obstacle, dynamic obstacle, root. The zoomed area corresponds to the leading 'large vehicle' as visualized in Figure 6.6. The images are based on [Bozatzidou, 2019].

6.5 OUTCOME

Within this chapter, a hierarchical semantic labeling approach based on an extension of the cross-entropy function is introduced. Therefore, a label hierarchy is defined as a tree structure which represents the relations between the different classes while grouping the classes together to more abstract classes. Compared to related work, the proposed hierarchical semantic labeling approach does not require an explicit formulation of the uncertainty or the probability estimate. It introduces the knowledge of the label hierarchy directly into the CNN, while the uncertainty is learned implicitly. As a result, the CNN is capable to ascend or descend the label hierarchy implicitly at inference time and can even predict the root class as the overall fallback class.

The results present the potential of this hierarchical approach which is able to reduce confusions between leaf classes by predicting parent classes of the label hierarchy. As a result, semantic labeling approaches can be taken one step further by implicitly encoding the uncertainty into the CNNs.

7 MULTI-MODAL STIXELS

CONTENTS

Within this chapter, the last step of the research pipeline (see Figure 1.5) is described. Parts of this chapter have previously appeared in [Piewak et al., 2018a], [Piewak et al., 2020a], and [Piewak et al., 2020b].

7.1 OVERVIEW

After generating a valuable LiDAR-based semantic labeling approach in Chapter 4, the last step of the research pipeline (see Figure 1.5) represented by the multi-modal Stixels can be approached.

Within the field of mobile robotics and autonomous driving, stringent requirements regarding accuracy, availability, and safety have led to the use of sensor suites that incorporate complimentary sensor types such as camera, LiDAR sensor, and RaDAR sensor. Each sensor modality needs to leverage its specific strengths to contribute to a holistic picture of the environment (see Section 1.1.4).

The sensor output typically involves quantities that are derived from raw measurements, such as detailed semantics (see Section 4.2) or object instance knowledge [Yang et al., 2016, Zhou and Tuzel, 2018]. The different representations provided by the various sensor types are typically fused into an integrated environment model as discussed in Section 1.1.2.

Fusing the massive amounts of data provided by multiple different sensors represents a significant challenge in a real-time application. For this reason, mid-level data representations have been proposed that reduce the amount of sensor data but retain the underlying information at the same time. A prime example of such a mid-level representation is the so-called Stixel-World [Badino et al., 2009, Pfeiffer, 2012, Schneider et al., 2016, Cordts et al., 2017, Hernandez-Juarez

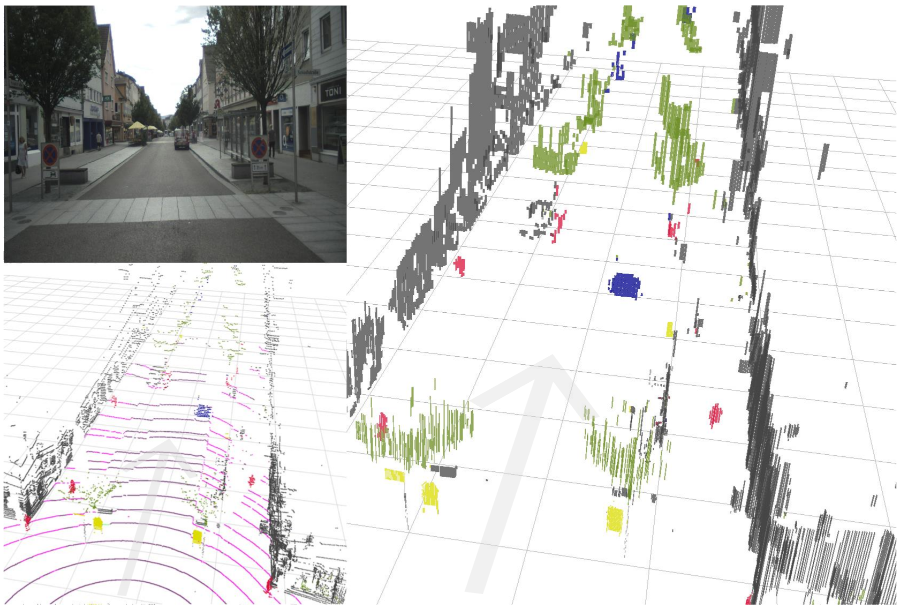


Figure 7.1: Example of a multi-modal Stixel scene visualized as 3D image (right) generated based on a camera image (top left) and a semantic point cloud (bottom left). Note that only *object* Stixels are visualized. The colors correspond to the Cityscapes semantic class color coding [Cordts et al., 2016]. The following semantic classes are visualized: road, sidewalk, person, rider, small vehicle, large vehicle, two-wheeler, construction, pole, traffic sign, vegetation, terrain. The images are based on [Piewak et al., 2018a].

et al., 2017] that provides a compact, yet geometrically and semantically consistent model of the observed environment. Thereby, a 3D scene is represented by a set of narrow vertical segments, the Stixels, which are described individually by their vertical extent, geometric surface, and semantic label. The Stixel concept was originally applied to stereo camera data, whereby the segmentation is primarily based on dense disparity data as well as pixel-level semantics obtained from a deep neural network [Schneider et al., 2016, Cordts et al., 2017, Hernandez-Juarez et al., 2017].

In this chapter, a transfer of the Stixel concept into the LiDAR domain to develop a compact and robust mid-level representation for 3D point clouds is proposed. Moreover, the Stixel concept is extended to a multi-modal representation by incorporating both camera and LiDAR sensor data into the model.

7.2 RELATED WORK

The multi-modal Stixel approach presented in this chapter combines LiDAR distance measurements with the point-wise semantic labeling information obtained from both - the LiDAR sensor and monocular camera. The approach is related to three different categories of existing work: Semantic labeling, sensor fusion, and compact mid-level data representations.

First, semantic labeling describes a range of techniques for the assignment of object class or object type to each measurement (e.g. pixel-wise) as explained in Section 1.1.3. The topic has been well explored within the camera domain [Garcia-Garcia et al., 2018, Long et al., 2015, Cordts, 2017, Sankaranarayanan et al., 2018]. In contrast, semantic labeling for 3D point clouds is a relatively recent topic as described in Chapter 4. As the multi-modal Stixel approach proposed in this chapter utilizes semantics from both LiDAR sensor and camera data, the LiLaNet (see Section 4.3.2) is applied to directly extract the detailed point-wise semantics from LiDAR data. This results in a class representation similar to the camera domain, whereby the efficient FCN architecture represented by Cordts [2017] is used.

Second, different fusion strategies can be applied to the multi-modal data of various sensors. Several approaches perform so-called low-level fusion by directly combining the raw data to obtain a joint sensor representation, which is then used for object detection [Gupta et al., 2014] or semantic labeling [Muller and Behnke, 2014]. A different method commonly used within the autonomous driving context is high-level fusion [Nuss et al., 2018], whereby the sensor data is processed independently and the results are later combined on a more abstract level. In this chapter, a novel fusion concept is presented which integrates the sensor data on mid-level, reducing the data volume while minimizing information loss. This representation can further be integrated into a more abstract environment model such as an occupancy grid [Nuss et al., 2018].

Third, the presented multi-modal Stixel approach is closely related to other compact mid-level representations in terms of the output data format. In particular, the Stixel-World obtained from camera sensors is referred, which has successfully been applied with [Benenson et al., 2012, Pfeiffer, 2012, Liu et al., 2015] and without [Levi et al., 2015] the use of stereoscopic depth information. The integration of camera-based semantic labeling information into the Stixel generation process was presented in [Schneider et al., 2016, Cordts, 2017, Hernandez-Juarez et al., 2017], thereby further improving robustness and promoting the semantic consistency of the result. The Stixel concept has also been adapted to other image-based sensor techniques, e. g. to a camera-based infrared depth sensor as shown in [Martinez et al., 2017]. Forsberg [2018] made use of a LiDAR scanner to obtain depth information for the Stixel generation process. Similar to an early idea in [Pfeiffer, 2012], the LiDAR point cloud is simply projected into the camera image to replace the original dense disparity information with the sparse LiDAR-based depth measurements. In contrast within this chapter, a LiDAR-specific sensor model that

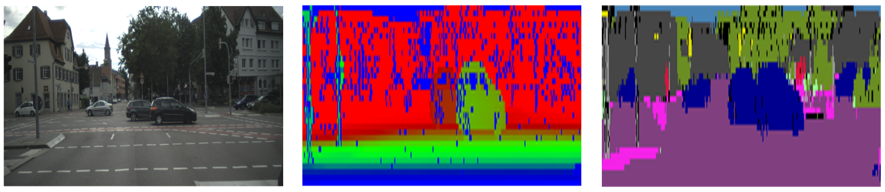


Figure 7.2: Example of a multi-modal Stixel scene visualized as 2D images (right, colors = Cityscapes semantic class [Cordts et al., 2016], each image column is separated into multiple semantic Stixels) and the corresponding LiDAR distance image (center, blue = close, red = far) projected to a cylindrical view. The corresponding camera image is shown on the left. The images are based on [Piewak et al., 2018a].

is particularly tailored to exploit the superior geometric accuracy of the LiDAR sensor over a stereo camera is introduced. Finally, semantics from both LiDAR and camera data are integrated into the Stixel generation process to obtain a high-quality, comprehensive mid-level 3D representation of the environment.

7.3 METHOD

The proposed Stixel model is inspired by the stereoscopic camera approaches of [Pfeiffer, 2012, Schneider et al., 2016, Cordts et al., 2017]. After a general definition of the Stixel representation, the transfer of the Stixel model to the LiDAR domain as well as the adapted Stixel generation process is described.

7.3.1 *Stixel Definition*

Stixels are segments which represent sensor data in a compact fashion while retaining the underlying semantic and geometric properties. Generally, the segmentation of an image represents a 2D optimization problem which is challenging to solve in a real-time environment. Instead, Stixels are optimized column-wise, which reduces the optimization task to a 1D problem that can be efficiently solved via dynamic programming [Pfeiffer, 2012]. As a result, each column is separated into rectangular stick-like segments $\boldsymbol{S}$ called Stixels. Within the LiDAR domain, the input data is represented as an ordered set of columns of the LiDAR scan, obtained from a cylindrical projection of the 3D measurements onto a 2D grid, as described in Section 4.3.1 and shown in Figure 7.2. Each Stixel $\boldsymbol{s}_i = (u, a, r, l, c)$ is represented by the bottom row index u and the top row index a, describing its vertical extent with regard to the vertically ordered measurements $\boldsymbol{M} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_h)$. Additionally, each Stixel has a semantic label l, a structural class c, and a distance r to the sensor or the ideal ground plane (depending on the structural

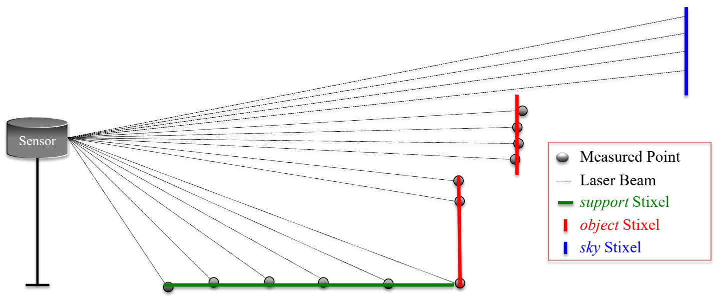


Figure 7.3: Exemplary Stixel extraction based on a vertical LiDAR scan column. The image is based on [Piewak et al., 2018a].

class c). There are three different Stixel structural classes, i.e. *support* ($\mathcal{G}$) for flat regions such as road surface or sidewalk, *object* ($\mathcal{O}$) for obstacles such as people or vehicles, and *sky* ($\mathcal{S}$) for areas without LiDAR measurements, as indicated in Figure 7.3.

7.3.2 *Stixel Model*

The vertically ordered (bottom to top) set of measurements $\boldsymbol{M} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_h)$ is processed column-wise (see Figure 7.2) and contains LiDAR depth measurements $\boldsymbol{D} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_h)$ as well as semantics from the camera $\boldsymbol{L}_{cam} = (\boldsymbol{l}_{cam_1}, \ldots, \boldsymbol{l}_{cam_h})$ and the LiDAR sensor $\boldsymbol{L}_{lidar} = (\boldsymbol{l}_{lidar_1}, \ldots, \boldsymbol{l}_{lidar_h})$, respectively. The extraction of semantics from the LiDAR sensor is performed using the LiLaNet architecture (see Section 4.3.2). The semantic information of the camera is associated with the 3D LiDAR points based on the *Autolabeling* process (see Chapter 3), which projects the LiDAR points into the image plane in order to associate the semantics provided by a state-of-the-art image-based FCN to each point.

Based on this definition, the posterior distribution $Pr(\boldsymbol{S}|\boldsymbol{M})$ of the Stixels $\boldsymbol{S}$ given the measurements $\boldsymbol{M}$ of a column is defined using the likelihood $Pr(\boldsymbol{M}|\boldsymbol{S})$ as well as the prior $Pr(\boldsymbol{S})$ as

$$Pr(\boldsymbol{S}|\boldsymbol{M}) = \frac{Pr(\boldsymbol{M}|\boldsymbol{S}) \cdot Pr(\boldsymbol{S})}{Pr(\boldsymbol{M})} \quad . \tag{7.1}$$

Here, the Stixels $\boldsymbol{S} = (s_1, \ldots, s_n)$ are vertically ordered in accordance with the measurement vector $\boldsymbol{M}$. Formulating the posterior distribution in the log-domain yields

$$Pr(\boldsymbol{S}|\boldsymbol{M}) = \mathrm{e}^{-\Pi(\boldsymbol{S},\boldsymbol{M})} \quad , \tag{7.2}$$

where $\Pi(\boldsymbol{S}, \boldsymbol{M})$ represents an energy function as defined by Cordts et al. [2017] as

$$\Pi(\boldsymbol{S}, \boldsymbol{M}) = \Theta(\boldsymbol{S}, \boldsymbol{M}) + \Omega(\boldsymbol{S}) - \log(Pr(\boldsymbol{M})) \quad . \tag{7.3}$$

Note that $\Theta(\boldsymbol{S}, \boldsymbol{M})$ represents the data likelihood, $\Omega(\boldsymbol{S})$ the segmentation prior, and $Pr(\boldsymbol{M})$ a normalization. In contrast to camera-based Stixel applications, as discussed in Section 7.2, the proposed approach of this thesis puts forward a LiDAR-specific sensor model to better integrate the accurate LiDAR geometry into the Stixel concept. This will be discussed within the next subsections.

7.3.2.1 *Prior*

The prior $\Omega(\boldsymbol{S})$ puts constraints on the Stixel model in terms of model complexity and segmentation consistency with

$$\Omega(\boldsymbol{S}) = \Omega_{mc}(\boldsymbol{S}) + \Omega_{sc}(\boldsymbol{S}) \quad . \tag{7.4}$$

The model complexity term $\Omega_{mc}(\boldsymbol{S})$ describes the trade-off between the compactness and the accuracy of the representation. The segmentation consistency $\Omega_{sc}(\boldsymbol{S})$ governs hard constraints on the Stixels concerning the relation of Stixels within a column. The formulation of these prior terms does not depend on the LiDAR measurements. As a result, the existing definitions of the camera domain are used. For further details, the reader is referred to [Cordts et al., 2017].

7.3.2.2 *Data Likelihood*

The data likelihood represents the matching quality of the measurements $\boldsymbol{M}$ to a given set of Stixels $\boldsymbol{S}$, considering three different data modalities: LiDAR geometry, LiDAR semantics, and camera semantics:

$$\begin{aligned} \Theta(\boldsymbol{S}, \boldsymbol{M}) = \sum_{\boldsymbol{s}_i \in \boldsymbol{S}} \sum_{\boldsymbol{m}_j \in \boldsymbol{M}_i^*} & \beta_{geo_{lidar}} \Theta_{geo}(\boldsymbol{s}_i, \boldsymbol{d}_j, \boldsymbol{d}_{j-1}) \\ & + \beta_{sem_{lidar}} \Theta_{sem_{lidar}}(\boldsymbol{s}_i, \boldsymbol{l}_{lidar_j}) \\ & + \beta_{sem_{cam}} \Theta_{sem_{cam}}(\boldsymbol{s}_i, \boldsymbol{l}_{cam_j}) \quad . \end{aligned} \tag{7.5}$$

Here, $\boldsymbol{M}_i^*$ represents a subset of the measurements $\boldsymbol{M}$ associated with a specific Stixel $\boldsymbol{s}_i$. The parameters $\beta_{geo_{lidar}}$, $\beta_{sem_{lidar}}$, and $\beta_{sem_{cam}}$ represent the weighting parameters of each modality. The different modalities are described within the following paragraphs.

LIDAR GEOMETRY The LiDAR geometry data likelihood consists of three elements defined as follows:

$$\begin{aligned} \Theta_{geo}(\boldsymbol{s}_i, \boldsymbol{d}_j, \boldsymbol{d}_{j-1}) = \Theta_{dist}(\boldsymbol{s}_i, \boldsymbol{d}_j) + \Theta_{gr}(\boldsymbol{s}_i, \boldsymbol{d}_j, \boldsymbol{d}_{j-1}) \\ + \Theta_{sens}(\boldsymbol{s}_i, \boldsymbol{d}_j) \quad . \end{aligned} \tag{7.6}$$

First, the relation of a LiDAR depth measurement $\boldsymbol{d}_j$ and the Stixel $\boldsymbol{s}_i$ is given by the term $\Theta_{dist}(\boldsymbol{s}_i, \boldsymbol{d}_j)$. This data likelihood is represented as a mixture of a normal distribution, encoding the sensor noise based on the variance σ, and a uniform distribution representing outlier measurements with an outlier rate of p_{out}. Note that this data likelihood is not LiDAR sensor specific, but rather specific to distance-measuring sensors except the parameters σ and p_{out}. Resulting is the definition of Cordts et al. [2017] applied.

In addition to the common depth likelihood definition $\Theta_{dist}(\boldsymbol{s}_i, \boldsymbol{d}_j)$, two additional likelihood terms are defined to take advantage of LiDAR-specific measurement properties: A ground term $\Theta_{gr}(\boldsymbol{s}_i, \boldsymbol{d}_j, \boldsymbol{d}_{j-1})$ and a sensor term $\Theta_{sens}(\boldsymbol{s}_i, \boldsymbol{d}_j)$. The ground term assesses the consistency of the data with an assumed ground model, based on the gradient between two measurements

$$\begin{aligned}\gamma(\boldsymbol{d}_j, \boldsymbol{d}_k) &= \arctan\left(\frac{\Delta p_{z_{jk}}}{d_{ground_{jk}}}\right) \\ &= \arctan\left(\frac{p_{z_k} - p_{z_j}}{\sqrt{p_{x_k}^2 + p_{y_k}^2} - \sqrt{p_{x_j}^2 + p_{y_j}^2}}\right) \quad . \end{aligned} \tag{7.7}$$

Note that a geometric LiDAR measurement $\boldsymbol{d}_j = (r_j, \alpha_{h_j}, \alpha_{v_j})$ is represented using polar coordinates and consisting of a measured distance r_j, a horizontal angle α_{h_j}, and a vertical angle α_{v_j}. Based on these polar coordinates, the Cartesian coordinates $(p_{x_j}, p_{y_j}, p_{z_j})$ are extracted.

The gradient γ obtained from the high-quality LiDAR measurements provides structural information of the environment to distinguish between flat surfaces such as ground (low gradient) and obstacles (high gradient). This information is encoded into an object existence probability using a parameterized hyperbolic tangent as

$$Pr_{ob}(\boldsymbol{d}_j, \boldsymbol{d}_{j-1}) = \frac{1 + \tanh(\beta_{gr,steep}(\gamma(\boldsymbol{d}_j, \boldsymbol{d}_{j-1}) - \beta_{gr,shift}))}{2} \quad . \tag{7.8}$$

Note that the parameters $\beta_{gr,steep}$ and $\beta_{gr,shift}$ adapt the sensitivity of the gradient model. Subsequently, the data likelihood based on the ground model is defined as

$$\Theta_{gr}(\boldsymbol{s}_i, \boldsymbol{d}_j, \boldsymbol{d}_{j-1}) = \begin{cases} -\log(1 - Pr_{ob}(\boldsymbol{d}_j, \boldsymbol{d}_{j-1})) & \text{if } \gamma \text{ is def. \& } c_i = \mathcal{G} \\ -\log(Pr_{ob}(\boldsymbol{d}_j, \boldsymbol{d}_{j-1})) & \text{if } \gamma \text{ is def. \& } c_i = \mathcal{O} \\ 0 & \text{if } \gamma \text{ is undef. or } c_i = \mathcal{S} \end{cases} \quad , \tag{7.9}$$

where c_i describes the structural class of the Stixel $\boldsymbol{s}_i$. Note that the data likelihood based on the ground model is set to zero when the gradient is undefined, which can be caused by missing reflections of the LiDAR sensor (e.g. if the LASER beam is pointing to the sky). However, both the vertical and horizontal angles of the polar coordinate of the so-called invalid measurement are still available.

In case of an invalid measurement, the data likelihood based on both the ground model and the depth matching cannot be processed. For this reason, the sensor term $\Theta_{sens}(\boldsymbol{s}_i, \boldsymbol{d}_j)$ is introduced to the likelihood formulation, which is based on the vertical distribution of measurement angles of the LiDAR sensor. It is assumed that a *sky* Stixel is more likely to occur at larger vertical angles, which is encoded into a parameterized hyperbolic tangent similar to Equation (7.8) as

$$Pr_{\mathcal{S}}(\alpha_{v_j}) = \frac{1 + \tanh(\beta_{sens,scale}(\alpha_{v_j} - \beta_{sens,shift}))}{2} \quad . \tag{7.10}$$

A similar definition is used with regard to small vertical angles and *support* Stixels by inverting the vertical angle $P_{\mathcal{G}}(\alpha_{v_j}) = P_{\mathcal{S}}(-\alpha_{v_j})$. Consequently, the sensor term contribution for invalid points is defined by

$$\Theta_{sens}(\boldsymbol{s}_i, \boldsymbol{d}_j) =
\begin{cases}
-\log(Pr_{\mathcal{S}}(\alpha_{v_j})) & \text{if } \boldsymbol{d}_j \text{ is invalid } \& \ c_i = \mathcal{S} \\
-\log(Pr_{\mathcal{G}}(\alpha_{v_j})) & \text{if } \boldsymbol{d}_j \text{ is invalid } \& \ c_i = \mathcal{G} \\
-\log(1 - Pr_{B}(\alpha_{v_j})) & \text{if } \boldsymbol{d}_j \text{ is invalid } \& \ c_i = \mathcal{O} \\
\infty & \text{if } \boldsymbol{d}_j \text{ is valid } \& \ c_i = \mathcal{S} \\
0 & \text{if } \boldsymbol{d}_j \text{ is valid } \& \ c_i \in \{\mathcal{G}, \mathcal{O}\}
\end{cases} \quad , \tag{7.11}$$

with $Pr_B(\alpha_{v_j}) = Pr_{\mathcal{S}}(\alpha_{v_j}) + Pr_{\mathcal{G}}(\alpha_{v_j})$. Note that a hard constraint is inserted to prohibit *sky* Stixels resulting from valid measurements.

SEMANTIC INFORMATION The semantic information obtained from the LiDAR data is utilized in a similar way as in the Stixel concept of the camera domain. Each semantic measurement $\boldsymbol{l}_{lidar_j}$ holds a probability estimate $Pr_{\boldsymbol{l}_{lidar_j}}(l)$ of each class l conditioned on the input data, which can be obtained from the underlying semantic labeling method represented by the LiLaNet (see Chapter 4). The definition of the semantic data likelihood is adapted from [Schneider et al., 2016] and [Cordts et al., 2017] as

$$\Theta_{sem_{lidar}}(\boldsymbol{s}_i, \boldsymbol{l}_{lidar_j}) = -\log(Pr_{\boldsymbol{l}_{lidar_j}}(l_i)) \quad . \tag{7.12}$$

To obtain high-resolution semantic information from the camera image, the efficient FCN architecture described by Cordts [2017] is used. Fusing this information into the proposed multi-modal Stixel approach enables the combination of high-resolution camera semantics with geometrically accurate information of the LiDAR sensor. For this purpose, the projection technique of the *Autolabeling* (see Chapter 3) is applied to extract the semantic information of the camera by projecting the LiDAR measurements into the semantically labeled image. Consider that this projection represents a dense measurement transfer due to the fact that each LiDAR measurement then holds additional semantic information from the camera domain $\boldsymbol{l}_{cam_j}$, which is processed similar to Equation (7.12) based on the probability $Pr_{\boldsymbol{l}_{cam_j}}(l)$ for each semantic class l with

$$\Theta_{sem_{cam}}(\boldsymbol{s}_i, \boldsymbol{l}_{cam_j}) = -\log(Pr_{\boldsymbol{l}_{cam_j}}(l_i)) \quad . \tag{7.13}$$

Note that this definition is independent of the LiDAR-based semantics, which enable the extraction of different domain-specific semantic classes l from the camera and LiDAR sensor. Especially the camera-based FCN [Cordts, 2017] extracts more semantic classes based on the higher resolution as well as the larger receptive field compared to the LiLaNet. Hence, the domain specific strengths of each sensor modality and the differing object appearance within the LiDAR sensor and the camera are combined to increase the semantic consistency of the multi-modal Stixel result.

7.3.3 *Stixel Generation*

Based on the proposed Stixel model, Stixels are generated by finding the maximum-a-posteriori solution of Equation (7.1). This is equal to the minimization of the energy function given in Equation (7.3). Note that the probability of the measurement $Pr(\boldsymbol{M})$ represents a scaling factor which is ignored within the optimization process. To solve this 1D column-wise optimization process, a dynamic programming approach is used as defined in the original Stixel formulation ([Pfeiffer, 2012] and [Cordts et al., 2017]).

7.4 EVALUATION

To evaluate the proposed multi-modal Stixel model, the manually annotated testing set of the VLP-32C dataset (see Section 3.4.1) is used. The dataset consists of manually annotated semantic LiDAR point clouds recorded from a vehicle in various traffic scenarios, and further includes corresponding image data captured by a front-facing monocular camera. This enables the usage of the LiDAR depth data, the LiDAR semantic data based on the LiLaNet as discussed in Chapter 4, and the camera semantic data transferred to the LiDAR with the *Autolabeling* process (see Chapter 3) as three input modalities for the multi-modal Stixel model. Furthermore, the dataset allows both a semantic evaluation of the proposed method based on the manually annotated semantic LiDAR data and a geo-

	Stereo Camera[1]	LiDAR Depth only	LiDAR Semantics only	Camera Semantics only	Multi-Modality
Outlier Rate in %	6.7	**0.62**	28.8	35.3	0.95
mIoU in %	66.5	61.8	70.0	60.8	**70.6**
Compression Rate in %	-	54.0	81.2	**85.3**	58.3

Table 7.1: Comparison of the original Stixel World (based on a stereoscopic camera), the different independent Stixel optimization modalities, and the combined multi-modal representation based on an equal weighting of the modalities ($\beta_{sem_{lidar}} = \beta_{geo_{lidar}} = \beta_{sem_{cam}} = 1$). The table is based on [Piewak et al., 2018a].

metric evaluation based on the LiDAR depth data. Due to the sensor configuration within the dataset, the evaluation is restricted to the area inside the field of view of the camera. Various performance metrics on a point-wise basis are evaluated to measure the geometric and semantic consistency as well as the compactness of the model:

1. Outlier Rate

   A distance deviation of the original LiDAR depth measurement to the associated Stixel relative to the original LiDAR depth measurement of more than 5% is declared as an outlier. Based on this formulation, the outlier rate is defined as the ratio of the amount of outliers to the number of total LiDAR points. Note that a small outlier rate represents a model with a high geometric consistency.

2. mIoU

   Based on the manually annotated semantic ground-truth, a mIoU of the Stixels to the ground-truth LiDAR points can be calculated as defined in Section 2.3. Note that a high mIoU represents a model with a high semantic consistency.

3. Compression Rate

   The data compression rate θ defines the ratio between the amount of stixels $num_{stixels}$ and the amount of original LiDAR points num_{points} via

$$\theta = 1 - \frac{num_{stixels}}{num_{points}} \quad . \tag{7.14}$$

   Note that a high compression rate represents a high compactness of the model.

The quantitative results are illustrated in Figure 7.4. First, the impact of the LiDAR semantic weight $\beta_{sem_{lidar}}$ is evaluated within

1 The results of the original Stixel-World (stereo camera) are added for comparison based on [Cordts et al., 2017]. No evaluation is done on the VLP-32C dataset.

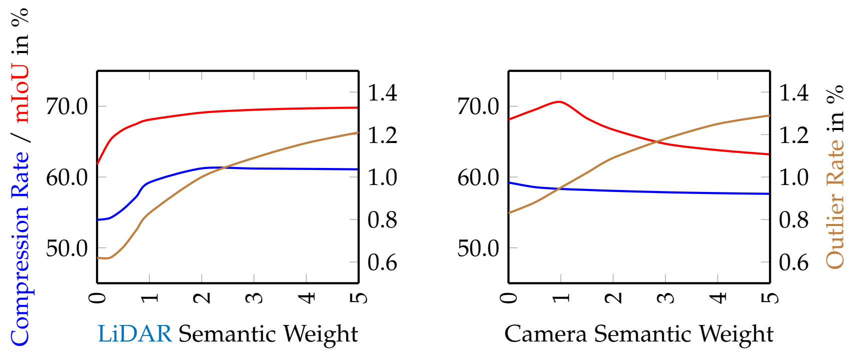


Figure 7.4: Impact of considering the semantic information within the multi-modal Stixel model with a constant LiDAR geometry weight ($\beta_{geo_{lidar}} = 1$). Left: Adaption of the LiDAR semantic weight $\beta_{sem_{lidar}}$ based on a deactivated camera semantics ($\beta_{sem_{cam}} = 0$). Right: Adaption of the camera semantic weight $\beta_{sem_{cam}}$ based on a LiDAR semantic weight $\beta_{sem_{lidar}} = 1$. Note that in both plots the left axis represents the mIoU (red) and the compression rate (blue) in %, while the right axis represents the outlier rate (brown) in %. The figure is based on [Piewak et al., 2018a].

the left plot of Figure 7.4, while the LiDAR geometry weight is set to $\beta_{geo_{lidar}} = 1$ and the camera semantics are deactivated ($\beta_{sem_{cam}} = 0$). It is observable that the semantic consistency is constantly increasing with an increase of the LiDAR semantic weight. At the same time, the compression rate raises as well as the outlier rate. Putting too much focus on the semantic input thus reduces the number of individual Stixels and yields a model purely tuned to the LiDAR semantics. In turn, consistency with the underlying geometry decreases.

Considering the multi-modality in the model by activating the camera semantics (right plot of Figure 7.4), the compression rate slightly decreases and the outlier rate further increases with an increase of the camera semantic weight. This represents a decrease concerning the geometric consistency as well as the data compression. The semantic consistency further improves until the weighting of the camera semantics $\beta_{sem_{cam}}$ reaches the weighting of the LiDAR semantics $\beta_{sem_{lidar}}$. However, the camera semantics on its own reaches a lower mIoU after the transfer to the LiDAR domain (see Table 7.1). This demonstrates the potential of the novel multi-modal Stixel approach, which creates a compact, geometrically and semantically consistent, mid-level representation by combining the advantages of different sensor domains to reach a higher accuracy than each modality on its own. The proposed method of equally weighting the different modalities represents the best combination with regard to the semantic consistency as well as a good compromise concerning the geometric consistency and the compression rate. This setup outperforms the original Stixel-World based on a stereoscopic camera regarding the geometric and the semantic consistency of the data representation (see Table 7.1) by combining the strength of each sensor modality. Note that the outlier rate of the LiDAR depth-only weighting and the multi-modality weighting is significantly smaller than the stereo camera approach due to the fact that the stereo camera generates distance estimates with a lower accuracy than a LiDAR sensor.

7.5 OUTCOME

In this chapter, the multi-modal Stixel-World is presented, a Stixel-based environment representation to directly leverage both camera and LiDAR sensor data. The design goal is to jointly represent accurate geometric and semantic information based on a multi-sensor system within a compact and efficient environment model. To this end, a LiDAR-specific sensor model is introduced that exploits the geometric accuracy of LiDAR sensors as well as a mid-level fusion technique to combine valuable semantic information from both camera and LiDAR sensor.

In the presented experiments, the benefits of the multi-modal Stixel-World over uni-modal representations in terms of representation and compression quality is demonstrated. The specific combination of the high resolution and semantic detail of camera data with the high distance accuracy of LiDAR data in the multi-modal Stixel-World results in a very powerful environment representation that outperforms the state-of-the-art. Moreover, the presented multi-modal Stixel approach can easily be extended to other sensor modalities as long as they can be projected densely into a commonly structured data format.

The result is a compact mid-level representation of LiDAR point clouds based on the multi-modal Stixel-World as presented as the last step of the research pipeline (see Figure 1.5). This representation reduces the amount of data per LiDAR point cloud by retaining the underlying geometric and semantic information of both camera and LiDAR sensor and increases the real-time capability of subsequent modules of autonomous driving platforms (see Figure 1.3).

8 CONCLUSION AND OUTLOOK

CONTENTS

Within this chapter, the main findings of this thesis are summarized and an outlook to future research is provided.

8.1 SUMMARY

In this thesis, an efficient research pipeline (see Figure 1.5) is proposed which describes the necessary modules to create a real-time capable LiDAR-based semantic scene understanding with high performance. Therefore, 4 steps are defined including a precise calibration of multi-modal sensor systems, a large-scale cross-modal training data generation, a high-quality semantic labeling of semi-dense LiDAR measurements, and a compact mid-level representation for semantic LiDAR measurements as well as a multi-modal mid-level fusion approach.

First, the precise calibration of a multi-modal sensor system represents the basis for further multi-modal approaches like a large-scale cross-modal training data generation or a multi-modal mid-level fusion. Different strategies are discussed in Section 2.4 as the first step of the research pipeline.

Based on a precise calibration, the cross-modal training data generation representing the second step of the research pipeline is discussed in Chapter 3. Thereby, the generation of large-scale datasets is of paramount importance for generating data-driven approaches like deep learning. Usually these datasets are generated in a manual fashion representing a cumbersome and cost extensive task especially within the 3D space of LiDAR sensors. For that reason, a fully automated process for large-scale training data generation called *Autolabeling* is introduced based on the transfer of high-quality image-based semantic labeling results of state-of-the-art approaches to LiDAR point clouds. Hence, a reference camera is mounted within a small spacial distance to a LiDAR sensor to transfer the labels based on a precise intrinsic and extrinsic calibration of both sensor modalities. Additionally, the temporal distance of both sensor measurements is taken into account to further improve the label transfer. However, the performance of the *Autolabeling* process is limited to the calibration performance, the performance of the state-of-the-art image-based semantic labeling approach, and the domain transfer from camera to LiDAR sensor, large-scale datasets for two state-of-the-art mobile LiDAR sensors (VLP-32C and VLS-128) of in total more than 840,000 frames including a small amount of manually annotated point clouds are created on a basis of 13 label classes. The result is a cost-efficient as well as time-efficient cross-modal approach for generating large-scale datasets is proposed.

The large amount of training data generated by the *Autolabeling* process enables the data-driven generation of high-quality semantic labeling approaches of semi-dense LiDAR measurements, which represents the third step of the research pipeline. In Chapter 4, the concept of pixel-wise image-based semantic labeling is transferred to the LiDAR domain. Therefore, the sparse 3D point cloud of a LiDAR sensor is transferred to a dense 2D LiDAR image by a lossless cylindrical projection of the 3D point cloud to leverage the potential of 2D CNNs. As a result, the LiLaNet, a novel CNN architecture for efficient LiDAR-based semantic labeling, is proposed. Additionally, a training technique is developed combining the automatically generated training dataset based on the *Autolabeling* process with a fine-tuning step based on small-scale manually annotated data. This yields a performance boost of up to 14 percentage points while keeping manual annotation efforts low. Furthermore, different optimization techniques like the filter reduction, dilated convolution layer, and factorization of convolution layers are applied to the LiLaNet to additionally increase the real-time capability as well as classification performance. The optimized LiLaNet is evaluated on the recently published SemanticKITTI Benchmark [Behley et al., 2019] while outperforming published state-of-the-art approaches for multi-class LiDAR-based semantic labeling at a real-time capable inference time. This real-time capability was exploited by integrating the LiLaNet into a research vehicle to further analyze the performance in a qualitative way.

The high quality semantically labeled LiDAR point cloud generated by the LiLaNet can be used by higher level modules of mobile robotics or autonomous driving platforms. Thereby, a LiDAR sensor generates thousands of points in a fraction of a second, which decreases the real-time capability of such modules. For this reason, Chapter 7 transferred, as the last step of the research pipeline, the Stixel-World of the stereoscopic camera domain to the LiDAR domain introducing a Stixel representation for LiDAR point clouds including a multi-modal mid-level fusion approach to combine monocular camera and LiDAR measurements. This multi-modal Stixel representation compresses the data of both camera and LiDAR sensor by retaining the underlying geometric and semantic information. This increases the real-time capability of downstream algorithms while minimizing the information loss. Furthermore, the benefits of the multi-modal Stixel-World over uni-modal representations are demonstrated by outperforming the original Stixel-World based on stereoscopic camera regarding the geometric and the semantic consistency of the data representation.

In addition to the different steps of the research pipeline, two different extensions for high-quality LiDAR-based semantic labeling are outlined in Chapters 5 and 6.

First, the portability of the LiLaNet between different LiDAR sensors types is analyzed in Chapter 5. There, the cylindrical projection to a 2D LiDAR image as used for the LiLaNet showed high sensor dependency. In contrast, a novel deep neural network architecture for semantic labeling of semi-dense LiDAR point clouds based on a pillar-like voxel representation called PiLaNet is proposed. Although this representation increases the computational complexity, it shows a higher portability between different LiDAR sensor types. Therefore, a cross-sensor evaluation based on the automatically generated VLP-32C and VLS-128 dataset is performed yielding an improvement of 10 percentage points in mIoU comparing to the LiLaNet. Furthermore, the PiLaNet can be fully transferred across different LiDAR sensor types with minimal adaption effort by fine-tuning the pre-trained network on a small target sensor dataset.

Second, the remaining misclassifications of the LiLaNet are analyzed in Chapter 6 showing mainly plausible misclassifications due to sensor specific constraints like the confusion between the class ‘large vehicle’ and the class ‘small vehicle’. These types of misclassifications are related to the uncertainty of the prediction of the CNN. Unfortunately, the uncertainty of a CNN is difficult to estimate while maintaining real-time capability and without requiring a completeness of the dataset. For this reason, an effective training technique is proposed encoding a label hierarchy into high-quality semantic labeling approaches. This presents the ability of the CNN to implicitly learn the label hierarchy and predict more abstract classes (e. g. ‘vehicle’ for ‘large vehicle’ and ‘small vehicle’) at a high uncertainty which occurs in large distances to the sensor or for underrepresented obstacles within the dataset.

Overall, the proposed research pipeline including the outlined extensions describes different steps to create a robust and real-time capable 3D semantic scene understanding with a high performance for LiDAR sensors. At the same time, the manual effort is low based on an automatic training data generation for large-scale datasets. The proposed 3D scene understanding raises the performance of environment perception of LiDAR sensors leading to an additional valuable sensor modality for mobile robotics and autonomous driving. Consequently, those systems gain performance, robustness, and redundancy increasing the safety as well as the comfort of future mobility.

8.2 OUTLOOK

This thesis proposed and discussed a research pipeline for high-quality, LiDAR-based semantic labeling including different extensions. Thereby, the evaluation of the results depicts further capabilities of the proposed approaches, which are discussed within the following subsections.

8.2.1 *Automatic Training Data Generation*

The results of the Automatic Training Data Generation present three reasons for a decreased performance at transferring the labels from the camera image to the LiDAR point cloud (see Section 3.4.3)

First, the projection of the LiDAR point cloud into the camera image showed some performance limitations due to slight calibration inaccuracies. These can be improved by generating a more precise offline calibration as well as a continuous online calibration of the multi-modal system. Furthermore, the projection misalignment can be reduced by adapting the multi-modal sensor system. For example, multiple cameras at different positions can be used to reduce the occlusion problem and to optimize the sensor system towards different obstacle distance (e. g. fish-eye cameras for small distances and telephoto lens cameras for large distances). Additionally, the projection can be improved by optimizing the ego-motion correction of each point towards each row of the rolling shutter image in case of using rolling shutter cameras.

Second, the results of the Automatic Training Data Generation showed a limitation by the performance of the image-based semantic labeling approaches. These image-based semantic labeling approaches are continuously improving. As a result, newer available image-based CNN models with a higher performance will improve the performance of the *Autolabeling* process. At the same time, the model uncertainty can be used either by extracting a probability estimate for weighting the training of the LiLaNet or by applying an implicit uncertainty representation as discussed in Chapter 6 to reduce the impact of wrong classifications (e. g. at object borders or at plausible misclassifications). Also, an execution of multiple image-based CNN architectures can be used to improve the overall performance.

Third, problems occur due to the domain transfer of the camera to the LiDAR sensor. These can be reduced by adapting the training technique of image-based semantic labeling approaches towards higher performance at object borders, e. g. a higher weighting of pixels at object borders within the error function of the learning rule can lead to a higher performance at object borders. This will directly influence the performance of the LiLaNet as discussed in Section 3.4.3.3.

In addition to the three reasons for the performance decrease, the results showed an overall performance limitation of the *Autolabeling* due to temporal inconsistencies caused by e. g. occlusion artifacts. Adding temporal information in terms of accumulating the LiDAR point clouds of each frame to a dense 3D point cloud might reduce these temporal inconsistencies of the *Autolabeling* process, increasing overall performance.

8.2.2 *LiDAR-based Semantic Labeling*

The LiDAR-based semantic labeling approach represented by the LiLaNet and the optimized LiLaNet already shows outperforming results on different datasets. However, the results depict different limitations which can be further discussed.

In Chapter 4, the performance of the LiLaNet is boosted with the large amount of automatic generated training data based on the *Autolabeling* process, which directly influences the performance of the LiLaNet. As a result, a higher performance of the *Autolabeling* process as proposed within the previous section should increase the performance of the LiLaNet. In addition, the fine-tuning of the LiLaNet with manually annotated point clouds strongly increases the classification performance. Therefore, an in-depth analysis on the necessary amount of manually annotated point clouds to increase the performance of the LiLaNet can be performed.

Additionally, due to a valid evaluation of the LiLaNet within this thesis, the specific network architecture is not changed related to the different datasets used. At the same time, the analysis of the portability between different LiDAR sensors (see Chapter 5) showed that the 2D LiDAR image used within the LiLaNet is rather sensor dependent. As a result, the network architecture of the LiLaNet can further be optimized towards different sensor types like the HDL-64 for the SemanticKITTI Benchmark (see Section 4.4.3).

Furthermore, two different extensions, the reduction of the sensor dependence and the hierarchical semantic labeling, can be pushed one step further to become more robust due to different sensor types and uncertainties of the prediction.

The reduction of the sensor dependence based on the pillar-like architecture of the PiLaNet (see Chapter 5) was compared to the LiLaNet. For that reason, the LiLaNet was used as a backbone CNN which already showed a higher portability between different LiDAR sensors. Furthermore, sensor independence can be expected by optimizing the field of view of the LiLaNet as the backbone CNN to the specific dimensions of the voxel space.

After an increased sensor independence, the PiLaNet can further be used to replace the *Autolabeling* process for new sensor types and generate a LiDAR to LiDAR label transfer. Similar to a teacher network, the computational complex PiLaNet can be used to automatically generate training data for light-weight and real-time capable sensor specific CNN architectures as the LiLaNet (see Figure 5.6 as illustration). This changes the motivation of the reduction of the sensor dependence from the reduced effort to transfer the knowledge of a CNN from one sensor to another (see Chapter 5) to an even lower sensor dependence independent of an affordable higher effort. For that reason, the PiLaNet can be trained in parallel on large (automatically labeled) datasets of different sensor types. This should increase the sensor independence of both the VFE and the backbone CNN. Additionally, this can be supported by **G**enerative **A**dversarial **N**etworks (GANs) to generate a sensor independent voxel representation.

The hierarchical semantic labeling as outlined in Chapter 6 presents an implicit uncertainty integration, while implicitly ascending and descending a label hierarchy related to the uncertainty of a CNN. The results show e. g. the prediction of parent classes at object borders due to higher uncertainty. This can further be improved by adapting the weights of the learning rule to create sharp obstacle borders within the LiDAR semantic labeling image.

Furthermore, the hierarchical semantic labeling is evaluated based on the IoU for a comparison with other non-hierarchical semantic labeling approaches. For an adaption of parameters or weights as mentioned before, an in-depth quantitative evaluation of hierarchical semantic labeling approaches is needed. Therefore, an evaluation metric similar to the IoU taking the label hierarchy into account has to be defined.

In addition, the hierarchical semantic labeling can be used for an active learning technique due to the implicit uncertainty representation. As a result, LiDAR point clouds, whereby parent classes are predicted at a high rate, can be manually annotated. This should directly lower the implicit uncertainty of the hierarchical semantic labeling approach and increase the overall performance.

8.2.3 *Multi-Modal Stixels*

The multi-modal Stixel-World as proposed in Chapter 7 represents the last step of the research pipeline (see Figure 1.5). Resulting is a performance increase of LiDAR-based semantic labeling approaches as suggested within the previous section that directly influences the multi-modal Stixel-World. Additionally, the extensions of LiDAR-based semantic labeling approaches as proposed in Chapters 5 and 6 can be integrated including the hierarchical semantic labeling which implicitly encodes the uncertainty of the prediction into a hierarchical label output. The quality of the probability estimates required for the data likelihood of LiDAR semantics can thus be improved.

Finally, the multi-modal Stixel-World operates on a dense measurement representation as the dense cylindrical projection of a single LiDAR point cloud or a dense camera image. This restriction can be reduced to increase the possible combinations of sensor modalities. For example, a combination of two LiDAR sensors is challenging due to the sparse projection of one LiDAR point cloud into the dense LiDAR image of the second point cloud. As a result, the multi-modal Stixel-World can further be extended to handle different sparse sensor representations.

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

H. Abu Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes. *International Journal of Computer Vision (IJCV)*, 126(9):961–972, 2018. (Cited on page 5.)

O. A. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *International Journal of Robotics Research*, 39(1):3–20, 2020. (Cited on page 33.)

I. Armeni, S. Sax, A. R. Zamir, and Others. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. In *arXiv preprint:1702.01105*, 2017. (Cited on page 31.)

H. Badino, U. Franke, and D. Pfeiffer. The Stixel World - A Compact Medium Level Representation of the 3D-World. In *Joint Pattern Recognition Symposium (DAGM)*, 2009. (Cited on pages 7 and 101.)

H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee. Intention-Aware Online POMDP Planning for Autonomous Driving in a Crowd. In *International Conference on Robotics and Automation (ICRA)*, 2015. (Cited on page 3.)

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 *International Conference on Computer Vision (ICCV)*, 2019. (Cited on pages 6, 13, 31, 41, 67, 68, 69, 87, and 114.)

J. Beltran, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. De La Escalera. BirdNet: A 3D Object Detection Framework from LiDAR Information. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2018. (Cited on page 72.)

R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. Fast Stixel Computation for Fast Pedestrian Detection. In *European Conference on Computer Vision (ECCV) Workshops*, 2012. (Cited on page 103.)

A. Bhoi. Monocular Depth Estimation: A Survey. In *arXiv preprint: 1901.09402*, 2019. (Cited on page 34.)

P. Biasutti, A. Bugeau, J.-F. Aujol, and M. Brédif. RIU-Net: Embarrassingly simple semantic segmentation of 3D LiDAR point cloud. In *arXiv preprint: 1905.08748*, 2019. (Cited on page 32.)

S. Bileschi. Fully Automatic Calibration of LIDAR and Video Streams From a Vehicle. In *International Conference on Computer Vision (ICCV) Workshops*, 2009. (Cited on page 29.)

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, New York, 2006. (Cited on pages 18 and 20.)

M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to End Learning for Self-Driving Cars. In *arXiv preprint: 1604.07316*, 2016. (Cited on page 33.)

G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009. (Cited on page 5.)

L. Caltagirone, S. Scheidegger, L. Svensson, and M. Wahde. Fast LIDAR-based Road Detection Using Fully Convolutional Neural Networks. In *Intelligent Vehicles Symposium (IV)*, 2017. (Cited on page 53.)

H.-J. Chien, R. Klette, N. Schneider, and U. Franke. Visual Odometry Driven Online Calibration for Monocular LiDAR-Camera Systems. In *International Conference on Pattern Recognition (ICPR)*, 2016. (Cited on page 30.)

M. Cordts. *Understanding Cityscapes: Efficient Urban Semantic Scene Understanding*. Phd thesis, Technische Universität Darmstadt, 2017. (Cited on pages 35, 44, 45, 47, 103, 108, and 109.)

M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on pages 4, 5, 6, 31, 34, 35, 38, 39, 41, 44, 45, 46, 47, 64, 67, 87, 89, 102, and 104.)

M. Cordts, T. Rehfeld, L. Schneider, D. Pfeiffer, M. Enzweiler, S. Roth, M. Pollefeys, and U. Franke. The Stixel World: A Medium-Level Representation of Traffic Scenes. *Image and Vision Computing*, 68: 40–52, 2017. (Cited on pages 8, 101, 102, 104, 105, 106, 107, 108, 109, and 110.)

C. Couprie, C. Farabet, L. Najman, and Others. Indoor Semantic Segmentation using depth information. In *International Conference on Learning Representations (ICLR)*, 2013. (Cited on page 52.)

A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. NieBner. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 31.)

N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE, 2005. (Cited on page 5.)

N. Das, S. Chaba, S. Gandhi, D. H. Chau, and X. Chu. GOGGLES: Automatic Training Data Generation with Affinity Coding. In *arXiv preprint: 1903.04552*, 2019. (Cited on page 34.)

Deutsches Institut für Normung. Straßenfahrzeuge - Fahrzeugdynamik und Fahrverhalten - Begriffe. In *DIN ISO 8855:2011*, 2011. (Cited on page 79.)

A. Dewan and W. Burgard. DeepTemporalSeg: Temporally Consistent Semantic Segmentation of 3D LiDAR Scans. In *arXiv preprint: 1906.06962*, 2019. (Cited on page 32.)

A. Dewan, G. L. Oliveira, and W. Burgard. Deep Semantic Classification for 3D LiDAR data. In *International Conference on Intelligent Robots and Systems (IROS)*, 2017. (Cited on pages 32 and 53.)

A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *Conference on Robot Learning (CoRL)*, 2017. (Cited on page 32.)

M. Everingham, L. van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge (VOC). In *International Conference on Computer Vision (ICCV) Workshops*, 2011. (Cited on page 5.)

M. Everingham, L. van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge (VOC). In *European Conference on Computer Vision (ECCV) Workshops*, 2012. (Cited on page 5.)

J. Fang, D. Zhou, F. Yan, T. Zhao, F. Zhang, Y. Ma, L. Wang, and R. Yang. Augmented LiDAR Simulator for Autonomous Driving. *Robotics and Automation Letters*, 5(2):1931–1938, 2020. (Cited on pages 32 and 33.)

D. Feng, C. Haase-Schutz, L. Rosenbaum, H. Hertlein, C. Glaser, F. Timm, W. Wiesbeck, and K. Dietmayer. Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *Transactions on Intelligent Transportation Systems (TITS)*, 2020. (Cited on pages 3, 4, 7, and 71.)

O. Forsberg. *Semantic Stixels fusing LIDAR for Scene Perception*. Master thesis, KTH Royal Institute of Technology Stockholm, 2018. (Cited on page 103.)

K. Fukushima. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, 36(4):193–202, 1980. (Cited on page 21.)

Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, 2016. (Cited on page 89.)

A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018. (Cited on pages 4, 7, 25, 71, 72, and 103.)

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012a. (Cited on pages 13, 31, 32, 53, and 67.)

A. Geiger, F. Moosmann, O. Car, and B. Schuster. Automatic Camera and Range Sensor Calibration using a single Shot. In *International Conference on Robotics and Automation (ICRA)*, 2012b. (Cited on page 29.)

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010. (Cited on pages 60 and 94.)

I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. (Cited on pages 14, 15, 16, 17, and 19.)

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Modern Neural Networks. In *International Conference on Machine Learning (ICML)*, 2017. (Cited on page 89.)

S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning Rich Features from RGB-D Images for Object Detection and Segmentation. In *European Conference on Computer Vision (ECCV)*, 2014. (Cited on pages 52 and 103.)

H. Habibi Aghdam and E. Jahani Heravi. *Guide to Convolutional Neural Networks*. Springer International Publishing, Cham, 2017. (Cited on pages 14, 15, 16, 18, 19, and 26.)

T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3D.NET: A New Large-Scale Point Cloud Classification Benchmark. *Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, IV-1/W1:91–98, 2017. (Cited on pages 6 and 31.)

X. Han. MR-based synthetic CT generation using a deep convolutional neural network method. *Medical Physics*, 44(4):1408–1419, 2017. (Cited on page 33.)

R. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. *Robotica*, 23(2):271–271, 2005. (Cited on page 37.)

K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *International Conference on Computer Vision (ICCV)*, 2015. (Cited on pages 60 and 80.)

E. Hemayed. A Survey of Camera Self-Calibration. In *International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2003. (Cited on page 29.)

D. Hernandez-Juarez, L. Schneider, A. Espinosa, and Others. Slanted Stixels: Representing San Francisco's Steepest Streets. In *British Machine Vision Conference (BMVC)*, 2017. (Cited on pages 8, 101, 102, and 103.)

L. Hobert, A. Festag, I. Llatser, L. Altomare, F. Visintainer, and A. Kovacs. Enhancements of V2X communication in support of cooperative autonomous driving. *IEEE Communications Magazine*, 53(12):64–70, 2015. (Cited on page 3.)

S. Hong, J. Oh, H. Lee, and B. Han. Learning Transferrable Knowledge for Semantic Segmentation with Deep Convolutional Neural Network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on page 72.)

C. Hubschneider, A. Bauer, J. Doll, M. Weber, S. Klemm, F. Kuhnt, and J. M. Zollner. Integrating End-to-End Learned Steering into Probabilistic Autonomous Driving. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2017. (Cited on page 33.)

F. N. Iandola, S. Han, M. W. Moskewicz, Khalid Ashraf, W. J. Dally, and K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. In *International Conference on Learning Representations (ICLR)*, 2017. (Cited on page 53.)

E. R. Kandel, J. H. Schwartz, T. Jessell, S. A. Siegelbaum, and A. J. Hudspeth. *Principles of Neural Science*. McGraw-Hill Medical, 2013. (Cited on page 14.)

B. Kang and T. Q. Nguyen. Random Forest With Learned Representations for Semantic Segmentation. *Transactions on Image Processing*, 28(7):3542–3555, 2019. (Cited on page 6.)

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015. (Cited on pages 21, 60, 80, and 94.)

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Conference on Neural Information Processing Systems (NIPS)*, 2012. (Cited on pages 16 and 23.)

V. R. Kumar, S. Milz, C. Witt, M. Simon, K. Amende, J. Petzold, S. Yogamani, and T. Pech. Monocular Fisheye Camera Depth Estimation Using Sparse LiDAR Supervision. In *Conference on Intelligent Transportation Systems (ITSC)*, 2018. (Cited on page 33.)

L. Ladický, C. Russell, P. Kohli, and P. H. Torr. Associative hierarchical CRFs for object class image segmentation. In *International Conference on Computer Vision (ICCV)*, 2009. (Cited on page 5.)

A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on pages 53, 73, 76, and 77.)

C. Laugier, I. E. Paromtchik, M. Perrollaz, M. Y. Yong, J.-D. Yoder, C. Tay, K. Mekhnacha, and A. Negre. Probabilistic Analysis of Dynamic Scenes and Collision Risks Assessment to Improve Driving Safety. *Intelligent Transportation Systems Magazine*, 3(4):4–19, 2011. (Cited on page 3.)

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE (Journal)*, 86(11):2278–2324, 1998. (Cited on pages 21, 22, 23, and 24.)

D. Levi, N. Garnett, and E. Fetaya. StixelNet: A Deep Convolutional Network for Obstacle Detection and Road Segmentation. In *British Machine Vision Conference (BMVC)*, 2015. (Cited on page 103.)

J. Levinson and S. Thrun. Automatic Online Calibration of Cameras and Lasers. In *Robotics: Science and Systems*, 2013. (Cited on page 29.)

J. Levinson, J. Askeland, J. Becker, and Others. Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV)*, 2011. (Cited on page 4.)

Y. Li, R. Bu, M. Sun, and Others. PointCNN: Convolution On X-Transformed Points. In *Conference on Neural Information Processing Systems (NIPS)*, 2018. (Cited on pages 52 and 72.)

T.-Y. Lin, M. Maire, S. Belongie, and Others. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*, 2014. (Cited on page 31.)

P. Lippe. *Hierarchical Multi-label Object Detection of Rare Classes for Autonomous Driving*. Bachelor thesis, Baden-Wuerttemberg Cooperative State University (DHBW), 2018. (Cited on pages 90 and 94.)

M.-Y. Liu, S. Lin, S. Ramalingam, and Others. Layered Interpretation of Street View Images. In *Robotics: Science and Systems*, 2015. (Cited on page 103.)

S. Liu, A. J. Davison, and E. Johns. Self-Supervised Generalisation with Meta Auxiliary Learning. In *Conference on Neural Information Processing Systems (NIPS)*, 2019. (Cited on page 72.)

J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (Cited on pages 25 and 103.)

D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, nov 2004. (Cited on page 5.)

A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *International Conference on Machine Learning (ICML)*, 2013. (Cited on page 16.)

K. E. Madawi, H. Rashed, A. E. Sallab, O. Nasr, H. Kamel, and S. Yogamani. RGB and LiDAR fusion based 3D Semantic Segmentation for Autonomous Driving. In *Conference on Intelligent Transportation Systems (ITSC)*, 2019. (Cited on page 32.)

M. Martinez, A. Roitberg, D. Koester, and Others. Using Technology Developed for Autonomous Cars to Help Navigate Blind People. In *International Conference on Computer Vision (ICCV) Workshops*, 2017. (Cited on page 103.)

O. Matan, C. J. C. Burges, Y. LeCun, and J. S. Denker. Multi-Digit Recognition Using a Space Displacement Neural Network. In *Conference on Neural Information Processing Systems (NIPS)*, 1991. (Cited on page 24.)

D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *International Conference on Intelligent Robots and Systems (IROS)*, 2015. (Cited on page 73.)

J. Mei, B. Gao, D. Xu, W. Yao, X. Zhao, and H. Zhao. Semantic Segmentation of 3D LiDAR Data in Dynamic Scene Using Semi-Supervised Learning. *Transactions on Intelligent Transportation Systems (TITS)*, 2019. (Cited on pages 4 and 53.)

H. H. Meinel. Evolving automotive radar - From the very beginnings into the future. In *European Conference on Antennas and Propagation (EuCAP)*, 2014. (Cited on page 4.)

F. M. Mirzaei, D. G. Kottas, and S. I. Roumeliotis. 3D LIDAR-camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization. *International Journal of Robotics Research*, 31(4):452–467, 2012. (Cited on page 29.)

K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su. PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (Cited on page 90.)

N. Muhammad and S. Lacroix. Calibration of a rotating multi-beam lidar. In *International Conference on Intelligent Robots and Systems (IROS)*, 2010. (Cited on page 29.)

A. C. Muller and S. Behnke. Learning depth-sensitive conditional random fields for semantic segmentation of RGB-D images. In *International Conference on Robotics and Automation (ICRA)*, 2014. (Cited on page 103.)

National Highway Traffic Safety Administration (NHTSA). NHTSA SAE Automation Levels, 2020. URL `https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety`. Accessed 2020/03/14. (Cited on page 3.)

N. Nourani-Vatani, R. López-Sastre, and S. Williams. Structured Output Prediction with Hierarchical Loss Functions for Seafloor Imagery Taxonomic Categorization. In *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2015. (Cited on pages 90, 92, and 93.)

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. *International Journal of Robotics Research*, 37(8):841–866, 2018. (Cited on pages 3 and 103.)

G. L. Oliveira, W. Burgard, and T. Brox. Efficient deep models for monocular road segmentation. In *International Conference on Intelligent Robots and Systems (IROS)*, 2016. (Cited on page 53.)

X. Pan, Y. Wu, and H. Ogai. Automatic Training Data Generation Method for Pixel-Level Road Lane Segmentation. In *International Conference on Genetic and Evolutionary Computing (ICGEC)*, 2018. (Cited on page 34.)

G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice. Automatic Extrinsic Calibration of Vision and Lidar by Maximizing Mutual Information. *Journal of Field Robotics*, 32(5):696–722, 2015. (Cited on page 29.)

D. Pfeiffer. *The Stixel World - A Compact Medium-level Representation for Efficiently Modeling Dynamic Three-dimensional Environments*. Phd thesis, Humboldt-Universität Berlin, 2012. (Cited on pages 7, 101, 103, 104, and 109.)

C. R. Qi, H. Su, K. Mo, and Others. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017a. (Cited on pages 52, 68, 69, 72, 73, and 76.)

C. R. Qi, L. Yi, H. Su, and Others. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Conference on Neural Information Processing Systems (NIPS)*, 2017b. (Cited on pages 52, 68, 69, and 72.)

M. A. Rahman and Y. Wang. Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation. In *International Symposium on Visual Computing (ISVC)*, 2016. (Cited on page 26.)

R. H. Rasshofer and K. Gresser. Automotive Radar and Lidar Systems for Next Generation Driver Assistance Functions. *Advances in Radio Science*, 3:205–209, 2005. (Cited on page 12.)

J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 90.)

K. Reif. *Fahrstabilisierungssysteme und Fahrassistenzsysteme*. Vieweg+Teubner Verlag, Wiesbaden, 1. edition, 2010. (Cited on page 1.)

G. Riegler, A. O. Ulusoy, and A. Geiger. OctNet: Learning Deep 3D Representations at High Resolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages 52 and 73.)

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. (Cited on page 20.)

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. (Cited on pages 6 and 31.)

S. Sankaranarayanan, Y. Balaji, A. Jain, and Others. Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on page 103.)

L. Schneider, M. Cordts, T. Rehfeld, and Others. Semantic Stixels: Depth is not enough. In *Intelligent Vehicles Symposium (IV)*, 2016. (Cited on pages 7, 101, 102, 103, 104, and 108.)

X. Shen. A survey of Object Classification and Detection based on 2D/3D data. In *arXiv preprint: 1905.12683*, 2019. (Cited on page 54.)

J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-class Object Recognition and Segmentation. In *European Conference on Computer Vision (ECCV)*, 2006. (Cited on page 5.)

J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. (Cited on page 5.)

A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on page 33.)

M. Siam, S. Elkerdawy, M. Jagersand, and S. Yogamani. Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2017. (Cited on pages 5 and 25.)

N. Silberman, D. Hoiem, P. Kohli, and Others. Indoor Segmentation and Support Inference from RGBD Images. In *European Conference on Computer Vision (ECCV)*, 2012. (Cited on pages 5, 6, and 31.)

C. N. Silla and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011. (Cited on pages 89 and 92.)

K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, 2015. (Cited on page 23.)

F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *Network*, 18(4):45–50, 2004. (Cited on page 29.)

M. Skilton and F. Hovsepian. *The 4th Industrial Revolution*. Springer International Publishing, Cham, 2018. (Cited on pages 14 and 17.)

Statistisches Bundesamt (Destatis). Unfallentwicklung auf deutschen Straßen 2017. Technical report, Berlin, 2018. (Cited on page 1.)

Statistisches Bundesamt (Destatis). Verkehr - Verkehrsunfälle - 2018. Technical Report 7, 2019a. (Cited on page 2.)

Statistisches Bundesamt (Destatis). Trend in the number of persons killed in road traffic accidents, 2020. URL `https://www.destatis.de/EN/Press/2020/02/PE20_061_46241.html`. Accessed 2020/03/14. (Cited on page 2.)

C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *International Conference on Computer Vision (ICCV)*, 2017. (Cited on pages 6, 25, and 31.)

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (Cited on pages 23 and 35.)

C. Szegedy, V. Vanhoucke, S. Ioffe, and Others. Rethinking the Inception Architecture for Computer Vision. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on pages 55, 58, and 66.)

R. Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer London, London, 2011. (Cited on pages 5, 21, and 22.)

L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese. SEGCloud: Semantic Segmentation of 3D Point Clouds. In *International Conference on 3D Vision (3DV)*, 2017. (Cited on page 52.)

M. Thoma. A Survey of Semantic Segmentation. In *arXiv preprint: 1602.06541*, 2016. (Cited on page 5.)

S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, and G. Hoffmann. Stanley, the Robot that Won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692, 2006. (Cited on pages 3 and 4.)

P. Tino, L. Benuskova, and A. Sperduti. Artificial Neural Network Models. In *Springer Handbook of Computational Intelligence*, volume 8, pages 455–471. Springer, Berlin, Heidelberg, 2015. (Cited on page 15.)

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *International Conference on Intelligent Robots and Systems (IROS)*, 2017. (Cited on page 33.)

L. Torrey and J. Shavlik. Transfer Learning. In E. S. Olivas, J. D. M. Guerrero, M. Martinez-Sober, J. R. Magdalena-Benedito, and A. J.

Serrano López, editors, *Handbook of Research on Machine Learning Applications*. IGI Global, Hershey (USA), 2009. (Cited on page 72.)

P. Tosteberg. *Semantic Segmentation of Point Clouds using Deep Learning*. Master thesis, Linköping University, 2017. (Cited on page 52.)

C. Urmson, C. Baker, J. Dolan, and Others. Autonomous Driving in Traffic: Boss and the Urban Challenge. *AI Magazine*, 30(2):17–28, 2009. (Cited on pages 3 and 4.)

R. Varga, A. Costea, H. Florea, and Others. Super-sensor for 360-degree Environment Perception: Point Cloud Segmentation Using Image Features. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2017. (Cited on page 34.)

Velodyne LiDAR Inc. Ultra Puck, 2019a. URL `https://velodynelidar.com/vlp-32c.html`. Accessed 2019/08/07. (Cited on pages 12 and 49.)

Velodyne LiDAR Inc. Alpha Puck, 2019b. URL `https://velodynelidar.com/vls-128.html`. Accessed 2019/08/07. (Cited on pages 12 and 49.)

Velodyne LiDAR Inc. HDL-64E, 2019c. URL `https://velodynelidar.com/hdl-64e.html`. Accessed 2019/08/07. (Cited on page 13.)

Verband der Automobilindustrie e. V. (VDA). Automatisierung. Von Fahrerassistenzsystemen zum automatisierten Fahren. Technical report, Potsdam, 2015. (Cited on page 2.)

T.-d. Vu, J. Burlet, O. Aycard, and Others. Grid-based localization and local mapping with moving object detection and tracking Grid-based Localization and Local Mapping with Moving Object Detection and Tracking. *Journal Information Fusion*, 12(1):58–69, 2011. (Cited on page 3.)

U. Wandinger. Introduction to Lidar. In *Lidar*, pages 1–18. Springer-Verlag, New York, 2006. (Cited on page 12.)

B. Wang, V. Wu, B. Wu, and K. Keutzer. LATTE: Accelerating LiDAR Point Cloud Annotation via Sensor Fusion, One-Click Annotation, and Tracking. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2019a. (Cited on page 34.)

F. Wang, Y. Zhuang, H. Gu, and H. Hu. Automatic Generation of Synthetic LiDAR Point Clouds for 3-D Data Analysis. *Transactions on Instrumentation and Measurement*, 68(7):2671–2673, 2019b. (Cited on pages 32 and 33.)

Y. Wang, T. Shi, P. Yun, L. Tai, and M. Liu. PointSeg: Real-Time Semantic Segmentation Based on 3D LiDAR Point Cloud. In *arXiv preprint: 1807.06288*, 2018. (Cited on page 53.)

Waymo LLC. On the road to Fully Self-driving - Waymo Safety Report. Technical report, 2018. (Cited on page 5.)

M. Weber, M. Huber, and J. M. Zollner. HDTLR: A CNN based Hierarchical Detector for Traffic Lights. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2018. (Cited on page 90.)

J. Wei, J. M. Snider, J. Kim, and Others. Towards a viable autonomous driving research platform. In *Intelligent Vehicles Symposium (IV)*, 2013. (Cited on page 4.)

S. Wirges, T. Fischer, C. Stiller, and J. B. Frias. Object Detection and Classification in Occupancy Grid Maps Using Deep Convolutional Networks. In *Conference on Intelligent Transportation Systems (ITSC)*, 2018. (Cited on page 4.)

R. Wolf and J. C. Platt. Postal Address Block Location Using A Convolutional Locator. In *Conference on Neural Information Processing Systems (NIPS)*, 1994. (Cited on page 24.)

B. Wu, A. Wan, X. Yue, and Others. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. In *International Conference on Robotics and Automation (ICRA)*, 2018. (Cited on pages 32, 62, and 69.)

B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer. SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud. In *International Conference on Robotics and Automation (ICRA)*, 2019. (Cited on pages 32, 33, 53, and 69.)

B. Yang, W. Luo, and R. Urtasun. PIXOR: Real-time 3D Object Detection from Point Clouds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on page 72.)

F. Yang, W. Choi, and Y. Lin. Exploit All the Layers: Fast and Accurate CNN Object Detector with Scale Dependent Pooling and Cascaded Rejection Classifiers. In *Conference on Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on page 101.)

Yi Yang, S. Hallman, D. Ramanan, and C. C. Fowlkes. Layered Object Models for Image Segmentation. *Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1731–1743, 2012. (Cited on page 5.)

J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Conference on Neural Information Processing Systems (NIPS)*, 2014. (Cited on page 72.)

F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *International Conference on Learning Representations (ICLR)*, 2016. (Cited on pages 57 and 58.)

M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision (ECCV)*, 2014. (Cited on pages 22, 24, 25, and 121.)

Q. Zhang and R. Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *International Conference on Intelligent Robots and Systems (IROS)*, 2004. (Cited on page 29.)

Y. Zhou and O. Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (Cited on pages 53, 73, 76, 77, and 101.)

H. Zhu, F. Meng, J. Cai, and S. Lu. Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. *Journal of Visual Communication and Image Representation*, 34:12–27, 2016. (Cited on pages 5 and 25.)

J. Ziegler, P. Bender, M. Schreiber, and Others. Making Bertha Drive - An Autonomous Journey on a Historic Route. *Intelligent Transportation Systems Magazine*, 6(2):8–20, 2014. (Cited on pages 3 and 4.)

PUBLICATIONS

CONFERENCE PROCEEDINGS

F. Piewak, T. Rehfeld, M. Weber, and J. M. Zöllner. Fully Convolutional Neural Networks for Dynamic Object Detection in Grid Maps. In *Intelligent Vehicles Symposium (IV)*, 2017. (Cited on page 3.)

N. Schneider, F. Piewak, C. Stiller, and U. Franke. RegNet: Multimodal Sensor Registration using Deep Neural Networks. In *Intelligent Vehicles Symposium (IV)*, 2017. © 2017 IEEE (Cited on pages 7, 11, 30, and 85.)

F. Piewak, P. Pinggera, M. Schäfer, D. Peter, B. Schwarz, N. Schneider, M. Enzweiler, D. Pfeiffer, and M. Zöllner. Boosting LiDAR-Based Semantic Labeling by Cross-modal Training Data Generation. In *European Conference on Computer Vision (ECCV) Workshops*, 2018b. (Cited on pages 7, 31, 38, 39, 44, 45, 51, 55, 56, 60, 62, 63, 64, and 85.)

F. Piewak, P. Pinggera, M. Enzweiler, D. Pfeiffer, and M. Zöllner. Improved Semantic Stixels via Multimodal Sensor Fusion. In *German Conference on Pattern Recognition (GCPR)*, 2018a. (Cited on pages 7, 85, 101, 102, 104, 105, 110, and 111.)

F. Piewak, P. Pinggera, and M. Zöllner. Analyzing the Cross-Sensor Portability of Neural Network Architectures for LiDAR-based Semantic Labeling. In *Intelligent Transportation Systems Conference (ITSC)*, 2019. © 2019 IEEE (Cited on pages 31, 36, 40, 71, 74, 75, 77, 78, 79, and 81.)

R. Heinzler, F. Piewak, P. Schindler, and W. Stork. CNN-Based Lidar Point Cloud De-Noising in Adverse Weather. In *International Conference on Robotics and Automation (ICRA)*, 2020a.

JOURNALS

R. Heinzler, F. Piewak, P. Schindler, and W. Stork. CNN-Based Lidar Point Cloud De-Noising in Adverse Weather. *Robotics and Automation Letters*, 5(2):2514–2521, 2020b.

SUPERVISED THESES

M. Schillinger. *Analysis and Optimization of Fully Convolutional Neural Networks for LiDAR-based Semantic Labeling*. Master thesis, Hochschule für Technik Stuttgart, 2018. (Cited on pages 51, 54, 55, 56, 65, and 66.)

C. Bozatzidou. *Hierarchische semantische LiDAR-basierte Szenensegmentierung*. Master thesis, Hochschule für Technik Stuttgart, 2019. (Cited on pages 87, 95, 96, 97, 98, and 99.)

PATENTS

F. P. J. Piewak and N. Schneider. Verfahren zur Kalibrierung von Sensoren eines Fahrzeugs, Publication Date: 2018/01/25. DE. Patent DE102017007765 (A1), 2018. (Cited on pages 11 and 30.)

F. P. J. Piewak, D. Peter, M. Enzweiler, N. Schneider, P. Pinggera, P. Schindler, R. Schweiger, U. Franke, M. Schäfer, and D. Pfeiffer. Verfahren zum Betreiben eines Fahrerassistenzsystems mit zwei Erfassungseinrichtungen, Publication Date: 2020/01/30. DE. Patent DE102018005969 (A1), 2020a. (Cited on page 101.)

F. P. J. Piewak, D. Peter, M. Enzweiler, N. Schneider, P. Pinggera, P. Schindler, R. Schweiger, U. Franke, M. Schäfer, and D. Pfeiffer. Method for Operating a Driver Assistance System having two Detection Devices, Publication Date: 2020/01/30. WO. Patent WO2020/020654 (A1), 2020b. (Cited on page 101.)

F. P. J. Piewak, C. Bozatzidou, and J. Uhrig. Verfahren zum Trainieren eines neuronalen Netzwerks zur Objektklassifizierung für eine elektronische Recheneinrichtung eines Kraftfahrzeugs, sowie elektronische Recheneinrichtung, Submission Date: 2020/01/31. DE. Patent DE102020000662 (A1), 2020c. (Cited on page 87.)