Martin Simon

Point Cloud Processing for Environmental Analysis in Autonomous Driving using Deep Learning

Point Cloud Processing for Environmental Analysis in Autonomous Driving using Deep Learning

Martin Simon

Universitätsverlag Ilmenau
2023

Impressum

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über
http://dnb.d-nb.de abrufbar.

Diese Arbeit hat der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau als Dissertation vorgelegen.

Tag der Einreichung:	25. März 2022
1. Gutachter:	Univ.-Prof. Dr.-Ing. Horst-Michael Groß (Technische Universität Ilmenau)
2. Gutachter:	Univ.-Prof. Dr.-Ing. J. Marius Zöllner (Karlsruher Institut für Technologie)
3. Gutachter:	Univ.-Prof. Dr. rer. nat. Bernhard Sick (Universität Kassel)
Tag der Verteidigung:	27. Februar 2023

Technische Universität Ilmenau/Universitätsbibliothek
Universitätsverlag Ilmenau
Postfach 10 05 65
98684 Ilmenau
https://www.tu-ilmenau.de/universitaetsverlag

ISBN 978-3-86360-272-7 (Druckausgabe)
DOI 10.22032/dbt.55809
URN urn:nbn:de:gbv:ilm1-2023000032

Kurzfassung

Eines der Hauptziele führender Automobilhersteller sind autonome Fahrzeuge. Sie benötigen ein sehr präzises System für die Wahrnehmung der Umgebung, dass für jedes denkbare Szenario überall auf der Welt funktioniert. Daher sind verschiedene Arten von Sensoren im Einsatz, sodass neben Kameras u. a. auch Lidar Sensoren ein wichtiger Bestandteil sind. Die Entwicklung auf diesem Gebiet ist für künftige Anwendungen von höchster Bedeutung, da Lidare eine genauere, von der Umgebungsbeleuchtung unabhängige, Tiefendarstellung bieten. Insbesondere Algorithmen und maschinelle Lernansätze wie Deep Learning, die Rohdaten über Lernzprozesse direkt verarbeiten können, sind aufgrund der großen Reichweite und der dreidimensionalen Auflösung der gemessenen Punktwolken sehr wichtig. Somit hat sich ein weites Forschungsfeld mit vielen Herausforderungen und ungelösten Problemen etabliert.

Diese Arbeit zielt darauf ab, dieses Defizit zu verringern und effiziente Algorithmen zur 3D-Objekterkennung zu entwickeln. Sie stellt ein tiefes Neuronales Netzwerk mit spezifischen Schichten und einer neuartigen Fehlerfunktion zur sicheren Lokalisierung und Schätzung der Orientierung von Objekten aus Punktwolken bereit. Zunächst wird ein 3D-Detektor entwickelt, der in nur einem Vorwärtsdurchlauf aus einer Punktwolke alle Objekte detektiert. Anschließend wird dieser Detektor durch die Fusion von komplementären semantischen Merkmalen aus Kamerabildern und einem gemeinsamen probabilistischen Tracking verfeinert, um die Detektionen zu stabilisieren und Ausreißer zu filtern. Im letzten Teil wird ein Konzept für den Einsatz in einem bestehenden Testfahrzeug vorgestellt, das sich auf die halbautomatische Generierung eines geeigneten Datensatzes konzentriert. Hierbei wird eine Auswertung auf Daten von Automotive-Lidaren vorgestellt. Als Alternative zur zielgerichteten künstlichen Datengenerierung wird ein weiteres generatives Neuronales Netzwerk untersucht.

Experimente mit den erzeugten anwendungsspezifischen- und Benchmark-Datensätzen zeigen, dass sich die vorgestellten Methoden mit dem Stand der Technik messen können und gleichzeitig auf Effizienz für den Einsatz in selbstfahrenden Autos optimiert sind. Darüber hinaus enthalten sie einen umfangreichen Satz an Evaluierungsmetriken und -ergebnissen, die eine solide Grundlage für die zukünftige Forschung bilden.

Abstract

One of the main objectives of leading automotive companies is autonomous self-driving cars. They need a very precise perception system of their environment, working for every conceivable scenario. Therefore, different kinds of sensor types are in use. Besides cameras, lidar scanners became very important. The development in that field is significant for future applications and system integration because lidar offers a more accurate depth representation, independent from environmental illumination. Especially algorithms and machine learning approaches, including Deep Learning and Artificial Intelligence based on raw laser scanner data, are very important due to the long range and three-dimensional resolution of the measured point clouds. Consequently, a broad field of research with many challenges and unsolved tasks has been established.

This thesis aims to address this deficit and contribute highly efficient algorithms for 3D object recognition to the scientific community. It provides a Deep Neural Network with specific layers and a novel loss to safely localize and estimate the orientation of objects from point clouds. First, a single shot 3D object detector is developed that outputs dense predictions in only one forward pass. Next, this detector is refined by fusing complementary semantic features from cameras and a joint probabilistic tracking to stabilize predictions and filter outliers. In the last part, a concept for deployment into an existing test vehicle focuses on the semi-automated generation of a suitable dataset. Subsequently, an evaluation of data from automotive-grade lidar scanners is presented. A Generative Adversarial Network is also being developed as an alternative for target-specific artificial data generation.

Experiments on the acquired application-specific and benchmark datasets show that the presented methods compete with a variety of state-of-the-art algorithms while being trimmed down to efficiency for use in self-driving cars. Furthermore, they include an extensive set of standard evaluation metrics and results to form a solid baseline for future research.

Danksagung

An dieser Stelle möchte ich allen beteiligten Personen danken, die mich bei der Anfertigung meiner Dissertation begleitet und unterstützt haben.

Mein besonderer Dank gilt Prof. Horst-Michael Groß für die ausgezeichnete Betreuung und kontinuierliche Unterstützung bei der Umsetzung der gesamten Arbeit. Im gleichen Zuge bedanke ich mich bei den Mitarbeiterinnen und Mitarbeitern der TU Ilmenau für die Organisation und den reibungslosen Ablauf.

Ein weiterer großer Dank gilt Stefan Milz, der die Arbeit durch seine Inspiration, Anleitung und tatkräftige Unterstützung von Anfang an geprägt hat.

Außerdem möchte ich mich bei der Valeo Schalter und Sensoren GmbH, insbesondere Johannes Petzold und Jörg Schrepfer, für die Ermöglichung einer externen Promotion recht herzlich bedanken.

Für das Korrekturlesen danke ich zudem David, Hauke und Jens.

Zuletzt danke ich meiner Familie, besonders meiner Frau Michaela, für ihre immerwährende Ermutigung mit lieben Worten und die Geduld bis zur Fertigstellung dieser Arbeit neben dem Arbeitsalltag.

Contents

Chapter 1

Introduction

During the last decades, humankind has become more and more fascinated by self-driving vehicles. Driverless cars that suddenly show up when needed as if ordered by magic could turn the worldwide traffic situation upside-down. Even small children can be safely carried as passengers while entertaining, watching movies, or sleeping during long trips. Driving licenses are no longer required, and a wide range of new services and applications would emerge. Todays mostly privately owned vehicles could be replaced by shared mobility with shuttles and robotaxis for human transportation and autonomous delivery trucks for the transportation of goods.

In recent years vehicles on public roads have developed into complex technical systems due to enormous safety requirements. With the aim to reduce accidents with critical personal injuries to zero, Advanced Driving Assistance Systems (ADAS) technologies such as adaptive cruise control or lane departure assistance are becoming widespread. Thus, the number of persons killed in road traffic in Germany fell by about 25 percent from 2010 ($365,000$) to $272,000$ in 2020 [Destatis 2021]. However, compared to fully Autonomous Driving, the primary goal is to support the driver with increased comfort and safety. The driver must always remain in control and observe his surroundings. In the context of automated driving, such systems are often referred to as Level 2 systems following the Society of Automotive Engineers (see Figure 1.1). Only from Level 3 „conditional automation“ the focus does change, taking temporary control of the vehicle. At the same time, a complete environmental perception is already assumed. The further the backup by a human driver is reduced, the higher the requirements grow. Especially the requirement of Level 5 to operate safely in every conceivable scenario is a key challenge since most existing ADAS systems are limited to specific operational design domains. There is a wide range of practical applications resulting from different vehicles, sensor setups, automation

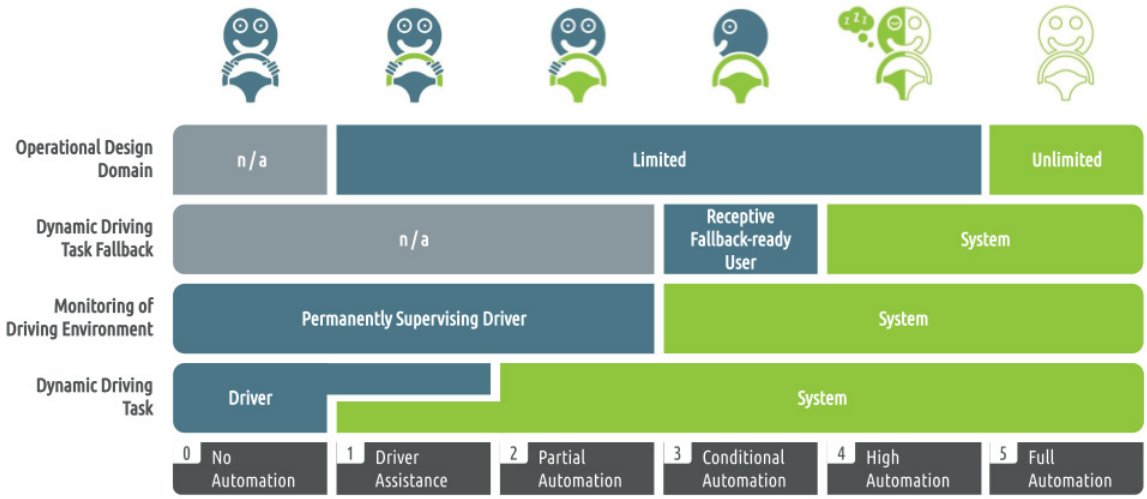


Figure 1.1: *Definition of Automation Levels according to the Society of Automotive Engineers (SAE) J3016™. The drawings in the first row indicate responsibility for 1) the control of steering with the steering wheel icon, 2) the control of acceleration/ deceleration related to the middle part of the steering wheel icon, and 3) the attention and control of the driver visualized by a smiley with and without hands. The blue and green colors indicate driver vs. vehicle responsibility, whereas white indicates no intended need for a human driver.*

levels, and objective capabilities leading to a huge field of research with worldwide interest and highest activity.

In most cases, a modular system architecture is developed according to the scheme sense-plan-act. Using multiple sensors, the surrounding environment has to be abstracted accurately in real-time to estimate trajectories for driving and finally control the actuators with low level commands, respectively. In contrast, the end to end approach aims at jointly solving sensing and planning with data in a Neural Network [Wayve 2022]. Parts of the research community, as well as commercial enterprises, focus on purely vision-based camera only approaches. However, the majority builds on combining multiple sensor modalities and redundancy. Figure 1.2 shows some prominent example setups. As a complementary alternative to cameras, lidar scanners provide direct depth measurements and deliver spatial point clouds in a 3D space of high accuracy (see Appendix A for more details). That is why lidar sensors' availability, advanced development, and quality of lidar scanners have also grown enormously. In essence, today, most test vehicles include them, and from many experts, lidar is considered an indispensable technology on the way to Level 5. Therefore, this dissertation deals with the processing of lidar point clouds to detect and classify objects in 3D from public road traffic for use in Level 3 to 5 systems.

Figure 1.2: *A sample collection of autonomous vehicles: All vehicles have very conspicuous sensor attachments to simplify perception except for the Valeo prototype. Images from [Cruise 2021; Zoox 2021; Waymo 2021; Aurora 2021; Valeo 2021; Level 5 2021].*

1.1 Motivation

Environmental perception is still at the top of the agenda, with many unresolved issues. Nevertheless, there has been tremendous progress so that human levels can already be achieved in individual subtasks, thanks to the latest deep learning methods. Breakthroughs in image processing based on Convolutional Neural Network (CNN) architectures, advanced learning, and training methods on massive data serve as the foundation for this great success. However, the characteristics of point clouds, e.g. from lidar, differ significantly from camera data: i) There is no fixed grid structure. ii) Points are unordered and important information is mainly contained in the neighboring relationship of individual points, which is not directly apparent from the data structure. iii) The point density extremely decreases at higher distances and strongly depends on aperture angles and the resolution of the sensor and object sizes.

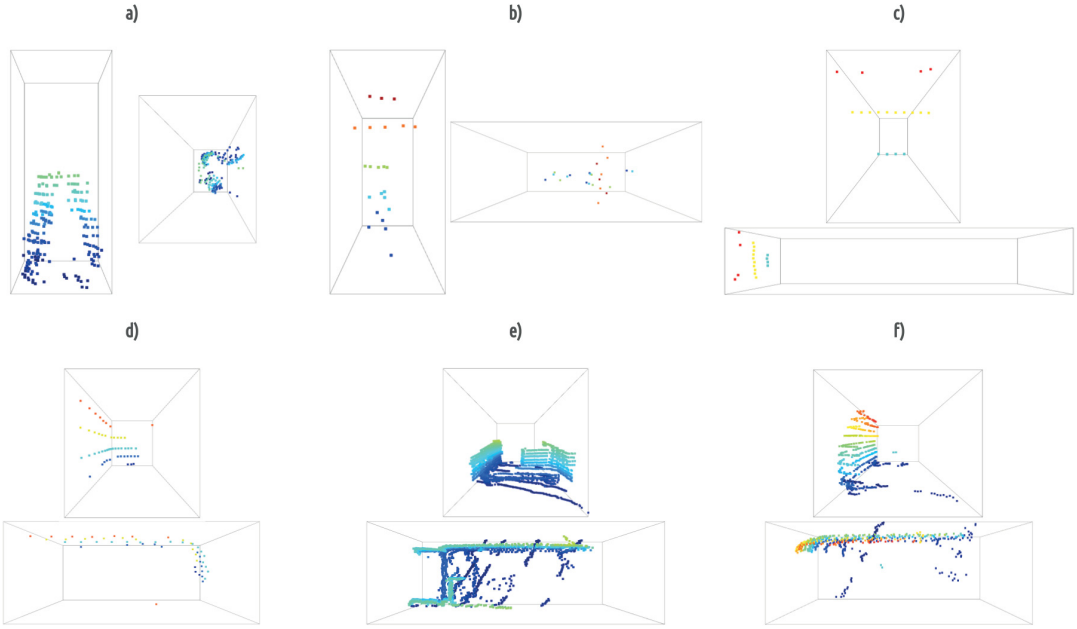


Figure 1.3: *Examples of point clouds from individual objects taken from an inner-city driving scenario, captured by an automotive-grade lidar sensor and visualized in third-person as well as top-down view: a) a nearby Pedestrian on the sidewalk, b) an oncoming Cyclist within* $30\,m$ *distance, c) a Bus ahead at a distance of* $100\,m$*, d) a Transporter parked on the other side of the road about* $40\,m$ *away, e) a Transporter with the rear doors open parked directly in front, and f) a mostly occluded Car parked in front of e) at a distance of* $20\,m$*. Points are colored by relative height inside the 3D bounding box, representing the ground truth of the objects to be recognized. Note that the sizes are scaled individually for better visibility.*

Hence, distant objects are typically mapped to only a few single points simply because of physics. However, some lidar sensors are designed for long-range applications with a small vertical field of view and detection ranges up to $250\,m$ and more. This also leads to many measurements of surfaces truncated at small heights, especially at close range. Further difficulties arise from highly dynamic occlusion, varying surface materials' reflection characteristics, and countless object poses and perspectives in which objects are visible. Some examples are depicted in Figure 1.3 to emphasize the challenges. However, robust detection of even small objects is essential for the task of self-driving.

On top, such application comes with further constraints. Efficiency and runtime play crucial roles with limited computing power and energy consumption, especially in electric vehicles. The highest reliability and robustness are necessary for safety reasons as the smallest errors at high speeds can result in fatal accidents. Secondarily, costs must also be optimized and the feasibility of huge quantities guaranteed to cover

the needs of the general public.

In addition to the aforementioned task-specific challenges, there is an endless variety of dynamic situations in public roads influenced by weather, temperature, season, day or night, regulations and conditions related to different countries, and so on. Moreover, the compliant behavior of other road participants is never completely ensured. Rare situations and corner cases can occur at any time, making reliable perception a colossal challenge. This problem also leads to challenges in data acquisition and curation. Massive amounts of well-balanced training data are needed to cover enough diversity for sufficient generalization using very Deep Neural Networks (DNNs) with sufficient capacity. Accordingly, other exciting research areas, such as transfer learning or the generation and use of artificial data, are expanding because of all these immense challenges. Still, while revolutionizing applications, the problem of object detection and classification on point clouds offers enormous potential for future research. Thus the best practices from machine learning and beyond are needed. In recent years large amounts of publications as well as huge funding and economical investments prove this.

1.2 Objective

This thesis addresses object recognition in autonomous cars, mainly using lidar sensors on public roads. Depending on the dataset, the application scenario includes urban and inner-city situations as well as highway driving at high speeds, i.e. all vehicles, pedestrians, and potential road users, whether stationary or moving, within a certain detection radius around the ego vehicle shall be detected. As seen in figure 1.2, most test carriers operate with one or more lidar sensors attached to roof mountings. In contrast, the application to a closer-to-production system shall also be analyzed within this work. The following requirements can be derived:

- **Accuracy:** Based on the four-wheel driving motion, the performance of the detector may vary depending on the location of the object. Objects in the driving area are particularly important, and poses shall be detected precisely up to a few centimeters and degrees for yaw rotation, respectively.
- **Robustness and generalization:** Highly dynamic and manifold scenarios with limited training data need to be handled. Minor noise and minor changes in the point clouds shall not disturb. Objects shall be detected according to wide ranges even though there are strongly varying point densities, occlusion,

and truncation. On the one hand, the detector shall extract robust local features from neighboring points but exploit global features with contextual information to compensate the aforementioned effects.

- **Real time capability:** In the context of autonomous vehicles, the detector must be designed so that the calculations can be carried out very quickly so that the overall runtime, including subsequent components for path planning and control, is still real-time. The goal is real-time recognition and high efficiency using as little computational resources as possible.

- **Flexibility and practicability:** Designed algorithms shall imply customizable solutions, be as sensor independent as possible, and be generally applicable to similar application scenarios. Due to the complexity and diversity, deep learning approaches are inevitable with the current state of the art.

Based on these requirements, the objective is to develop methods for the perception that constitute a basic building block of an overall application for highly automated or autonomous vehicles suitable for the real world. The main focus is on a preferably efficient and real-time 3D object detection that estimates object poses in real-world data with high accuracy and classifies them into categories such as car, truck, pedestrian, and cyclist simultaneously. On top of this, the application to a specific sensor set shall be analyzed, including the semi-automated construction of a dataset suitable for evaluation. In contrast to state-of-the-art, there is a particular focus on unambiguous recognition of the orientation of objects, as autonomous vehicles proactively plan ahead to maneuver in time at appropriate velocities based on this information. Most existing work and related metrics do not take this sub-problem into account and are therefore limited to detecting an ambiguous angle without the exact indication of the direction. Here, the current state-of-the-art should be extended by a robust estimation of object orientations.

1.3 Contributions

This work contains the following original contributions:

- One of the first deep learning based real-time multi-class object detectors on point clouds using novel layers and DNN architectures is presented [Simon et al. 2018]. Based on an extension from image processing, a novel loss in complex space is developed to address ambiguities in regression of object orientation.

The presented approach achieves very high efficiency with robust performance on single frames evaluated on a benchmark dataset.

- The following enhancements are presented to increase further the detection performance, and robustness [Simon et al. 2019]: by migrating to voxel-based processing of point clouds, purely data-driven learning occurs, in contrast to manually calculated input channels; pioneering work on feature level fusion of camera and lidar to incorporate useful semantic features from camera; modular multi-target feature tracking decoupled from plain network predictions stabilizing over time; novel scoring for a direct object to object comparison based on affine transformations as an alternative to state of the art [Simon et al. 2019; Yogamani et al. 2019] with higher efficiency and flexibility.

- A concept for efficient integration into an application-specific setup is introduced. Semi automated use of the aforementioned object detector as well as tracking in the human annotation process strongly increase the efficiency in obtaining specific training data. This approach has added value for potentially all supervised learning methods. Additionally, it has been partially incorporated into the publication of a dataset for research [Yogamani et al. 2019]. It underlines the generalization and flexibility of the approaches chosen in the context of this work.

- For directed dataset augmentation, the first generative network for conditional conversion of point clouds into synthetic image content is presented [Milz et al. 2019]. The main advantage over current state-of-the-art is the ability to improve diversity where needed while maintaining realistic geometrical and semantic constraints.

Parts of the described methods and included experiments have already been published at International Conferences. Below is a listing of the publications with a brief summary of their contents.

Publications directly related to this work

M. Simon, S. Milz, K. Amende, and H.-M. Groß (2018). „Complex-YOLO: Realtime 3D Object Detection on Point Clouds“. In: *European Conference on Computer Vision (ECCV)*, pp. 1-14

and

M. Simon and S. Milz (2018). „Echtzeit 3D Objekterkennung mit Punktwolken“. In: *34. VDI/VW Gemeinschaftstagung Fahrerassistenzsysteme und automatisiertes Fahren 2018*. VDI Verlag GmbH, pp. 125-136

To recognize objects in 3D space based on point clouds, an existing approach from image processing was adapted and extended as presented in chapter 3. A particular focus was on reliable regression of object rotations as well as efficiency using a single-stage model to ensure real-time capabilities. As of March 2022, there have already been more than 290 citations, demonstrating the notably strong influence on the research community.

M. Simon, K. Amende, A. Kraus, J. Honer, T. Sämann, H. Kaulbersch, S. Milz, and H.-M. Groß (2019). „Complexer-YOLO: Realtime 3D Object Detection and Tracking on Semantic Point Clouds“. In: *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*

The aforementioned model was improved and extended, e.g. by fusing lidar with a camera incorporating semantic texturing features and Multi Object Tracking (MOT) (chapter 4). In addition, an efficient score for comparing 3D objects was presented, which highlighted individual components depending on use case and application (section 4.3).

S. Milz, M. Simon, K. Fischer, M. Pöpperl, and H.-M. Groß (2019). „Points2Pix: 3D Point-Cloud to Image Translation using Conditional GANs“. In: *German Conference on Pattern Recognition (GCPR).* vol. 3. 1, pp. 387-400

Designed an approach for conditional translation of point clouds into camera images as presented in chapter 5. The method builds on a Generative Adversarial Network (GAN) architecture and is able to generate realistic images for point clouds of single objects, constraint with their viewpoint, and supervised with background patches in order to control the visual appearance while keeping spatial properties. The author of this work equally contributed.

Publications as co-author directly related to this work

T. Sämann, K. Amende, S. Milz, C. Witt, M. Simon, and J. Petzold (2018). „Efficient Semantic Segmentation for Visual Bird's-eye View Interpretation". In: *International Conference on Intelligent Autonomous Systems (IAS)*, pp.679-688

Improvements of the ENet model [Paszke et al. 2016] for semantic segmentation applied on fisheye camera images regarding runtime and efficiency have been published: parallelization of the arg max layer on GPU, and channel pruning.

S. Yogamani, J. Horgan, G. Sistu, P. Varley, D. O. Dea, M. Uricar, S. Milz, M. Simon, K. Amende, C. Witt, H. Rashed, S. Chennupati, S. Nayak, S. Mansoor, X. Perrotton, and P. Perez (2019). „WoodScape: A multi-task, multi-camera fisheye dataset for Autonomous Driving". In: *IEEE International Conference on Computer Vision (ICCV)*, pp.9308-9318

A dataset for automotive fisheye cameras was published, covering, among others, nine tasks like object detection, segmentation, depth estimation, or soiling detection. The proposed semi-automated annotation toolchain subsection 5.3.3 was used to create parts of the annotations. The author of this thesis has made particular contributions to the chapter related to 3D object detection.

Further publications without a direct reference to this work

V. R. Kumar, S. Milz, C. Witt, M. Simon, K. Amende, and J. Petzold (2018). „Near-field Depth Estimation using Monocular Fisheye Camera: A Semi-supervised learning approach using Sparse LIDAR Data". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Deep Vision: Beyond Supervised learning*

and

V. R. Kumar, S. Milz, C. Witt, M. Simon, K. Amende, J. Petzold, S. Yogamani, and T. Pech (2018). „Monocular fisheye camera depth estimation using sparse lidar supervision". In: *IEEE 2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2853-2858

A CNN model for near field depth estimation on monocular fisheye cameras was published semi-supervised via lidar point clouds. A major focus was on occlusion correction in point clouds resulting from varying sensor mounting positions.

K. Fischer, M. Simon, S. Milz, H.-M. Groß, and P. Mäder (2021). „StickyPillars: Robust and Efficient Feature Matching on Point Clouds using Graph Neural Networks“. In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 313-323

A DNN architecture composed of Graph Neural Network, attention mechanism, and optimal transport problem for 3D feature matching on point clouds has been presented. The model is intended to be used as middleware for registration, odometry estimation, and localization problems.

M. Reuse, M. Simon, and B. Sick (2021). „About the Ambiguity of Data Augmentation for 3D Object Detection in Autonomous Driving“. In: *IEEE International Conference on Computer Vision Workshops (ICCV)*, pp. 979-987

Experiments with data augmentation for 3D object detectors on point clouds are presented and compared for different network architectures and two datasets.

F. Poucin, A. Kraus, M. Simon (2021). „Boosting Instance Segmentation with Synthetic Data: A Study to Overcome the Limits of Real World Data Sets“. In: *IEEE International Conference on Computer Vision Workshops (ICCV)*, pp. 945-953

A simple yet effective approach to the use of synthetic data during the training of a network for the task of instance segmentation on images was presented.

K. Fischer, M. Simon, S. Milz, and P. Mäder (2022). „StickyLocalization: Robust End-to-End Relocalization on Point Clouds using Graph Neural Networks“. In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 2962-2971

A novel end-to-end Graph Neural Network-based solution for relocalization and loop closing on point clouds was presented. Here, a descriptor-based feature pre-selection and a distance-based matching loss were introduced.

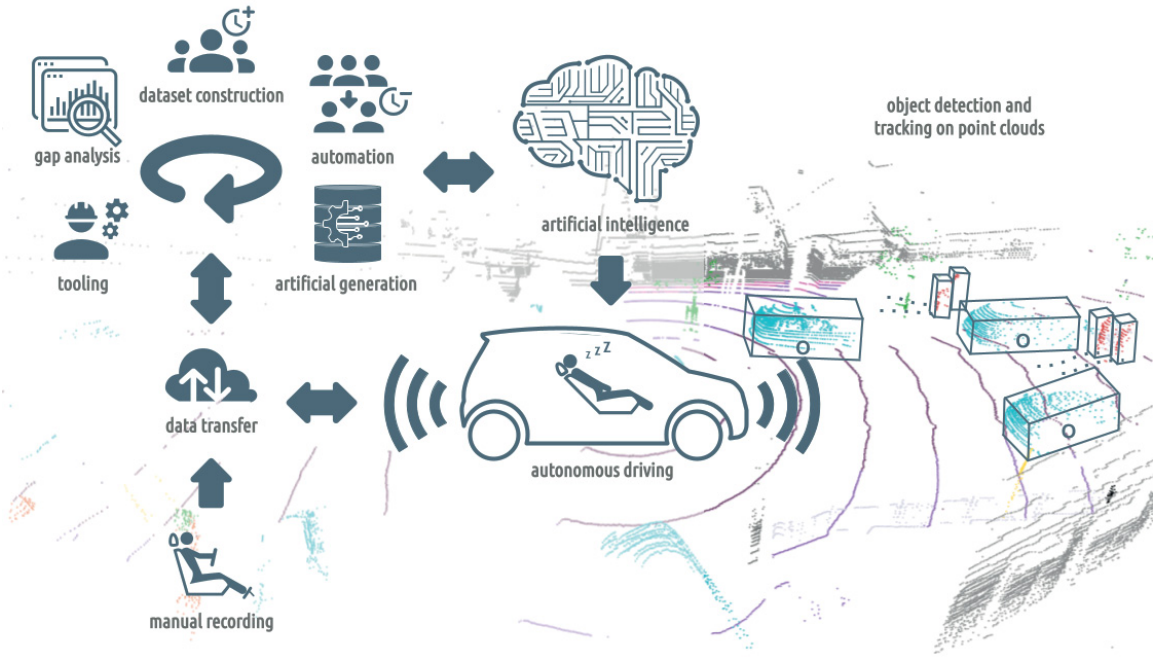


Figure 1.4: *An overview of this thesis at a glance: The main focus is on object detection and tracking on point clouds based on artificial intelligence. Furthermore, a concept for integration into an autonomous vehicle is presented, including the acquisition and evaluation of application-specific data.*

1.4 Outline

This thesis is structured as follows, and an overview is given in Figure 1.4.

In chapter 2, relevant theoretical background with fundamental deep learning techniques and concepts are reviewed. Chapter 3 and 4 provide a detailed explanation of the proposed object detector extended with camera features as well as modular multi-target feature tracking and presents the evaluation results of the approaches on a benchmark dataset. To demonstrate that the model is also suitable for an application-specific setup, chapter 5 examines a concept for integration, including the acquisition of a required dataset. Furthermore, the outputs are evaluated and discussed in comparison to former results. To overcome the limits of datasets and their costly acquisition, chapter 5 also contains the proposed method for conditional point cloud to image translation to generate systematic synthetic data. Finally, chapter 6 summarizes the results of this work, and depicts the possibilities for future work. The appendix contains more details on lidar technology, recent related datasets, and more insights into the evaluation metrics used to validate the methods proposed in this thesis.

Chapter 2

Background

The following chapter describes necessary foundations that are relevant for this work. In section 2.1, the basics of deep learning for computer vision are described in a nutshell. Section 2.2 introduces the fundamentals of deep generative models. This is followed by a review of state of the art for object detection in computer vision in section 2.3. Lastly, section 2.4 presents the basics of multi-object tracking.

2.1 Deep Neural Networks

Basic knowledge of Deep Neural Networks (DNNs) is necessary for understanding the dissertation. Therefore, the core concept is briefly presented in this section.

A DNN is a powerful functional approximator that can define arbitrary differentiable functions to predict outputs according to provided inputs. These computational models are inspired by the human brain named for the depth of layers nested together in a structural way and optimized with respect to any differentiable loss function using Stochastic Gradient Descent (SGD). As a result of a variety of architectures, layers, and advanced methods for optimization, as well as the possibility to efficiently process huge datasets in a machine learning process, DNN models are state of the art in many domains. The following subsections describe the essential concepts of DNN model types and techniques related to this work.

2.1.1 Neural Networks

The term Neural Network (NN) is linked to information processing in biological systems, as NNs are composed of artificial neurons, conceptually derived from biological neurons. Each neuron has inputs, such as features from external data or outputs from other neurons, and produces a single output sent to multiple connected neurons. In

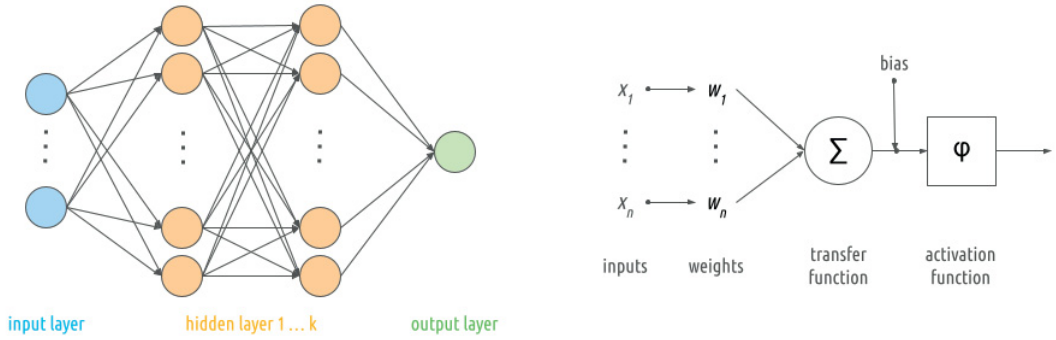


Figure 2.1: *Left: Simplified arrangement of a synthetic multi-layer Neural Network containing multiple connected neurons. Neurons in one layer are not connected to each other, only to all neurons in the previous layer. Thus, efficient matrix multiplication can be used to calculate the activation of all neurons in one layer. Right: Diagram for a single synthetic neuron inspired by the human brain. All inputs x are multiplied with a weight w, summed up, and an additive bias parameter is added. Finally, an activation function φ generates the output signal.*

the simplest case, it has an input layer, output layer, and one or more hidden layers with data flow only in a forward direction, as visualized in Figure 2.1. Here, neurons in one layer are only connected to all neurons in the previous layer, sometimes also called fully connected feed-forward NN or Multi Layer Perceptron (MLP). The precise mathematical form for a layer is as follows:

$$f(x) = \varphi(Wx + b) \tag{2.1}$$

where a NN can be described as a repeated matrix multiplication with weight matrix W combined with element-wise nonlinear activation function φ and additive bias b. All network parameters are obtained efficiently with the technique of error backpropagation.

An activation function is a mathematical function added into a Neural Network in order to enable the network to learn complex patterns. In this way, Neural Networks are capable to learn the non linearity that is mostly required to solve real-world problems. Activation functions used throughout this thesis are Rectified Linear Unit (ReLu), leaky ReLu [Maas et al. 2013] and parameterized ReLu [He et al. 2015a], as visualized in Figure 2.2. Those functions are common activation functions with piecewise linearity as well as they are very easy to compute. Unlike ReLu, the variants also allow to let negative values pass through the network. The functions are defined by the following equations:

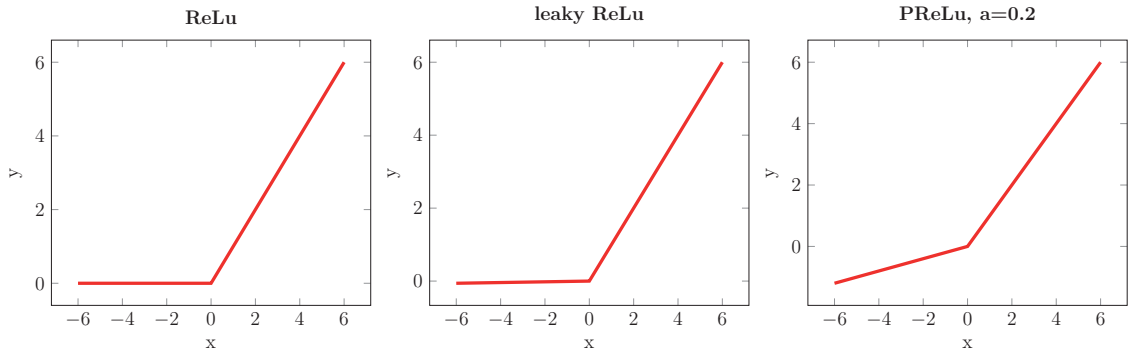


Figure 2.2: *Plots of the commonly used ReLu activation functions.*

$$ReLu(x) = max(0, x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.2}$$

$$leakyReLu(x) = max(0, x) + 0.01 * min(0, x) = \begin{cases} x & x \geq 0 \\ 0.01 * x & x < 0 \end{cases} \tag{2.3}$$

$$PReLu(x) = max(0, x) + a * min(0, x) = \begin{cases} x & x \geq 0 \\ a * x & x < 0 \end{cases} \tag{2.4}$$

2.1.2 Convolutional Neural Networks

The methods developed in this thesis primarily use a Convolutional Neural Network (CNN). Therefore this subsection briefly reviews the basic building blocks and concepts.

For high dimensional inputs with spatial topology, like images or audio data, fully connected NN are poorly suited since they have a huge number of parameters that are optimized individually. Thus, these models tend to overfit and ignore the strong spatial correlation included in the data. Therefore, specific CNN architectures were designed to address this issue. As seen in Figure 2.3, a typical CNN is constructed of multiple stacked convolutional layers followed by some sort of pooling to further compress the size of local feature maps. In this way, neurons in a CNN are only connected to other neurons with respect to spatial relationships while sharing parameters with a massive decrease in their number, which also helps to address overfitting.

At the core of a CNN are convolutional layers that convolve inputs with a set of filters, as also displayed in Figure 2.3. Here the local neighborhood of the input is processed in a sliding window manner parallelized over parallel GPU cores. Consequently, the filter kernels are applied over the entire input given additional stride

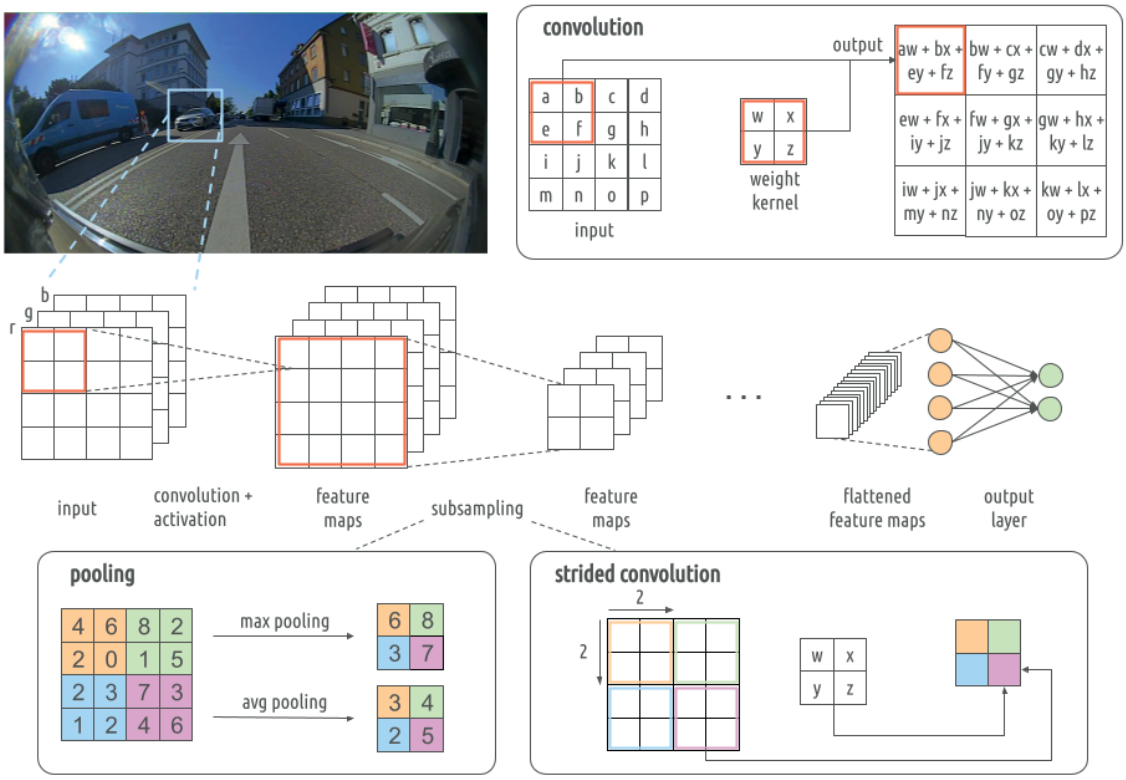


Figure 2.3: *Typical structure of a CNN for image classification: The network consists of alternating layers of convolutional filter kernels and subsampling, such as pooling or strided convolutions, to extract high dimensional feature maps. In this example, a fully connected output layer predicts classification scores based on the last feature maps.*

and padding parameters. Similarly, pooling functions like maximum or average are utilized in order to combine local features into a more compact feature representation with respect to the underlying activation. Both elements can be flexibly combined using different sets of parameters.

2.1.3 Channel Pruning

Since efficiency is one of the main requirements of this thesis, channel pruning is used to optimize a network for a fusion of point clouds with semantic segmentation from cameras mentioned in chapter 4. Therefore this section explains the necessary basics.

Although deep CNN architectures primarily operate with shared weights, they require an extensive amount of computations depending on the number of layers and input resolution. A common approach for structured simplification of such models is compression with channel pruning, where the feature map width gets reduced in order

to shrink the overall network [Wen et al. 2016]. Here, the number of input channels gets reduced by using redundancy while the output is maintained. [He, Zhang, and Sun 2017] proposed an iterative two-step algorithm to prune each layer of common CNN architectures like deep residual networks [He et al. 2016]. First, channels are selected for pruning according to the least absolute shrinkage and selection operator regression [Tibshirani 1996]. Thereby, the goal is to retain as much information as possible with fewer channels. Then, the output feature maps are reconstructed with linear least squares. Corresponding channels and filters are removed once the channels are selected and pruned. Furthermore, the filters in the previous layer used to calculate the pruned features can also be removed. Overall, there is a trade-off between acceleration and increase of error introduced by the degree of channel pruning.

2.2 Deep Generative Models

In recent years, there has been a growing interest in NN based methods for Domain Translation, which will be investigated in chapter 5 to provide an approach for directed data generation. Therefore, this section classifies the related work and presents the underlying state of the art on deep generative models.

Domain Translation can be categorized into the field of Domain Adaptation as a branch of Transfer Learning. An overview is given in Figure 2.4. Given a source domain D_S and learning task T_S, and a target domain D_T and learning task T_T, Transfer Learning aims to help improve the learning of the target predictive function $f_T(\bullet)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$ [Pan and Yang 2009; Lin and Jung 2017]. A domain D consists of a feature space X and a marginal probability distribution $P(X)$, where $X = \{x_1, \ldots, x_n\} \in X$ and a task can be denoted as $T = \{Y, f(\bullet)\}$, where Y is the label space and $f(\bullet)$ the learned objective predictive function [Pan and Yang 2009].

Domain Adaptation can be referred to as if the feature spaces are the same, while the marginal probability distributions of source and target data are different. The most dominating examples are Variational Autoencoder [Kingma and Welling 2013] and a Generative Adversarial Network (GAN) [Goodfellow et al. 2014]. For instance, [Chen et al. 2018; Inoue et al. 2018] both deal with the task of object detection for different domains such as image style, illumination, or object appearance, where training and test data come from different distributions. Adversarial adaptation

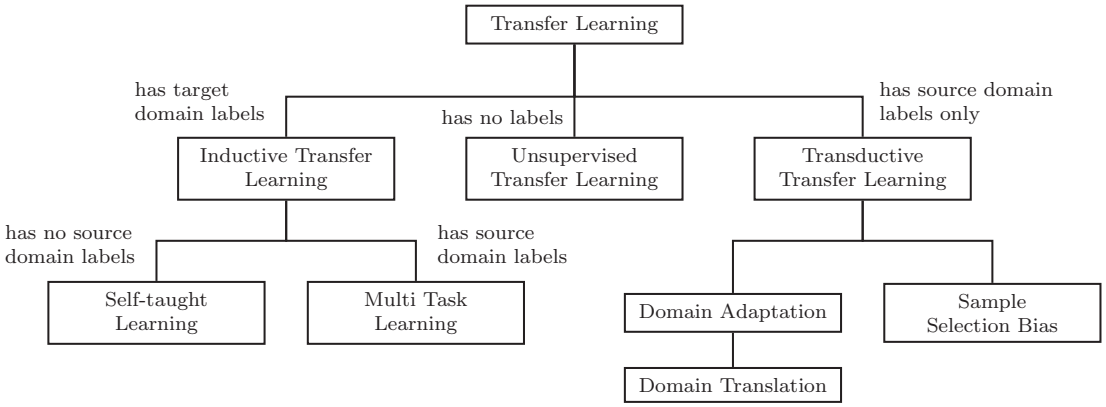


Figure 2.4: *Classification of Domain Translation used to generate synthetic training data from a learned data distribution as a subbranch of transfer learning. Figure taken and modified from [Pan and Yang 2009].*

either focuses on discovering domain invariant representations or mapping between unpaired domains.

Finally, Domain Translation, sometimes also referred to as domain mapping, deals with the problem of finding a meaningful correspondence between two domains to get such mapping. The following subsections describe the basics of GANs as well as Image to Image Translation. More in-depth information related to Transfer Learning can be found in [Pan and Yang 2009; Sun et al. 2015; Wilson and Cook 2018].

2.2.1 Generative Adversarial Networks

Since GAN architectures are state of the art for Domain Translation, this subsection briefly introduces the core concepts.

The aim of GANs is to generate new samples based on a learned variational distribution in data. Here, these networks use an adversarial training process inspired by game theory to learn such distributions. Initially proposed by [Goodfellow et al. 2014], GANs consist of two competing NNs trained in a minimax game. A generator G is trained to generate realistic samples and additionally takes random noise as input. Here, the aim is to learn a distribution in the target domain, i.e. to generate diverse outputs. On the other hand, a discriminator D takes the generated samples as input and tries to distinguish real from fake samples. Therefore, the generator needs to be able to learn the distribution of the training data, and the discriminator needs to be able to classify. Despite tremendous results, GAN variants usually suffer from several problems. Often the generator keeps generating the same type of style to fake the discriminator, called mode collapse or training instability. Additionally, the

random input noise can be suppressed, resulting in a lack of diversity or a possible lack of regularization with the target domain [Lee et al. 2018]. The original GAN architecture has no control over the generated outputs due to the random input noise. Therefore, [Mirza and Osindero 2014] introduced ConditionalGANs, where additional data is fed as a condition onto both the generator and discriminator. Given this setup, [Regmi and Borji 2018] were able to generate synthetic data, [Hoffman et al. 2018] developed a model for Domain Adaptation, and [Gonzalez-Garcia et al. 2018] worked on cross-domain disentanglement based on Image to Image Translation.

2.2.2 Image to Image Translation

The developed method from chapter 5 examines a method for automatically generating image data from point clouds for training while trying to keep specific properties and attributes. Therefore, the underlying fundamentals are explained as follows.

Compared to image generation, where images are directly generated from random noise, image translation generates an image from another existing image, modifying specific attributes like meaningful features or style. Hence, it can be formulated as a disentanglement problem, separating the content which needs to be preserved across domains from appearance to be changed. In [Isola et al. 2017], a paired image to image translation maps an image from the input to the output domain, requiring pairs of corresponding images in different domains for training to formulate a reconstruction loss. The structure is penalized at the scale of image patches according to [Li and Wand 2016]. In general, a mapping function G called generator, based on a condition c and a random noise input z generates the output image y:

$$G : \{c, z\} \rightarrow y \quad y \in \mathbb{R}^{w \times h \times 3} \quad c = \begin{cases} \in \mathbb{R} & \rightarrow \text{label to image} \\ \in \mathbb{R}^t & \rightarrow \text{text to image} \\ \in \mathbb{R}^{w \times h \times 3} & \rightarrow \text{image to image} \end{cases} \tag{2.5}$$

A competing discriminator D tries to distinguish between real images and created fake ones. Derived from this setup, the loss can be described as:

$$L_{cGAN}(G, D) = \mathbb{E}_{c,y}\{\log(D(c, y))\} + \mathbb{E}_{c,z}\{\log(1 - D(c, G(c, z))\} \tag{2.6}$$

In a follow-up, [Wang et al. 2018] increased the resolution of the generated output with a coarse to fine generator and mapped among multiple domains [Choi et al. 2018].

In contrast, unsupervised approaches for unpaired settings were proposed based on cycle consistency [Zhu, Park, et al. 2017; Kim et al. 2017; Yi et al. 2017]. Here,

the reconstruction of the inverse mapping from the output back to the input domain is checked. Other autoencoder-based models like [Liu et al. 2017; Huang, Liu, Belongie, et al. 2018] learn a shared latent representation using two encoders, two generators, and two discriminators with shared weights of the encoders. Furthermore, [Tripathy et al. 2018] trained with paired and unpaired settings simultaneously and [Zhu, Zhang, et al. 2017; Almahairi et al. 2018; Lee et al. 2018; Liu et al. 2019] introduced extensions for multiple domains as well as multi-modal synthesis. Additionally, detailed improvements and extensions were presented in [Tang et al. 2019; Zhang, Pfister, et al. 2019; Gokaslan et al. 2018; Wu et al. 2019; Liang et al. 2018].

2.3 Object Detection in Computer Vision

In chapter 3, a model for 3D object detection on point clouds is developed based on recent advances in computer vision. Therefore this section very briefly reviews the related state of the art.

The term object detection refers to a challenging fundamental problem in computer vision. For several decades there has been an active area of research in order to localize and classify instances of objects according to given categories. An overview of 2D object detection is given in Figure 2.5, starting from traditional methods shown in green, up to more recent approaches based on DNNs. Milestones up to the time of the work of this thesis are visualized in orange, and more recent works are shown in blue, after the dashed line.

First traditional methods like [Viola and Jones 2001; Viola and Jones 2004] are based on handcrafted features like [Lowe 1999] with sophisticated feature representations on sliding windows. For a long time the main bottleneck was limited computational resources. After AlexNet [Krizhevsky et al. 2012] became public in 2012, CNNs repeatedly outperformed all previous results aided by superior computers with parallel processing on GPUs and increasingly large datasets. During this period of rapid evolution, various architectures were developed with increasing depth and enhanced methods for training and regularization. [Simonyan and Zisserman 2015] proposed to increase the number of layers with multiple very small 3×3 convolution filters called VGG. Inspired by the breakthrough in image classification, the family of region-based CNNs [Girshick et al. 2014; Girshick 2015; Ren et al. 2015] emerged with a Region Proposal Network (RPN). Here, region bounds and scores at each location on a regular grid are simultaneously predicted using multi scale anchors. This is implemented as a sliding window with $n \times n$ convolution followed by two

Figure 2.5: *Timeline of important milestones in 2D object detection (traditional methods in green, related methods in orange, and methods upon this work in blue, additionally separated by the dashed line): SIFT [Lowe 1999], Cascades [Viola and Jones 2001], HOG [Dalal and Triggs 2005], SURF [Bay et al. 2006], AlexNet [Krizhevsky et al. 2012], RCNN [Girshick et al. 2014], Fast RCNN [Girshick 2015], Faster RCNN [Ren et al. 2015], 3DOP [Chen et al. 2015], VGG [Simonyan and Zisserman 2015], GoogLeNet [Szegedy et al. 2015], YOLO [Redmon et al. 2016], ResNet [He et al. 2016], Mono3D [Chen et al. 2016], SSD [Liu et al. 2016], MS-CNN [Cai et al. 2016], YOLOv2 [Redmon and Farhadi 2017], DenseNet [Huang et al. 2017], Mask RCNN [He, Gkioxari, et al. 2017], MobileNet [Howard et al. 2017], ResNeXt [Xie et al. 2017], RetinaNet [Lin, Goyal, et al. 2017], YOLOv3 [Redmon and Farhadi 2018], MobileNetv2 [Sandler et al. 2018], Cascade RCNN [Cai and Vasconcelos 2018], CornerNet [Law and Deng 2018], Group Norm [Wu and He 2018], CenterNet [Duan et al. 2019], Res2Net [Gao et al. 2019], Cascade RPN [Vu et al. 2019], FreeAnchor [Zhang, Wan, et al. 2019], Cascade Mask RCNN [Cai and Vasconcelos 2019], EfficientDet [Tan et al. 2020], DETR [Carion et al. 2020], Deformable DETR [Zhu et al. 2020], Dynamic RCNN [Zhang et al. 2020], Gen. Focal Loss [Li et al. 2020], CentripetalNet [Dong et al. 2020], YOLOF [Chen et al. 2021], PVT [Wang, Xie, et al. 2021], Scaled YOLOv4 [Wang, Bochkovskiy, et al. 2021], Swin Transformer [Liu et al. 2021].*

sibling fully connected 1×1 convolution layers, one for the regression parameters and one for the classification scores, respectively. In parallel, You Only Look Once (YOLO) [Redmon et al. 2016] implemented the regression of spatially separated 2D bounding boxes associated with class probabilities in a single network. Therefore a CNN generates downsampled features over the whole image and predicts all related regression targets for each cell directly from a last fully connected layer. The unified multi-scale network from [Cai et al. 2016] tried to cover objects at many scales using CNNs to exploit feature maps of several resolutions. Here, the aim is to have variable sizes for the receptive fields to match various object sizes. Similarly, [Liu et al. 2016] discretized a set of default anchors over different aspect ratios and scales and predicted scores for the presence of objects for each default box as well as offsets for fine-grained, precise predictions. Furthermore, [He et al. 2016] introduced a learning framework with shortcut connections between pairs of convolutional layers, namely ResNet. Inspired by updates from prior work, [Redmon and Farhadi 2017] proposed YOLOv2, sometimes also called YOLO9000, with anchors similar to Faster RCNN [Ren et al. 2015] and multi-scale and joint training using a hierarchical view of object classification to make use of classification datasets. In a follow-up, the concepts of residual learning from [He et al. 2016], and multi-scale feature pyramids from [Lin, Dollár, et al. 2017], were integrated to form YOLOv3 [Redmon and Farhadi 2018]. Other methods aim at higher efficiency, e.g. [Howard et al. 2017] with depthwise separable convolution for efficiency, [Sandler et al. 2018] with an inverted residual block with the linear bottleneck to further reduce computations and [Xie et al. 2017] with a simpler and more efficient convolutional architecture based on inception-like blocks from [Szegedy et al. 2015]. RetinaNet [Lin, Goyal, et al. 2017] tackles the imbalance problem of foreground and background during object detection with a reshaped cross-entropy loss, called focal loss. Unlike most other approaches where batch normalization [Ioffe and Szegedy 2015] is used, [Wu and He 2018] propose group normalization. Here feature channels are divided into groups. Within each group the mean and variance are computed and used for normalization. Instead of directly regressing bounding boxes, [Law and Deng 2018] formulates them as a pair of key points without anchors for dense prediction.

More recent work, which could not be considered for the relevant problems after the publications [Simon et al. 2018; Simon et al. 2018], can be structured as follows. The dashed line in Figure 2.5 indicates this split. Several methods like [Cai and Vasconcelos 2018; Vu et al. 2019; Cai and Vasconcelos 2019; Zhang et al. 2020] operate in a multi-stage fashion, first predicting coarse object proposals followed by a sequential

refinement stage. Following [Law and Deng 2018], in [Duan et al. 2019] objects are modeled as a set of key points using a multi-task loss for all other regression targets to form bounding boxes. [Dong et al. 2020] introduced a mechanism to shift predicted key points of object corners and to perform corner matching based on both the predicted and shifted corners. As alternatives, [Li et al. 2020] presented a loss for joint representation of classification and localization parameters, while [Zhang, Wan, et al. 2019] used an object anchor matching for flexible anchor assignment and learning. In contrast, multiple approaches are proposed for better efficiency based on multi-scale features [Gao et al. 2019; Tan et al. 2020; Wang, Bochkovskiy, et al. 2021]. In addition, [Chen et al. 2021] proposed a dilated encoder and uniform matching based on k nearest neighbors to solve the imbalance of positive anchors. Recently transformers [Vaswani et al. 2017] have also found their way into computer vision. Here, [Carion et al. 2020] construct a transformer encoder-decoder architecture with parallel feature decoding and formulate a set-based global loss with bipartite matching. In a follow-up [Zhu et al. 2020], the attention modules were restricted to key sampling points around a reference point. Additionally, [Liu et al. 2021] and [Wang, Xie, et al. 2021] proposed transformer-based backbones with patch wise feature embedding.

2.4 Multi Object Tracking

In chapter 4, a Multi Object Tracking (MOT) concept will be added to the developed single frame object detector on point clouds from chapter 3. Therefore this section reviews the related methods used for MOT.

The major goals of MOT, or Multi Target Tracking (MTT), are to detect and localize object instances of interest under the presence of noise-corrupted data from sensors and unknown origin while filtering out sensor noise and assigning the same instances to unique target tracks over a certain period of time. Due to the complexity of this task and its usefulness for real-world applications, MOT is one of the most active research areas. There are various approaches with different strategies. On the one hand, sequence learning methods were proposed following the great success of DNNs. Unlike MLPs, a Recurrent Neural Network (RNN) makes use of previous outputs to be reused as inputs while having hidden states. Here, historical information is processed by shared weights across time. Drawbacks are slower processing and challenges with the training of particularly deep networks due to vanishing or exploding gradients. Therefore, Gated Recurrent Unit (GRU) [Chung et al. 2014] and Long Short Term Memory (LSTM) [Hochreiter and Schmidhuber 1997] added

layers as gates which feature subtasks such as a) update: how much matters the past now? b) relevance: keep or drop previous information? c) forget: delete a cell?, and d) output: how much to output to a cell? Training these networks however remains challenging, especially for the complex task of MOT. Hence, there are only a few approaches such as [Ning et al. 2017] or [Kahou et al. 2017].

On the other hand, the task of MOT can be solved in two phases. First, objects of interest are detected by an algorithm, and second, identical objects are associated and modeled over time, commonly based on Bayesian statistics. Compared to end-to-end learned methods, this approach offers the opportunity for step-wise abstraction and the incorporation of rules or assumptions based on physics and expert knowledge. Natural processes like birth, death, or the motion of objects and their states can be modeled independently via probabilities using a sequence of measurements. Thus, tracking by detection has been widely explored and can be categorized into online and offline tracking. In contrast to online methods, offline tracking makes use of global information about the future, e.g. like [Lee et al. 2016; Frossard and Urtasun 2018], which makes them unsuitable for real-time applications. According to [Vo et al. 2015], the most popular methods can be clustered into the Joint Probabilistic Data Association Filter (JPDAF) [Bar-Shalom et al. 2011], Multiple Hypothesis Tracking (MHT) [Blackman and Popoli 1999] and Random Finite Set (RFS) based filters [Mahler 2014]. In many cases, a Kalman filter [Kalman 1960] is used for single state estimation as a closed-form solution to Bayes recursion for linear cases. Similarly, several filter approximations have been proposed to solve nonlinear problems [Julier and Uhlmann 1997; Doucet et al. 2000; Wan and Van Der Merwe 2000; Ristic et al. 2003].

As an emerging paradigm, the MOT problem can be formulated as a dynamic multi-target state estimation, in which the multi-target state is modeled as RFS. Following this approach, [Reuter et al. 2014] focuses on highly efficient approximations of multi-object states based on a Labeled Multi-Bernoulli Random Finite Set (LMB RFS). A coordinated turn model solves the here considered problem of tracking maneuvering targets, according to [Roth et al. 2014]. The basics of LMB RFS are briefly described in the following subsection. At the same time, subsection 2.4.2 reviews the basics of an Unscented Kalman Filter (UKF) that is used in this work to estimate object states with the nonlinear coordinated turn motion. Finally, a brief review of the coordinated turn model can be found in subsection 2.4.3.

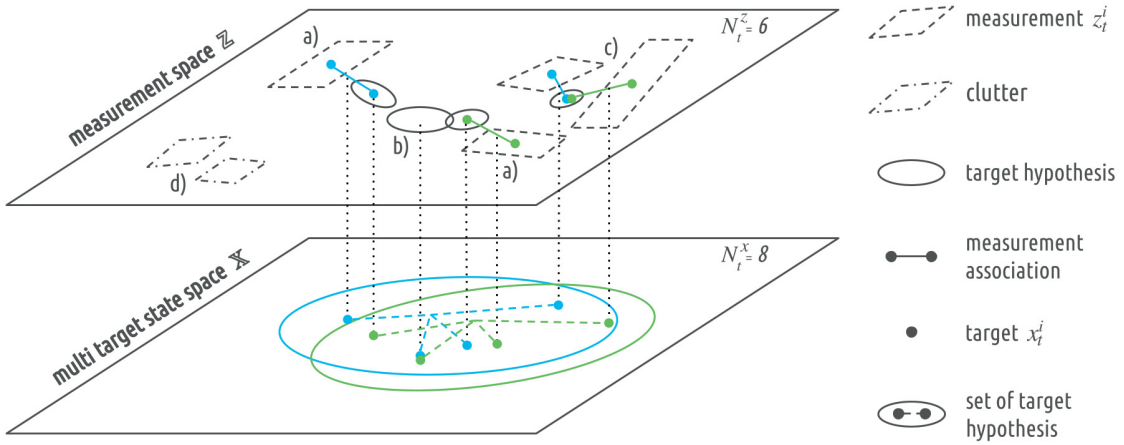


Figure 2.6: *Simplified illustration of the basics of multi-object tracking via Labeled Multi-Bernoulli Random Finite Set (LMB RFS): Several sets of hypotheses are generated, maintained or rejected at each time step t in order to approximate the multi-target distribution based on the individual measurements and modeled associations. Here, only two sets are visualized in blue and green to highlight the different cases: a) a measurement where a target hypothesis can be associated, b) a target hypothesis without an associated measurement, c) measurements with ambiguous association, and d) clutter from noisy measurements.*

2.4.1 Multi Object Tracking via Labeled Multi- Bernoulli Random Finite Sets

Since efficiency and real-time capability are major requirements in this work, the aforementioned developed tracking concept is based on Labeled Multi-Bernoulli Random Finite Set (LMB RFS) [Reuter et al. 2014]. The basic idea is illustrated in Figure 2.6 with the help of a small example, and relevant equations are described as follows, reproduced from [Simon et al. 2019].

In order to approximate the underlying Bayes multi-target filter, [Reuter et al. 2014] proposed a Labeled Multi-Bernoulli Filter to propagate the posterior density of the multi-target state recursively in time.

Let the state x_t^i of the ith target at discrete time t be a random variable. The set of all targets at time step t is a subset of the state space $\mathbb{X}$ denoted by:

$$\mathbf{X}_t = \left\{x_t^i\right\}_{i=1}^{N_t^x} \subset \mathbb{X} \tag{2.7}$$

In contrast, the set cardinality $N_t^x = |\mathbf{X}_t|$ is a discrete random variable at time t. The set of all measurements at time t is again modeled as a random set with set

cardinality $N_t^z = |\mathbf{Z}_t|$ and denoted by:

$$\mathbf{Z}_t = \left\{ z_t^i \right\}_{i=1}^{N_t^z} \subset \mathbb{Z} \tag{2.8}$$

Individual measurements z_t^i are either target-generated or clutter and the true origin is assumed unknown. Next, the set of all measurements including the time step t is:

$$\mathbf{Z}^t = \bigcup_{\tau=1}^{t} \mathbf{Z}_\tau \tag{2.9}$$

The particular choice of indices is arbitrary, as both the above sets are without order. Targets and measurements are modeled as LMB RFS as proposed in [Reuter et al. 2014; Bryant et al. 2018]. A Bernoulli RFS is a set that is either empty with probability $1 - r$ or contains a single element. Here, the probability density may be written as:

$$\pi(X) = \begin{cases} 1 - r, & \text{if } X = \emptyset, \\ r\, p(x), & \text{if } X = \{x\} \end{cases} \tag{2.10}$$

with $p(\cdot)$ a spatial distribution on $\mathbb{X}$, as described in [Reuter et al. 2014]. A Multi-Bernoulli RFS is then the union of independent Bernoulli RFSs, i.e. $X_{\mathrm{MB}} = \bigcup_i X_{\mathrm{B}}^{(i)}$. In essence, a Multi-Bernoulli RFS is well-defined by the parameters $\{r^{(i)}, p^{(i)}\}_i$.

Labeled RFSs allow the estimation of both the targets state and their individual trajectories. For this reason the target state is extended by a label $l \in \mathbb{L}$, i.e. each single target state is given by $\mathbf{x} = (x, l)$ and in turn the multi-target state $\mathbf{X}$ lives on the product space $\mathbb{X} \times \mathbb{L}$ with $\mathbb{L}$ a discrete space. Note that this definition does not enforce the labels l to be distinct. [Vo and Vo 2011] introduced the so-called distinct label indicator:

$$\Delta(\mathbf{X}) := \delta_{|\mathbf{X}|}(|\mathcal{L}(\mathbf{X})|) \tag{2.11}$$

that enforces the cardinality of $\mathbf{X}$ to be identical to the cardinality of the projection $\mathcal{L}(\mathbf{X}) = \{\mathcal{L}(\mathbf{x}) : x \in \mathbf{X}\}$, $\mathcal{L}(\mathbf{x}) = l$. Together with Equation 2.10, it follows that the probability density of a LMB RFS is well-defined by the parameter set $\{r^{(l)}, p^{(l)}\}_{l \in \mathbb{L}}$ and the cardinality distribution yields:

$$\rho(n) = \prod_{i \in \mathbb{L}} (1 - r^{(i)}) \sum_{L \in \mathcal{F}_n(\mathbb{L})} \prod_{l \in L} \frac{r^{(l)}}{1 - r^{(l)}} \tag{2.12}$$

with $\mathcal{F}_n(\mathbb{L})$ the set of all subsets of $\mathbb{L}$ containing n elements. The core objective of the multi-target tracking is to approximate the multi-target distribution $f_{t|t}(\mathbf{X}_t|\mathbf{Z}^t)$ in each time step t. This is achieved with the multi-target Bayes filter:

$$f_{t|t}(\mathbf{X}_t|\mathbf{Z}^t) = \frac{f_t(\mathbf{Z}_t|\mathbf{X}_t) f_{t|t-1}(\mathbf{X}_t|\mathbf{Z}^{t-1})}{\int f_t(\mathbf{Z}_t|\mathbf{X}_t) f_{t|t-1}(\mathbf{X}_t|\mathbf{Z}^{t-1}) \delta\mathbf{X}_t} \tag{2.13}$$

and the Chapman-Kolmogorov prediction:

$$f_{t+1|t}(\mathbf{X}_{t+1}|\mathbf{Z}^t) = \int f_{t+1|t}(\mathbf{X}_{t+1}|\mathbf{X}_t) f_{t|t}(\mathbf{X}_t|\mathbf{Z}^t) \delta\mathbf{X}_t \tag{2.14}$$

with $f_t(\mathbf{Z}_t|\mathbf{X}_t)$ the multi-target measurement set density and $f_{t+1|t}(\mathbf{X}_{t+1}|\mathbf{X}_t)$ the multi-target transition density.

2.4.2 Unscented Kalman Filter

In this work, the state of an object is modeled as a normal distribution. However, since the coordinated turn motion model is nonlinear, the UKF [Julier and Uhlmann 1997] is used to approximate the transition in the Bayes theorem efficiently.

An UKF is a recursive state estimator that takes the system state as a probability distribution. The state x_t can contain non-observable variables that are estimated by the filter based on normal distributions. For this purpose, the filter establishes a connection between the state estimation $bel(x_t)$, the measurement data z_t, and the control u_t via a system model. The current state estimate $bel(x_t) = p(x_t|z_{1:t}, u_{1:t})$ is recursively derived from the previous estimate $bel(x_{t-1})$ and the current observation z_t. The term u_t can be ignored because the system does not know the control input. While the initial Kalman filter [Kalman 1960; Barker et al. 1995] is only a closed-form solution for linear systems with additive Gaussian noise, the UKF approximates the n dimensional Gaussian distribution using a set of specific weighted samples referred to as sigma points $\{\mathcal{X}_i\}_{i=0}^{2n}$ and the actual mean $\hat{\mathbf{x}}$. The Unscented Transformation therefore captures and transforms mean and covariance using the weighted sigma points. The sigma points and corresponding weights can be denoted as:

$$\mathcal{X}_0 = \hat{\mathbf{x}} \tag{2.15}$$

$$\mathcal{X}_i = \hat{\mathbf{x}} + \left(\sqrt{(n+\lambda)\mathbf{P}}\right)_i, \quad i = 1, ..., n \tag{2.16}$$

$$\mathcal{X}_i = \hat{\mathbf{x}} - \left(\sqrt{(n+\lambda)\mathbf{P}}\right)_{i-n}, \quad i = n+1, ..., 2n \tag{2.17}$$

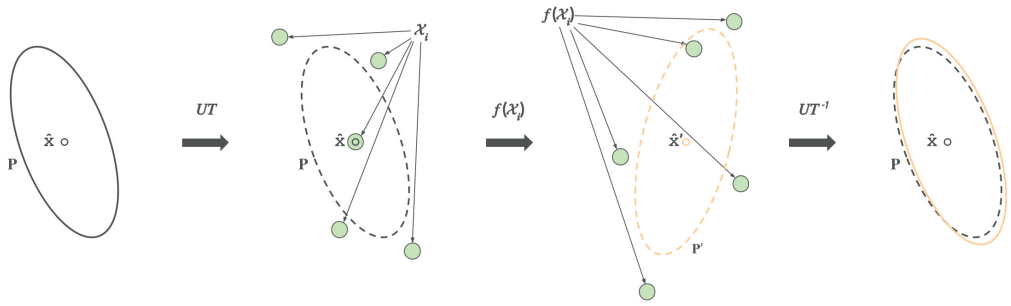


Figure 2.7: *A simple sample for mean and covariance propagation based on the Unscented Transformation (UT) in a 2-dimensional system: Only 5 sigma points $\mathcal{X}_i$ are required to estimate the underlying Gaussian distribution accurately. Further details are shortly described in subsection 2.4.2. Figure taken and modified from [Wan and Van Der Merwe 2000].*

where $\lambda = \alpha^2(n + \kappa) - n$, such that α and β describe the spread of the sigma points with respect to the mean. Every sigma point has two weights:

$$W_0^m = \frac{\lambda}{n + \lambda} \tag{2.18}$$

$$W_0^c = \frac{\lambda}{n + \lambda} + (1 - \alpha^2 + \beta) \tag{2.19}$$

$$W_i^m = W_i^c = \frac{1}{2(n + \lambda)}, \quad i = 1, ..., 2n \tag{2.20}$$

where W_i^m is used to calculate the mean and W_i^c to reconstruct the covariance. Based on these sigma points, a nonlinear function $f(\cdot)$ is applied to extract the reproduced mean $\hat{\mathbf{x}}'$ and covariance $\mathbf{P}'$ with additional predictive noise $\mathbf{Q_t}$:

$$\hat{\mathbf{x}}' = \sum_{i=0}^{2n} W_i^m f(\mathcal{X}_i) \tag{2.21}$$

$$\mathbf{P}' = \sum_{i=0}^{2n} W_i^c \big(f(\mathcal{X}_i) - \hat{\mathbf{x}}'\big)\big(f(\mathcal{X}_i) - \hat{\mathbf{x}}'\big)^T + \mathbf{Q_t} \tag{2.22}$$

An example is shown in Figure 2.7. Following this, the update may be stated as:

$$\hat{\mathbf{z}}_t = \sum_{i=0}^{2n} W_i^m h(\mathcal{X}_i) \tag{2.23}$$

$$\mathbf{S}_t = \sum_{i=0}^{2n} W_i^c \big(h(\mathcal{X}_i) - \hat{\mathbf{z}}_t\big)\big(h(\mathcal{X}_i) - \hat{\mathbf{z}}_t\big)^T + \mathbf{R_t} \tag{2.24}$$

$$\mathbf{C}_t = \sum_{i=0}^{2n} W_i^c \big(\mathcal{X}_i - \hat{\mathbf{x}}'\big)\big(h(\mathcal{X}_i) - \hat{\mathbf{z}}_t\big)^T \tag{2.25}$$

$$\mathbf{K}_t = \mathbf{C}_t \mathbf{S}^{-1} \tag{2.26}$$

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}' + \mathbf{K}_t(\mathbf{z}_t - \hat{\mathbf{z}}_t) \tag{2.27}$$

$$\mathbf{P}_t = \mathbf{P}' - \mathbf{K}_t \mathbf{S}_t \mathbf{K}_t^T \tag{2.28}$$

where $\hat{\mathbf{z}}_t$ is the predicted observation at state t, $\mathbf{S}_t$ the related covariance and $\mathbf{R}_t$ additive measurement noise. Thus, the Kalman gain $\mathbf{K}_t$ is calculated based on the cross-covariance $\mathbf{C}_t$ between the predicted state and predicted observation. Finally, mean and covariance are updated.

2.4.3 Coordinated Turn Motion Model

As the performance of an UKF can show a significant dependence on the choice of state coordinates, [Roth et al. 2014] proposed to use Coordinated Turn models in order to track maneuvering targets. Therefore the following basics are reproduced from [Roth et al. 2014]. Coordinated Turn models have their origin in curvilinear particle motion [Li and Jilkov 2003] and describe a horizontal motion at nearly constant speed along with circle segments. Hence, a target can be described by a position with respect to a fixed Cartesian frame x and y, a velocity vector v and heading angle $\phi = atan2(v_y, v_x)$. In this way, the yaw rate is defined as the time derivative of ϕ with $\dot{\phi} = \frac{d\phi}{dt}$. Thus, the target state is defined by $x_t = \begin{bmatrix} x & y & v & h & \dot{\phi} \end{bmatrix}^T$. Moreover, targets can be driven with linear (longitudinal) a and rotational acceleration α, corresponding to inputs of the dynamic rotation (steering), acceleration, and deceleration of objects. Here, speed and yaw rates are modified. An illustration of the involved parameters is given in Figure 2.8.

In contrast to linear Coordinated Turn variants with known yaw rates, nonlinear ones allow for varying speed as well as yaw rates [Li and Jilkov 2003], as required for this work. However, they imply the use of a nonlinear transition function in the state estimation, e.g. given an UKF. The transition for a target state at time t to the next time step $t + 1$ is defined by:

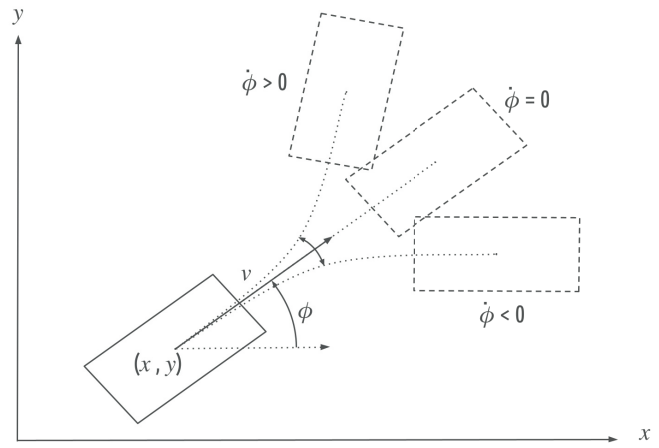


Figure 2.8: *Illustration of state components of the Coordinated Turn model: The target object is depicted in the xy plane, with the magnitude of a velocity vector v and heading angle ϕ. The yaw rate $\dot{\phi}$ can be denoted as the time derivative of the heading angle. Here, three paths with different yaw rates are outlined (dotted lines).*

$$x_{t+1} = f(x_t) + G(x_t)w_k \tag{2.29}$$

Assuming a Cartesian velocity, the discrete-time state estimation $f(x_t)$ with sample time T is given by:

$$f(x_t) = \begin{bmatrix} x + \frac{2v}{\dot{\phi}} \sin(\frac{\dot{\phi}T}{2}) \cos(h + \frac{\dot{\phi}T}{2}) \\ y + \frac{2v}{\dot{\phi}} \sin(\frac{\dot{\phi}T}{2}) \sin(h + \frac{\dot{\phi}T}{2}) \\ v \\ h + \dot{\phi}T \\ \dot{\phi} \end{bmatrix} \tag{2.30}$$

Then, the additive noise mapping of individual state parameters $G(x_t)$ can be denoted as:

$$G^1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ T & 0 \\ 0 & 0 \\ 0 & T \end{bmatrix}, \qquad G^2 = \begin{bmatrix} \frac{T^2}{2}\cos(h) & 0 \\ \frac{T^2}{2}\sin(h) & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix} \tag{2.31}$$

with noise sampled from the acceleration inputs $w = [a\ \alpha]^T$. Here, a and α are treat as uncorrelated random variables with zero mean and standard deviations σ_a and σ_α in order to obtain a stochastic motion model with uncertainties from the unknown parameters.

Chapter 3

Single Shot 3D Object Detection on Point Clouds

Object detection is a key problem in robotics and computer vision as it has many applications, including Autonomous Driving. This task aims to automatically localize and classify instances of objects that semantically belong together. Objects are mainly described with simple geometric shapes, such as cuboid bounding boxes. However, in particularly complex scenarios from Autonomous Driving, it may be extremely challenging due to diversity, visibility, or unforeseen environmental changes. In addition, a distinction is often introduced between static, e.g. vehicles permanently parked at the roadside, and dynamic moving objects, in order to predict future motion of other road participants. This is mainly realized via tracking concepts with the addition of the temporal domain. However, this chapter will first present a novel single frame object detector on point clouds without tracking as a technical baseline for the following chapter 4, where a tracking concept is added.

Through preprocessing into a regular grid structure, input points can be processed by Convolutional Neural Network (CNN) architectures utilizing recent advances from image processing. The resulting output features are processed by a specific subnetwork, which directly outputs objects in one forward pass. Furthermore, the overall network is trained with a new kind of loss. It achieves competing results to state of the art proven on a benchmark dataset while being much faster and computationally more efficient. The content is derived primarily from the published work on [Simon et al. 2018] and [Simon and Milz 2018]. The first section motivates the usefulness and explains the technical difficulties. Section 3.2 summarizes methods from state-of-the-art related to object detection on point clouds relevant for this work. The focus of this chapter is section 3.3, which presents the model architecture as well as all related innovations in the training procedure for optimization and evaluations on

a benchmark dataset. Afterwards, section 3.5 provides an update on state of the art with more recent approaches that were developed after the publications in this chapter. Finally, section 3.6 gives a summary.

3.1 Motivation

Among other perception tasks like classification or pointwise semantic segmentation, object detection, sometimes also called recognition, has some particular advantages for use in autonomous vehicles, as it can be used directly for downstream modules such as tracking without extensive further processing. Objects in 3D space are typically described as oriented bounding boxes, where additional information, e.g. categories or flags like static vs. dynamic, can be assigned. In this way, all relevant information is wrapped in a simplified native and highly efficient fashion.

Object detection on point clouds covering public driving scenarios is very challenging due to many aspects. First of all, inferred coherent parameters, like a pose, dimensions, category, all have completely different meanings, value ranges, and scaling. This complicates the regression significantly because all those parameters are combined to form a single object. On top, the number of objects in a scene is unknown and may vary dramatically. As already mentioned in 1.1, lots of challenges and difficulties come from the use case or rather the application itself. Object viewpoints have a wide variety, with point clouds capturing only the front surfaces. They can be found throughout the region of interest in variable quantities. Plus, there may be partial occlusion, further increasing the diversity. On the other hand, the objects of interest are already very dissimilar, resulting in strong inter- and intraclass variations. A huge truck on the road needs to be recognized just like a small child on the sidewalk. Accompanying this, environmental factors such as weather conditions, daytime variations, or material properties affect the sensor's perception, also generating sensor noise. Ultimately, runtime and hardware requirements are severely limited, so the highest efficiency must be achieved.

In recent years, deep learning has also revolutionized object detection methods. The high level of attention in the research community has led to breakthroughs in image processing, especially which have already been widely used in real-world applications, such as Video Surveillance, Robot Vision, and Autonomous Driving prototypes. All of this forms the basis to be transferred to point clouds. Particularly because more and more public datasets now contain not only large-scale point clouds as a reference to camera images but also target dedicated tasks operating on point

clouds or fusion concepts in order to use an alternative modality complementary to a camera. According to the advantages mentioned above, 3D object detection is among them as one of the most complex tasks with great potential for several real-world applications.

3.2 Related Work

This section reviews the state-of-the-art related to 3D object detection on point clouds up to the time of the work of this chapter. Further work done after or in parallel with the publication can be found in section 3.5.

Most recent approaches are built on findings from image processing utilizing CNN architectures. However, the naive transfer of 2D object detectors based on image inputs to the point cloud domain lacks several aspects. On the one hand, the third spatial dimension results in a computational burden while increasing the size of the inputs, i.e. the memory consumption grows. On the other hand, processing with CNN is highly inefficient since point clouds are usually very sparse with unequal spatially distributed local densities. The first approaches explored voxelization. [Zeng Wang and Posner 2015] use a sliding window in all three dimensions over 3D voxels with fixed dimensional feature vectors consisting of points, or zero. Voting based on a classifier, e.g. Support Vector Machine (SVM), scores objects of interest at each position. This voting is repeated for several discrete orientation steps to deal with rotations. Overlapping predictions are suppressed via Non Maximum Suppression (NMS) [Neubeck and Van Gool 2006]. Following [Zeng Wang and Posner 2015], [Engelcke et al. 2017] proposed a CNN based on sparse 3D convolutions applied to all non-empty voxel cells in a sliding window manner. Furthermore, fixed-size bounding boxes were assumed to remove the need to regress object dimensions. Thus, such networks are designed to be class-specific. [Li 2017] used the typical encoding decoding scheme with strided 3D convolutions to generate two output branches: an objectness map predicting if a region belongs to an object and a bounding box map predicting all regression parameters defining bounding boxes.

Another useful representation of point clouds, as range often called front view images, was explored in [Li et al. 2016]. First, a conversion is done using cylindrical projection where all pixels of a 2D representation are filled with the distance in the ground plane and their height value. Similar to [Li 2017], the processing is done via CNN, whereas bounding boxes are encoded as 24D vectors containing eight corner points.

More recently, multiple fusion approaches with heavy processing pipelines combine point clouds with camera images. In [Chen et al. 2017], point clouds are encoded as multi-view representation with a birdview, containing height, intensity, and density features, and front view with height, distance, and intensity features. Inspired by [Ren et al. 2015] and [Simonyan and Zisserman 2015], a CNN first generates 3D proposals from birdview with an RPN and projects them to front view and image inputs. Subsequently, a deep fusion network combines the features region-wise and jointly predicts object class as well as bounding box parameters, again encoded as eight corners. Similarly, [Ku et al. 2018] operated on birdview representation. Both camera and lidar inputs are processed by two modified VGG16 [Simonyan and Zisserman 2015] sub-networks with bilinear upsampling followed by an anchor-based RPN. The outputs are fused via an element-wise mean operation at feature level and two task-specific fully connected layers to regress axis-aligned object proposals. Finally, a second stage network is used to create final predictions, encoding bounding boxes as four corners at the ground, two height offsets, and yaw angle, respectively. Here feature crops are generated from projecting predictions of the first network into the two input views followed by an element-wise mean operation and three fully connected layers. Duplicate predictions are filtered by NMS and the overall architecture is trained end to end. Furthermore, [Qi et al. 2018] leverage a mature 2D object detector in the image domain to propose and classify 2D object regions. For each proposal from camera, a frustum is generated from corresponding bounding boxes, collecting all points falling into it and normalizing rotations such that the center axis of the frustum is orthogonal to the image plane. This is followed by two PointNets [Qi, Su, et al. 2017], one for instance segmentation and one for bounding box regression and a small sub-network to learn a transformation of the instance center to the actual object center of the complete object in 3D.

A discretization as a voxel or range image obscures natural 3D patterns and invariances of 3D data. Therefore, [Zhou and Tuzel 2018] kept a fixed number of points in each occluded voxel cell. These points are then processed by multiple stacked Voxel Feature Encoding layers based on a preprocessing and a linear layer with batch normalization, Rectified Linear Unit (ReLu) activation as well as max pooling to get highly descriptive pointwise features, inspired by [Qi, Su, et al. 2017]. On top, resulting features are sequentially processed by 3D convolutional middle layers with batch normalization and ReLu, followed by an RPN.

State of the art focuses on different representations of point clouds to utilize existing and newly proposed network architectures. An overview is given in Figure 3.1.

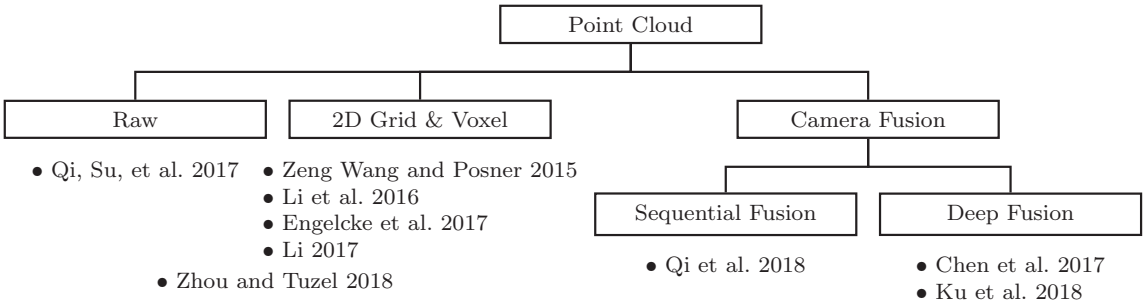


Figure 3.1: *Classification of related approaches for 3D object recognition based on point cloud representation up to the time of the work in this chapter.*

Using range images, nearby features are not necessarily adjacent in 3D space. Thus, models need to recover the spatial geometric relationship. Fusion approaches suffer from large processing pipelines in order to solve the complex task of object detection step-wise. This leads to high accuracies but is mostly unsuitable for real-time applications due to heavy computations. Voxels are used as a powerful alternative to structure unordered point clouds to be used in CNN models. However, those methods are often less efficient since typically more than 90 percent of voxels are empty due to sparsity. This chapter takes this deficit and presents a novel real-time detector by extending [Redmon and Farhadi 2017] from the image domain. Inspired by [Chen et al. 2017], birdview representation of point clouds is used in this work because it has several advantages with regard to the target application. First, objects usually lie on the ground plane with a small variance in the vertical direction. Second, the physical sizes of objects are preserved. Lastly, occlusion problems are mostly avoided since objects occupy a different space.

3.3 Baseline Model for 3D Object Detection on Point Clouds

This section describes the proposed baseline model for 3D object detection based on point clouds, named Complex-YOLO. It follows the single-shot detection paradigm „You Only Look Once (YOLO)“ assisted by a CNN for direct regression of objects trained end to end. Hence it forms the foundation for the subsequent chapter where tracking is added. A diagram of this approach is presented in Figure 3.2. As input, a single point cloud is transformed into birdview representation first and contains only three feature channels. Subsequently, a feature encoder network composed of convolutions and a task-specific Region Proposal Network are applied in order to

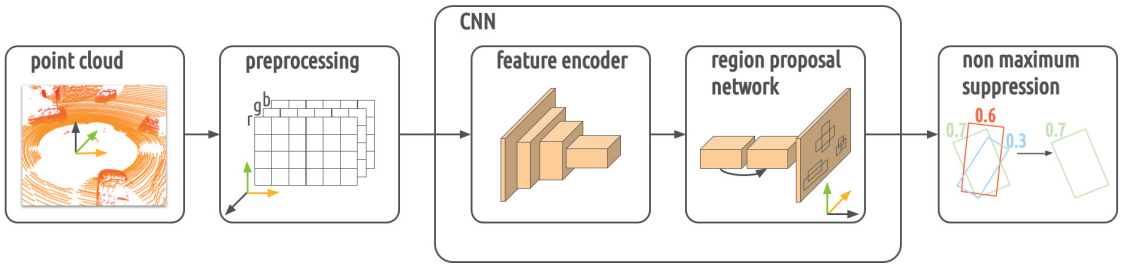


Figure 3.2: *Overview of the 3D object detection model. A single CNN takes a point cloud transformed into birdview image representation as input and computes bounding boxes in one forward pass. The network architecture is adopted from image processing and extended with additional regression parameters for object orientations in complex space.*

output dense predictions in one single forward pass. The following subsections explain each part in more detail.

3.3.1 Point Cloud Preprocessing

Initially, a point cloud $\mathcal{P}$ of a single time frame is cropped to a predefined area of interest Ω:

$$\mathcal{P}_\Omega = \{\mathcal{P} = [x, y, z]^T | x \in [x_{min}, x_{max}], y \in [y_{min}, y_{max}], z \in [z_{min}, z_{max}]\} \tag{3.1}$$

The origin of $\mathcal{P}$ is considered at the position of the generating sensor, x points towards the front, y points right and z is up, as indicated in Figure 3.2. With the help of the mapping function $\mathcal{S}_j = f_{\mathcal{PS}}(\mathcal{P}_{\Omega i}, g)$ with $\mathcal{S} \in \mathbb{R}^{m \times n}$, each point $\mathcal{P}_\Omega$ with index i gets mapped into a specific grid cell $\mathcal{S}_j$, described as a set:

$$\mathcal{P}_{\Omega i \to j} = \{\mathcal{P}_{\Omega i} = [x, y, z]^T | \mathcal{S}_j = f_{\mathcal{PS}}(\mathcal{P}_{\Omega i}, g)\} \tag{3.2}$$

where g is the predefined size of a grid cell, equal in both directions, resulting in m rows and n columns. For instance, a typical size g lies between $0.06\,m$ to $0.40\,m$, resulting in grid resolutions of roughly $m = n \in [200, 1334]$ for an area of interest of $80\,m \times 80\,m$. A more detailed configuration with all parameters can be found in the experiments section (see subsection 3.4.2). As per requirement efficiency, each cell is filled with only three handcrafted feature channels inspired by [Chen et al. 2017]:

$$\begin{aligned} z_b(\mathcal{S}_j) &= \max(\mathcal{P}_{\Omega i \to j} \cdot [0, 0, 1]^T) \\ z_g(\mathcal{S}_j) &= \max(I(\mathcal{P}_{\Omega i \to j})) \\ z_r(\mathcal{S}_j) &= \min\left(1.0, \log(N+1)/\log(64)\right) \quad N = |\mathcal{P}_{\Omega i \to j}| \end{aligned} \tag{3.3}$$

Algorithm 3.1 Birdview Transformation

Require:
points $\mathcal{P}$, rows m, cols n, stepsize g, range $x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}$

$grid \leftarrow init(m, n) \in \{\}$
$bv \leftarrow Matrix(m, n, 3) \in 0.0$
$\mathcal{P} \leftarrow \mathcal{P}[\mathcal{P}_x > x_{min} \wedge \mathcal{P}_x < x_{max}, \mathcal{P}_y > y_{min} \wedge \mathcal{P}_y < y_{max}, \mathcal{P}_z > z_{min} \wedge \mathcal{P}_z < z_{max}]$

for $i \leftarrow 0$ to $(|\mathcal{P}| - 1)$ **do**
    $u \leftarrow (m-1) - round(\frac{\mathcal{P}_{ix} - x_{min}}{g_x})$
    $v \leftarrow (n-1) - round(\frac{\mathcal{P}_{iy} - y_{min}}{g_y})$
    if $u \geq 0 \ \wedge \ u < m \quad \wedge \quad v \geq 0 \ \wedge \ v < n$ **then**
        $grid[u][v] \leftarrow grid[u][v] \cup \{\mathcal{P}_i\}$

for $u \leftarrow 0$ to $(m-1)$ **do**
    for $v \leftarrow 0$ to $(n-1)$ **do**
        $\mathcal{P}_{uv} \leftarrow grid[u][v]$
        if $|\mathcal{P}_{uv}| > 0$ **then**
            $bv[u][v][0] \leftarrow round(\frac{max(\mathcal{P}_{uv} \cdot [0,0,1]^T) - z_{min}}{z_{max} - z_{min}})$
            $bv[u][v][1] \leftarrow max(I(\mathcal{P}_{uv}))$
            $bv[u][v][2] \leftarrow min(1.0, \frac{\log(|\mathcal{P}_{uv}|+1)}{\log(64)})$

return bv

Here, each point contains an intensity value $I(\mathcal{P}_\Omega)$ used to calculate the maximum intensity z_g, N describes the number of points within a single grid cell $\mathcal{S}_j$ used to calculate the normalized density z_r and z_b encodes the maximum height. The pseudo-code is provided in Algorithm 3.1.

3.3.2 Feature Encoder Backbone

From a transformed point cloud in birdview representation with dimensions $m \times n \times 3$, an efficient and straightforward CNN can be applied to learn latent features. The first part of the model is adopted from Darknet-19 [Redmon and Farhadi 2017]. Generally, only 3×3 and 1×1 convolutions mixed by max-pooling layers are used. Figure 3.3 presents the underlying architecture in detail.

First, the input is fed into two blocks of 3×3 convolutions, followed by max-pooling with stride for downsampling in parallel increasing the number of channels. This is followed by three blocks of three convolutions with additional max-pooling layers, either two 3×3 mixed with one 1×1 convolutions or all three using 3×3 filter kernels. The resulting feature maps are convolved by another block of five

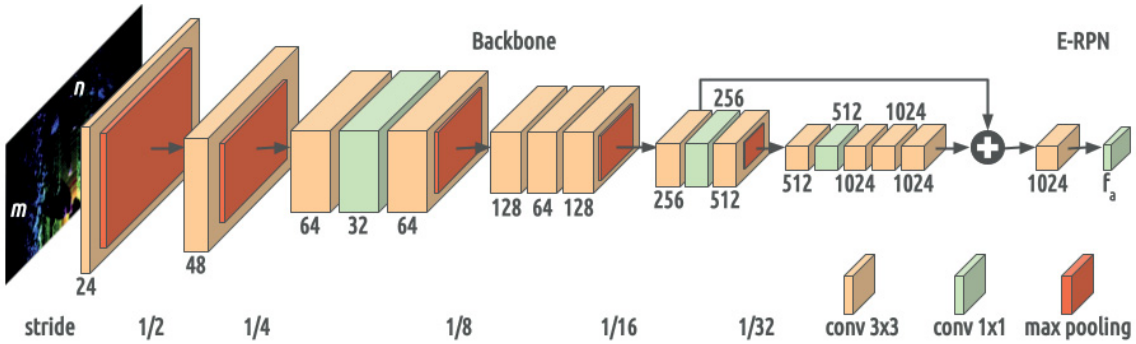


Figure 3.3: *The structure of the CNN model for 3D object detection adopted from [Redmon and Farhadi 2017]. Point clouds are transformed into birdview image representation and processed by the network consisting of convolutions as well as max-pooling layers, including one skip connection as indicated by the black arrow. Overall, the input resolution gets downsampled by* $\frac{1}{32}$ *in both directions with final outputs after the Euler Region Proposal Network (E-RPN). Here,* f_a *denotes the number of filters in the last convolutional layer.*

convolutional layers. Afterwards, a skip connection concatenates residual features to enable the usage of fine-grained features from a previous layer. Finally, two last convolutions, 3×3 and 1×1 generate outputs that can be directly interpreted as detected objects by the Euler Region Proposal Network (E-RPN). Throughout the network, leaky ReLu activation is used except the last layer, where linear activation is used instead. Similarly, batch normalization is used for regularization in order to stabilize the training and speedup convergence, except for the last layer. As seen in the Yolov2 model [Redmon and Farhadi 2017], 1×1 filters are periodically mixed into the blocks of convolutions to ensure strong compression of the feature representation. To this end, the number of channels doubles after most pooling steps, reaching up to $1,024$ channels while downsampling to $\frac{1}{32}$.

Compared to the original Darknet-19, the initial number of filters in early layers and the overall number of convolutional blocks are slightly reduced. As found by experiments, this saves computations and speeds up runtime while losing nearly zero performance. Potentially due to sparsity and less diversity in the inputs compared to real camera images.

3.3.3 Euler Region Proposal Network

The last layer of the network is the E-RPN detection layer. The layer itself does not contain additional learnable parameters, as it simply parses the final feature map of the last convolutional layer and outputs oriented bounding boxes in 3D space

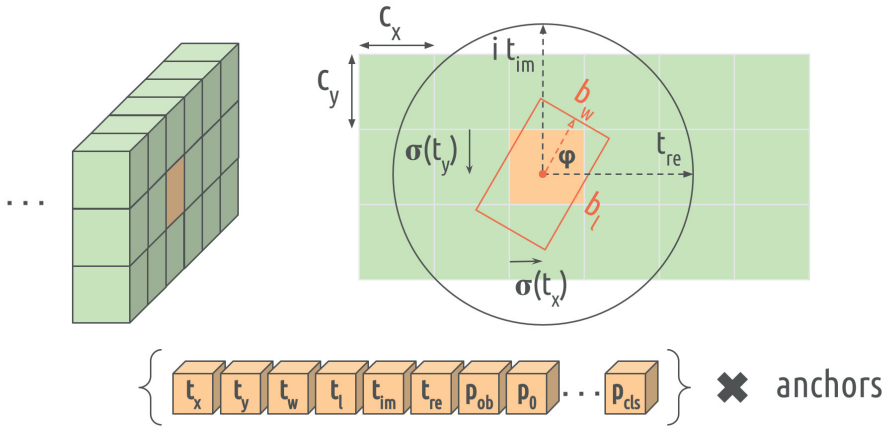


Figure 3.4: *Outline of the bounding box interpretation used within the E-RPN. The last feature map is conditioned to learn offsets from hand-picked priors, based on the anchoring principle introduced in [Ren et al. 2015] and [Redmon and Farhadi 2017]. For each grid cell, multiple predictions are calculated from the output features with respect to the corresponding anchor values.*

through birdview back projection. Similar to [Redmon and Farhadi 2017], anchors are used as priors to simplify the problem. All anchor values are defined according to the distribution of the underlying dataset. Instead of predicting coordinates directly, only offsets at every location in the last feature map relative to the location of the grid cell are estimated, as shown in Figure 3.4. For each grid cell, multiple predictions are generated using the 3D position assuming a flat surface $b_{x,y}$, width b_w and length b_l as well as a confidence probability p_{ob}, class scores $p_0 ... p_{cls-1}$ and finally the orientation b_ϕ in conjunction with their related anchors. Object heights b_h are fixed values for each class, determined from the underlying training dataset. Thus, making it easier for the network to learn. Overlapping predictions can be filtered using NMS.

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_l &= p_l e^{t_l} \\ b_\phi &= arg(|z|e^{ib_\phi}) = \arctan_2(t_{im}, t_{re}) \end{aligned} \tag{3.4}$$

Compared to state-of-the-art, the key innovation lies in the introduction of two additional regression targets t_{im} and t_{re}, that correspond to the phase of a complex number. This is interpreted as imaginary and real fraction corresponding to the object's yaw angle b_ϕ, similar to [Beyer et al. 2015]. Again anchors are added for both

targets to simplify the problem to learn. On the one hand, this avoids singularities. On the other hand, this results in a closed mathematical space, which has a favorable impact on generalization of the model. Consequently, all regression parameters are directly linked into the loss function without ambiguities.

3.3.4 Loss Function

In order to train the presented model, an appropriate loss function is used. Based on the concepts from [Redmon et al. 2016] and [Redmon and Farhadi 2017], the network optimization is done via an extended multi-part loss $\mathcal{L}$, built on the sum of squared errors of individual parts. The existing loss $\mathcal{L}_{\text{Yolo}}$ is extended by an orientation part $\mathcal{L}_{\text{Euler}}$, to get use of complex numbers with closed mathematical space for angle estimations.

$$\mathcal{L} = \mathcal{L}_{\text{Yolo}} + \mathcal{L}_{\text{Euler}} \tag{3.5}$$

The extended part of the loss $\mathcal{L}_{\text{Euler}}$ assumes that the difference between prediction and ground truth, i.e. $|z|e^{\mathrm{i}b_\phi}$ and $|\hat{z}|e^{\mathrm{i}\hat{b}_\phi}$ is always located on the unit circle with $|z| = 1$ and $|\hat{z}| = 1$. Therefore, the absolute value of the squared error gets minimized to get a real valued loss:

$$\mathcal{L}_{\text{Euler}} = \lambda_{coord} \sum_{i=0}^{\frac{m \cdot n}{32}} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left| (e^{\mathrm{i}b_\phi} - e^{\mathrm{i}\hat{b}_\phi})^2 \right| \tag{3.6}$$

$$= \lambda_{coord} \sum_{i=0}^{\frac{m \cdot n}{32}} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(t_{im} - \hat{t}_{im})^2 + (t_{re} - \hat{t}_{re})^2 \right] \tag{3.7}$$

where λ_{coord} is a scaling factor to ensure stable convergence in early phases and $\mathbb{1}_{ij}^{obj}$ denotes that the jth bounding box predictor in cell i has highest Intersection Over Union (IOU) compared to ground truth for that prediction, considering their anchors respectively. Consequently, the comparison between ground truth G and predicted box P_j with IOU $\frac{P_j \cap G}{P_j \cup G}$, where $P_j \cap G = \{x : x \in P_j \wedge x \in G\}$, $P_j \cup G\{x : x \in P_j \vee x \in G\}$ is adapted to consider rotations as well. In particular, this is required at the inner core of the loss to calculate $\mathbb{1}_{ij}^{obj}$ and $\mathbb{1}_{ij}^{noobj}$, impacting all individual parts. With the help of 2D polygon geometries, the IOU is calculated for rotated rectangles created from parameters b_x, b_y, b_w, b_l and b_ϕ of ground truth and prediction.

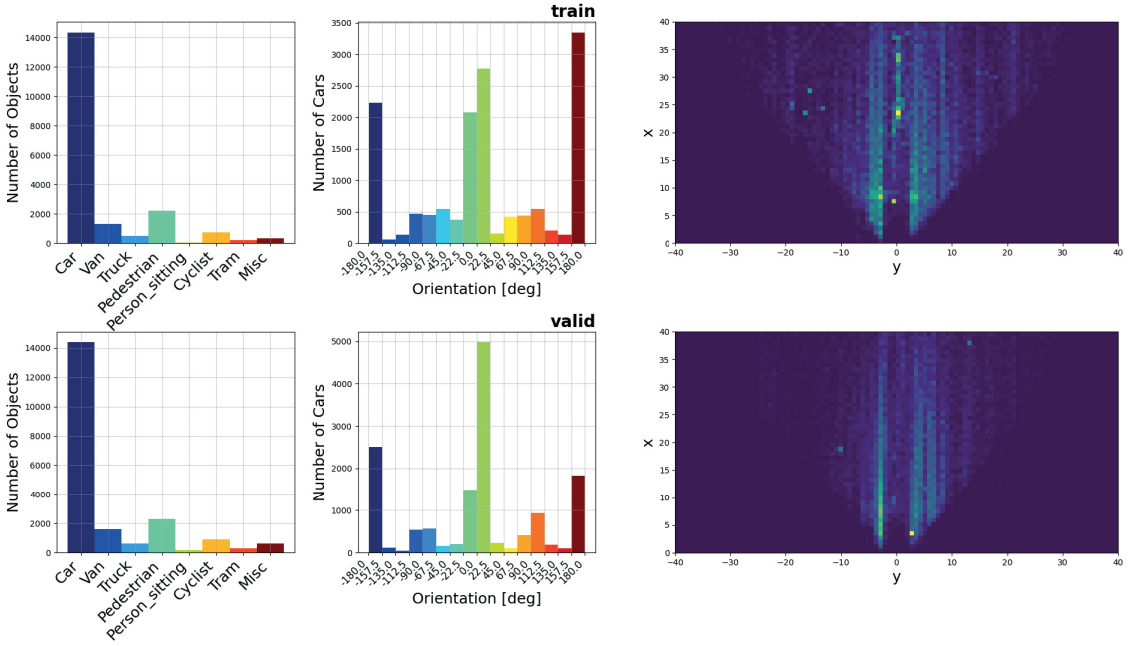


Figure 3.5: *Object occurrence and orientation statistics of the KITTI dataset used for training (first row) and validation (second row): Here, the number of objects per class, the number of Cars per yaw angle, and the spatial distribution of object centers in a normalized 2D birdview histogram are visualized, respectively. Note the strong imbalance with an order of magnitude more Cars compared to the other classes. As a result, the respective direction of a Car driving is clearly reflected in the orientation plot as well as driving lanes occurring in both heatmaps.*

3.4 Experiments

This section describes the experiments conducted to evaluate the performance of the proposed model. First, the datasets, as well as details for training and optimization, are presented. Then, the evaluation metrics and results for object detection are reported.

3.4.1 Datasets

The pioneering KITTI object detection dataset [Geiger et al. 2012] is used to assess the performance of the proposed method. This dataset consists of $7,481$ training and $7,518$ testing samples, including point clouds and images. Ground truth annotations done by humans are publicly available for training only. An exhaustive description is provided in section B.3.

Following the protocol proposed by [Chen et al. 2017; Zhou and Tuzel 2018;

Qi et al. 2018], the training set is split into $3,712$ samples for training and $3,769$ for validation while keeping recorded sequences separated in order to avoid samples from the same sequence being included in both the training and validation sets, obtaining the distributions in Figure 3.5. Both the class and the angle distributions contain severe imbalances. As ground-truth objects are only available for the area visible in the image plane, there are two blind spots to the left and right of the ego vehicle, where supervision can only be applied via augmentation. Nevertheless, the training and validation splits are approximately equally distributed in occurrence and orientation.

3.4.2 Training and Optimization Details

Before training, all point clouds are transformed into birdview images, as described in 3.3.1, where $x \in [0\,m, 40\,m]$, $y \in [-40\,m, 40\,m]$, $z \in [-2\,m, 1.25\,m]$, $g = 0.08\,m$, $m = 512$ and $n = 1024$. Simultaneously global rotation is performed for augmentation. Rotations around the origin of the point cloud from $-30°$ to $30°$ with step size 5 are randomly applied to point clouds and ground truth in order to scale the size of the training dataset. Examples from different scenarios of the training dataset are visualized in Figure 3.6. With the number of anchors set to 5 and using all 8 classes, the number of filters in the last convolutional layer is set to $f_a = 75$. Here, anchors are related to: 1) *Car* size, heading towards the front, 2) *Car* size, heading towards the back, 3) *Cyclist* size, heading towards the front, 4) *Cyclist* size, heading towards the back and 5) *Pedestrian* size with heading to the left. These values are defined based on the distribution of ground truth objects within the training dataset. All described prediction parameters from subsection 3.3.3 are included in x, resulting in $16 \times 32 \times 5$ predictions simultaneously. Each one also contains probabilities for object occurrence and classification used afterwards for filtering with a threshold of 0.6. By design, there should be no overlapping predictions. Therefore, NMS with a threshold of 0.2 is applied to the final output.

Following the original YOLOv2 [Redmon and Farhadi 2017], the model is trained with Stochastic Gradient Descent (SGD) using random weight initialization with a weight decay of 0.0005 and momentum 0.9. After the first epochs, the learning rate is slowly scaled up starting from 0.0001 and then gradually decreased for up to 300 epochs to prevent model instability causing the training to diverge early on. Using a constant batch size of 4, the learning rate is adjusted through quantitative testing along with other meta parameters.

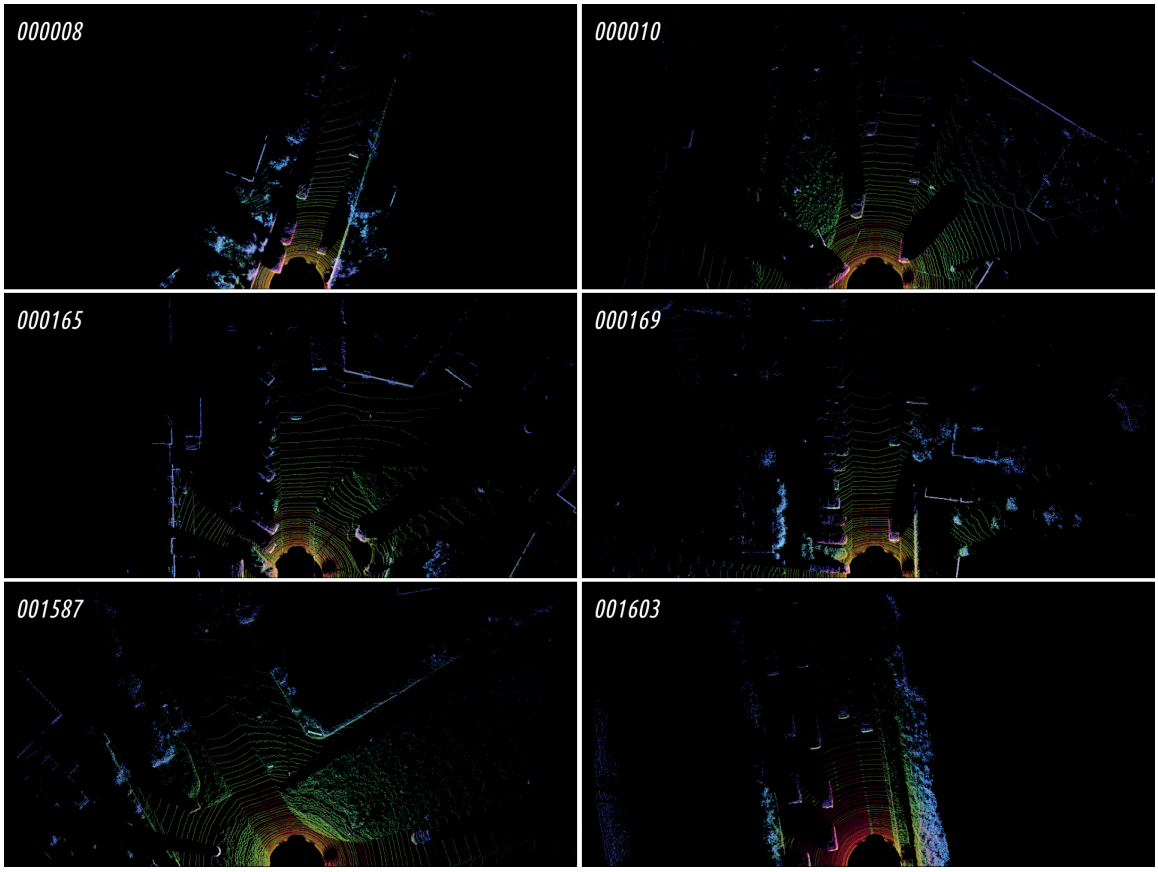


Figure 3.6: *Diverse samples of point clouds transformed into birdview images: 000008) road within a village with parked vehicles on the sides of the road, 000010) road with vegetation and parked vehicles, 000165) inner city with diagonal parking, 000169) outskirts with crowded parking spaces, 001587) landscaped intersection with low traffic and 001603) highway with traffic jam. Each color represents one of the feature channels as described in subsection 3.3.1 while pixels without any point are displayed in black as they are empty. Typical scan patterns from lidar are clearly visible. For instance, the point density decreases heavily with distance (also shown as the red color channel), as well as the characteristic shadows behind objects caused by occlusion. Similarly, the expected diversity and immense complexity need to be taken into account while having many empty pixels.*

3.4.3 Applied Evaluation Metrics

The evaluation of the object detection performance follows the PASCAL criteria [Everingham et al. 2010] also used for 2D object detection. Here, an interpolated Average Precision (AP) metric is used to approximate the shape of the precision-recall curve by averaging. This is done separately for the categories 2D (bounding box overlap in image space), 3D (3D bounding box overlap), and birdview (rotated bounding box overlap in birdview space), i.e. depending on the function to distinguish

Method	Modality	Car			Pedestrian			Cyclist		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
[Li et al. 2016]	Lidar	40.14	32.08	30.47	-	-	-	-	-	-
[Chen et al. 2017]	Lidar	86.18	77.32	76.33	-	-	-	-	-	-
[Chen et al. 2017]	Lidar+Mono	86.55	78.10	76.67	-	-	-	-	-	-
[Qi et al. 2018]	Lidar+Mono	88.16	84.02	76.44	**72.38**	**66.39**	**59.57**	**81.82**	**60.03**	56.32
[Zhou and Tuzel 2018]	Lidar	**89.60**	**84.81**	**78.57**	65.95	61.05	56.98	74.41	52.18	50.49
Complex-YOLO	Lidar	84.95	76.37	75.77	44.23	42.21	38.47	66.36	60.02	**59.63**

Table 3.1: *Performance comparison for birdview object detection based on Average Precision percentages where higher is better. The results are obtained on the validation dataset with the same setup. A few values were not reported originally.*

Method	Modality	Car			Pedestrian			Cyclist		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
[Li et al. 2016]	Lidar	15.20	13.66	15.98	-	-	-	-	-	-
[Chen et al. 2017]	Lidar	71.19	56.60	55.30	-	-	-	-	-	-
[Chen et al. 2017]	Lidar+Mono	71.29	62.68	56.56	-	-	-	-	-	-
[Qi et al. 2018]	Lidar+Mono	**83.76**	**70.92**	63.65	**70.00**	**61.32**	**53.59**	**77.15**	56.49	53.37
[Zhou and Tuzel 2018]	Lidar	81.97	65.46	62.85	57.86	53.42	48.87	67.17	47.65	45.11
Complex-YOLO	Lidar	76.01	70.11	**65.42**	40.86	38.16	37.17	64.74	**58.09**	**57.92**

Table 3.2: *Performance comparison for 3D object detection based on Average Precision percentages where higher is better. The results are obtained on the validation dataset with the same setup. A few values were not reported originally.*

true from false predictions. Furthermore, the evaluation is subdivided into *Car*, *Pedestrian* and *Cyclist* with respect to existing object classes. All details can be found in Appendix C.

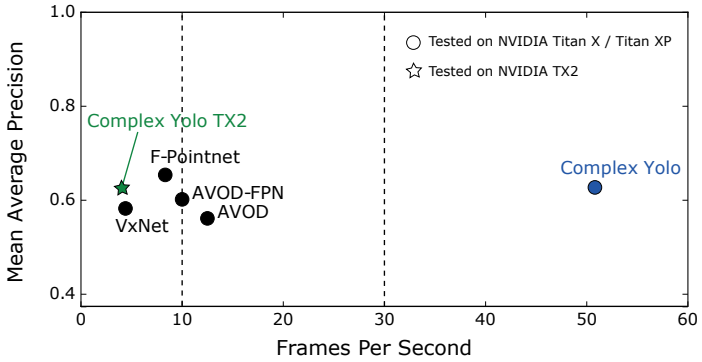


Figure 3.7: *Comparison of 3D detection performance in relation to runtime. Being one of the first works, efficiency has been increased significantly. Note that the mAP values slightly differ compared to Table 3.2, because a different dataset split was used. Reproduced from [Simon et al. 2018].*

3.4.4 Results

Several related models have been developed preliminary to this work, which are included in Table 3.1 and Table 3.2 for comparison. Both tables report the aforementioned AP scores on the validation dataset. As can be seen, the higher difficulty level of the 3D task constantly decreases results compared to birdview evaluation. Most similar to this work are the models of [Chen et al. 2017]. In contrast to their work, no second point cloud representation and third camera-input are used, offering a much lighter model while removing the need for additional fusion layers. Unlike VoxelNet [Zhou and Tuzel 2018], where models are trained separately and fine-tuned to single class predictions, Complex-YOLO performs multi-class prediction with only one model trained on all classes in parallel, mainly due to efficiency and the intended use in Autonomous Driving applications. The developed model of this work still manages to achieve comparable accuracy. On top, with an average runtime of 50.4 frames per second measured on comparable hardware, Complex-YOLO outperforms all other published methods in terms of efficiency (runtime) by an impressive margin of 400%. A comparison can be found in Figure 3.7 with mAP over the classes *Car*, *Pedestrian* and *Cyclist* plotted against frames per second. Even on Nvidia TX2 [NVIDIA Corporation 2017] as a dedicated edge device, 4 frames per second can be achieved without specialized optimization.

In addition to the mentioned quantitative results, Figure 3.8 presents some selected qualitative results as well. Six different driving scenarios from KITTI are presented with point clouds in birdview and associated camera images for reference. Objects detected by the developed model are shown as oriented bounding boxes in

both modalities, colored by class. A small vector from the center of each object with the same color indicates the detected orientation. Although the training data is biased due to the majority of *Cars* driving straight ahead or coming towards as seen in Figure 3.5, the model generalizes the localization and orientation of vehicles on these unseen samples. For instance, all directions in the upper left sample are recognized correctly, even though one of the vehicles is parked against the driving lane. This does not work quite as well for smaller objects like *Pedestrians* (yellow) or *Cyclists* (green), that are barely visible in the point cloud.

Figure 3.8: *Qualitative results with projection into image space for illustrative purposes, reproduced from [Simon et al. 2018]. More details are described in subsection 3.4.4.*

Nevertheless, the occurrence is approximated almost correctly in many cases, as in the upper right and middle left examples. Here, groups of small objects combined

with partial occlusions complicate the recognition. Both examples with multiple *Pedestrians* or *Cyclists* in a confined space lead to a few misdetections. This effect is further enhanced by extremely sparse point clouds, where the network cannot extract many details. The model generally predicts up to fine-grained objects of multiple classes but commonly makes subtle mistakes. As can be seen in the middle right and lower left samples, the assumed mean values for the height of the *Van* (red) and *Misc* (blue) class does not correspond to the actual height in the image. Likewise, the model tends to overestimate the confidence of detections and is limited to the area visible by the camera as mapped in the underlying training data from KITTI. For example, the first *Car* on the left of the picture from the lower-right sample is excluded from the region of interest, and therefore no detection is available. Furthermore, the limited range becomes visible on the highway in the lower-left example, where vehicles ahead can no longer be detected. However, the model often gets the right intention and is also capable of predicting *Cars* parking diagonally presented in the lower right sample of Figure 3.8. This suggests that useful features are learned from the birdview channels that abstract the nearby environment in addition to the object itself. As already seen from its predecessor [Redmon and Farhadi 2017] in image processing, the performance also strongly depends on the resolution of the birdview grid. Therefore, the resolution was selected rather high contrary to the efficiency criterion.

3.4.5 Ablation Study

To justify the formulation of the proposed method for object detection, an ablation study is performed. Following former experiments, the described network configurations are used in this study. The model is trained again from scratch, firstly with modified point cloud preprocessing to single input channel features only, and secondly with modified regression of object orientations. Here, a single regression parameter is learned instead of the complex space, and all related parts are adapted accordingly. After training, the models were applied to the validation dataset using the existing ground truth for evaluation. In Table 3.3, a summary of this analysis is provided, highlighting the negative impact of missing feature channels and complex formulations. Both birdview and 3D performance significantly drop in AP when using only one of the three input channels compared to all at once. The intensity channel consistently yields results that are up to 50 percentage points lower than the entire input. Several environmental influences, for instance, highly reflective materials or sunlight,

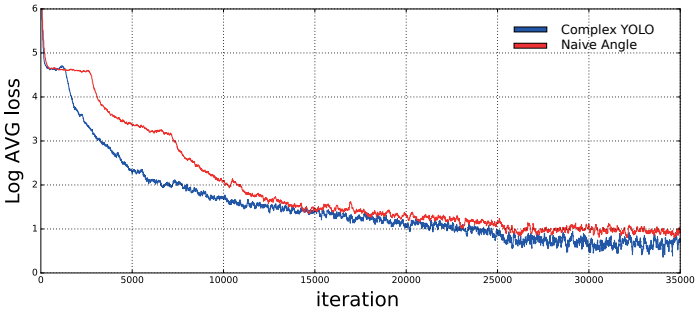


Figure 3.9: *An analysis of the proposed model using different strategies to predict object orientations. The plots illustrate the loss when using the complex formulation in blue compared to direct regression in red averaged over three attempts.*

Method	Birdview Car			3D Car		
	Easy	Mod.	Hard	Easy	Mod.	Hard
Height Channel	70.92 -14.03	55.09 -21.28	48.98 -26.79	57.45 -18.56	43.26 -26.85	38.14 -27.28
Intensity Channel	49.94 -35.01	40.04 -36.33	35.05 -40.72	29.98 -46.03	21.61 -48.50	20.63 -44.79
Density Channel	65.55 -19.40	52.14 -24.23	46.28 -29.49	43.04 -32.97	32.75 -37.36	28.24 -37.18
Naive Single Angle	68.13 -16.82	56.09 -20.28	55.02 -20.75	66.42 -9.59	53.98 -16.13	53.02 -12.40
Complex-YOLO	**84.95**	**76.37**	**75.77**	**76.01**	**70.11**	**65.42**

Table 3.3: *Analysis of the impact of individual feature channels and complex orientation estimation on the overall performance (AP). The model was trained separately with different single input channels in birdview images using a simple strategy with direct angle regression.*

lead to erroneous measurements plus sensor noise, potentially decreasing the performance. Although density features contain rich information for near-field, they are limited to a certain distance, only providing intermediate results. In contrast, overall, the height channel seems to contain the most entropy, which is also visually apparent to the human eye when looking at birdview samples.

Additionally to the results with naive regression of orientation, Figure 3.9 shows the course of the loss functions on average compared to the proposed solution. In early iterations, much faster convergence and, in total, a lower loss is achieved, emphasizing the beneficial properties of Complex-YOLO. Unlike other methods from state-of-the-art, a well-defined heading without ambiguities is learned.

3.5 Related Work upon the Baseline Model

Due to a worldwide interest, there has been rapid progress in the field of 3D object detection on point clouds. In parallel to the developed baseline model in this chapter, further methods for 3D object detection quickly contributed to state of the art. The trend here is mainly towards birdview, or rather voxel representation of point clouds, due to computationally friendly properties and re-usage of existing CNN architectures. Likewise, BirdNet [Beltran et al. 2018] generates three handcrafted birdview feature channels, with maximum height, mean intensity, and density, respectively. Instead of YOLO, Faster R-CNN [Ren et al. 2015] is adopted, using a VGG16 [Simonyan and Zisserman 2015] backbone for feature extraction. Moreover, the RPN [Ren et al. 2015] is extended by classification into N discrete bins corresponding to quantized yaw rotations, followed by softmax normalization. In this way, orientations are calculated using a weighted average of the predicted angle bins and the most probable neighbor. In contrast, [Luo et al. 2018] and [Yang et al. 2018] threat the height dimension in birdview space as channels for regular convolutions, similar to voxels. This comes with reduced loss of information during quantization but additional computations. Object orientations are learned with single regression targets θ normalized by $atan2(\sin\theta, \cos\theta)$. Again, architectures like VGG16 [Simonyan and Zisserman 2015] and SSD [Liu et al. 2016] from image processing were adapted. On top, [Luo et al. 2018] exploits spatiotemporal features by stacking multiple consecutive frames into the input voxels. Here, all 3D points from past frames are transformed into the current vehicle coordinate system while increasing the point density and giving cues about the motion of single objects. Unlike those methods, SECOND [Yan et al. 2018] follows VoxelNet [Zhou et al. 2018], using the proposed Voxel Feature Encoding (VFE) layers as an intermediate representation between the regular voxel and raw points. A sparse convolutional feature extractor is used, followed by an SSD [Liu et al. 2016] architecture as RPN with anchors. In [Lang et al. 2019], point clouds are organized in voxels with infinite height, called pillars, and processed by a feature encoding based on PointNet [Qi, Su, et al. 2017] as well as VFE layers [Zhou et al. 2018] and a 2D CNN followed by an SSD [Liu et al. 2016] detection head. In contrast, [Yang et al. 2019] developed a proposal generation module to first generate proposals from spherical anchors based on PointNet [Qi, Su, et al. 2017; Qi, Yi, et al. 2017]. Features are then generated for each proposal using a VFE layer. Another two-stage method from [Shi et al. 2019] is based on a bottom-up proposal generation with PointNet encoder [Qi, Yi, et al. 2017] for foreground-background segmentation and canonical

box refinement. In a follow-up, [Shi, Wang, Shi, et al. 2020] predict and aggregate intra object parts derived from bounding boxes for better supervision. Furthermore, [Shi, Guo, et al. 2020] combine raw point processing with VFE from [Zhou et al. 2018] and [Kuang et al. 2020; Ye et al. 2020] proposed multi-scale features in a voxel feature pyramid network, also based on VFE. Similarly, [Zheng, Tang, Chen, et al. 2021] fuse semantic features from different abstraction levels and align confidences from classification with the regression of localization parameters. Additionally, post-processing with distance variant NMS weighted by IOU is applied. [Yang et al. 2020] operates on raw point clouds, based on global feature generation with set abstraction layers from [Qi, Yi, et al. 2017] and fusion sampling, where objects are predicted relative to selected candidate points. Other specific work such as [He et al. 2020] proposed an auxiliary network for point-wise supervision and a part-sensitive warping scheme to align the classification with the regression branch in the detection head. [Liu et al. 2020] developed a triple attention module for joint channel-, point- and voxel-wise feature generation and coarse to fine regression. In [Zheng, Tang, Jiang, et al. 2021], shape aware data augmentation is used in a student-teacher setup based on a pair of single-stage detectors, together with orientation aware- and consistency loss. Unlike others, [Shi and Rajkumar 2020] transform point clouds to a fixed radius near-neighbors graph and make usage of a Graph Neural Network consisting of a Multi Layer Perceptron (MLP) with residual connections. Furthermore, [Yin et al. 2021] first formulate the task as keypoint detection, followed by a regression to other attributes and refinement with additional point features using an MLP.

Another area of research focuses on the fusion of point clouds with camera images. The idea is to benefit from meaningful contextual information of dense camera textures while remaining accurate spatial information of raw 3D point clouds. In addition to MV3D [Chen et al. 2017], Frustum-PointNet [Qi et al. 2018] and AVOD [Ku et al. 2018] as already presented in section 3.2, [Xu et al. 2018] proposed a feature level fusion, where point clouds and images are independently processed by a PointNet [Qi, Su, et al. 2017] and CNN, respectively. Then, a fusion network combines the resulting outputs and finally predicts 3D bounding boxes. More recently, [Liang et al. 2019] proposed a multi-task formulation of 2D and 3D object detection, depth completion, and ground estimation, where lidar and camera features are fused at various levels in a dense fusion scheme. In [Pang et al. 2020], individual candidate detections from lidar and camera are converted into a set of consistent joint detections and post-processed by a 2D CNN for refinement. Similarly, [Yoo et al. 2020] separate the feature generation based on VoxelNet [Zhou et al. 2018] and ResNet

[He et al. 2016] for camera. Features are then projected into birdview representation and processed by a gated feature fusion based on PointNet [Qi, Su, et al. 2017] for proposal generation and refinement.

Overall, a trade-off between efficiency and accuracy can be seen when selecting the architecture and input representation for 3D object detection algorithms. Most current work achieves high accuracy due to the complex proposal and refinement stages. However, it has too much computational effort and too high runtime in order to use them for AD applications. Lightweight methods such as one-stage birdview processing typically have the best runtimes with fewer computations but tend to be less accurate compared to voxels or raw processing.

3.6 Conclusion

This chapter presented the developed model for 3D object detection on point clouds. The model is a single shot CNN, formulating object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. Point clouds are preprocessed into compact birdview images containing three feature channels only. The entire model can be optimized end to end on raw point cloud datasets. Quantitative evaluation of this method on the KITTI benchmark yielded results on par with state of the art while clearly outperforming in terms of efficiency. Although the results are encouraging, the model is subject to a few limitations. First, it can only generate pseudo 3D objects since no object height, and offsets in height dimension are learned. Adding such regression targets strongly affects the overall performance since the model seems to be very sensitive to the interaction of single parameters. Finally, there is a trade-off between efficiency and loss of information when using only three handcrafted features in a regular 2D grid structure as inputs. Instead of processing all points, some potentially beneficial properties of point clouds are harshly trimmed down, comparable to lossy compression. However, current approaches and complex architectures are poorly suited for efficient operation on point clouds. Therefore, the baseline model will be improved in the next chapter before a concept for tracking over time is developed. There, the input representation will be changed to voxels with more channels and supplemented with contextual features from the image space to improve the feature extraction and allow for full 3D detection.

Chapter 4

Joint Object Detection and Tracking on Point Clouds

The method for 3D object detection presented in the preceding chapter, Complex-YOLO, is exclusively based on the point cloud of the current time step. Consequently, predictions of two successive frames often vary considerably. From small deviations in the pose to random rotations of the heading, uncontrolled errors often occur. This indicates uncertainties and network failures when compared with earlier predictions since unnatural behavior is modeled. Similar to video processing, a key to solve this problem lies in the use of spatiotemporal information, well known as tracking. Therefore, in this chapter, the developed algorithm for joint object detection and tracking, namely Complexer-YOLO, is presented. As a baseline, the model described in chapter 3 is updated with new building blocks from state of the art upon completion from section 3.5. Furthermore, a novel fusion with visual features from camera is introduced. Finally, resulting predictions are jointly tracked to stabilize recognized objects and filter outliers, formulated as Multi Object Tracking (MOT).

The first section indicates the usefulness for real-world applications and technical problems. Then, section 4.2 reviews methods from state of the art related to MOT. In section 4.3, the developed model for joint object detection and tracking is presented and evaluated on a real-world benchmark dataset in section 4.4. Finally, section 4.6 gives a summary.

4.1 Motivation

A human driver constantly observes the surroundings carefully while driving. Driving safely requires anticipation and precaution, as mistakes by other participants always have to be expected, and longer breaking distances of several meters may be necessary

depending on the speeds and weather conditions. Therefore, the potential behavior of other road users is continuously estimated, based on prior knowledge or experience, but also, especially with the help of recent motion and flow. Just in a few seconds of observation, velocities and headings can be determined with high accuracy and directly used for decision making. In some cases, particular objects, humans, or animals are actually only perceived by their behavior or movement over time. For instance, cyclists when pedaling or the gait of a human being, contain typical patterns that humans can easily recognize. In both cases, the visual appearance emerges more clearly from the background, and the movements attract much more attention. In a computer, this task can be referred to as tracking. Compared to humans, it is very challenging to recognize highly complex scenarios and find the right associations in a sequence. Most methods in this area utilize the tracking by detection paradigm, where first objects are localized with a detector, after which an algorithm associates them over time. This becomes more challenging due to occlusions, vague or even missing detections, or target interactions, particularly on busy public road scenarios and crowded areas. Despite the difficulty of this task, especially recent deep learning methods enabled great progress in visual recognition and tracking. Detection rates and accuracy have grown enormously, allowing tracking algorithms to better assess appropriate associations. However, it remains a challenging problem, as detectors and trackers still have subtle flaws. Instead of fully utilizing the historical context, as humans do, detector and tracker are fully decoupled during the processing pipeline. For instance, Figure 4.1 visualizes a sequence with decoupled predictions from the algorithm in chapter 3. As can be seen, the detected truck in frame *000080* has a significant discrepancy compared to other time steps and ground truth. Surely, this leads to difficulties for the usage in a tracking algorithm. Overall, there are so far only a few tracking algorithms with a focus on the spatial dimension of point clouds, as it gets even more complicated with another degree of freedom. However, real-world applications such as Autonomous Driving require robust spatiotemporal perception. In essence, existing deep learning based detectors are not sufficient, as meaningful information is supposed to be used from the temporal dimension, similar to humans. Therefore, a broad field of research with great potential has emerged, trying to challenge the complexity of real-world scenarios.

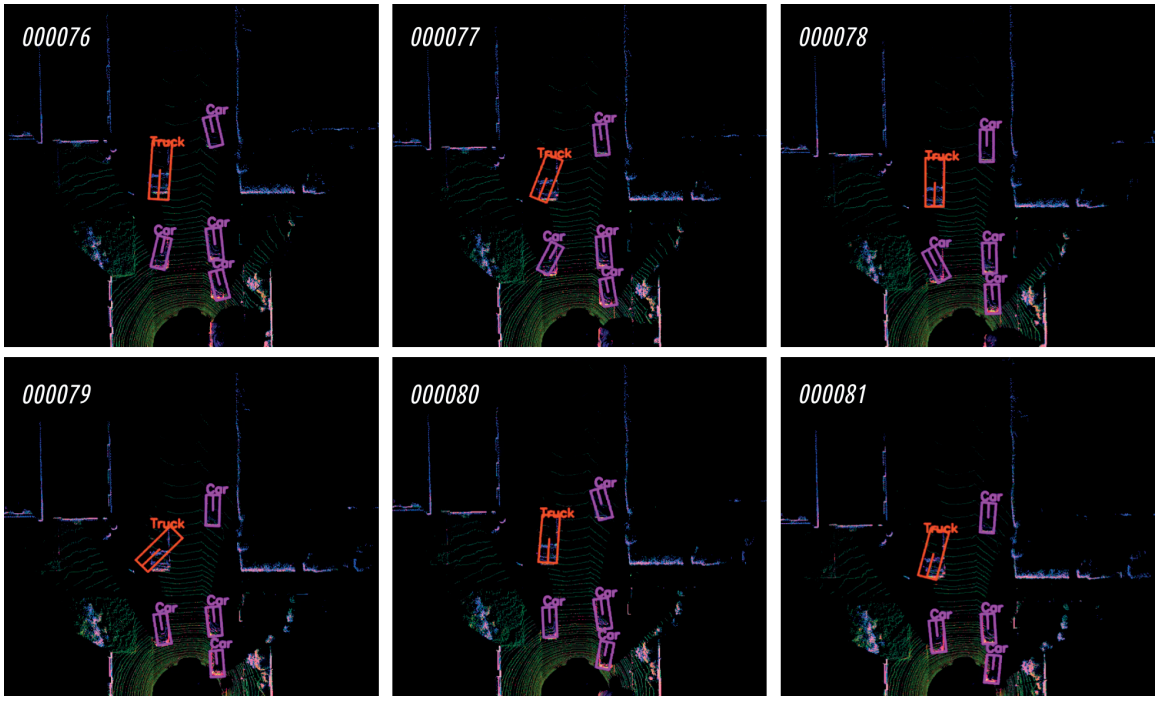


Figure 4.1: *Visualization of Complex-YOLO individually applied to the KITTI raw sequence from 2011-09-26 number 0022 in birdview perspective using the color channels described in subsection 3.3.1. All related objects on the street are constantly detected. However, there are some notable outliers and inaccuracies. For instance, the orientation of the Truck (red bounding box) and some other Cars jitters from one frame to the next. Without further processing, this would often lead to emergency breaking in Autonomous Driving mode to avoid collisions with ghost objects that are predicted into the ego lane.*

4.2 Related Work

This section supplements the theoretical background from chapter 2 and the related work from chapter 3 with a brief summary of existing MOT algorithms and the relevant basics of MOT used for joint object detection and tracking.

Following the aforementioned tracking by detection paradigm, MOT is usually accomplished in two stages. Objects of interest are first detected by an algorithm, and then identical objects are associated over time utilizing object hypotheses from the detector. A common strategy is to leverage global information regarding the detections [Lee et al. 2016; Frossard and Urtasun 2018]. Here, entire sequences are optimized offline with information from both past and future. In contrast, online models must be used for real-time applications, where only the past is known, and outputs are generated for each frame. Recent online approaches focus on the tracking

of 2D objects. In IMMDP [Xiang et al. 2015], a framework based on reinforcement learning with multiple Markov decision processes was demonstrated using monocular camera detections from Faster R-CNN [Ren et al. 2015]. With the help of a binary classifier, pairwise similarity scores are calculated between lost targets and non tracked detections. Then, the Hungarian algorithm [Munkres 1957] is used to predict assignment scores. Unlike IMMDP, [Lenz et al. 2015] formulate a min-cost flow optimization, inspired by [Zhang et al. 2008] and the dynamic shortest path algorithm for data association [Berclaz et al. 2011; Pirsiavash et al. 2011]. Again, the algorithm was evaluated given 2D monocular camera detections from [Wang et al. 2013]. First partial extensions into 3D space were presented in [Scheidegger et al. 2018] and [Sharma et al. 2018]. The former operates on detections from a Convolutional Neural Network (CNN) architecture [Krishnan and Larsson 2016; Yu et al. 2017], extended by the regression of a distance from the camera center to the center of the detected object, trained with smooth L1 loss from [Girshick 2015]. Moreover, detected objects are used as input measurements modeled as Random Finite Set (RFS) [Mahler 2007] to recursively estimate a Poisson Multi Bernoulli Mixture density [Williams 2015]. The latter associates pairwise detections from [Murthy et al. 2017], including 2D bounding boxes, 3D shape, and object poses using key points of discriminative parts. This is done via a likelihood matrix between targets of different time steps, minimizing several cost functions.

The work in this chapter differs from all of these approaches in that the model goes beyond the 2D space and instead directly operates on 3D detections. This comes with higher complexity due to the additional spatial dimension but allows far better interpretability and re-usability for environmental perception in real-world applications. Based on the concepts from [Vo and Vo 2011; Reuter et al. 2014; Granström et al. 2017; Bryant et al. 2018], targets and measurements are modeled as Labeled Multi-Bernoulli Random Finite Set (LMB RFS) (see subsection 2.4.1). Furthermore, object movements are described as coordinated turn motion model [Roth et al. 2014] (see subsection 2.4.3) with innovation calculated using an Unscented Kalman Filter (UKF) [Julier and Uhlmann 1997] (see subsection 2.4.2).

4.3 Model

This section presents the developed algorithm for joint object detection and tracking. A conceptual diagram is visualized in Figure 4.2. Following recent updates from state-of-the-art, the core concepts from chapter 3 are adopted, whereas some details are

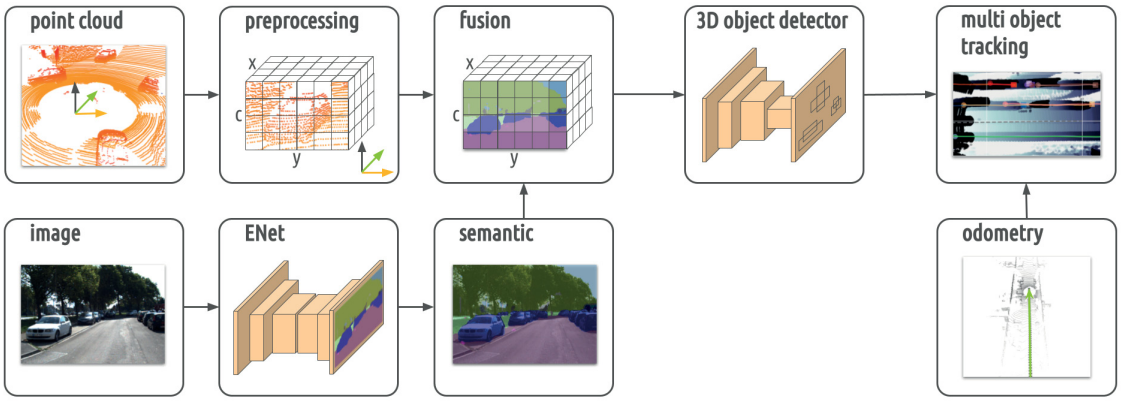


Figure 4.2: *Overview of the joint 3D object detection and tracking pipeline. Point cloud inputs are transformed into regular 3D voxel representation (described in subsection 4.3.2) and fused with contextual features extracted from camera images using ENet [Paszke et al. 2016] (see subsection 4.3.1). These are fed into another single CNN to generate 3D object predictions (described in subsection 4.3.3). Finally, the predicted parameters are threatened as measurements for multi target feature tracking over time using odometry (see subsection 4.3.4).*

refined in order to improve subtle failure modes. Unlike Complex-YOLO, Complexer-YOLO first transforms point clouds into occupancy voxels in order to increase the number of input channels. Inspired by camera fusion approaches from section 3.5, dense visual features from semantic segmentation are projected into the voxel cells to enrich the input feature space further. Although more computations are needed, substantially less information is lost during preprocessing. This is followed by an upgraded version of the CNN from chapter 3, again extended with two additional regression targets representing object heights and height offsets, respectively. Finally, resulting detections are further refined with MOT for temporal consistency and to filter out inaccuracies. In the following subsections, each part is explained in more detail.

4.3.1 Efficient Visual Semantic Segmentation

The developed network to generate semantic maps out of images is built on ENet [Paszke et al. 2016]. ENet adopts ResNet [He et al. 2016], together with PRelu [He et al. 2015b], different forms of convolutions, like regular, dilated, strided or deconvolutions and spatial dropout [Tompson et al. 2015] to efficiently classify pixels. Hence, it was created for low latency operations and aimed for high efficiency in accordance with the designation. First, the network heavily reduces the input sizes

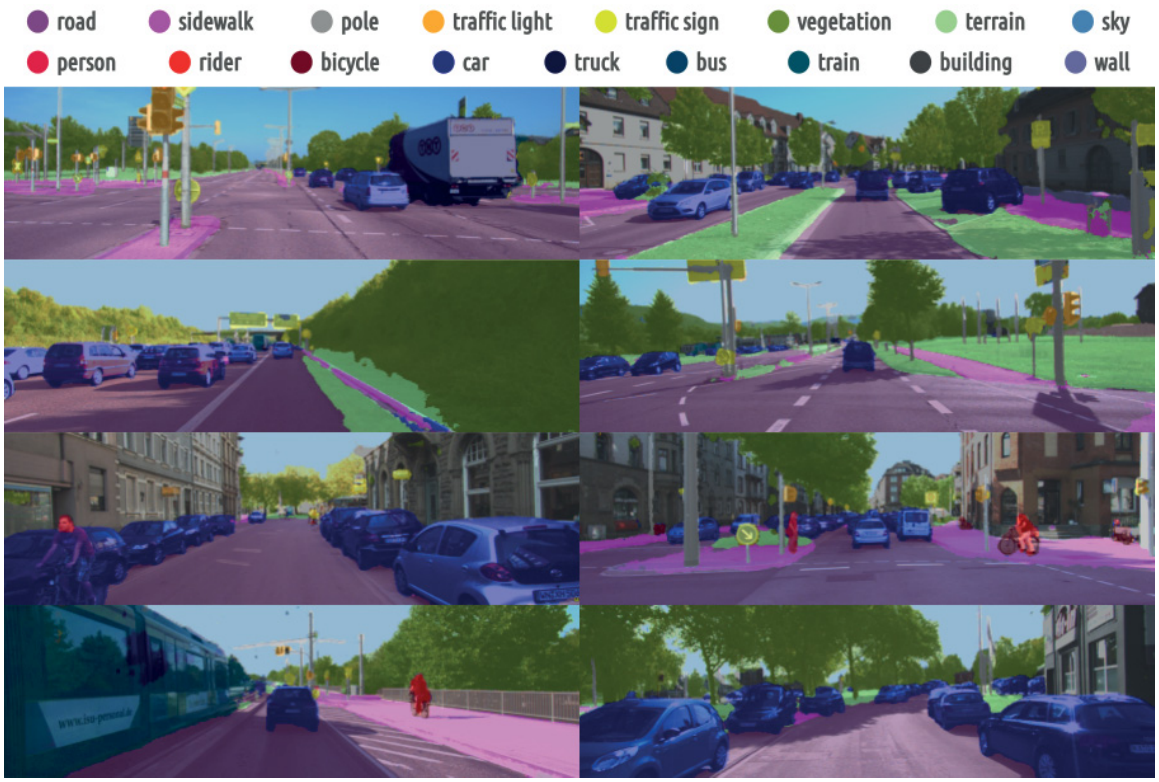


Figure 4.3: *Visual semantic segmentation results from ENet [Paszke et al. 2016] on diverse KITTI samples drawn semi-transparently over the camera images: The network was retrained as described in subsection 4.4.2. Although ENet is very efficient, it achieves high quality in the classification of individual pixels.*

while compressing visual information, using early downsampling. Second, blocks of convolutions are decomposed into smaller ones based on factorization. To further increase the efficiency, the last layer of the network that performs the argmax operation was entirely parallelized using threading. This allows calculations to execute concurrently for every single pixel of the last feature map. In addition, channel pruning was performed, as described in subsection 2.1.3. More details of the training and optimization of the ENet can be found in subsection 4.4.2. Despite a significant reduction of computations and memory requirements, the accuracy decreases only marginally. Resulting semantic maps (see Figure 4.3) are used during preprocessing of the point clouds to fill generated voxels, as described in subsection 4.3.2.

4.3.2 Point Cloud Preprocessing

Similar to subsection 3.3.1, a point cloud $\mathcal{P}$ of a single time frame is cropped to a predefined area of interest Ω, as shown in the first two upper blocks in Figure 4.2.

Instead of a 2D grid, the mapping function $f_{\mathcal{PS}}$ is extended by one dimension with $\mathcal{S} \in \mathbb{R}^{m \times n \times c}$, where c is the number of channels along the height dimension. Hence, each point $\mathcal{P}_\Omega$ with index i gets mapped into a specific voxel cell $\mathcal{S}_j$, as in Equation 3.2. Again, a typical resolution is in range $m = n \in [200, 1334]$ and $c \in [5, 50]$ with $g_z \in [0.10\,m, 1.00\,m]$ for an area of interest of $80\,m \times 80\,m \times 5\,m$. A more detailed configuration with all parameters can be found in the experiments section (see subsection 4.4.2).

After voxels and semantic maps (subsection 4.3.1) are prepared in parallel, they are merged based on point cloud to image projection, as illustrated in the top middle block in Figure 4.2. For all non-empty voxels, every point inside a voxel cell gets projected into pixel coordinates in image space using matrix multiplications. First, points are transformed into camera coordinates based on extrinsic calibrations. This is followed by multiplication with a camera projection matrix based on intrinsic calibrations. The resulting pixel coordinates are then used to address class labels in the semantic map from subsection 4.3.1. Finally, one resulting feature per voxel cell is calculated given the *argmax* over all resolved class indices from all points inside a voxel. Here, a normalized floating value in range $[1, 2]$ is assigned in case the voxel is non-empty and visible to the camera, zero otherwise. The pseudo-code is provided in Algorithm 4.1. In this way, contextual information with visual features from a camera is efficiently fused into the voxel map while only increasing complexity from boolean occupancy to floating-point occupancy plus visual semantic. At the same time, irregular point clouds are quantized into regular voxel grids, allowing to use CNN architectures for further processing. This configuration was found by quantitative experiments to work best compared to pure voxel occupancy, voxels filled with intensity values of the reflectivity from lidar sensors, or a birdview approach (as presented in chapter 3). Further details are described in subsection 4.4.5.

4.3.3 3D Object Detector

In the next step of the processing pipeline, the fused voxels are fed into a network for single-shot 3D object detection. The network from chapter 3 consists of a feature encoder backbone and a Region Proposal Network (RPN) with its specific loss function. Following updates from state-of-the-art, both parts are adapted in this work. Inspired by [Redmon and Farhadi 2018], a hybrid approach between the presented network in subsection 3.3.2 and deep residual networks [He et al. 2016] is used for feature extraction. Hence, the network uses successive blocks of 1×1 and 3×3

Algorithm 4.1 Voxel Generation and Camera Fusion

Require:
points $\mathcal{P}$, image img, rows m, cols n, channels c, stepsize g,
range $x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}$

$grid \leftarrow init(m, n, c) \in \{\}$
$voxel \leftarrow init(m, n, c) \in 0.0$
$semantic \leftarrow \text{ENet}(img)$

for $i \leftarrow 0$ to $(|\mathcal{P}| - 1)$ **do**
  $u \leftarrow (m-1) - round(\frac{\mathcal{P}_{i_x} - x_{min}}{g_x})$
  $v \leftarrow (n-1) - round(\frac{\mathcal{P}_{i_y} - y_{min}}{g_y})$
  if $u \geq 0 \ \wedge \ u < m \quad \wedge \quad v \geq 0 \ \wedge \ v < n$ **then**
    $channel \leftarrow round(\frac{\mathcal{P}_{i_z} - z_{min}}{z_{max} - z_{min}}(c-1))$
    $grid[u][v][channel] \leftarrow grid[u][v][channel] \cup \{\mathcal{P}_i\}$

for $u \leftarrow 0$ to $(m-1)$ **do**
  for $v \leftarrow 0$ to $(n-1)$ **do**
    for $channel \leftarrow 0$ to $(c-1)$ **do**
      $\mathcal{P}_{uvk} \leftarrow grid[u][v][channel]$
      if $|\mathcal{P}_{uvk}| > 0$ **then**
        $\mathcal{P}_{\text{cam}} \leftarrow \text{CamProj}(\mathcal{P}_{uvk})$
        if $|\mathcal{P}_{\text{cam}}| > 0$ **then**
          $voxel[u][v][channel] \leftarrow norm(mode(semantic[\mathcal{P}_{\text{cam}}]))$
return voxel

convolutions with residual shortcut connections. Strided convolutions are utilized for downsampling instead of max-pooling layers. Throughout the network, leaky ReLu activation is used, except for the last layer. The overall architecture is significantly larger compared to subsection 3.3.2, as shown in Table 4.1. There are 49 convolutional layers in total, instead of 59 as in [Redmon and Farhadi 2018]. As found by experiments, the multi-scale prediction introduced by [Redmon and Farhadi 2018] decreases the performance of the network. Unlike the image space, where objects appear with varying scales, the voxel (birdview) representation of point clouds keeps the metric space with priors about the physical dimensions of objects. Therefore, the developed network only works with single-scale predictions. As a result, the aforementioned 10 convolutional layers are removed at the end of the architecture without losing performance. On top of that, less filter kernels are required in early layers to extract meaningful features.

Based on the expanded number of input channels and enhanced capacity in the

Layer	Output Size	Filter Size	Stride	Activation
input	$m \times n \times c$	-	-	-
conv1	$m \times n \times 21$	$3 \times 3, 21$	1	leakyReLu
conv2	$\frac{m}{2} \times \frac{n}{2} \times 42$	$3 \times 3, 42$	2	leakyReLu
conv3_x	$\frac{m}{2} \times \frac{n}{2} \times 64$	$\begin{bmatrix} 1 \times 1, 32 \\ 3 \times 3, 64 \end{bmatrix} \times 1$	1	leakyReLu
conv4	$\frac{m}{4} \times \frac{n}{4} \times 128$	$3 \times 3, 128$	2	leakyReLu
conv5_x	$\frac{m}{4} \times \frac{n}{4} \times 128$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	1	leakyReLu
conv6	$\frac{m}{8} \times \frac{n}{8} \times 256$	$3 \times 3, 256$	2	leakyReLu
conv7_x	$\frac{m}{8} \times \frac{n}{8} \times 256$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 256 \end{bmatrix} \times 8$	1	leakyReLu
conv8	$\frac{m}{16} \times \frac{n}{16} \times 512$	$3 \times 3, 512$	2	leakyReLu
conv9_x	$\frac{m}{16} \times \frac{n}{16} \times 512$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 512 \end{bmatrix} \times 8$	1	leakyReLu
conv10	$\frac{m}{32} \times \frac{n}{32} \times 1024$	$3 \times 3, 1024$	2	leakyReLu
conv11_x	$\frac{m}{32} \times \frac{n}{32} \times 1024$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 1024 \end{bmatrix} \times 2$	1	leakyReLu
conv12	$\frac{m}{32} \times \frac{n}{32} \times f_a$	$1 \times 1, f_a$	1	linear

Table 4.1: *The architecture of Complexer-YOLO. Building blocks with residual connections are shown in brackets, with the numbers of blocks stacked. Downsampling is performed with strided convolutions. The number of filters in the last convolutional layer conv12 is denoted by f_a.*

backbone network, the RPN is extended with the following aspects. First, additional regression parameters are added for the height of objects h and the offset from the ground z, respectively. In this way, the network learns fine nuances instead of the rough approximation with class-wise fixed height values from the former chapter. Differences in the elevation level are mainly observed on steep inclines or downhill slopes in the surrounding area, where offsets of several meters can occur at longer distances. Therefore, the multi-part loss $\mathcal{L}$ is extended by $\mathcal{L}_{\text{Z}}$, the squared errors of the object height h and coordinate z between ground truth and prediction:

$$\mathcal{L} = \mathcal{L}_{\text{Yolo}} + \mathcal{L}_{\text{Euler}} + \mathcal{L}_{\text{Z}} \tag{4.1}$$

$$\mathcal{L}_{\mathrm{Z}} = \lambda_{\mathrm{coord}} \sum_{i=0}^{\frac{m \cdot n}{32}} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(h_i - \hat{h}_i)^2 + (z_i - \hat{z}_i)^2 \right] \tag{4.2}$$

where λ_{coord} denotes a scaling factor to ensure stable convergence in early phases, i is the index over all grid cells, j is the index of object predictions with B the number of object predictions per cell. Then, the developed score S_{srt} is used to define $\mathbb{1}_{ij}^{obj}$ instead of Intersection Over Union (IOU). This is a profound adaptation, as the whole training process is supervised and thus directly depends on the comparison with ground truth. On the one hand, IOU ignores opposite orientations of two objects modulo π. Though estimated orientations are well defined through the complex space, using IOU, the network is encouraged to predict objects rotated by π, since wrong predictions are not penalized. On the other hand, calculating the IOU for rotated bounding boxes in 3D requires several operations. This results in heavy computations, considering the huge amount such calculations are invoked during training and inference. Thus, inspired by affine transformations, the S_{srt} score between two arbitrary 3D objects described as bounding boxes A and B can be formulated as a composite of independent scores for scaling S_s, rotation S_r and translation S_t, defined as follows:

$$S_s = 1 - \min\left(\frac{3 - (s_x + s_y + s_z)}{w_s}, 1 \right), \quad w_s \in (0, 1] \tag{4.3}$$

$$s_i = \begin{cases} s_i, & \text{if } s_i < 1 \\ \frac{1}{s_i}, & \text{otherwise} \end{cases}, \quad i \in \{x, y, z\} \tag{4.4}$$

$$S_r = \max\left(1 - \frac{\theta}{w_r \pi}, 0\right), \quad w_r \in (0, 1] \tag{4.5}$$

$$\theta = \pi - \Big| |\theta_x| - \pi \Big| + \pi - \Big| |\theta_y| - \pi \Big| + \pi - \Big| |\theta_z| - \pi \Big| \tag{4.6}$$

$$S_t = \max\left(1 - \frac{\|t\|^2}{w_t(diag(A) + diag(B))}, 0\right), \quad w_t \in (0, 1] \tag{4.7}$$

where s_i denotes ratios of the sizes of the objects in x, y, z dimensions, θ denotes the sum of the difference from the orientation angles $\theta_{x,y,z}$, t is the vector of the difference between the two objects centers, and $diag()$ denotes the length of the spatial diagonals. The weighting factors w_s, w_r, and w_t can be used to control how strict each of the individual scores is, whereas towards 0 is more strict. S_t is calculated

with respect to the sizes of the two objects using the diagonals. For tiny objects, small translations already have a bigger impact and vice versa for large objects.

All previous scores S_s, S_r, S_t are limited to $[0, 1]$ and can be combined into the final score S_{srt} using a weighted average:

$$S_{srt} = \alpha\, S_s + \beta\, S_r + \gamma\, S_t, \quad \alpha + \beta + \gamma = 1 \tag{4.8}$$

where α, β, γ are the weights of the three sub scores, respectively.

In essence, the developed network operates on voxelized point clouds fused with semantic features from a camera, as described in subsection 4.3.2, extracts features based on the adapted CNN architecture, and detects 3D objects in one forward pass. Remaining parts are inherited from chapter 3. The detector can be optimized end to end, while using the S_{srt} score for object to object comparison with predefined weights w_s, w_r, w_t and α, β, γ.

4.3.4 Multi Object Tracking

In the last part of the developed model, predictions are tracked based on the LMB RFS [Reuter et al. 2014; Bryant et al. 2018] approach (see subsection 2.4.1). This step is related to the right upper block in Figure 4.2 and requires additional odometry to compensate the ego motion. Hence, 3D bounding boxes, i.e. center coordinates $c = [x, y, z]$, dimensions $s = [l, w, h]$ and yaw orientation ϕ, are interpreted as Gaussian noise corrupted measurements z_t^i, $i \in \{1, \ldots, N_t^z\}$ with $z = [c, s, \phi]$ of the positional parameters (see Equation 2.8), extended by x_t^i, $i \in \{1, \ldots, N_t^x\}$ as extended targets [Granström et al. 2017] (see Equation 2.7), with the measurement noise covariance matrix:

$$R = \begin{bmatrix} 0.5^2 & 0 & \ldots & 0 \\ 0 & 0.5^2 & \ldots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \ldots & 0.1^2 \end{bmatrix} \tag{4.9}$$

Target movements are assumed based on the coordinated turn motion model [Roth et al. 2014] (see subsection 2.4.3). Here, the process noise covariance consisting of the standard deviation of the acceleration σ_a and the yaw rates derivative σ_α is:

$$Q = \begin{bmatrix} \sigma_a^2 & 0 \\ 0 & \sigma_\alpha^2 \end{bmatrix} \tag{4.10}$$

Following the coordinated turn model, the extended target state mean $\bar{x}_t^i$ contains the same parameters as the measurements z as well as the motion parameters velocity v and yaw rate $\dot{\phi}$. Hence, the ith target at time t can be described as:

$$\bar{x}_t^i = [c_t^i, s_t^i, \phi_t^i, v_t^i, \dot{\phi}_t^i] \tag{4.11}$$

with state covariance matrix $\bar{P}_t^i$. Based on the measurement equation:

$$z = H \cdot \bar{x} \tag{4.12}$$

with measurement matrix $H = (\mathrm{I}_7 \quad 0) \in \mathbb{R}^{7\times 9}$, where I_7 is the identity matrix of dimension 7, a Bayesian filter can be formulated, using an Unscented Kalman Filter (UKF) (see subsection 2.4.2) for innovation. Here, the application of an UKF for the nonlinear motion model is desirable, while the update for the linear measurement model can also be simplified to a default Kalman Filter. During an update step, each predicted target is associated with each measurement at that step, where the update is performed according to the defined measurement model. Based on the update likelihood, association probabilities on which targets are kept or discarded can be modeled in a heatmap. Formally, the association probability $p_a(x^i, z^j)$ of the measurement z^j and the state x^i can be used to formulate the non-assignment probability as:

$$p_{na}(z^j) = 1 - \sum_{x^i \in \mathbf{X}} p_a(x^i, z^j) \tag{4.13}$$

Given a threshold P_{na}, the birth of a new target from an unexplained measurement can be assumed, if $p_{na} > P_{na}$. From the mean of the cardinality distribution in Equation 2.12, the number of targets to be extracted N_e can be derived. Finally, all targets are filtered according to their existence probabilities $r^{(l)}$ and extracted as final outputs of the model.

4.4 Experiments

In the following experiments, the developed model that uses voxelized point cloud inputs, the presented specific CNN and multi-object tracking are evaluated. First, underlying datasets, as well as details for training and optimization, are presented. Second, the evaluation metrics and both quantitative and qualitative results for multi-object tracking are reported. Finally, an ablation study provides more details w.r.t. the proposed S_{srt} score and voxel input features fused with semantic textures.

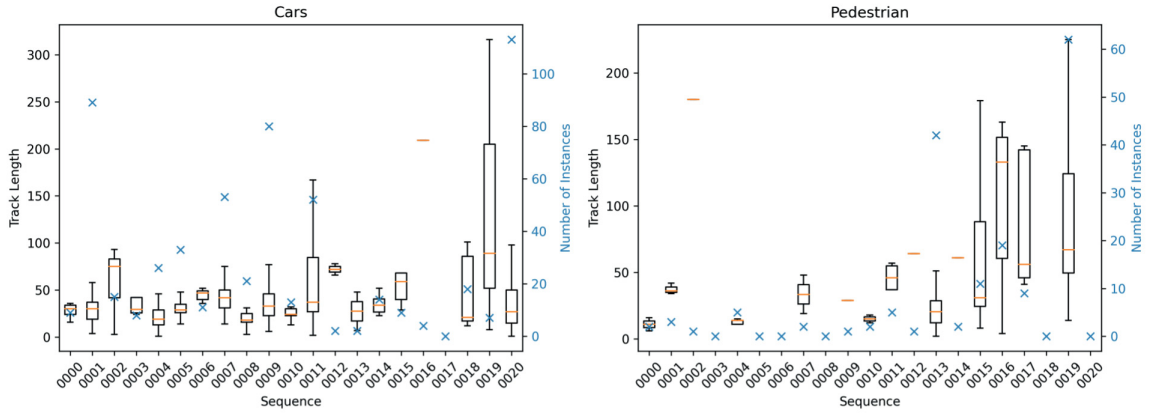


Figure 4.4: *Statistics of the training sequences in the KITTI tracking dataset for the classes Car and Pedestrian: A box plot visualizes a summary of the length of existing tracks over time without outliers for better visibility. This is complemented by the number of object instances for each individual sequence (blue).*

4.4.1 Datasets

In addition to the object detection datasets mentioned in subsection 3.4.1, the KITTI object tracking dataset [Geiger et al. 2012] is used to assess the performance of the proposed joint object detection and tracking method. This dataset consists of 21 sequences for training and 29 sequences for testing with annotated 3D bounding box instances over time for the classes *Car* and *Pedestrian*. Like object detection datasets, ground truth is only publicly available for training sequences. Therefore, only the training data is analyzed here. The evaluation of the test sequences is done directly on a web server by uploading final detections. More details can be found in section B.3.

Overall, there are $8,008$ samples for training with 579 instances of *Cars* as well as 167 instances of *Pedestrians*, and $11,095$ samples for testing, respectively. In Figure 4.4, the distribution of all training sequences is presented. As can be seen, most sequences differ significantly from each other in the number of occurring object instances as well as in the length of individual tracks. Some sequences contain instances of only one of the two classes. Furthermore, the length of a sequence varies from 78 to $1,059$ frames, resulting in the duration of tracks ranging from 0 to 643 frames and approximately 60 frames on average.

On top, the Cityscapes dataset [Cordts et al. 2016] is used to train the network described in subsection 4.3.1, as well as the KITTI semantic segmentation dataset [Alhaija et al. 2018] for fine-tuning. Jointly the two datasets contain $5,200$ samples

with dense pixel-wise annotations. More details about these datasets are addressed in Appendix B.

4.4.2 Training and Optimization Details

To optimize the Complexer-YOLO model, several training stages are carried out. First, the image pipeline for the extraction of semantic features is trained. Using the same set of optimization parameters as in [Paszke et al. 2016], the modified ENet is pre-trained on Cityscapes [Cordts et al. 2016]. Next, the training is continued for 10 epochs on the KITTI semantic segmentation dataset [Alhaija et al. 2018] in order to fine-tune the network in the target domain. Again, the same training procedures and optimization parameters are used as in the original version. Thus, the weights are fixed, and this network can be used to generate the semantic maps in the input images.

Second, point cloud inputs are transformed into voxel representation, as described in subsection 4.3.2, where $x \in [0m, 60m], y \in [-40m, 40m], z \in [-2.73m, 1.27m]$ with $g = 0.08m \times 0.08m \times 0.19m$, resulting in $m = 768, n = 1024, c = 21$ input features. At the same time, the corresponding generated semantic maps are inserted as final input for the object detection network. This network is trained from scratch for $140k$ iterations, with a step wise decayed learning rate at $20k$, $80k$ and $120k$ iterations, respectively. These parameters are adjusted through quantitative testing. Furthermore, the S_{srt} score parameters are adjusted to $w_s = 0.3$, $w_t = 1.0$ and $w_r = 0.5$ to be more sensitive for errors in scale or orientation compared to translation. The sub score weights are defined as $\beta = 0.4, \alpha = \gamma = 0.3$, with slightly increased weight for orientation. An exhaustive comparison of S_{srt} to IOU can be found in subsection 4.4.5. All other parameters like the learning rate, batch size and so on are similar to subsection 3.4.2, except the number of classes. Here, the classes *Misc* and *Tram* were removed due to the strong imbalance in the tracking dataset and the few *Sitting Persons* are merged with *Pedestrians*. So, there are 5 classes in total resulting in $f_a = 70$ filters of the last convolutional layer. Additionally to global yaw rotation, random flipping along the x axis is used for augmentation, adopted from [Yang et al. 2018].

Lastly, for tracking of the outputs from the object detection network, the noise of the motion turn model described in subsection 4.3.4 is defined as $\sigma_a = 18.89$ and $\sigma_\alpha = 1.49$, found by quantitative testing. In this way, the tracker is emphasized to put more weight on measurements from the network instead of model priors.

4.4.3 Applied Evaluation Metrics

To assess the performance of the proposed joint object detection and tracking approach, the HOTA [Luiten et al. 2020] and CLEAR MOT [Bernardin and Stiefelhagen 2008; Li et al. 2009] evaluation metrics are reported. These metrics are commonly used in MOT and include Higher Order Tracking Accuracy (HOTA), Detection Accuracy (DetA), Association Accuracy (AssA), Detection Recall (DetRe), Detection Precision (DetPr), Association Recall (AssRe), Association Precision (AssPr), Localization Accuracy (LocA) as well as Multi-Object Tracking Accuracy (MOTA), Multi-Object Tracking Precision (MOTP), Id Switches (IDs), False Positives (FP), False Negatives (FN), Mostly Tracked (MT), Mostly Lost (ML) and Trajectory Fragments (Frag). MOTA, MOTP, and especially HOTA combine the subtasks of accurate detection, localization, and association into single combined metrics. All these metrics are integrated into the KITTI benchmark suite [Geiger et al. 2012] and can be performed for the test dataset via an upload of the predictions with a predefined format. Further details and definitions can be found in Appendix C.

4.4.4 Results

The overall runtime of the model is 11.5 frames per second, measured with an experimental implementation on a GTX1080Ti GPU and Intel Xeon CPU. All aforementioned metrics are reported in Table 4.2 and Table 4.3 on the KITTI tracking test dataset [Geiger et al. 2012], together with the cited performance of approaches from [Sharma et al. 2018], [Xiang et al. 2015], [Scheidegger et al. 2018] and [Lenz et al. 2015], but these models are based on inputs from cameras. Since the evaluation also takes place in the image plane based on 2D IOU, all tracked 3D detections have to be back-projected. This explains the inferior performance of Complexer-YOLO, because every single task, detection, localization, and association, is substantially more challenging with an additional spatial dimension. Nevertheless, the proposed model achieves remarkable results without working natively in the image domain. For instance, three out of four image methods yield lower performance than the achieved Association Precision (AssPr) of 85.23%, indicating well-predicted trajectories keeping track to ground truth. Unlike all other methods, the developed model is capable of recognizing and tracking several classes simultaneously. However, for small objects like *Pedestrians*, the level of difficulty is significantly higher, so the achieved results also drop dramatically.

Method	Type	HOTA	DetA	AssA	DetRe	DetPr	AssRe	AssPr	LocA
[Sharma et al. 2018]	2D	63.75 %	**72.87 %**	56.40 %	**76.58 %**	**85.38 %**	59.05 %	**86.70 %**	**86.90 %**
[Xiang et al. 2015]	2D	**68.66 %**	68.02 %	**69.76 %**	71.47 %	83.28 %	**74.50 %**	82.02 %	84.80 %
[Scheidegger et al. 2018]	2D	59.12 %	65.43 %	54.28 %	69.87 %	80.68 %	57.28 %	83.89 %	83.94 %
[Lenz et al. 2015]	2D	50.92 %	58.57 %	44.51 %	63.69 %	75.67 %	46.47 %	81.23 %	81.44 %
CY (Car)	3D	49.12 %	62.44 %	39.34 %	67.58 %	76.86 %	40.72 %	85.23 %	81.47 %
CY (Pedestrian)	3D	14.08 %	24.91 %	8.15 %	27.21 %	52.62 %	8.63 %	59.39 %	68.64 %

Table 4.2: *Quantitative tracking results of Complexer-YOLO (CY) on the KITTI tracking test dataset [Geiger et al. 2012], based on the HOTA tracking metrics from [Luiten et al. 2020] (see section C.2), in comparison with state of the art camera-based online tracking approaches.*

Method	Type	MOTA↑	MOTP↑	IDs↓	FP↓	FN↓	MT↑	ML↓	Frag↓
[Sharma et al. 2018]	2D	**82.68 %**	**85.50 %**	934	**4283**	741	**72.61 %**	**2.92 %**	581
[Xiang et al. 2015]	2D	82.75 %	82.78 %	**211**	5300	**422**	60.31 %	12.15 %	**201**
[Scheidegger et al. 2018]	2D	79.23 %	81.58 %	485	5634	1024	62.77 %	6.46 %	554
[Lenz et al. 2015]	2D	70.78 %	78.78 %	770	7360	1918	49.23 %	9.38 %	847
CY (Car)	3D	72.61 %	78.49 %	1952	5809	1658	56.92 %	5.85 %	1103
CY (Pedestrian)	3D	11.99 %	62.31 %	1555	15000	3820	2.41 %	38.49 %	1649

Table 4.3: *Quantitative tracking results of Complexer-YOLO (CY) on the KITTI tracking test dataset [Geiger et al. 2012], based on the CLEAR MOT metrics from [Bernardin and Stiefelhagen 2008] (see section C.2), in comparison with state of the art camera-based online tracking approaches.*

Furthermore, HOTA results are visualized in Figure 4.5 with plots for the classes *Car* (left) and *Pedestrian* (right). For both *Car* and *Pedestrian*, Complexer-YOLO is rather conservative with higher precision, but lower recall as some objects are not detected or associated. However, only moderate performance is achieved for *Pedestrian*, as both recognition from super sparse point clouds and association in 3D are very challenging. In both cases, the performance quickly drops with an IOU threshold α above 0.6 for *Car*, and already from 0.4 for *Pedestrians*. This indicates inaccuracies in the predicted bounding boxes but also introduced from back-projection into image space, where the evaluation takes place. With roughly 0.15 (*Car*) and 0.50 (*Pedestrian*) False Negatives, not all objects from ground truth are detected even for very low IOU thresholds (α), as seen in Detection Recall (DetRe). In contrast, Localization Accuracies (LocA) highlight the higher accuracies for true positive detections with approximately 0.81 for *Car* and 0.69 for *Pedestrian*. On top, there are

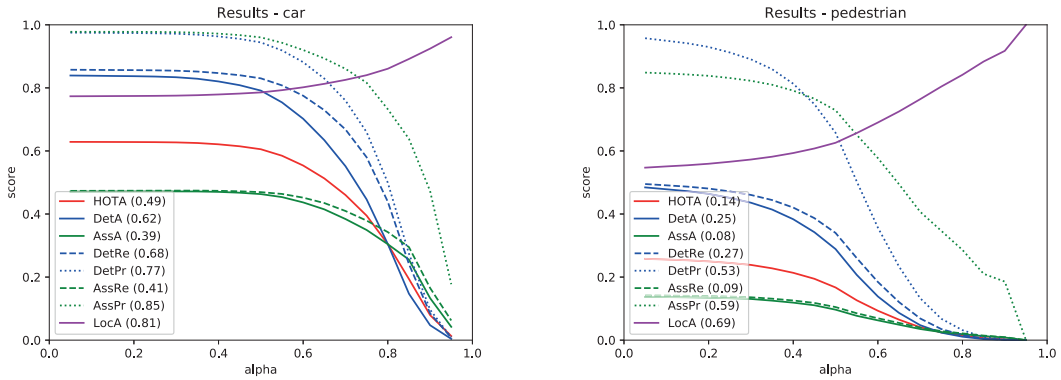


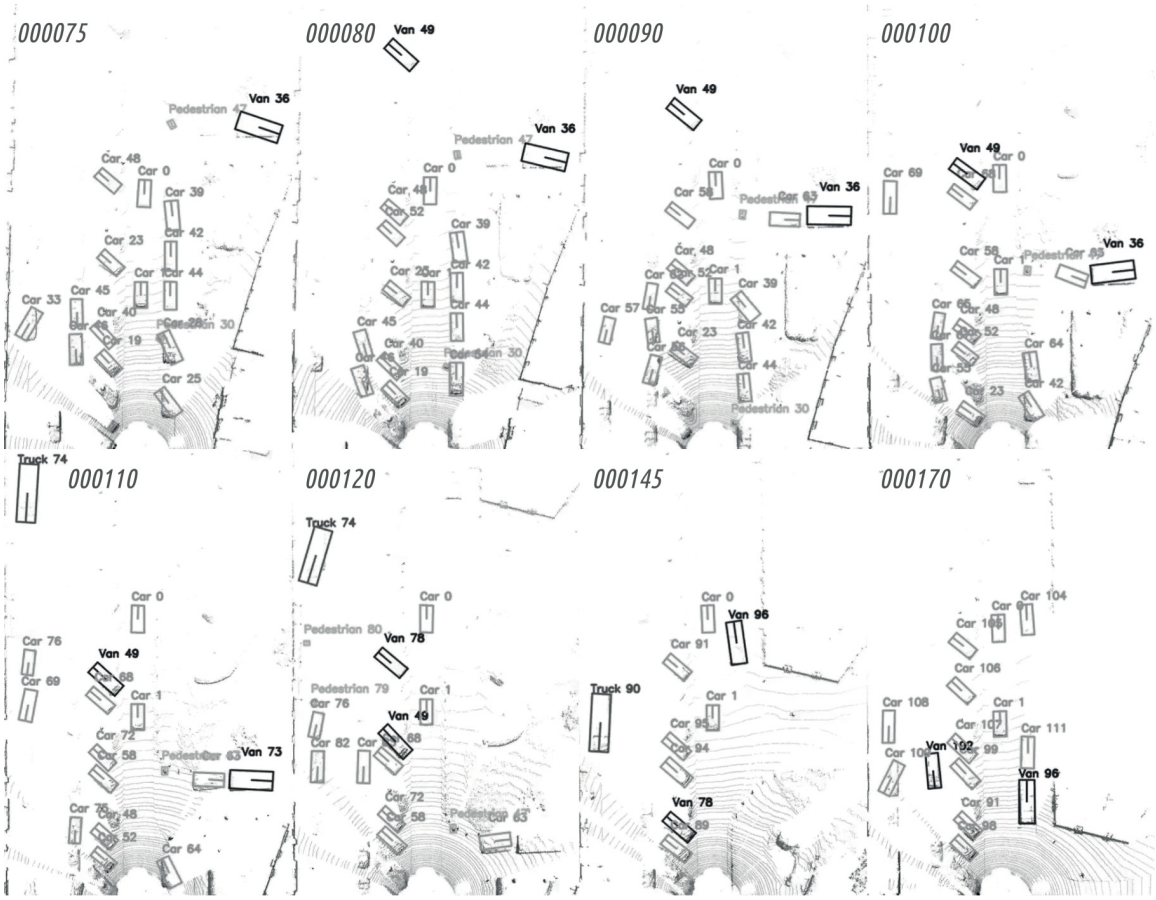
Figure 4.5: *Analysis of the tracking performance of the presented approach on the KITTI tracking test dataset [Geiger et al. 2012] for the classes Car (left) and Pedestrian (right), based on the HOTA tracking metrics [Luiten et al. 2020] (described in subsection 4.4.3 and section C.2). The approximated area under the respective curve is given in brackets, while alpha (α) corresponds to the used IOU thresholds. More details are discussed in subsection 4.4.4. Reproduced from* ***http://www.cvlibs.net/datasets/kitti/eval_tracking.php*** *.*

very few false-positive associations, as the Association Accuracy (AssA) is similar to the Association Recall (AssRe). Likewise, the detections with Detection Accuracy (DetA) scored only slightly below the Detection Recall (DetRe). Nevertheless, due to short-term fragmentations and Identity Switches, there are several false-negative associations affecting the association performance. Hence, Association Recall (AssRe) and Association Accuracy (AssA) scores the worst with $0.41, 0.39$ for *Car* and $0.09, 0.08$ for *Pedestrian* respectively. In total, the calculated HOTA scores are 0.49 (*Car*) and 0.14 (*Pedestrian*).

As can be seen from the example sequence in Figure 4.6, even in such a heavily crowded environment, the developed model discovers the majority of objects over a duration of 11.5 seconds. Here, the tracking compensates for temporary occlusions and outliers from single-shot detection, whereby point clouds contain almost zero points in some regions. This also helps to stabilize the correct estimation of the orientation of individual objects compared to single-shot detection. However, missing or false detections lead to a few identifier switches while also decreasing the Association Precision (AssPr), because predicted tracks become fragmented.

Another example is visualized in Figure 4.7. The figure shows tracked trajectories of a short highway sequence with traffic jam from KITTI in top-view perspective. Here, the ego vehicle is driving at low speed for a few seconds in the right lane (dashed trajectory), while other vehicles in all three regular lanes come to a standstill (colored

Figure 4.6: *Qualitative tracking results of Complexer-YOLO on the KITTI raw sequence from 2011-09-26 number 0104 in grayscale. The figure is divided into two parts, second one (b) on the next page, to improve readability.*
a) *Both images in the top row provide an overview of the scene depicted at two time steps (000055 and 000090) corresponding to a frame rate of* $10\,Hz$*. In contrast, the other rows contain different point cloud frames from a successive sequence in birdview perspective starting at 000055. Here, bounding boxes, class labels, and unique tracking identifiers predicted by the presented approach are visualized. Besides the complexity of the scene with many different objects, partial or complete occlusion is the biggest challenge as it heavily affects the detection step in the processing pipeline. In particular, wrongly interpreted orientations causing incorrect tracks afterwards, e.g. Car* 22 *in 000055, Van* 5 *in 000060, Car* 33 *in 000075 and so on. In contrast, many objects such as the Cars driving in front of the ego vehicle with identifiers* 0 *or* 1*, even the Pedestrians* 30 *or* 47 *are detected and tracked throughout their existence. The detection range is considerably reduced for smaller objects so that Pedestrians are reliably detected only up to a distance of approximately* $25\,m$*. Although Pedestrians at longer ranges are sporadically detected, the tracker filters conservatively in most cases because the accuracy of these detections is not sufficient. Considering the extremely small number of points in the point clouds in such regions, a fairly good performance can be achieved. Especially up to mid-range objects in the direct field of view of the ego vehicle are tracked robustly, e.g. Cars* $15, 16, 25, 27$ *or* 28 *from 000055 onward.*



b) *The second part of Figure 4.6 shows the continued sequence with some challenges in the permanent association of inaccurate or missing single shot detections. For instance, the Van on the right with identifier* 36 *switches to* 73 *from frame 000100 to 000110. Another example is between frame 000120 and 000145, where the track of the Truck with identifier* 74 *is lost for a short period and gets reborn as identifier* 90*. In both cases, the objects are completely occluded by tall parking vehicles on the sides of the road for a few frames while driving through this passage.*

trajectories). At the same time, two vehicles in front switched to the emergency lane and kept rolling (overlaying green and bright blue trajectories). Despite heavy occlusions introduced by the first row of vehicles in the middle lane, most trajectories are reliably detected in the left and middle lanes. However, sometimes the tracker cannot fully compensate for missing detections even in this sequence and creates new tracks visible as very short and fragmented trajectories. Nevertheless, the developed tracking greatly improves the temporal consistency compared to single-shot detection.

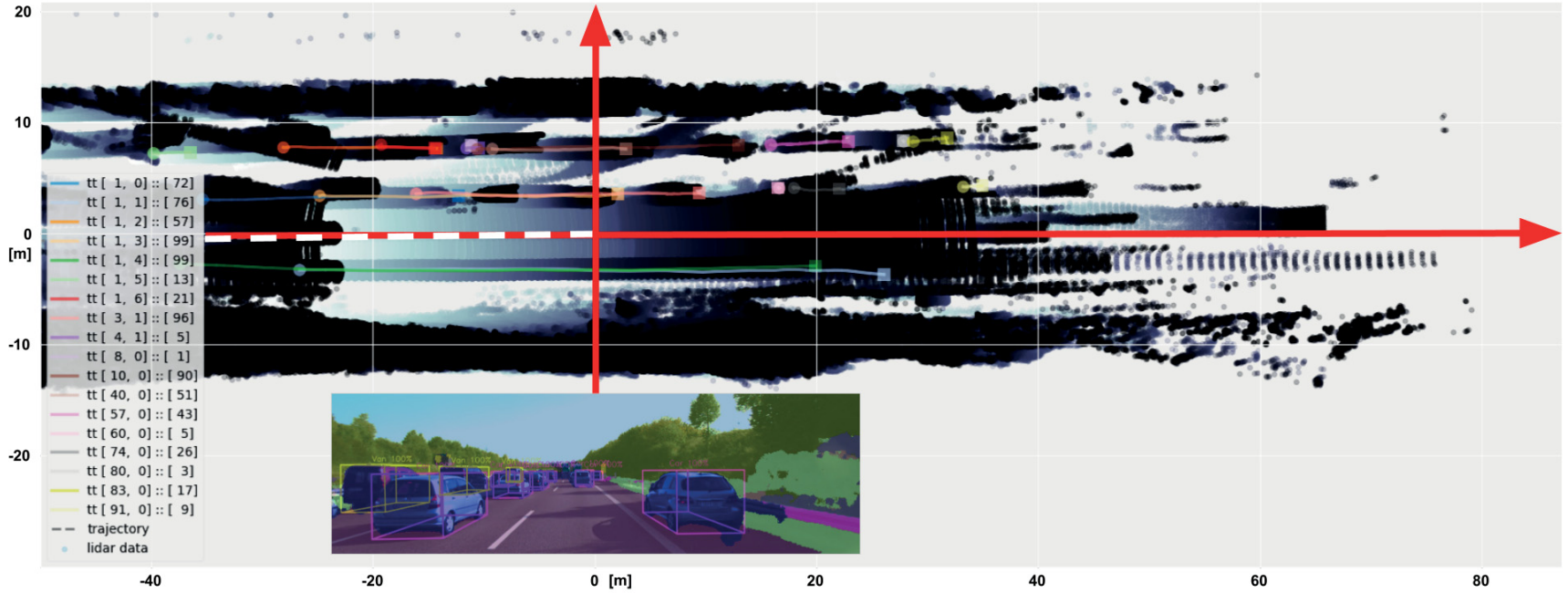


Figure 4.7: *Visual results of tracked trajectories on one of the KITTI tracking sequences for a highway scenario: The dashed line in white represents the ego movement from past up to the current frame centered at the origin (red coordinate axes). Colored lines correspond to tracked objects with a circle indicating birth and a square for death, respectively. The legend is according to the format: target track* $[T, N] :: [\Delta T]$*, where* T *is the time frame of birth,* N *is a counter for born tracks per frame and* ΔT *is the length of the track. In addition, the background of the whole image is filled with color, representing the density of scan points from lidar accumulated over the full sequence, where white denotes no points and black indicates many superimposed points. Note that all brighter regions correspond to very shortly seen or unseen surroundings from the lidar sensor with fewer points due to occlusion. In parallel to the ego trajectory, there are particularly sparsely populated regions due to other driving vehicles, making the detection and tracking challenging. The camera image shows one of the last frames for reference overlaid with the detected segmentation map and bounding boxes. Reproduced and modified from [Simon et al. 2019].*

Feature	IOU 0.7	SRT 0.7
Birdview	28.64	30.02
Occupancy	31.93	33.24
Intensity	32.39	33.57
Semantic	**34.14**	**35.43**

Table 4.4: *Ablation study of different input feature representations with mAP values (in %) reported for 3D object detection on the KITTI validation set. Reproduced from [Simon et al. 2019].*

4.4.5 Ablation Study

In this section, an extensive ablation study is presented to determine the contribution of several design decisions. Further experiments are performed on a fixed setup for training, based on the KITTI datasets introduced in subsection 3.4.1. First, the resolution of the voxel input channels c is analyzed starting from cubic voxels with $g = 0.08\,m$ resulting in $c = 50$ height channels, down to $g = 0.40\,m$ with $c = 10$. It seems the network is not able to utilize fine-grained features, or degraded density hinders the generation of meaningful features. Therefore, $c = 21$ is found as the best trade-off between accuracy and runtime.

In addition, several input features are tested and reported in Table 4.4, instead of voxels filled with semantic features from camera. Here, the mean Average Precision (mAP) values present the 3D object detection performance averaged over the classes *Car*, *Pedestrian* and *Cyclist*, when using voxels with occupancy, intensity, i.e. the calibrated reflectivity from lidar sensors (see Appendix A), or fused semantic features. Plus, the first row is related to a birdview input with 3 channels only, as presented in chapter 3. All runs are individually trained and repeated twice, once with IOU, once with S_{srt}. In both cases, the threshold for correct detections is set to 0.7 for all classes. Although the generated semantic maps contain subtle flaws with wrongly classified pixels, this feature adds about 2% mAP. This gain mostly comes from improvements of small objects such as *Pedestrians*, where the visual context from images is utilized. Furthermore, S_{srt} gives 1.3% improvement on mAP as well as up to 20% speedup during inference and halved the time required for training. An exhaustive assessment of the S_{srt} score in comparison to the commonly used IOU is provided in Appendix D.

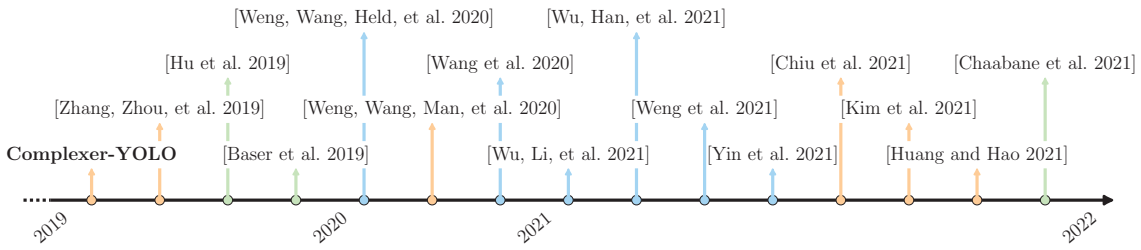


Figure 4.8: *Systematic timeline of related work upon Complexer-YOLO [Simon et al. 2019]: All presented methods aim to track in 3D space while using different inputs: camera and lidar (orange), lidar only (blue) and camera only (green).*

4.5 Related Work upon Complexer-YOLO

In parallel to the method developed in this chapter, several other approaches have been proposed for MOT with respect to point clouds and 3D processing, after it was published in [Simon et al. 2019]. An overview is given in Figure 4.8. For instance, [Hu et al. 2019] proposed a motion learning module based on Long Short Term Memory (LSTM) for MOT of vehicles in monocular images. Similarly, [Chaabane et al. 2021] proposed a joint detection and tracking using an LSTM architecture to filter physically implausible matches. In [Baser et al. 2019], a feature association network based on a CNN and cosine similarity maps are used to learn object associations fully.

In contrast, some works only operate on point cloud inputs such as [Weng, Wang, Held, et al. 2020], where lidar detections in 3D are predicted, followed by a Kalman filter [Kalman 1960] and Hungarian algorithm [Munkres 1957]. Furthermore, [Wu, Li, et al. 2021] presented a tracklet proposal network to first generate proposals, and then refine and associate them for MOT, where the refinement uses consistency constraints in the spatiotemporal feature space. [Wu, Han, et al. 2021] proposed a data association guided by prediction confidences generated from a constant acceleration motion model, while [Weng et al. 2021] developed a joint 3D tracking and motion forecasting based on a Graph Neural Network (GNN) and generative trajectory prediction. Two different approaches have been presented by [Wang et al. 2020] and [Yin et al. 2021]. The former proposed an end-to-end network to generate foreground masks, 3D bounding boxes, and point wise tracking association displacements, operating on two adjacent point cloud input frames and the latter formulated objects as key points, which are first detected and tracked and then refined with additional point features. Here, the tracking is realized by a simple greedy closest point matching.

However, a majority of other methods also focus on the fusion of images from camera with point clouds from lidar. First, [Zhang, Zhou, et al. 2019] proposed a

multi-modality feature extraction and fusion with PointNet [Qi, Su, et al. 2017] and VGG [Simonyan and Zisserman 2015]. Second, [Weng, Wang, Man, et al. 2020] again used a GNN for multi-modal feature learning from images and point clouds in parallel. In [Chiu et al. 2021], an association with Mahalanobis distance combined with feature distances was presented. Furthermore, [Kim et al. 2021] integrated all observations from camera and lidar at object level into a two-stage association framework. Another joint detection and tracking network was proposed in [Huang and Hao 2021], where bounding boxes are predicted in the first step together with association scores for further processing in a comprehensive data association module.

Overall, there are countless works in the field of MOT for Autonomous Driving (AD) with different approaches. On the one hand, end-to-end methods based on LSTM, or more recently GNN architectures try to detect and associate objects with more discriminative feature engineering jointly. On the other hand, the tracking by detection paradigm offers a powerfully flexible and modular alternative also used in this work. Most importantly, recent advances in 3D object detection with consistently improving results can be reused to simplify the association problem.

4.6 Conclusion

For use in real-world applications, MOT is often one of the most practical components to complement object detection for environmental perception. In addition to the recognition and localization of objects, the focus of this task is on association over time, while behavior and motions are estimated. To address this task, this chapter presented one of the first methods for joint object detection and tracking on point clouds, entirely in 3D. To this end, point cloud inputs as well as generated semantic maps from camera are fused into a 3D voxel representation and processed by a specific CNN for 3D object detection. Then, detections are tracked following the Labeled Multi-Bernoulli Random Finite Set (LMB RFS) [Reuter et al. 2014; Bryant et al. 2018] approach, allowing motion models to be incorporated and uncertainties to be modeled over time. The model is inspired by chapter 3 but includes a novel metric for the object to object comparison that can be inserted into any Neural Network (NN) for object detection to be more efficient and flexible. Experiments on the KITTI benchmark demonstrate the power and efficiency of the model with respect to baselines related to previous work natively operating in the image domain. Unfortunately, at the time of this work, no alternative datasets existed for applications in

AD to allow more fair comparisons directly in 3D. However, also qualitative experiments show visually pleasing results. In future work, an advanced combination of Deep Neural Network (DNN) architectures with statistical models for tracking can be extremely beneficial. However, different motion models are needed for different object categories. For instance, wheeled vehicles move completely in different ways compared to pedestrians. Similarly, innovations from state-of-the-art to individual components of the model, as seen in section 4.5, can be integrated to boost the overall performance.

Chapter 5

Concept for Integration into an Application-Specific Scenario

After a method for 3D object detection on point clouds was developed in chapter 3 and extended by tracking in chapter 4, the integration into an application-specific scenario is outlined in this chapter. A thorough development of a complete system for Autonomous Driving (AD) is beyond the scope of this work, as further detailed challenges such as path planning and control arise. Despite the fact that the presented methods achieve high recognition rates, the scenarios on public roads are way too complex and require a smart combination of additional perception, e.g. from cameras. However, there are already significant differences in the raw point clouds resulting from specific experimental vehicles or sensor technology. Additionally, the deployment requires advanced optimization for suitable target hardware and an extended implementation for online processing, as well as an integration part with input and output interfaces. Therefore, the focus in this chapter is on a concept for integration in order to assess the potential performance of such a system. An existing test vehicle is presented, where prerequisites and conditions are analyzed. Derived from this, an application-specific dataset is created. Here, the developed methods from former chapters are used for automation in a developed application to efficiently and accurately label ground truth through interaction with humans. With the help of the generated dataset, the performance on point clouds from different sensors is assessed by training and optimization of the model.

In contrast to the semi-automated manual data acquisition, a novel method for the synthetic generation of additional training data is also presented. Here, a Generative Adversarial Network (GAN) is developed that learns a distribution of the 3D characteristics of objects and translates it to the image space to generate visual

representations. Thus, existing datasets can be expanded explicitly and selectively. The content is derived primarily from the published work in [Milz et al. 2019].

The first section in this chapter explains the technical difficulties and motivates the usefulness. Second, the target system with interfaces for integration is described in more detail. Then, section 5.3 presents the process of how the application-specific dataset is created in a semi-automated fashion supported by the developed methods from the previous chapters. This dataset is used for experimental evaluation in section 5.4. As an alternative in obtaining training data, section 5.5 explores the generative approach based on a GAN. Finally, section 5.6 gives a summary.

5.1 Motivation

During the past several years, there has been worldwide increased research in the field of AD. Compared to the single tasks of object detection or tracking based on point clouds from the previous chapters, this area is by far broader and more complex. No existing sensor modality and no existing system alone fulfill all the required properties for solving the existing problems. Although deep learning approaches enable enormous progress with higher detection rates, better accuracy and fewer false predictions, there is still a long way to go before human drivers are no longer needed. There are dozens of challenges and disadvantages associated with such methods. First and foremost, the massive demand of data for training, optimization, validation, and testing to ensure the safety-critical function. Likewise, significantly reduced performance is to be expected in cross-sensor deployment, as the generalization capability is limited and any change in the target domain remains a challenge. For instance, each of the test vehicles in Figure 1.2 has different sensors with varying mounting positions. Therefore, the need for specialization and adaptation heavily arises. However, the predominant approach is to use expensive prototypes to simplify the challenges as much as possible, especially in environmental perception. For example, in most cases, there are roof structures with a huge number of sensors for research and development, which are probably completely unsuitable for the mass market and series production. The price of the sensors alone can easily exceed the price of the actual vehicle many times over. For this reason, a field of research has emerged, mitigating the costs with cheaper sensors and hardware suitable for the generality while adding individual functions to increase driving comfort, safety, or partially to take over control and relieve the driver temporarily. This area paves the way step by step towards AD. Here, the toughest challenges are the more challenging conditions due to the use

of cheaper hardware and sensor technology, which are additionally integrated into current vehicles as invisibly as possible. On the one hand, the systematic introduction of more and more automation is becoming easier. At the same time, general acceptance is increased, and reliance on machines builds up.

5.2 Application Scenario of the Autonomous Car

An autonomous vehicle typically consists of a system of individual components based on sensors, perception, planning, and control that must be optimally coordinated with each other. Additionally, there are defined interfaces between all modules at different levels of abstraction. For instance, the presented object detection with bounding box abstraction is generated from sensor inputs and forwarded to the planning components. As already indicated, the overall function is a safety-critical system since there is a continuous interaction with other road users and public traffic participants. Therefore, it is highly sensitive e.g. to false detections, and the design has to be very conservative, better to stop instead of causing fatal accidents. On the other hand, unnatural, harsh braking lowers the user's acceptance significantly. Consequently, a high degree of reliability is required, which is aimed for through redundancy and fusion concepts. Here, the developed object detection and joint tracking methods can be used as sub-components of perception. Safe recognition of road users is the first step in predicting movements and behavior. Precise localization of objects is of high importance since it will have a direct influence on navigation and driving. In addition to other abstraction levels, such as maps with lanes, traffic lights, traffic signs, etc., the object level is one of the main components for environmental modeling in dynamic road scenarios, as elementary information can be processed quickly and efficiently.

The following subsections provide more details of the target setup, where the developed methods will be integrated. First, the vehicle setup with used sensors is described. Second, subsection 5.2.2 highlights the challenges compared to state-of-the-art, resulting from the different setups. Finally, the concept for the integration of the developed methods is outlined in subsection 5.2.3.

5.2.1 Experimental Vehicle and Sensors

The underlying test vehicle is based on the Volkswagen Passat B8, converted with an additional battery and the most advanced charging alternator for the power supply to industrial computers with GPU cards mounted via PCI express extensions. As

CPU	16 × Custom *Carmel* ARM64
GPU	2 × Volta iGPU, 2 × Turing dGPU
Accelerator	2 × Deep Learning Accelerator
Extras	Stereo and Optical Flow Engine Image Signal Processor Programmable Vision Accelerator CAN Interfaces 16 × GMSL(R) Camera ports
TDP	500W

Table 5.1: *Nvidia Drive AGX Pegasus: Technical Hardware Specifications.*

an alternative embedded platform, Nvidia Drive AGX [NVIDIA Corporation 2021] can be used, where GPUs are also available, see Table 5.1 for detailed specifications. Electronically longitudinal and lateral control of the test vehicle is established via CAN signals over an AutoBox device [dSPACE GmbH 2021]. Individual modules of the entire system can be distributed to the computers so that communication takes place via Ethernet sockets.

For perception, more than thirty sensors of different modalities are mounted in or around the vehicle, and connected to the PCs. Figure 5.1 illustrates the sensor mountings. In most cases, prototype development sensors with raw data access via Ethernet are used since traditional automotive-grade sensors typically output pre-processed data only. In addition to the well-known surround-view fisheye cameras, radars, ultrasounds, a front camera, and inertial DGPS are also installed to cover a broad experimental setup.

This work focuses on point clouds generated by the lidar scanners, 6× Valeo Scala Gen1 around the test vehicle (darker green) and one Scala Gen2 (brighter green) mounted in front. Obviously, the front and back of the vehicle are covered with more sensors to have multiple overlapping regions and redundancy. There is a small area on each side that is not visible by the Valeo lidars as the vehicle cannot move to the sides. Additionally, the two Velodynes mounted on top of the vehicle can be used for reference similar to the benchmark dataset used in former chapters. For comparison, sample point clouds of all lidars from one of the recordings are shown in Figure 5.2. Valeo Scala sensors have a limited horizontal field of view of about 140°, whereas Velodyne captures full 360° by rotation around its own axis. The different number of layers (laser diodes) and resulting point density is clearly visible from 4 (Scala1) up to 64 (Velodyne HDL-64E). Overall, all lidars typically suffer from internal reflections

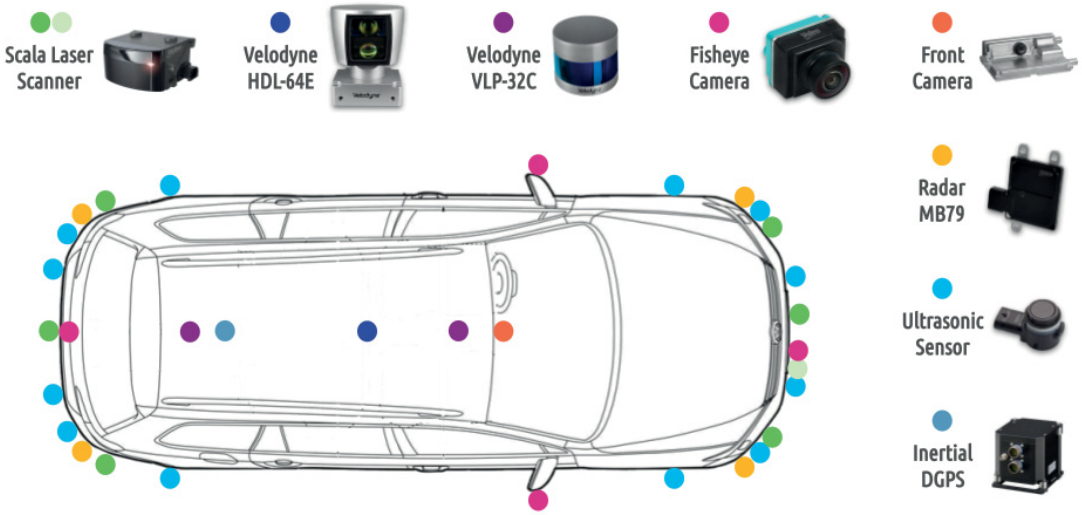


Figure 5.1: *Overview of sensors mounted on the Volkswagen Passat B8 platform. This vehicle setup is used for research and development in the area of AD. The focus of this work is on the lidars (Scala and Velodyne).*

and noise from near field reflections $< 1m$, but output measurements directly in 3D.

5.2.2 Delimitation to State of the Art

All methods developed in this work and from state of the art are also suitable for use in the presented setup. However, decreased performance must be expected without specific optimization and fine-tuning. The main difference concerns the underlying sensor outputs and their mounting positions. Compared to the Velodyne lidars, the Scalas have a substantially fewer number of laser diodes (Generation 1 has 4 and 2 has 16 layers) and, in addition, the horizontal and vertical field of view is smaller. These lidars are designed to be mounted at lower levels, e.g. at the bumpers, to enable cleaner integration into existing vehicle designs. Therefore, especially in the near field, the viewing capacity is strongly limited in the height dimension to cover long ranges. Moreover, the number of points measured is reduced by a factor of 42 due to the aforementioned restrictions, but this fluctuates heavily depending on the scenario and internal lidar parameters. For instance, multiple echos with different thresholds can be used to compensate for extreme conditions, but algorithms must be robust against the additional sensor noise.

Furthermore, the occlusion problem is considerably increased by low sensor installations. Where roof attachments like Velodyne lidars partially overlook objects with small blind spots behind, the Scalas look almost parallel to the ground at objects

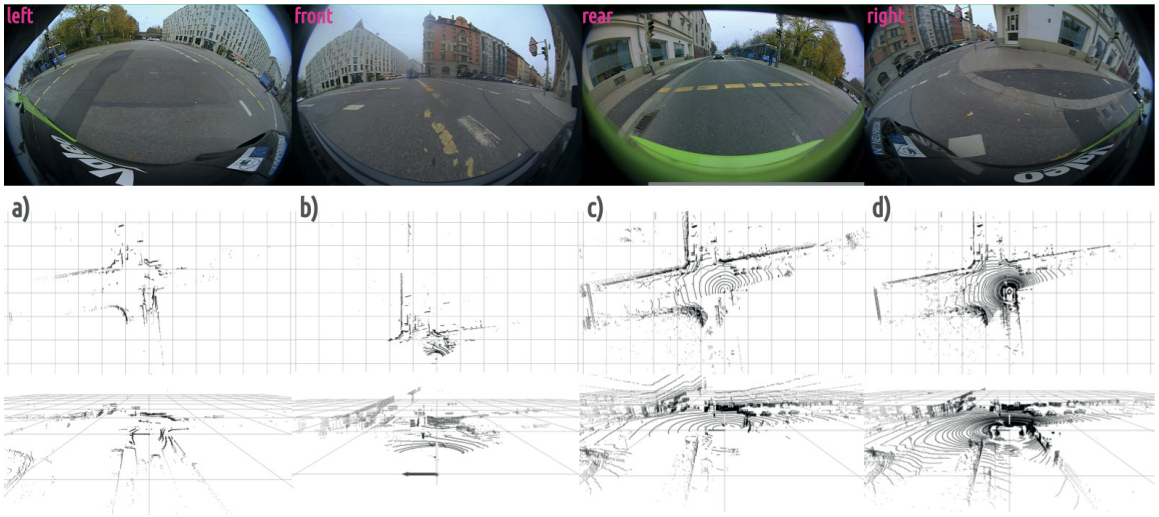


Figure 5.2: *Point cloud samples from four different lidar sensors: a) 6× Valeo Scala1, b) Valeo Scala2, c) Velodyne VLP-32C, and d) Velodyne HDL-64E of an inner-city environment. The top row contains fisheye camera images for a 360° overview of the scene, whereas each point cloud is rendered from top-view at the middle row and a 3D perspective at the bottom. All presented lidars are based on rotating mirrors, whereas Valeo Scalas have a restricted horizontal field of view. Therefore, a) contains measurements of all available Scala1 sensors transformed into unified coordinates. Both automotive-grade Valeo Scala sensors are mounted low in the front bumper and Velodyne sensors on top of the recording vehicle. This allows the upper sensors to see beyond some objects of moderate height, offering several advantages for the resulting point clouds.*

so that everything behind is completely occluded. Estimating the upper surface of objects is much more complex, as no measured points are generated from the surface, as seen from below. Subsequently, there are cases where individual laser beams of the lower layers pass below objects and only appear much later on solid surfaces. For instance, higher vehicles like SUVs or trucks offer plenty of ground clearance and thus make point clouds rather challenging.

Overall, due to the aforementioned characteristics, a significantly reduced detection performance compared to the results from the previous chapters can be expected. Likewise, insufficient generalization can be expected because besides the highly complex public environments, also the properties of the point clouds deviate greatly. However, with the help of optimization on application-specific data, acceptable detection rates can be achieved in order to improve existing driving and comfort applications and step-wise extend automation.

5.2.3 Model Integration

In order to use the developed method for 3D object detection from chapter 4 for the AD system, certain CPU and GPU computing power are required. Among other modules and calculations, the model can flexibly run on the target hardware described in subsection 5.2.1. In case distributed computers are used for individual modules of the system, proper communication between each node must be ensured, e.g. via Ethernet. As mentioned before, the model needs to be optimized and fine-tuned for the specific sensor setup to achieve the best potential performance. This is described in more detail in subsection 5.4.2.

For online processing, the inputs are captured from streams via Ethernet interfaces and decoded into point clouds in Cartesian space. The associated timestamps and sensor calibration are used to transform individual point clouds into a global one when using multiple lidar sensors. This is followed by the preprocessing described in subsection 4.3.2. For simplification, occupancy voxels are used instead of semantic features generated from cameras because an additional synchronization and integration of the presented semantic segmentation model is required. The developed tracking approach from chapter 4 can be used optionally, depending on which tracking or fusion the system is based on. Here, a serial connection to the vehicle CAN bus is used to retrieve odometry and ego-motion signals. Specifically, vehicle yaw rates and velocities are used. All inputs are temporarily stored in memory using ring buffer implementations.

The inference of the model is triggered regularly at $10Hz$ as a trade-off for efficiency vs. accuracy. This is the native frame rate of default Velodyne settings, whereas intermediate frames from Scala with $25Hz$ are ignored. In this way, point cloud frames are selected based on the nearest neighbors of the timestamps without further corrections. The output interface consists of all predicted objects, namely the center coordinates, the dimensions, yaw orientation, uncertainty, and the inferred class label with fixed type and precision. If tracking is used, estimated objects velocities can be added to enable better prediction of individual motion for consuming planning modules. This output is sent via socket communication either locally or over the Ethernet interface, depending on which computer setup is used.

5.3 Application-Specific Dataset

The preparation of a qualified dataset is extremely time-consuming and costly, as ground truth must be generated for the optimization of the networks in addition to the raw recordings. There are high demands on quality and quantity with many sources of errors down to low-level package loss due to network traffic, jitter and others. After all, every publicly available large-scale dataset for AD has been carefully developed over the years. An overview and further details of public benchmark datasets can be found in Appendix B. But still, complex scenarios from public roads are often extremely rarely present and cannot be recreated sufficiently in a closed test environment. Moreover, specific data is always needed to achieve reasonable performance for various use cases. Therefore, recording sessions in public are carried out with the experimental vehicle at different locations and at different times to create an application-specific dataset. From as much material as possible, helpful, interesting sequences tailored to the applications are manually selected in postprocessing and extended with ground truth for 3D objects generated by a semi-automated annotation process with humans in the loop. Therefore, a dedicated software application was developed to efficiently label the data, integrating the methods from previous chapters for automation.

The individual steps for creating the application-specific dataset are described in detail below. First, a special recording system and the recording sessions are described in subsection 5.3.1. Then, the annotation process with the developed GUI application is presented in subsection 5.3.2 and subsection 5.3.3. Here, methods from chapter 3 and chapter 4 are adapted and reused for partial automation. Finally, subsection 5.3.4 provides an overview with statistical information regarding the outcoming dataset.

5.3.1 Raw Recording

The recordings were made with the vehicle presented in subsection 5.2.1. All sensors are calibrated to reference coordinates and connected to a single recording PC. The powerful NI PXI platform [National Instruments Corporation 2021] with multiple interface modules was used to guarantee enough performance and to avoid additional synchronization over multiple nodes. Recordings are controlled by a human within a recording toolchain based on the NI LabView environment. Hence, start and stop are triggered manually during driving, while the software performs regular sanity checks with online status indicators to abort on failure modes. Once a recording is started, incoming packages with low-level raw data are cached and saved to a binary format

without further processing or decoding. In this way, a superior bandwidth of more than 500 Megabytes per second of data can be processed. This is required since raw data from most of the sensors must be saved immediately.

With this setup, a human driver maneuvered the vehicle through several major cities in Germany at busy times while triggering the recordings. To this end, Stuttgart in June 2018, the area of Nuremberg and Erlangen in July 2018 as well as Munich in November 2019 were picked with several hours of recordings, each. The raw data of sequences with 5 to 45 minutes duration are backed up on persistent memory after each day of recording.

5.3.2 Ground Truth Generation

Assuming a frame rate of $25Hz$ over all raw sequences, more than two million frames are recorded in total. A major part is of no use for training and optimization of the developed perception models, as almost identical contents repeat continuously. For instance, each time the vehicle has to wait at a traffic light or stop sign, the sensors capture the same environments for a few seconds. At a frame rate of $25Hz$, the variations from one frame to the next are also minimal. Therefore, individual sequences are carefully selected by qualitative evaluation from humans to generate a balanced dataset with high diversity, also covering some rare cases. Those sample sequences have a length of 50 seconds up to 200 seconds, as a trade-off between diversity for single frame object detection and tracking needs. Each sequence is extracted from the corresponding raw data with an existing proprietary toolchain and decoders, synchronized to a frame rate of $2Hz$. This is done using nearest-neighbor interpolation with the time of arrival timestamps generated from the LabView environment, where the Velodyne clock is used as the master.

Based on these exported sequences, a software application is developed to support humans in 3D object labeling in order to generate ground truth. Hence, the GUI-based tool is separated into three areas. First, a menu bar contains global functions to load and save a session and to replay a sequence of data. The main part is the workspace, where interactive windows with points clouds and additional camera images are visualized. On top, property bars present fields with quantitative details corresponding to a focused bounding box and allow for interactive manipulation. Additionally, a scene graph lists all existing instances grouped by classes and identifiers. All areas are fully adjustable dockable bars or sub-windows and can be modified by drag and drop, see Figure 5.3 for an example. Furthermore, data from additional or

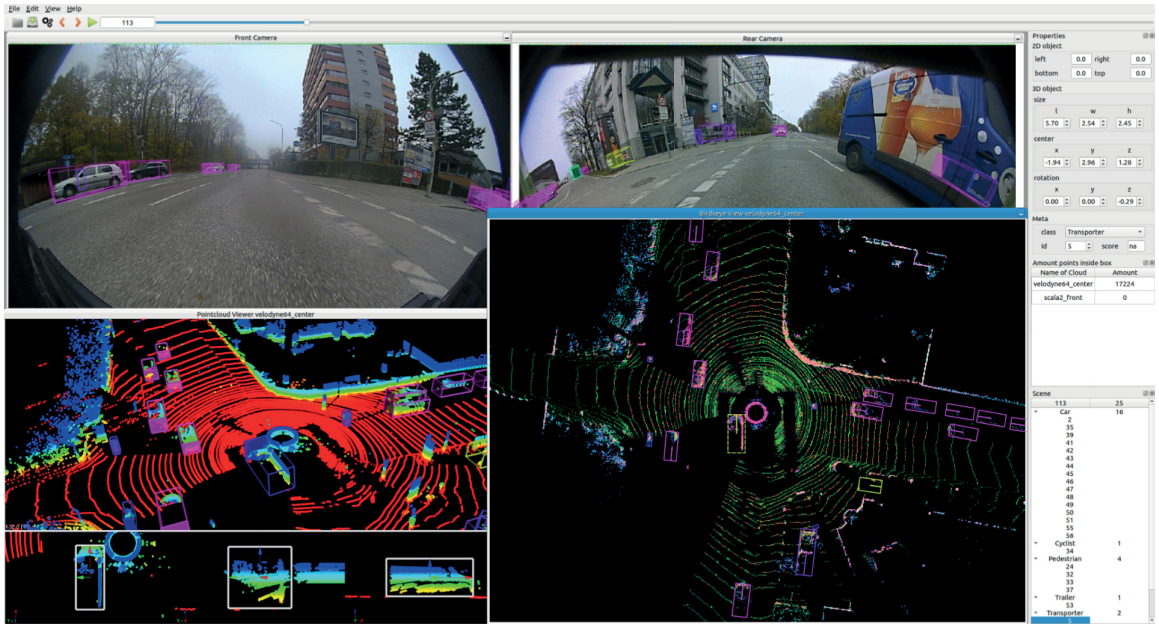


Figure 5.3: *The GUI of the annotation tool with sample data: Here, four fully adjustable sub-windows with fisheye front and rear as well as birdview and 3D renderings from the Velodyne point cloud are arranged inside the workspace below the menu bar. The 3D view is further subdivided into an interactive perspective view in the upper part. The lower part shows a small subset of points related to the focused bounding box of a Transporter instance in different orthogonal views (top, behind, side). This is especially helpful for fine-grained corrections. A height-based color coding is used from red (low) to blue (high) while ignoring* 10% *of the lowest and highest z values to filter outliers. In contrast, birdview coloring is similar to the color channels described in subsection 3.3.1, except the red channel is based on an angular feature calculated by normals of each point to make vertical surfaces stand out clearly. While* 25 *objects are already annotated, bounding boxes (small rectangles with direction indicators, colored by class) can be added and manipulated in all visualizations as well as menu and property bars to generate ground truth efficiently. Everything is saved automatically to JSON format.*

fewer sensors can be loaded in order to be flexible for changes in the available sensor setup. Ultimately, the application outputs generated bounding boxes to Java Script Object Notation (JSON) format.

Although the update rate is relatively low at two frames per second, objects of intermediate frames can also be interpolated. In complex dynamic environments, this results in minor errors due to interpolation in favor of wide variety and diversity. Moreover, the accuracy also depends on the presented synchronization, as labels once generated in point clouds from Velodyne are transferred, e.g. to Scalas or cameras. The relatively slow rotation around its own axis results in small deviations during a

run. For simplification, no further corrections are applied here.

5.3.3 Semi Automated Annotation

Although the developed tool involves the placement and manipulation of objects in 3D smoothly via interaction with mouse and keyboard shortcuts, an extremely large number of individual actions needs to be performed manually until satisfactory accuracy is achieved. For instance, the virtual camera must be permanently aligned or single object parameters precisely adjusted to respective objects. In most cases, the point clouds are extremely sparse and require the use of a camera or checking through parts of the sequence where the objects are perfectly visible. All this leads to an average processing time of over one minute to complete a single frame, but the use of automation reduces the time by 50%.

For this reason, the methods for 3D object detection and tracking from former chapters are reused and integrated into the annotation tool. An overview of the overall workflow is visualized in Figure 5.4. Initially with the use of a keyboard shortcut or menu entry, the inference of the model from chapter 3 can be triggered so that the current point cloud flows through the network as input, and detections are directly added as bounding boxes. Figure 5.5 provides a sample annotation.

Consequently, the number of actions is reduced, the better the predictions are since only minor corrections need to be done. Second, during forward annotation of a sequence, objects are considered as measurements to be tracked. Hence, predictions are automatically inserted into the next frame, assuming that all objects are manually corrected by humans before switching to the next frame. Here, the logic of the tracking model is slightly adapted to fit this use case as the measurements are considered to be perfect. On the one hand, the object dimensions from the current frame are taken over unchanged into the predictions of the next frame since ground truth values are already assigned. Similarly, birth and death of a target track as well as existence and target thresholds are adapted, i.e. a new target with full existence probability is created for every new measurement and directly removed as soon as the measurement disappears. Figure 5.6 visualizes a few results of the prediction. Whenever bounding boxes of the current frame are corrected, the switch to the next frame automatically updates the internal state of the tracker by using these boxes as current measurements together with odometry. At the same time, another prediction step is performed to generate new bounding boxes that are immediately shown after the next frame is loaded. As can be seen, static or non-moving objects like parked

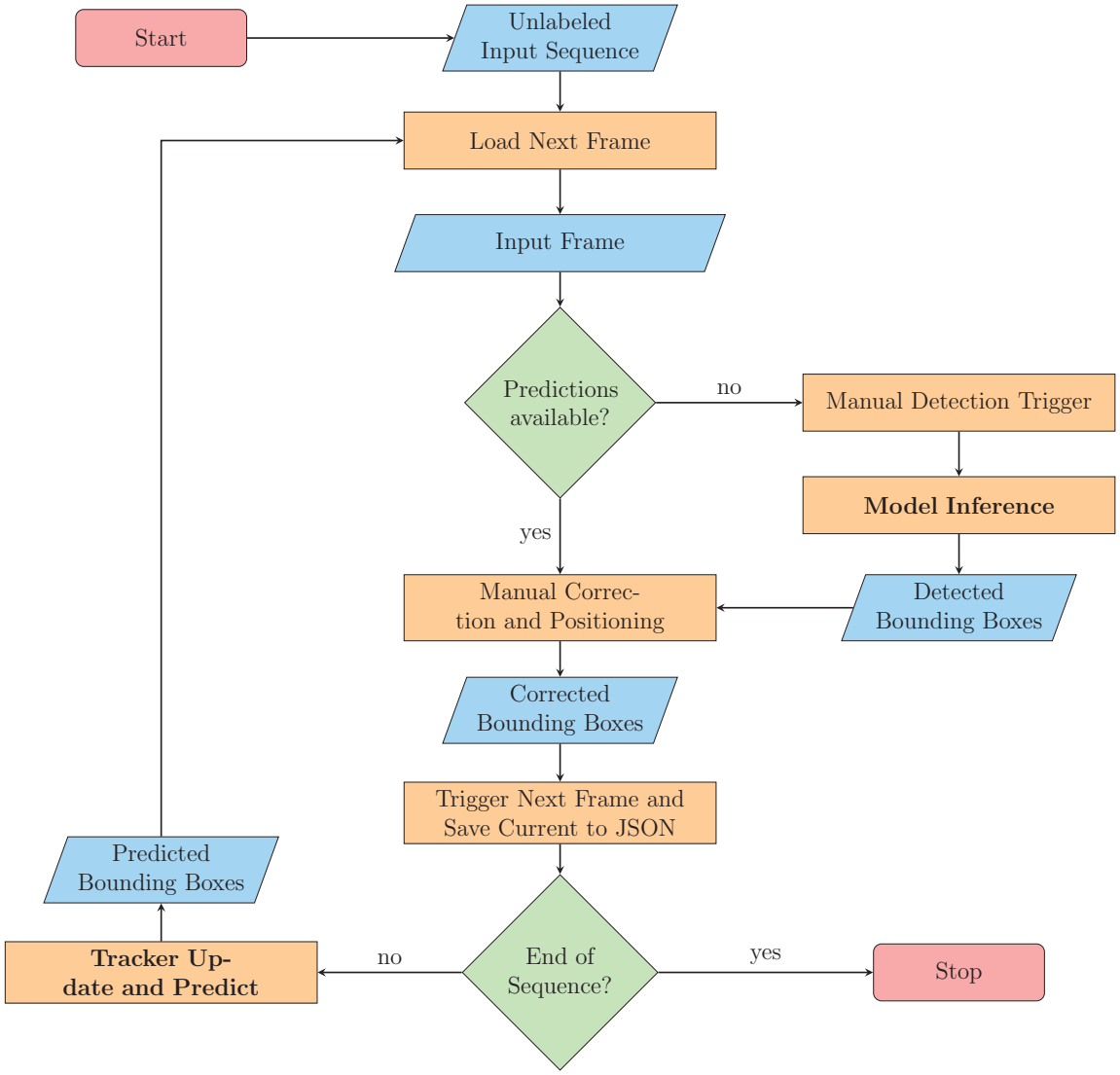


Figure 5.4: *Workflow diagram for a semi-automated annotation of an unlabeled sequence of data samples: All high-level processing steps are visualized as rectangles in orange, with the integrated intelligent algorithms for automation marked in bold. Related inputs or outputs are drawn as blue trapezoids, and conditional branches are shown as diamonds in green. For the first input frame, initial detections can be generated by one click to reduce the number of manual actions. This is followed by a manual correction step, with the removal of wrong, creation of missing objects, and fine-grained manual adaptations. Furthermore, the tracking automatically predicts bounding boxes for the next frame when a frame switch is triggered through another click.*

cars are labeled perfectly since the relative ego-motion was already corrected. In contrast, dynamic moving objects require minor corrections. Here, the tracker also estimates the velocity of a dynamic object after it is tracked at least two times. For

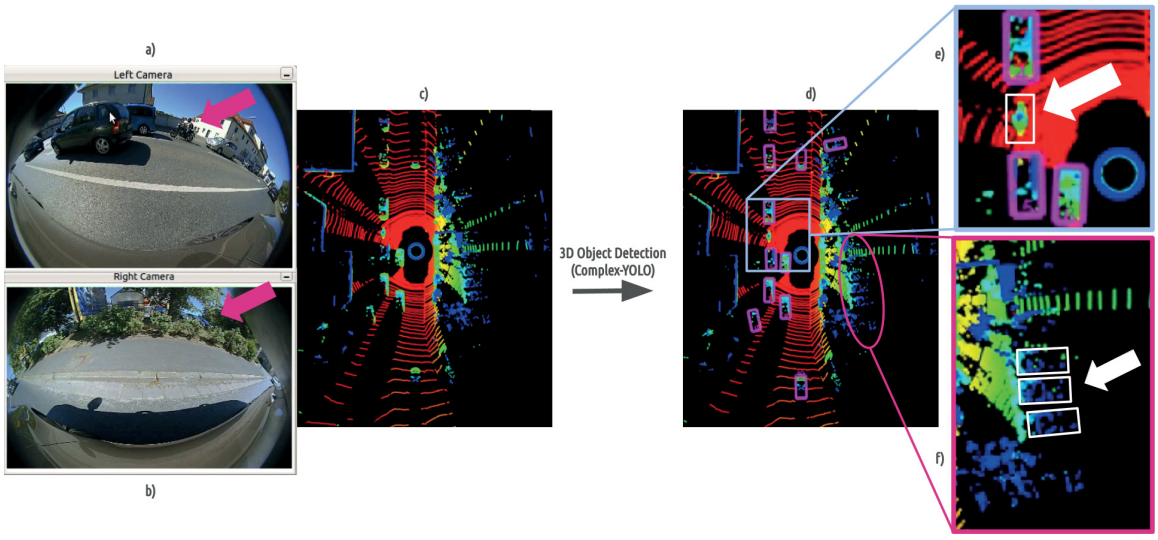


Figure 5.5: *Automated annotation using the developed object detection model from chapter 3. For a given sample, left a) and right b) fisheye images as well as the point cloud c) are visualized. The same height-based coloring presented in Figure 5.3 is used here. When using Complex-YOLO, all visible bounding boxes are generated on the fly with the inference of the model in d), triggered by one click in the annotation tool. Afterwards, minor manual corrections are made, based on reference cameras or by scrolling forward or backward in the sequence. For instance, the motorcycle zoomed-in e) or the vehicles parking behind the bushes f), both indicated by white arrows, are not detected. Hence, missing bounding boxes visualized in white need to be added manually. This significantly reduces the manual effort as only a few bounding boxes need to be created from scratch and thus ensures vastly reduced time and costs. As reference, there are also magenta arrows in the fisheye images pointing to the missing objects.*

instance, both upper red arrows point to a car driving in front of the ego vehicle in the same direction and an oncoming *Truck*, that are first predicted with several meters offset to the point cloud in *000005*, but accurately predicted in *000010* and *000015*.

Both methods complement each other in the annotation process, as objects can be initially created by the detector and thus tracked until the end of the sequence with small step-by-step corrections. However, the performance of automation heavily depends on the accuracy of the detections as well as predictions and, therefore, on the complexity of the scene. Objects are sometimes temporarily fully occluded, which can only be solved by forward and backward annotation over multiple frames. For instance, the parked car in Figure 5.6 represented by the yellow arrows cannot be seen in the point cloud of frame *000000*, while being clearly visible afterwards. This results in an even higher number of actions needed for highly accurate annotation of

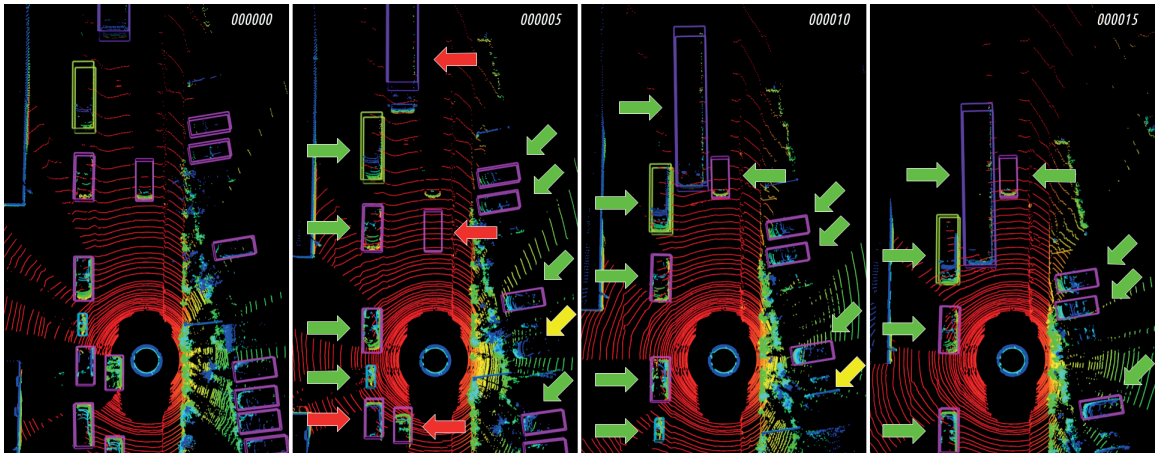


Figure 5.6: *Automated annotation using the adapted tracking method from chapter 4 illustrated on a sample sequence from Velodyne HDL-64E at 2Hz (every 5th frame with delta 0.5s). In frame 000000, corrected bounding boxes are fed into the tracking algorithm as initial measurements. Then, frames 000005, 000010 and 000015 show predictions from tracking following sequential forward and manual correction cycles described in Figure 5.4. Here, green arrows indicate predictions almost without the need for manual correction, while red and yellow arrows represents bounding boxes either to be reworked or missing due to occlusion in 000000. In this way, mainly subtle corrections of dynamic objects (red arrows) are left for an annotator instead of the full annotation of the frame.*

ground truth. If many wrong or partially wrong objects are automatically generated, the number of actions remains nearly similar. When switching to a frame other than the next, tracking is temporarily disabled, and the internal state of the tracker is reset. Overall, the effort for annotation is significantly reduced by automation, and the full process is heavily accelerated, which makes the annotation tool also suitable for larger amounts of data.

5.3.4 Dataset Analysis and Annotation Statistics

After several people were trained with the presented application on sample data, all sequences described in subsection 5.3.2 were annotated. The first part is in a version without automated tracking. Here, no unique object instance identifiers were assigned throughout the sequences either, but all appearing objects were annotated throughout the sequences, even with temporary occlusion. For this purpose, the sequences were used forward and backward as well as reference images from cameras. Additionally, the Scala2 was only available for later recordings in Munich. Overall, three-stage reviews were carried out to validate and ensure high quality. During validation,

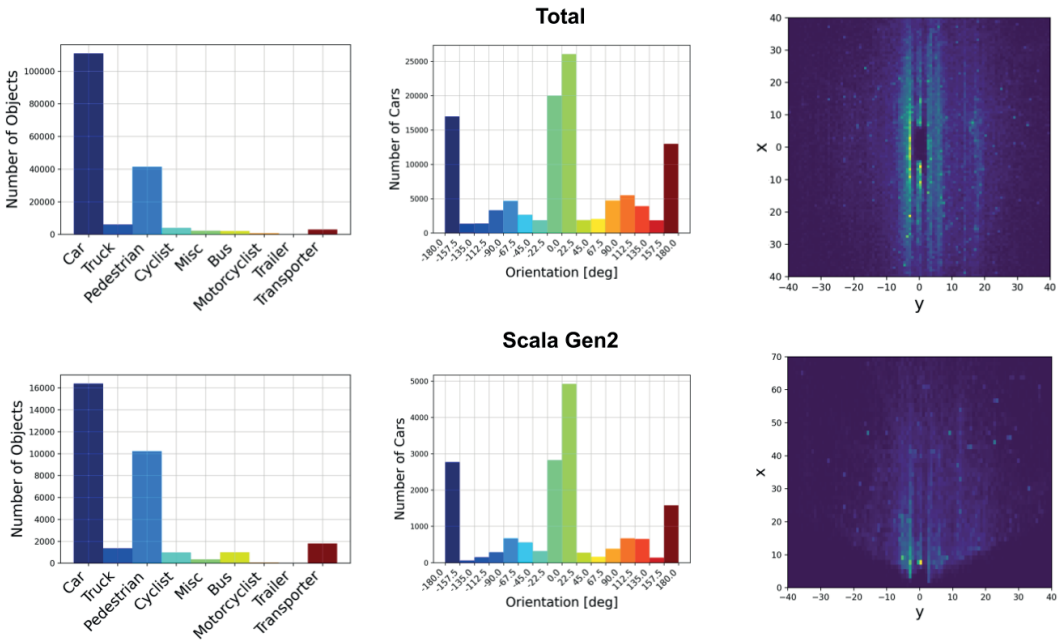


Figure 5.7: *Object occurrence and orientation statistics of the generated application-specific dataset: The first row presents overall quantities, whereas the second row is related to the described part with Scala2, only. The first two columns show histograms with the number of objects per class and the distribution of yaw rotation angles. They are followed by the spatial distribution of object centers in a normalized 2D histogram in a birdview perspective. While Velodyne64 and the combined Scala1 point clouds in the upper plot cover the full surrounding of the car with the origin at the rear axle, the field of view from the front Scala2 is clearly reflected in the lower heatmap.*

inaccuracies between annotations from Velodyne applied to Scala point clouds were observed. As illustrated, the root causes are sensor noise and synchronization because higher relative speeds of more than $100 km/h$ occur. Therefore, the labels for Scala2 were corrected manually and saved separately. In addition, a script-based check for plausibility and basic errors was performed, using the temporal context within the sequences. For instance, an object instance can be determined by its identifier, if available, or by its 3D position in successive frames. Thus, movements can be calculated, consistent classes and dimensions, or the deviation of the yaw orientation can be monitored. Moreover, all single object parameters are verified for reasonable values per class.

A couple of statistical quantities of the generated application-specific dataset are shown in Figure 5.7. In total, $13,922$ frames are included at $2Hz$ with roughly $170k$ annotated objects, whereas $3,322$ frames contain Scala2 with roughly $31k$ objects, as well. Essentially, the distributions of classes and yaw angles are similar to the KITTI

dataset from Karlsruhe presented in subsection 3.4.1. As can be seen, real object occurrence probabilities of major German cities are reflected. There is a high density of objects, especially on road surfaces and in the near field of up to $20\,m$. The sensor setup is designed exactly for this purpose as the majority of sensors cover the front and the rear of the ego vehicle.

Parts of this setup, from tooling to the overall automated annotation process, have also been incorporated into the publication of the Valeo Woodscape dataset [Yogamani et al. 2019], a fisheye camera open-source dataset that comprises of over $10k$ images captured by multiple vehicles using automotive-grade surround-view fisheye cameras and annotations for a variety of computer vision tasks, assisted by point cloud reference.

5.4 Experiments on the Application-Specific Dataset

This section describes the experiments on the generated data conducted to assess the object detection performance for the application-specific sensor setup. First, the derived datasets for evaluation are presented, followed by details related to the training and optimization of the model. Finally, after the metrics are shortly described in subsection 5.4.3, subsection 5.4.4 reports the achieved results.

5.4.1 Datasets

Based on the recorded and annotated sequences described in section 5.3, three individual datasets are constructed and split into subsets for training, validation, and testing. The first one consists of point clouds from Velodyne64 for reference and comparison. Second, the Scala1 dataset is based on fused point clouds from all six Scala1 sensors. Therefore, calibrations are used to transform all point clouds into common vehicle coordinates using nearest neighbor time synchronization. All object annotations are propagated from Velodyne64 by transformation into the same vehicle coordinates based on the sensor calibrations, without further corrections for synchronization or jitter. Both, Velodyne64 and Scala1 datasets are randomly subdivided into 38 sequences ($7,847$ samples) for training, 13 sequences ($3,086$ samples) for validation and 14 sequences ($2,989$ samples) for testing. There was at least one minute of driving between single sequences during the recordings, and routes were only traveled through once, so individual sequences considerably differ from each other.

The third dataset is composed of point clouds from Scala2. As shown in Figure 5.7, only the front of the ego vehicle is visible with small blind spots to both sides. Due to the availability of the sensor during the recording sessions, only a part of the aforementioned datasets is covered. This results in 12 sequences (2,446 samples) for training, 3 sequences (388 samples) for validation and 4 sequences (488 samples) for testing.

5.4.2 Training and Optimization Details

A simplified variant of the developed model from section 4.3 is trained repeatedly on all three datasets end to end without additional camera inputs and tracking. Here, point clouds are preprocessed to occupancy voxels instead. Subsequently the Convolutional Neural Network (CNN) architecture generates object outputs, as visualized in Figure 5.8. According to the datasets, there are two different sets of parameters. First, voxelization is done for Velodyne and Scala1, where $x \in [-39.68\,m, 39.68\,m]$, $y \in [-39.68\,m, 39.68\,m]$, $z \in [-2\,m, 4\,m]$, $g = 0.16\,m \times 0.16\,m \times 0.20\,m$, resulting in $m = n = 496$ and $c = 30$. Second, as Scala2 covers the front only with $x \in [0\,m, 69.12\,m]$, $y \in [-39.68\,m, 39.68\,m]$, $z \in [-2\,m, 4\,m]$, $g = 0.16\,m \times 0.16\,m \times 0.20\,m$, with $m = 432$, $n = 496$ and $c = 30$. All annotated objects with less than 5 points within the bounding box in the respective point cloud are filtered and ignored, since many objects are occluded in the Scala point clouds due to the low mounting positions. Based on the resulting sets of objects, the regression anchors are defined by class wise mean values of length, width and height, times the two most appearing yaw orientations. Hence, there are 12 anchors in total, with $f_a = 180$ filters in the last convolutional layer. In addition to an extended data augmentation, all other training parameters from section 4.4 are retained. Since Velodyne64 and Scala1 have ground truth labels for the complete environment, global rotation is used in 360° without restriction.

5.4.3 Applied Evaluation Metrics

Following the 3D object detection metrics introduced in section 3.4, Average Precision (AP) metrics are used to assess the performance of the model. On the one hand, this evaluation also enables cross-comparisons to chapter 3. However, there are some significant distinctions due to the labeling policies. Unlike KITTI, the application-specific dataset does not rely on camera reprojection and is not limited to the area visible to the front camera. Therefore, there are no similar difficulty levels of easy,

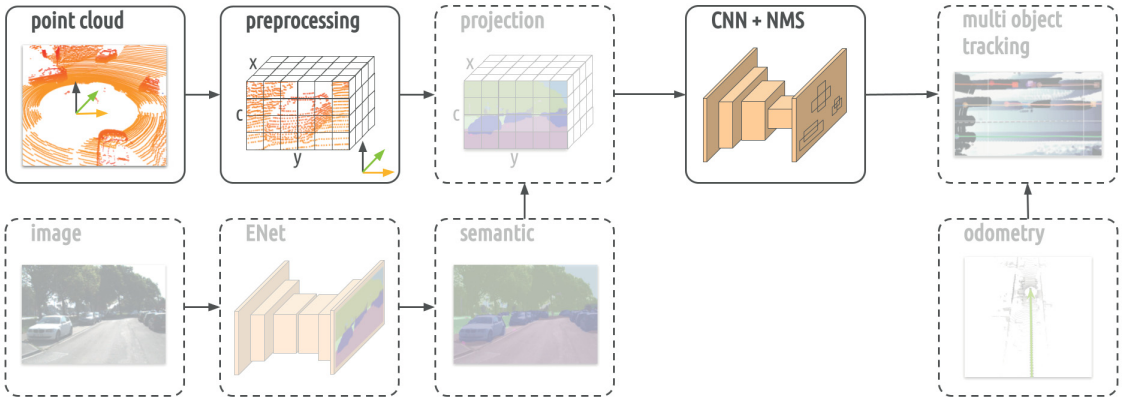


Figure 5.8: *Overview of the experimental pipeline for 3D object detection on the application-specific setup. This model is a section of Figure 4.2, operating only on point cloud inputs without downstream tracking. All unused building blocks are shown transparently with dashed lines.*

moderate, and hard. Additionally, there are no occlusion or truncation labels. Each individual object is labeled throughout the sequences without *Dontcare* areas, only filtered out if less than 5 points are within the bounding box, respectively. The evaluation is based on the covered areas of the sensor data, according to the region of interests defined in subsection 5.4.2. Furthermore, results are reported for 3D and birdview bounding box overlap classified according to the Intersection Over Union (IOU) thresholds 0.3, 0.5, and 0.7. More details regarding the definition of AP can be found in Appendix C.

5.4.4 Results

The difficulty level of 3D object detection on the application-specific setup multiplies, primarily due to the low mounting positions of the sensors as well as lower vertical resolution compared to Velodyne64. However, the developed model discovers a vast amount of objects even with more sparsity and extremely complex inner-city scenarios as shown in Figure 5.9. The example scenes from Munich include multi-lane traffic scenarios and huge road intersections with heavy traffic. Additionally, there are countless pedestrians and cyclists on the sidewalks. Here, the density of correctly detected objects varies according to the capabilities of the underlying point clouds. Nevertheless, in some cases, the model is able to localize objects even with a few point measurements.

Additionally, an exhaustive quantitative evaluation is provided in Table 5.2 as well as Figure 5.10 for 3D and Figure 5.11 for birdview analysis. In both cases, the model

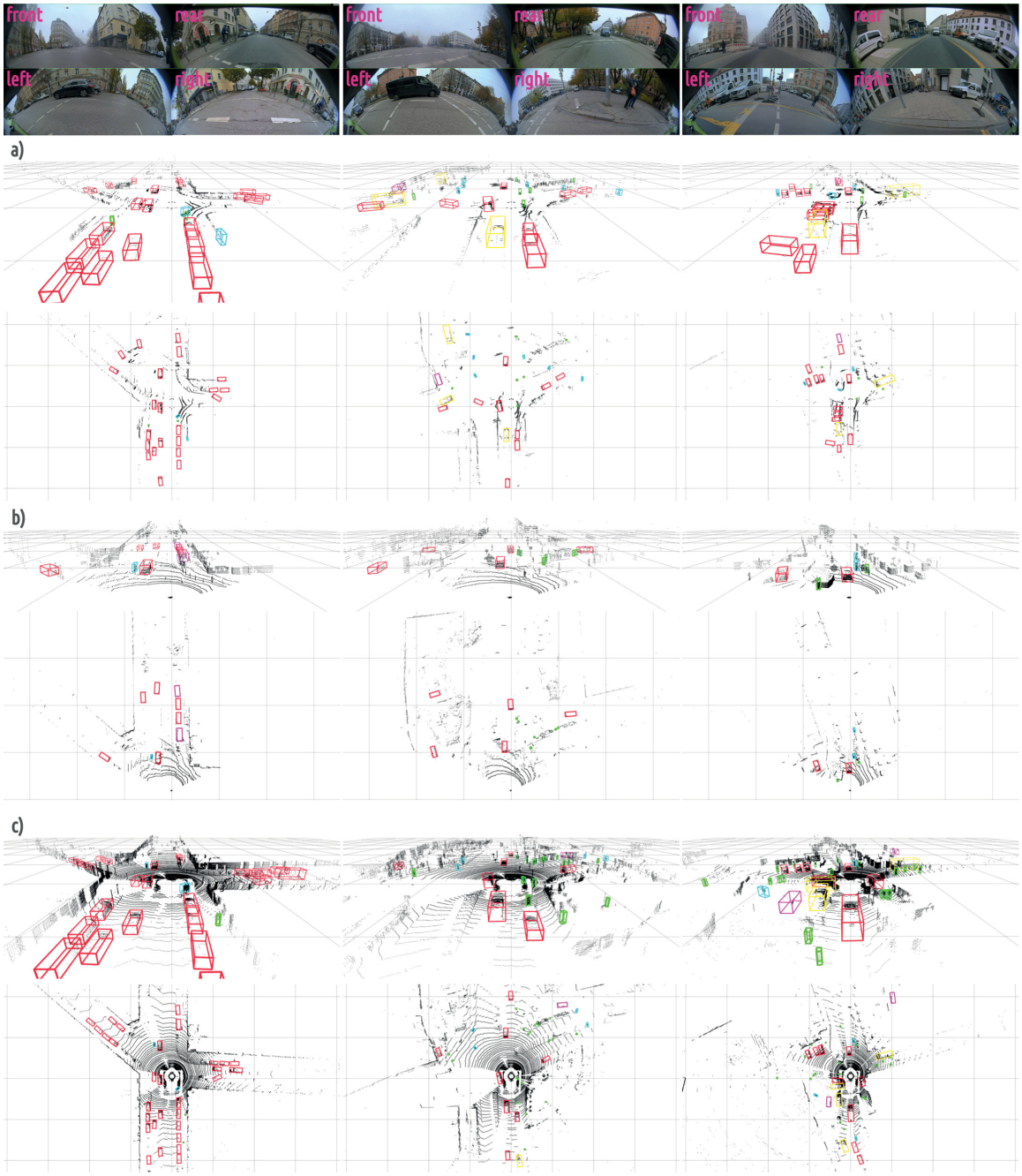


Figure 5.9: *Qualitative 3D object detection results on point clouds from a)* 6× *Valeo Scala1, b) Valeo Scala2, and c) Velodyne HDL-64E, representing crowded test samples from Munich per column. Visualized are recognized objects as bounding boxes in 3D and birdview perspective, colored by classification: red for Car, green for Pedestrian, yellow for Truck, cyan for Cyclist, blue for Bus and magenta for Transporter. All grid cells have a size of* $20 \times 20\,m$*, representing the ground surface in order to give an estimate of the proportions. In addition, fisheye camera images are visualized for an impression of the scenes (top).*

Data	**3D IoU**	**mAP**	Car	Pedestrian	Cyclist	Transporter	Bus	Truck
Velodyne64	0.3	42.98	**80.34**	47.32	50.09	2.41	30.30	**47.43**
	0.5	38.68	77.59	37.26	47.17	2.37	23.81	43.86
	0.7	21.81	62.26	4.06	25.98	2.07	13.20	23.30
Scala1	0.3	19.90	56.82	4.53	15.10	0.58	24.90	17.47
	0.5	14.69	46.04	0.97	9.63	0.51	17.03	13.96
	0.7	5.58	20.67	0.05	1.73	0.18	5.48	5.39
Scala2 (front)	0.3	**49.86**	76.08	**54.00**	**55.28**	**28.55**	**54.51**	30.71
	0.5	42.89	70.41	39.56	46.59	26.61	47.22	26.97
	0.7	19.10	43.24	2.87	21.70	13.10	21.54	12.17
Data	**BEV IoU**	**mAP**	Car	Ped.	Cycl.	Transp.	Bus	Truck
Velodyne64	0.3	43.13	**80.47**	47.70	50.10	2.41	30.30	**47.79**
	0.5	40.97	79.41	42.28	48.50	2.40	27.21	46.03
	0.7	29.73	72.79	10.97	34.58	2.37	19.09	38.60
Scala1	0.3	20.69	59.01	5.49	15.44	0.58	25.05	18.58
	0.5	18.05	54.42	1.94	11.48	0.56	24.52	15.38
	0.7	11.06	39.59	0.19	4.17	0.51	11.46	10.42
Scala2 (front)	0.3	**50.65**	76.97	**54.66**	**56.56**	**28.79**	**56.23**	30.71
	0.5	47.29	75.59	49.07	49.02	28.47	52.91	28.69
	0.7	32.15	68.26	11.94	35.30	24.71	30.71	22.00

Table 5.2: *Performance comparison of the presented approach for 3D and birdview (BEV) object detection based on AP percentages, where higher values are better. Values are reported for different IOU thresholds* $0.3, 0.5, 0.7$*, and the best result of each column is marked in bold. Note that Scala2 has a restricted region of interest. As can be seen, there is a significant drop in performance depending on the IOU threshold in most cases. This indicates fine-grained weaknesses in the regression of individual parameters, while the coarse localization of objects is well learned. Furthermore, sparse point clouds with far less vertical sampling from Scala1 drop significantly in performance. Except for Scala2, the model is not able to recognize Transporters well, as in most cases, confusion with Cars occurs due to the similarity and biased distribution across the classes.*

is capable of classifying and localizing most object occurrences but makes subtle mistakes due to ambiguities from sparsity, partial occlusion, or interaction with other objects. For instance, Figure 5.12 shows prominent cases, where the *Car* in front has just a couple of point measurements from the rear, and the pole of a traffic light interacts with a nearby *Pedestrian*. A similar situation can be observed with groups

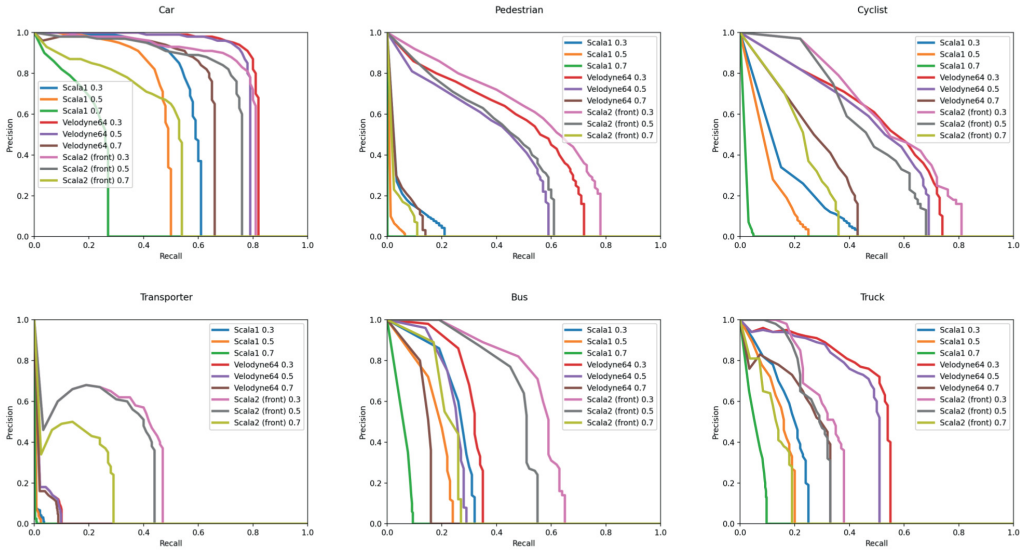


Figure 5.10: *Comparison of 3D object detection results on Scala1, Scala2, and Velodyne, visualized as class-wise PR curves. As can be seen, there is a substantial variance across IOU thresholds* $(0.3, 0.5, 0.7)$*, classes, and underlying sensors. While the model clearly performs the best for Cars on all three sensor sets, it struggles with other classes due to varying reasons.*

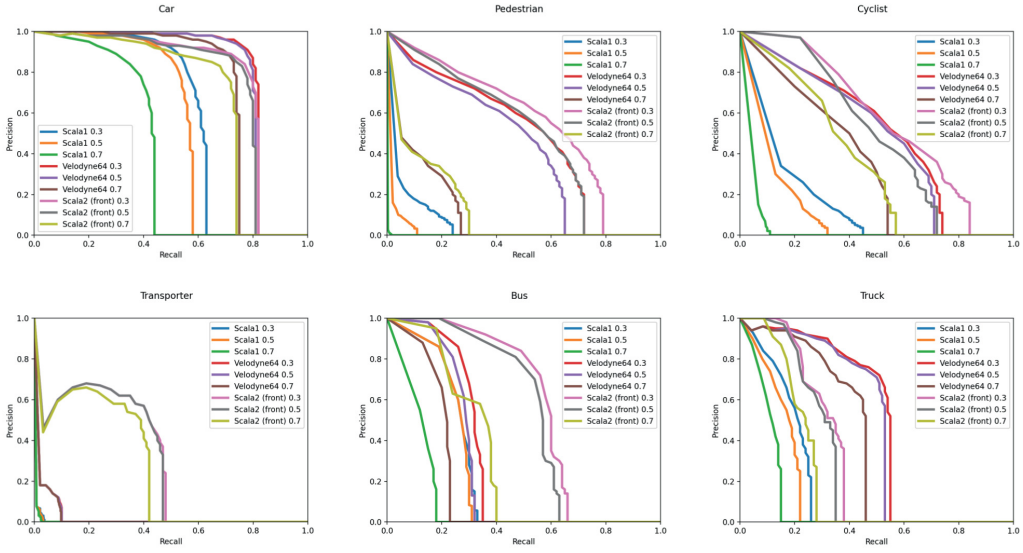


Figure 5.11: *Comparison of birdview object detection results on Scala1, Scala2 and Velodyne, visualized as class-wise PR curves for IOU thresholds* $(0.3, 0.5, 0.7)$*. All plots follow the trends and are consistently better or equal to the 3D metrics presented in Figure 5.10. Hence, the discrepancies within the graphs of one sensor are also smaller. However, there remains a strong variance across the classes and sensors corresponding to the distribution in training data as well as the density of point clouds.*

of small objects like *Pedestrians*, e.g. waiting at a bus stop. Furthermore, many objects are truncated, especially in near to mid-range, as they are partly outside the field of view of the sensors. In such cases, required information about the surface and contours of objects is not adequately represented in the measured point clouds. Therefore, the detection accuracy also decreases significantly, especially at higher distances. This can be seen in the remarkably increased performance for smaller IOU thresholds of 0.5 and 0.3. Moreover, results for Scala1 are generated with the limitation of inaccurate ground truth, as all labels for training and evaluation are interpolations from Velodyne64 with small errors from synchronization.

In general, similar trends can be seen as in the evaluation on benchmark data from chapter 3. *Cars* work best by far, whereas small and relatively rare objects are troublesome. The reported performance on Velodyne64 is in total considerably lower due to differences in the evaluation metrics described in subsection 5.4.3. Overall, Scala2 is on par with Velodyne64, helped by a few reasons. First, the region of interest is limited to the front only. Second, the diversity and the distance from training to test data are lower because only recordings from Munich on the same day are included instead of multiple. Compared to Velodyne64 and Scala2, there is a major drop in performance for Scala1 throughout all categories. First of all, the density of points is reduced to a minimum with missing crucial information through neighboring relations. As a result, the model can hardly abstract more descriptive features, and predictions become uncertain and inaccurate. Furthermore, the higher the relative velocity of the objects, the less accurate the ground truth propagated from Velodyne64, since all point clouds were recorded at different frame rates and synchronized by nearest neighbor timestamps. Here, up to $1\,m$ offset occurs in worst case scenarios leading to potential misalignment during training and evaluation. In addition, by design Scala2 and especially Scala1 barely cover objects in the height dimension at close range, often showing only the lower half of objects, as presented in Figure 5.12.

Overall, the performance of the class *Transporter* drops significantly across all classes for Velodyne64 ($mAP < 0.30$) and Scala1 ($mAP < 0.03$). Here, the root cause is mainly due to an inconsistency in labeling. Parts of the ground truth labels mixed up *Transporter* with *Car* because of the similarities. In essence, the model tends to predict *Car* instead of *Transporter*, as the number of instances is also extremely predominant. On the other hand, all *Transporter* wrongly classified as *Car* barely stand out in the amount of *Car* instances but generate some false positives as well. All corrupted labels were corrected for Scala2 during the individual annotation.

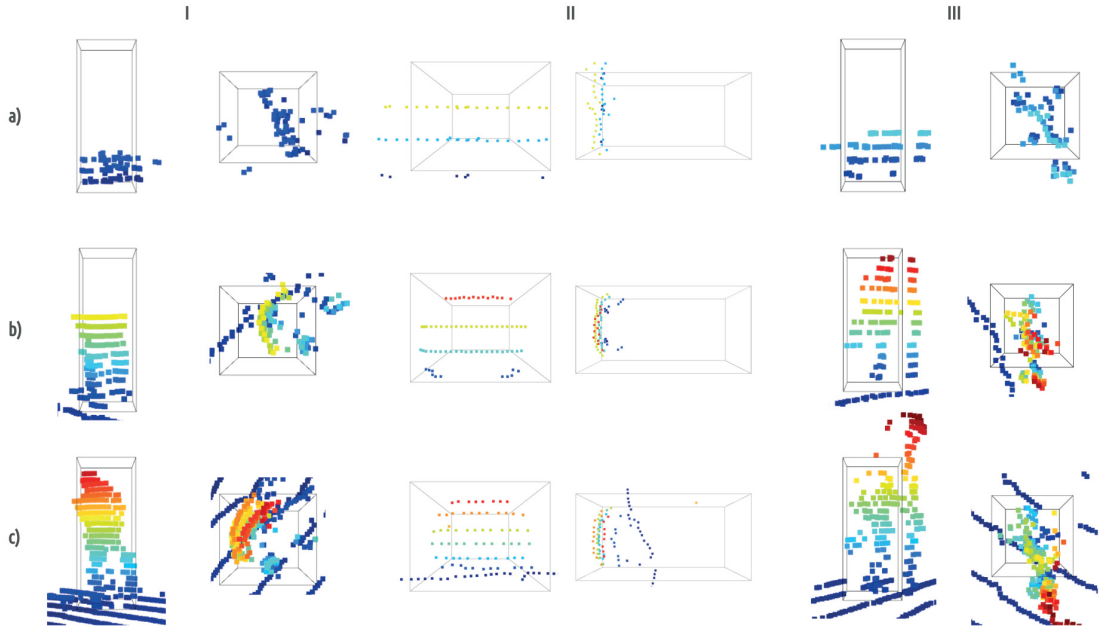


Figure 5.12: *Examples of common failure modes from the test split: I) a partly incomplete mid-range pedestrian, II) a vehicle directly in front at a distance of approx. 45 m and III) a pedestrian close to the pole of a traffic light. Point clouds from a) 6× Scala1, b) Scala2, and c) Velodyne64 are cropped around the object bounding boxes from ground truth, each presented in behind and top-view perspective, colored by height. In Scala1 point clouds a), the low vertical resolution and strongest sparsity becomes a major challenge as object surfaces are mostly unseen or truncated. This improves with Scala2 b), where for example, at least half of the pedestrian I) is covered in b). The problem of incomplete object surfaces is further highlighted in II), where only a few patterns of the back represent the object shape making recognition more difficult. In addition, III) highlights the potential problem of delimiting nearby objects from each other. Overall, these problems get even worse at higher distances with far fewer points. Note that the sizes are scaled individually for better visibility and in a) bounding boxes are inaccurate, as they are interpolated from Velodyne64 annotations without further corrections.*

Therefore, a performance of a completely different order of magnitude was achieved. But still, many *Transporter* instances are wrongly classified as *Car* or even *Truck*, because the underlying point clouds have very similar characteristics.

5.5 Point Cloud to Image Translation

As the diversity and balance of data is a major key for robust models with good generalization, domain translation techniques can be utilized in order to enhance sparse data instead of laborious manual acquisition. Most supervised learning methods are

based on the assumption that training and testing data are drawn from the same distribution. If one violates this constraint, a model trained on a source domain will most likely experience a drop in performance when tested on the target domain due to differences between domains [Patel et al. 2015], which might quickly result in fatal accidents in the case of self-driving cars. Therefore, a careful preparation of large-scale datasets with multiple sensor modalities is particularly important, since they lead to time-consuming and costly efforts. All major public datasets have taken up to one year for recordings alone and more years are required to prepare suitable sequences for training, testing, and validation. A comprehensive list of related public datasets can be found in Appendix B. Although there are more than a dozen such datasets nowadays, they cover only a fraction of possible real-world scenarios. Hence, in order to provide alternatives, the research field of transfer learning has been established, empowered by deep learning.

This section presents the developed approach, called Points2Pix, for conditional image generation from learned point cloud characteristics based on a Generative Adversarial Network (GAN). The content is mainly derived from [Milz et al. 2019]. First, state of the art before the publication of this work is reviewed in subsection 5.5.1. Thus, the developed model is described in subsection 5.5.2. This is followed by experiments conducted on two fundamentally different benchmark datasets in subsection 5.5.3 and a brief conclusion with a broader impact.

5.5.1 Related Work

After the breakthrough in image processing with the help of GAN architectures described in subsection 2.2.1, only a few methods were adapted for point cloud processing. As a precursor, [Wu et al. 2017] deals with single image 3D reconstruction, inspired by [Wu et al. 2016]. Here, two encoding decoding networks for 2.5D sketch estimation were proposed, including normal, depth and silhouette, followed by 3D shape estimation. The output is a 3D voxel-based reconstruction of the input image. The first deep generative models based on point clouds were introduced in [Achlioptas et al. 2018], where three architectures were presented: i) an autoencoder based on PointNet [Qi, Su, et al. 2017], that was trained first, then fixed, and a GAN trained in the latent feature space of the bottleneck from the autoencoder. Both generator and discriminator consist of simple Multi-Layer Perceptrons (MLPs) with a single and two hidden layers, respectively, ii) a GAN with raw point cloud inputs, where the generator consists of five fully connected ReLu layers and a discriminator similar to

the autoencoder of the previous approach, and iii) Gaussian mixture models fitted on the latent space from the autoencoder, used to sample and decode with the decoder from the autoencoder. In contrast, [Li et al. 2019] showed that common discriminators from image processing are not suitable for point sets. Therefore, architectures based on PointNet [Qi, Su, et al. 2017] are used as discriminator. Furthermore, learning a feature descriptor of point cloud inputs to be used as an additional input into the generator was proposed to overcome this problem.

The state of the art at the time of this work is mainly focusing on deep generative models to explore several variants for domain adaptation. There is a wide range of subtasks, with image to image translation being the most significant one. However, only very few work exists dealing with point clouds, and none of them aims to translate properties from point clouds into images that can be used to enhance datasets. Therefore, this section addresses this deficit with a novel generative model parameterized via additional input conditions described as follows.

5.5.2 Conditional Generative Model

The basic idea of the developed model (Points2Pix) is to learn and convert knowledge of 3D objects from a distribution of training data into loosely generated images. At the same time, important properties like the pose or shape of the object shall be parsed by the model and translated into the output. In addition, an arbitrary background image patch is used to parameterize the environment while again containing the 3D point cloud projected into the image plane in order to enhance the control over the image synthesis further. Thus, by entering and manipulating these predefined conditions, completely new samples with desired properties can be generated. For instance, any number of images can be created for a rare vehicle, retaining the concise features and basic characteristics as specified in the conditions. Simultaneously, the input of the background image patch allows for variations of the main visual context, such that the environment in the generated synthetic image is also flexible.

Inspired by [Isola et al. 2017], Points2Pix aims to generate realistic images in a conditional GAN setup, where the main input comes from a point cloud of a single object instead. A high-level overview of the full pipeline is visualized in Figure 5.13. Hence, the task includes the translation of structural and geometric properties from the point cloud domain and merges them with textures from a learned image distribution out of a training dataset. This is solved by a generator consisting of two

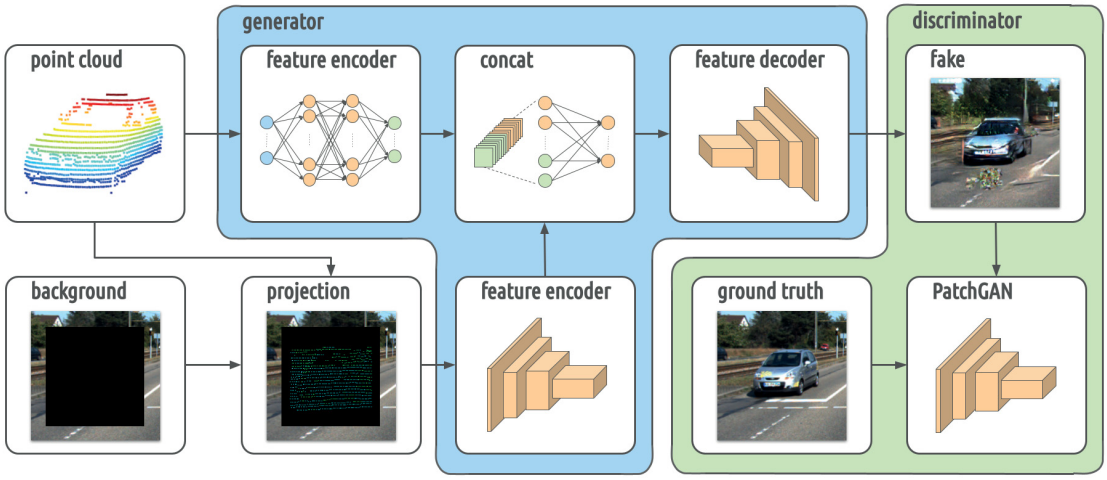


Figure 5.13: *Overview of the Points2Pix model architecture for point cloud to image translation. This figure highlights the competing adversarial formulation of a jointly trained generator (blue), based on raw point cloud input as well as a background image patch combined with points mapped from 3D into 2D via camera projection and a discriminator (green). While the aim of the generator is to synthesize a complete image (fake) by translating 3D characteristics from the point cloud and surroundings from the background image patch, the PatchGAN [Isola et al. 2017] discriminator learns to classify the result at the scale of the local image patches into real or fake in order to enforce the generator to output fine-grained realistic textures.*

combined sub-networks as well as three input conditions, jointly trained with the discriminator network. The first part is a point feature encoder based on PointNet [Qi, Su, et al. 2017], to generate global 3D features capturing the shape and description of the underlying object. Second, a UNet autoencoder [Ronneberger et al. 2015] is used to generate the images from the input patch filled with pixels from the background condition as well as the projected points. Here, features from the point encoder and image encoder are concatenated at the innermost part. Both the generator and the discriminator networks, as well as the loss to jointly train them, are explained in more detail in the following paragraphs.

Generator. A defining feature for the underlying problem is to fuse features from point clouds with features from the image space while mapping a high-resolution input to a high-resolution output grid. In addition, structural and geometric details mapped from the point cloud need to be aligned with the output, while the visual appearance can be drawn from the distribution in training data. Therefore, the proposed generator $G(c_1, c_2, c_3)$ with conditions c_1, c_2, c_3 is adapted from [Isola et al.

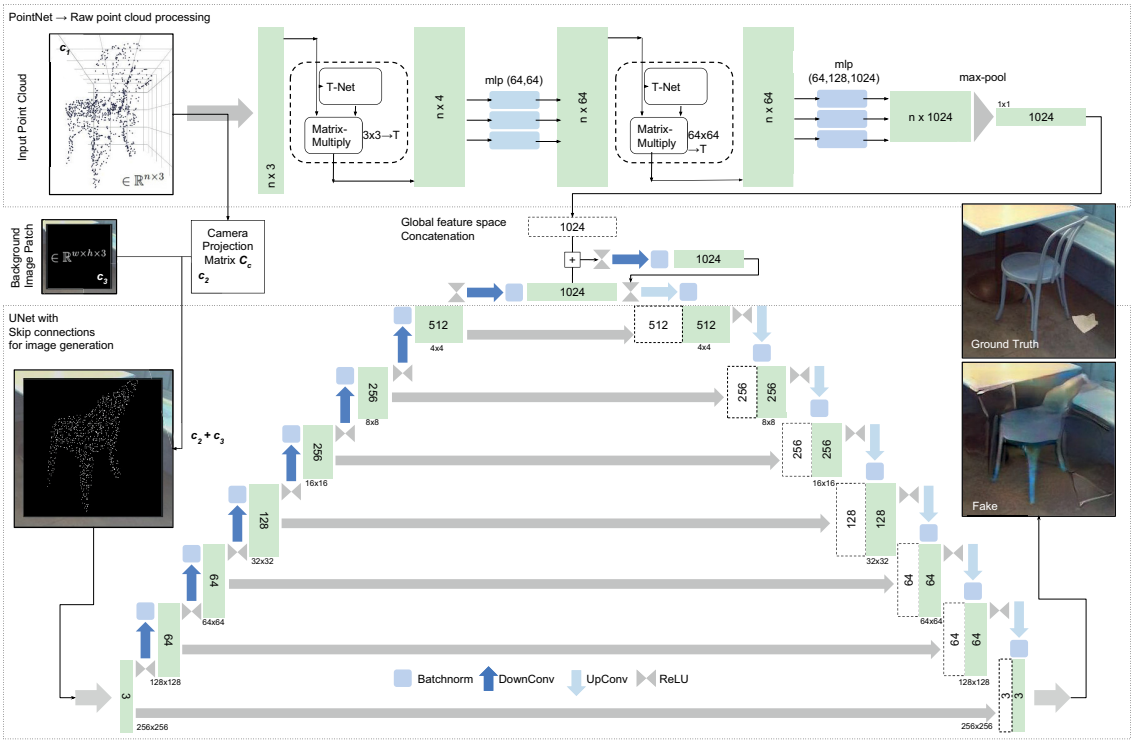


Figure 5.14: *Detailed architecture of the generator. The overall pipeline is based on three input conditions and can be split into three areas: top) point cloud feature encoding based on [Qi, Su, et al. 2017], middle) feature-level fusion, and bottom) convolutional UNet architecture from image processing [Ronneberger et al. 2015]. Reproduced from [Milz et al. 2019].*

2017] based on UNet [Ronneberger et al. 2015] and extended by a PointNet [Qi, Su, et al. 2017]. A detailed illustration of the network architecture can be found in Figure 5.14.

All three conditions are obtained in a preprocessing step based on the underlying dataset. An overview of the assembly can be found in Figure 5.15. First, in $c_1 \in \mathbb{R}^{n\times 3}$ the raw point cloud of an object is cropped from the global scene and randomly sampled to n points. Here, n points are chosen with equal probability as a subset of all points inside a 3D bounding box, while individual points can occur multiple times if there are less than n points contained in total. Afterwards, a PointNet [Qi, Su, et al. 2017] is used, where two stages of feature space transformations followed by fully connected MLPs are carried out, and a final max pooling layer aggregates global point cloud features. These features are combined with intermediate camera features at the innermost part of the UNet [Ronneberger et al. 2015] in order to provide meaningful features for the rendering of 3D characteristics.

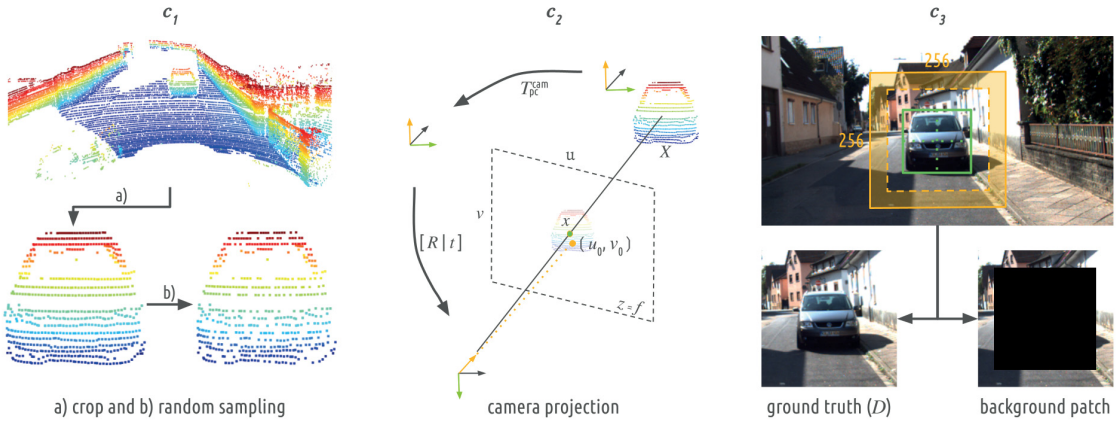


Figure 5.15: *An overview of how the conditions c_1, c_2, c_3 are generated on a sample from KITTI [Geiger et al. 2012]. While the point cloud of an object is used both directly as input c_1 and indirectly to generate further input features via back projection into the image plane c_2, a background patch c_3 as well as the ground truth image to be used for the discriminator is also cut out of the camera image. Therefore, ground truth bounding boxes are used in 3D and 2D (green box) to extract the relevant regions. Point clouds are colored by height. More details and equations are described in the generator paragraph of subsection 5.5.2.*

The second condition c_2, i.e. an image with features projected from the same point cloud input as in c_1, is generated through a camera projection matrix P using homogeneous coordinates:

$$kx = PT_{\text{pc}}^{\text{cam}}X \tag{5.1}$$

where k is a constant, $x = (u, v, 1)^T$ are pixel coordinates, $X = (X, Y, Z, W)^T$ are point coordinates from the point cloud extended with homogeneous coordinate W, $T_{\text{pc}}^{\text{cam}}$ is an extrinsic transformation matrix from point cloud into a world coordinate system and P can be denoted as:

$$P = K\,[I \mid 0] \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} = K\,[R \mid t] \tag{5.2}$$

where R is a 3×3 rotation matrix and t a 3-dimensional translation vector, both representing the rotation and translation of the camera from world reference, and a matrix K containing the intrinsic camera parameters defined by:

$$K = \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

with the coordinates of the principal point u_0, v_0, a skew factor s, $\alpha_u = \frac{f}{unit_w}$, the width of a pixel in world units $unit_w$, $\alpha_v = \frac{f}{unit_h}$, the height of a pixel in world units $unit_h$ and f being the focal length. One color channel of the resulting image coordinates x normalized by k is filled with the Euclidean distance from the origin of the camera to point coordinates X in world reference. This results in a sparse image similar to a depth image and can be calculated by:

$$y_g(x) = \frac{\|T_{\text{pc}}^{\text{cam}} X\|}{d_{\max}} \tag{5.4}$$

where $d_{\max}$ is a normalizing scaling factor and the resulting green color channel is denoted by y_g. In case point clouds contain an additional intensity value for each point, e.g. the reflectivity from most lidar sensors (see Appendix A), a second color channel (blue) can be filled with $y_b(x) = I(X)$.

Finally, the third condition c_3 is an arbitrary image patch with the surrounding of an object cropped from camera images as visualized in Figure 5.15. First, an area of 256×256 pixels is extracted as ground truth for the discriminator D centered at the center of a given 2D bounding box from ground truth for objects. Then, the pixel values from the inner area of this ground truth image are removed, which are to be regenerated later by the network. Only the border is kept at a predefined width of 15 pixels to constraint environmental rendering. Thus, for each visible object for which ground truth bounding boxes are also present in 2D and 3D, an input sample with c_1, c_2, c_3 can be generated. Objects that are close to the edge of the image are filtered out. During training, the image background patch is compliant to the ground truth reference for the discriminator D, while in test mode, background patches can be randomly selected to produce completely new image creations. In order to increase efficiency, c_2 is merged into the empty image center in c_3, before feeding both into the UNet [Ronneberger et al. 2015]. After point and image features are encoded in parallel, both are fused at the innermost part. This is followed by several up-sampling layers to generate the final output image.

Discriminator. The assessment of the output from the generator is performed by an adapted version of the Markovian discriminator from [Isola et al. 2017]. It consists of 5 convolutional layers with batch and instance normalization as well as dropout layers and tries to distinguish between fake $D\left[G(c_1, c_2, c_3)\right]$ and real images $D[y]$ at the scale of $N \times N$ patches. In contrast to [Isola et al. 2017], the input of the discriminator is restricted to the generated image, and the conditions c_2, c_3 as the

raw point clouds c_1 cannot be processed by a CNN without further preprocessing. To effectively distinguish fake from real images, while being able to model high- and low-frequency structures in parallel, an additional L_1 term is applied by shifting a window across the generated image, similar to a convolution as originally proposed by [Li and Wand 2016].

The overall objective of the conditional generative model can be formulated as an additive combination of the loss from the generator L_G and the discriminator L_D. During training, both are trained simultaneously while iteratively reducing L_G and increasing L_D ideally. In other words, the output from the generator improves with more realistic images, while the discriminator progressively struggles to distinguish them from real ones. Following Equation 2.6 and incorporating the conditions c_1, c_2 as well as c_3, the basic loss can be described as:

$$L_{cGAN}(G, D) = \mathbb{E}_y[\log(D(c_2, c_3, y))] + \mathbb{E}_{c_1,c_2,c_3}[\log(1 - D(c_2, c_3, G(c_1, c_2, c_3))] \quad (5.5)$$

Compared to Equation 2.6, the random noise input z is removed but implicitly incorporated as dropout layers during training, similar to [Isola et al. 2017]. In order to force the generated output to be compliant with the conditions, the aforementioned L_1 difference between the generated output and the ground truth is added. Therefore, the final loss can be denoted as:

$$L_{\text{Points2Pix}} = L_{cGAN}(G, D) + \lambda_{L_1} \cdot \mathbb{E}_{c_1,c_2,c_3,y}\left[||y - G(c_1, c_2, c_3)||_1\right] \quad (5.6)$$

with weighting factor λ_{L_1}, as proposed by [Wang et al. 2018].

5.5.3 Experiments

This subsection introduces the datasets, all underlying training and model parameters, as well as the metrics used to evaluate the developed model. To emphasize the generalization, the results are presented on two fundamentally different datasets, where alternating conditions like different backgrounds or random rotations to the point cloud input prove the capabilities to generate high diversity. On top, two simplified network architectures, called $Points2Pix_{\text{no\_pointnet}}$ and $Points2Pix_{\text{half\_unet}}$, are examined to determine the influence of individual input conditions as well as the capabilities of synthesizing images with the help of a Deep Neural Network (DNN) trained as a generator. An overview of the different network variants is illustrated in Figure 5.16.

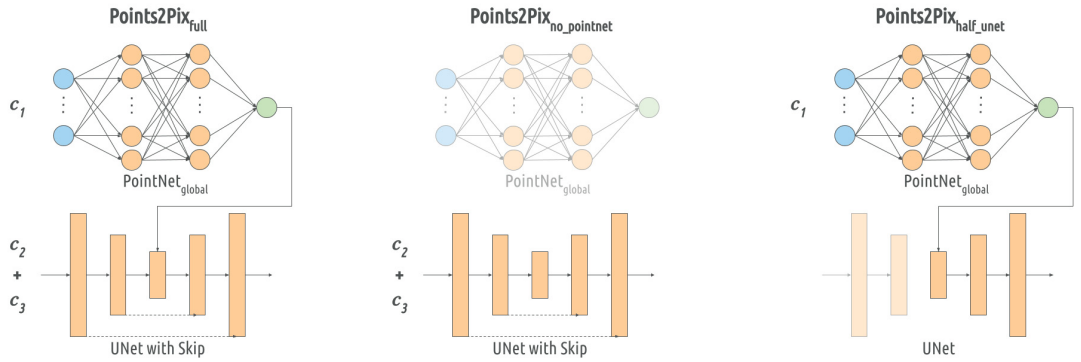


Figure 5.16: *Architectural review of the proposed Points2Pix generator (left), a derivative without the raw point cloud condition c_1 (middle) and a variant without the features encoded from the camera projection c_2 as well as the background patch c_3 (right). Thus, both $Points2Pix_{\text{no\_pointnet}}$ and $Points2Pix_{\text{half\_unet}}$ each operate only in image space or with point cloud input, while the full Points2Pix model benefits from all input conditions $G(c_1, c_2, c_3)$ by implicitly learning a mapping between the spaces. Reproduced and modified from [Milz et al. 2019].*

Datasets. As in chapter 3, the KITTI object detection dataset [Geiger et al. 2012] is used to evaluate the model for the target environment of Autonomous Driving. Again, the same splits are used with $3,712$ samples for training and $3,769$ samples for validation. In preprocessing, the data for single objects is generated by cropping their area in the images using the 2D bounding box labels and the points from the point cloud belonging to the object using the 3D bounding box labels, respectively. Here, strongly occluded or truncated objects are skipped using the existing labels. In addition, objects with more than $d_{\text{max}} = 60\,m$ distance or with less than 700 points are filtered out. All images are resized to match 256×256 pixels using bicubic interpolation. Based on this setup, more than $20k$ samples are extracted for training using *Cars* only.

Furthermore, the indoor dataset SunRGBD [Song et al. 2015] is used to prove generalization. In contrast, RGB-D sensors are used to capture this dataset, instead of lidar. Therefore, the projection in condition c_2 only contains the radial depth y_g, since no additional intensity values $I(X)$, i.e. the reflectivity of materials measured by a lidar, are available. By applying the same preprocessing with $d_{\text{max}} = 4\,m$, $3,267$ samples are generated with *Chairs*, *Tables*, *Desks*, *Pillows*, *Sofas* and *Garbage Bins*. Due to very limited size, $2,940$ samples are used for training and only 327 samples for validation.

Training and Optimization Details. The developed model and the variants described in Figure 5.16 are trained from scratch on both training datasets for 100 epochs using the Adam optimizer [Kingma and Ba 2015] with a batch size of 4, learning rate of of 0.0002 and momentum $\beta_1 = 0.5, \beta_2 = 0.999$ such as $\lambda_{L_1} = 100$. All point cloud inputs are randomly sampled to $n = 1,024$ points, i.e. individual points can occur multiple times for point clouds from objects with fewer points than n (see paragraph generator in subsection 5.5.2). The background image patch c_3 with the environmental surroundings contains 15 pixels from the outer border to the inner of the image, leaving a square of 226 pixels to be generated (see c_3 in Figure 5.15). For the generator, batch normalization [Ioffe and Szegedy 2015] and ReLu activation are used, while the discriminator is based on leakyReLu and a sigmoid activation after the last convolutional layer. These values were acquired through quantitative testing.

Evaluation Metrics. According to the desired properties of generative models, several measures have been proposed to estimate the performance. However, there is no clear consensus about which measure is the best for comparison [Borji 2019]. Therefore, in order to assess the performance of the developed model, three specific scores, namely a classification score S_c, an object-based score S_{iou} and a diversity score S_d, are used, inspired by [Wang and Gupta 2016]. Hence, a YOLOv3 [Redmon and Farhadi 2018] detector trained on ImageNet [Deng et al. 2009] and Microsoft coco [Lin et al. 2014] is utilized to detect 2D objects in the generated images as well as the original ones. Then, the output is compared to the existing ground truth from manual annotations while assuming that such a detector will work if the images are realistic enough.

First, a number of correctly detected classes S_c can be denoted as:

$$S_c = \frac{TP_{fake}}{TP_{real}} \tag{5.7}$$

where a true positive TP is a correctly classified detection independent of the related bounding box when compared to the ground truth class label. Moreover, $fake$ denotes the generated image and $real$ the original target image. Second, the IOU can be considered for all TP detections, resulting in:

$$S_{iou} = \frac{BB_{\text{fake}} \cap BB_{\text{real}}}{BB_{\text{fake}} \cup BB_{\text{real}}} \quad | \; S_c = 1 \tag{5.8}$$

such that only positive results in terms of classification are evaluated $S_c = 1$, where BB_{fake} is the detected 2D bounding box in the generated image and BB_{real} the

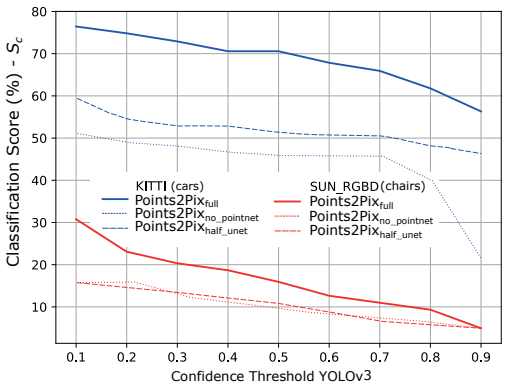


Figure 5.17: *Classification scores S_c plotted over the confidence from an object detector for Cars (KITTI [Geiger et al. 2012]) and Chairs (SunRGBD [Song et al. 2015]). Here, confidence describes a lower bound for the estimated certainty of each predicted object. A more detailed description is given in the results paragraph in subsection 5.5.3. Reproduced from [Milz et al. 2019].*

detected 2D bounding box in the original image, respectively. The third score S_d measures the diversity based on the average S_{iou}, when alternating the background condition $c_3 \rightarrow \{1...N\}$:

$$S_d = \frac{1}{N} \sum_{i=1}^{N} S_{iou_i} \tag{5.9}$$

In this way, $N = 10$ different background patches are randomly selected as input to condition the diversity of the generated output images while verifying the detection from YOLOv3.

Results. In this paragraph, the aforementioned quantitative metrics as well as qualitative results are reported for both datasets. In Figure 5.17, the classification score S_c can be found for *Cars* on KITTI as well as *Chairs* on SunRGBD. Here, $Points2Pix_{full}$ corresponds to the full model with $G(c_1, c_2, c_3)$, while $Points2Pix_{\text{no\_pointnet}}$ is a version without features from the input point cloud c_1 and $Points2Pix_{\text{half\_unet}}$ without c_2 and c_3 based on the point cloud input only, as depicted in Figure 5.16. As can be seen, the results on KITTI are significantly better. This is mainly due to uncertainties in YOLOv3 [Redmon and Farhadi 2018] for smaller, underrepresented classes such as *Chairs*, which leads to frequently misclassified or misdetected objects. Nevertheless, the network benefits from all input conditions with consistently up to 20 percent better results. On the one hand, this emphasizes an implicitly learned mapping from

Dataset	Class	S_{iou}			S_d		
		0.3	0.5	0.7	0.3	0.5	0.7
KITTI	Car	0.76	0.77	0.77	0.71	0.70	0.68
		S_{iou}			S_d		
		0.1	0.2	0.3	0.1	0.2	0.3
SunRGBD	Sofa	0.52	0.77	0.77	0.16	-	-
	Table	0.70	-	-	0.24	0.22	-
	Chair	0.60	0.58	0.58	0.45	0.37	0.33

Table 5.3: *Quantitative results with S_{iou} and S_d scores for KITTI and SunRGBD data: The values are presented for several detection confidence thresholds, i.e.* 0.3*,* 0.5*, and* 0.7 *in case of KITTI, as well as* 0.1*,* 0.2 *and* 0.3 *for SunRGBD to address uncertainties from the YOLOv3 network. A minus indicates no detections for the associated class. Reproduced from [Milz et al. 2019].*

point cloud to image space. On the other hand, the knowledge from the distribution in the training data is better reflected by the learned weights resulting in more meaningful feature representations. A top performance of roughly $C_s = 76$ percent is reached for *Cars* at a low confidence threshold of 0.1. Moreover, the results decrease with an increasing confidence threshold, again indicating uncertainties of the underlying detector. However, YOLOv3 is able to detect most of the objects in the generated $fake$ images, demonstrating that the developed model produces sufficient structures and patterns similar to real images.

In addition, Table 5.3 presents both, the IOU based detection scores S_{iou} and the diversity scores S_d. Again, similar behavior can be observed in comparison to C_s, while the impact of the confidence threshold is less. This indicates that the detector is also capable of predicting the bounding boxes on generated $fake$ images with high accuracy compared to the bounding boxes detected on real images. Likewise, Figure 5.18 gives an impression with pairs of the recognized objects in $real$ (right) and $fake$ (left) images, each with detected object on $fake$ in green and on $real$ in red. Here, the behavior of the detector is extremely similar in all examples, which also demonstrates positive characteristics of the generated images similar to real ones.

In most cases, the model is able to seamlessly continue patterns from the background patch c_3 such as road markings or shadows from the sunlight. Furthermore, in some cases, the basic colors of the objects are changed, thus creating more diversity. However, some objects still have somewhat unnatural structures, which also lead to slightly reduced performance of the detector. In particular, the variation of the background patch c_3 (see Figure 5.15) partly degrades the results of S_d seen in Table 5.3.

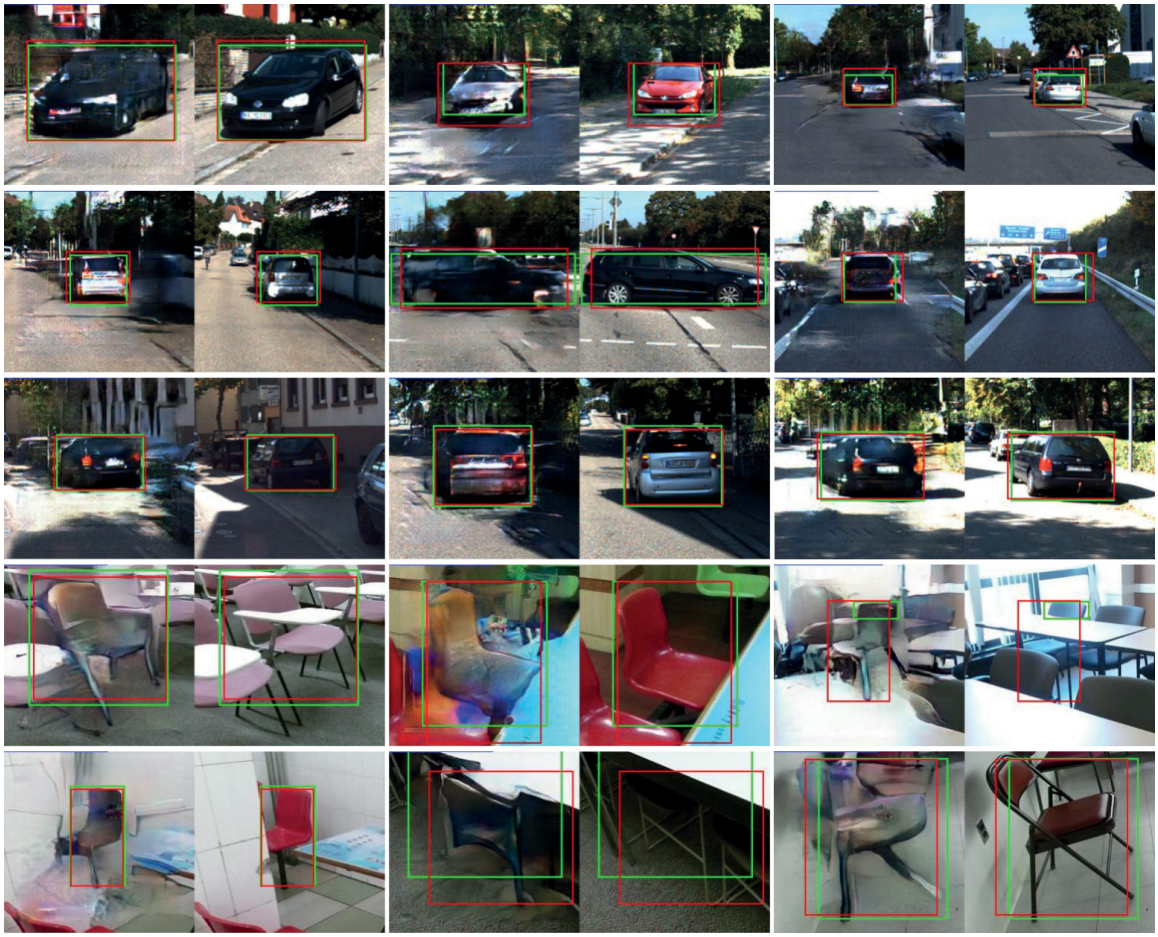


Figure 5.18: *Examples of generated images on the validation sets with results from object detection compared to corresponding real images. Multiple pairs of fake (left) and real (right) images are visualized from Cars or Chairs decorated with bounding boxes detected by YOLOv3 [Redmon and Farhadi 2018]. Green bounding boxes are detected in fake, while the red ones are from real images. A more detailed description is given in the results paragraph in subsection 5.5.3. Reproduced from [Milz et al. 2019].*

For instance, Figure 5.19 shows different examples with varying backgrounds. It can be clearly seen that properties from the point cloud c_1 as well as the back projection in c_2 are translated into the generated images, making all objects easily recognizable. Similarly, positions and orientations, as well as the viewpoints, are transferred very accurately from input conditions into the output. Notably, the basic brightness, contrast, and daylight are still seamlessly integrated, even though completely different scenes have been generated. In contrast, some areas of the images are often blurred, resulting in less fine-grained details and structures. Especially colored objects from indoor scenarios in SunRGBD confuse the generation of the environment and lead to

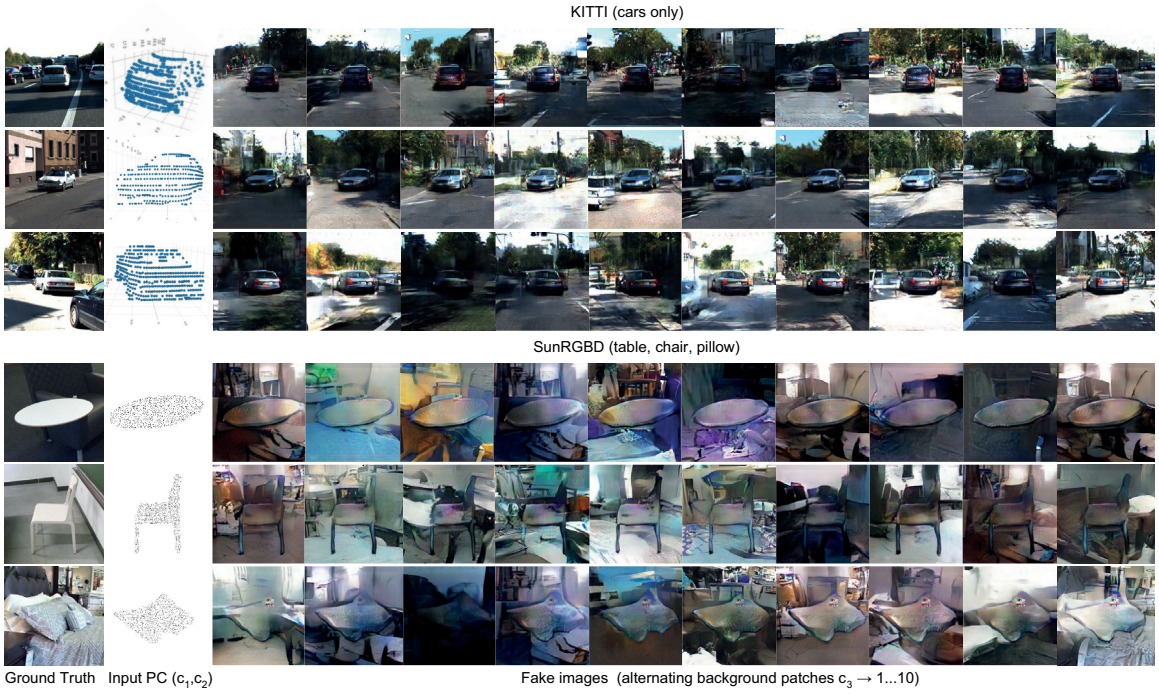


Figure 5.19: *Results with varying background condition c_3. Generated images from four different object classes with ten different background inputs from the validation sets are visualized. The left column shows the original image, followed by the corresponding point cloud used as constant input conditions c_1 and c_2. Throughout all of the samples, the 3D characteristics are translated and clearly visible in 2D, while the environmental patterns are mostly realistic. A more detailed discussion is given in the results paragraph in subsection 5.5.3. Reproduced from [Milz et al. 2019].*

unnatural large, almost monochrome colored areas. In addition, some of the outputs tend towards abstraction or art, potentially due to the significantly smaller amount of training samples. Nevertheless, 3D patterns from point clouds are learned and translated into the image space with clearly visible patterns and characteristics in 2D, while the main content in the images integrates very well into the forced backgrounds.

Another way to increase the flexibility of data generation using the developed model is to modify the point cloud input, affecting the raw point condition c_1 and the image projection c_2. In particular, transformations like global rotation can be used to generate specific orientations of the objects. For instance, Figure 5.20 presents some qualitative results with partly over-stressed rotation, where the objects are upside down. Although the generated images are a bit blurry around the objects, the intended contours and shape are clearly visible.

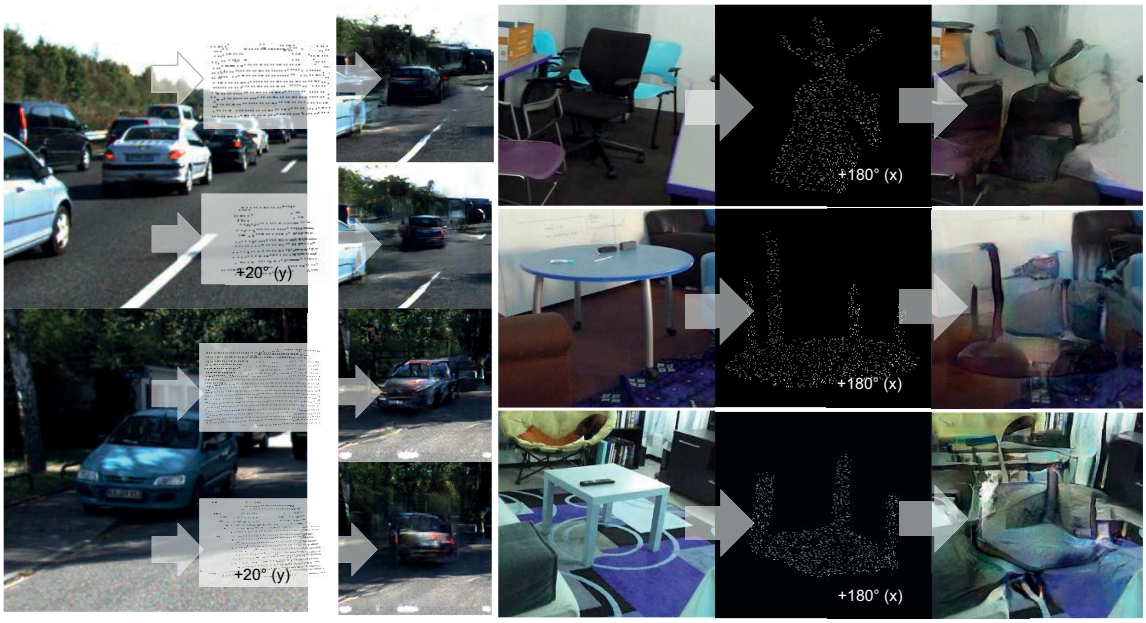


Figure 5.20: *Results when modifying the input point cloud in raw point condition c_1 and image projection c_2. The left part shows two examples of Cars from the KITTI validation set when rotating the point cloud by 20° around the up axis. The right part contains examples from the SunRGBD validation set with a flipped-up axis. Although the generated objects are slightly blurred, the 3D representation can be successfully translated into the image. Reproduced from [Milz et al. 2019].*

5.5.4 Conclusion

In this section, a pioneering approach for point cloud to image translation called Points2Pix was presented. An existing conditional image to image model from [Isola et al. 2017] was adapted to generate highly realistic images mainly based on 3D properties from point cloud inputs. Therefore, a novel network architecture based on PointNet [Qi, Su, et al. 2017] combined with UNet [Ronneberger et al. 2015] was proposed. Here, three conditions, c_1 a raw point cloud of an object of interest, c_2 an image back-projection of c_1 and a background image patch c_3, are used as input, while the network learns a distribution of the characteristics in training data as well as a translation from 3D to 2D. After the network is trained, a targeted data generation can be implemented to extend existing datasets or to increase and balance diversity simply by modifying one of the input conditions. While keeping important characteristics and structures from 3D, the conditional setup is very helpful in generating a variety of diverse data. However, the output resolution is limited to 256×256, and despite a special error function, blurred areas can occur. This leaves room for further development.

Nevertheless, the model can be used for 3D texturing, lidar and camera fusion, cross sensor domain adaptation, as well as data augmentation, since most AD datasets focus on high redundancy through multi-modal sensors. Building upon this work, several other methods have already been developed. An overview for image to image translation was given in [Alotaibi 2020]. Moreover, an approach for generative image in-painting was presented in [Shao et al. 2020]. Additionally, multiple methods for rendering images from point clouds were developed, such as [Atienza 2019; Kim et al. 2020; Cortinhal et al. 2021; Peters and Brenner 2020]. Likewise, a view synthesis from colored point clouds, i.e. images are rendered from arbitrary viewpoints of a scene, was presented in [Song et al. 2020]. Similarly, [Haiderbhai et al. 2020] generated X-ray images from generic point clouds, while [Caccia et al. 2019] modeled a reconstruction of compressed point clouds from lidar. Also, many perception tasks were enhanced with domain adaptation techniques, e.g. [Saito et al. 2019; Kim et al. 2019; Wang et al. 2019; Zhao et al. 2019; Choi et al. 2019].

5.6 Conclusion

This chapter contains a concept for the integration of the developed methods from chapter 3 and chapter 4 in Advanced Driving Assistance Systems (ADAS) or Autonomous Driving (AD) systems, consisting of an analysis of the application-specific setup as well as a semi-automated construction of a dataset suitable to train and evaluate the developed methods in comparison to the performance on a scientific benchmark dataset. Hence, the essential steps and difficulties for a real-world deployment are highlighted, and the potential of the developed model is explored. However, full integration requires further development to solve revealed problems in detail. First, the implementation and adaptation for embedded hardware are missing, as a specific API has to be used in order to utilize acceleration modules fully. Parts from chapter 4 were neglected for simplification of the experimental evaluation. The proposed fusion with semantic features from camera requires the highest accuracy in sensor synchronization as well as further supervised training data in the image domain. Inaccuracies have already been identified during synchronization based on software timestamps, making manual corrections of ground truth for Scala2 necessary instead of simple propagation from Velodyne64. Although automated annotation of data has modified requirements for detector and tracking, the developed methods could be reused with minor adaptations in a real-world application assisting humans during annotation.

As a result, enormous time and cost savings can be achieved, and moreover, errors from humans can be prevented to increase the quality of the generated ground truth.

Additionally, a model to synthesize images from point clouds based on a Generative Adversarial Network (GAN) was investigated. This is particularly helpful when it comes to multi-modal data acquisition, which is common for AD. Conditions such as a point cloud from single objects, e.g. a vehicle or a section of an image from the target environment, can be given as input to the network in order to provide specific data. As an alternative, such methods are beneficial, especially because rare or underrepresented cases can simply be synthesized rather than laboriously recorded in the real world. On top, the diversity and balance of training datasets strongly influence the convergence and generalization of learning-based approaches such as Deep Neural Networks DNNs. Overall, the conducted experiments on the constructed dataset confirm that the proposed methods also work when operating on restricted automotive-grade sensors in real-world scenarios. The results achieved on point clouds from Scala2 are particularly very promising. Future work will focus on the extension to multiple Scala2 or next-generation sensors, covering the full 360° environment while increasing the size of the dataset. The concepts developed in this work form a solid basis for further development and improvements.

Chapter 6

Summary and Outlook

6.1 Summary

This thesis dealt with the recognition of objects in 3D point clouds for Autonomous Driving (AD). Highly automated driving on public roads provides the foundation for the field of environmental perception, currently one of the most exciting fields of research. The first promising results from computer vision have been fuelled by recent advances in deep learning. However, the pursuit of progressively improved automation up to full autonomy in such safety-critical systems requires complex processing chains with additional redundancy. In particular, the spatiotemporal detection of the surrounding environment plays a central role in the planning and control of vehicles. Therefore, lidar technology has emerged to produce accurate measurements of depth forming point clouds. Nevertheless, both sensor technology and processing are still mainly the subject of research, practically unsuitable for the usual private ownership of vehicles. Based on this, the primary objective of this work was also the application to a closer-to-production system with even more challenging conditions due to requirements for sensor mountings with smooth integration into the vehicle design as well as reasonable hardware and sensor constraints.

In the first part (chapter 3), a novel highly efficient model for object detection on point clouds called Complex-YOLO [Simon et al. 2018] was developed, based on You Only Look Once (YOLO) [Redmon et al. 2016], a Convolutional Neural Network (CNN) from image processing. As one of the first works of its kind, the developed method considerably contributes to the research community with an above-average number of citations. Particular focus was put on a lightweight input representation of point clouds in birdview image space, together with a specific loss function to incorporate robust estimation of the orientation directly into the regression layer using

the complex space. Thus, state-of-the-art has been extended by essential building blocks for 3D object detection with higher efficiency compared to existing approaches.

Based on this baseline, a downstream probabilistic Multi Object Tracking (MOT) was presented (chapter 4), which is the first of its kind to operate directly in a global 3D space with point clouds from lidar. In contrast, sequential processing of point clouds directly in a Deep Neural Network (DNN) is challenging in several aspects. First, recurrent architectures quickly become very complex with many layers. Additionally, the incorporation of regression together with the association over time into a learning process with differentiable loss is severely limited and often unstable. Therefore, the developed approach offers a convenient alternative following the tracking-by-detection scheme using an online tracking based on Labeled Multi-Bernoulli Random Finite Set (LMB RFS) [Reuter et al. 2014; Bryant et al. 2018]. This facilitates the optimization of DNNs as a robust detector and at the same time allows the use of temporal information and target instantiation based on realistic physical assumptions. Moreover, as one of the first works, a feature-level fusion with point clouds converted into voxels and features from camera images was presented. Thus, the capability of the baseline detector is strongly increased by voxels filled with a normalized floating number calculated from back-projected class labels. Hence, an ENet [Paszke et al. 2016] was trained to generate these class labels formulated as pixel-wise semantic segmentation on camera images. This offers the opportunity to efficiently learn improved features that represent complex semantic relationships that are hardly present in irregular point clouds. In addition, a novel measure for object to object matching called Scale Rotation Translation Score (S_{srt}) was introduced. In comparison to the commonly used Intersection Over Union (IOU), S_{srt} comes with higher efficiency and better flexibility since individual components can be weighted. In this way, application-specific requirements, e.g. a prioritization for correct regression of individual object parameters like orientation, localization or scale, can also directly be mapped into the learning process.

In the last part of this thesis (chapter 5), the embedding and integration of the developed methods into an application-specific setup was examined. Accordingly, a dataset was prepared in order to assess the performance with point clouds from automotive-grade lidar sensors in a close-to-production vehicle. This is one of the biggest challenges for data-driven approaches in real-world safety-critical environments, as vast amounts of well-curated data are a prerequisite. Therefore, the developed algorithms were smoothly integrated into a software application for human annotations and iteratively reused for automation. Although the requirements differ

significantly from perception in autonomous vehicles, costs and effort could be effectively reduced, also showing their potential for generalization. Overall, the achieved performance alone is not sufficient for AD level 5. Nevertheless, a significant contribution can already be made to a holistic system based on redundancy, where the developed methods can be used as single or complementary components in parallel to others. Here, the immense potential for scaling via data, the potential exchange of individual building blocks, or more advanced sensor technology is particularly decisive. All shown concepts are also valid for use in real-world comfort and driving applications with partial automation for driver assistance. In addition, an alternative approach for directed cross-sensor data generation was designed (section 5.5), extending state of the art with a novel conditional setting for point cloud to image translation, i.e. image synthesis. This kind of solution with initial results is particularly promising to solve problems related to the acquisition of datasets needed for deep learning approaches. Not only can costs and efforts be reduced, but the aspects of increasing the balance and diversity in the data must also be taken into account, which is very important for the performance of supervised learning methods. However, fundamental research is needed for a more goal-oriented synthesis of data of the highest quality and even more flexibility for real-world applications.

6.2 Conclusion and Outlook

In the context of this work, a comprehensive contribution could be made to the objective set, namely efficient object recognition in 3D point clouds for AD, and thus to push the frontier of environmental perception by expanding deep learning for point cloud processing. Spatial perception is one of the key challenges to realizing self-driving vehicles, with lidar sensors, among others, making tremendous progress in producing accurate and dense measurements of the environment. Although new methods have emerged, there are many opportunities for optimization and expansion.

Further improvements can also be made to the developed model for joint object detection and tracking itself. For instance, modular building blocks like the backbone for feature generation can be exchanged by new trends from state-of-the-art. Furthermore, as many raw points as possible should flow through the network via an efficient representation in order to discard as little information as possible at this early stage to enrich the learning of valuable features. Moreover, there are several directions to investigate. First, any improvement of the detector simplifies the tracking, as the association strongly depends on the underlying predictions. Here, a comparison of

different strategies for the fusion with features from camera would be interesting. Also, multiple point clouds can be used as input into the network at the same time, as a temporal sliding window, in order to make even better use of temporal information. Graph Neural Networks [Wu et al. 2020] or transformers from natural language processing [Vaswani et al. 2017] are particularly well suited for this purpose. However, efficiency must still be increased. Another option is to simplify the regression in the detector, e.g. to transfer the estimation of dimensions to the MOT. Similarly, dedicated motion models for pedestrians and cyclists could contribute to the state estimation. To further increase efficiency and for use in embedded systems, runtime optimizations can be investigated both in the CNN and in pre- or postprocessing.

One of the main concerns throughout this thesis is the dataset issue when applying deep learning algorithms to real-world problems. Due to the complexity and diversity of available driving scenarios, the quantity for sufficient generalization multiplies to an unrealistic order of magnitude. For instance, several traffic environments exist like a highway, inner-city, landscape, or urban areas resulting in varying speeds, driving maneuvers, and density of road participants, as well as environmental conditions such as weather or time of the year and day, plus worldwide diversity for traffic signs and rules in different countries. Also, the dynamic environment of public roads changes permanently, especially due to construction sites or new types of participants, e.g. next-generation vehicles. Additionally, meaningful differences in raw data from diverse or evolving sensor technology as well as varying mounting positions and viewpoints introduce severe challenges. Recently, there have been rapid advances exemplified by many new public benchmark datasets, as can be seen in Appendix B. Similarly, more data from the application-specific setup can be generated in further iterations while using the newest heavy multi-stage offline detectors and tracking for better automation. In parallel, it would be beneficial to adapt the setup to have a mix of long-, mid-, and short-range sensors to minimize the truncation of objects. Furthermore, the dependency on data or sensors can be minimized by cross dataset or sensor learning as well as domain adaptation techniques, as another exciting field for research.

Together with the methods developed in this thesis, future efforts will support the objective of shaping the future of mobility worldwide with the safest ways of moving people or goods.

List of Abbreviations

AD	Autonomous Driving
ADAS	Advanced Driving Assistance Systems
AP	Average Precision
CNN	Convolutional Neural Network
DNN	Deep Neural Network
E-RPN	Euler Region Proposal Network
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
IOU	Intersection Over Union
JPDAF	Joint Probabilistic Data Association Filter
JSON	Java Script Object Notation
LMB RFS	Labeled Multi-Bernoulli Random Finite Set
LSTM	Long Short Term Memory
MHT	Multiple Hypothesis Tracking
MLP	Multi Layer Perceptron

MOT	Multi Object Tracking
MTT	Multi Target Tracking
NMS	Non Maximum Suppression
NN	Neural Network
ReLu	Rectified Linear Unit
RFS	Random Finite Set
RNN	Recurrent Neural Network
RPN	Region Proposal Network
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
UKF	Unscented Kalman Filter
VFE	Voxel Feature Encoding
YOLO	You Only Look Once

List of Figures

List of Tables

Appendix A

Light Detection and Ranging - Lidar

The following sections provide information about the basics of lidar imaging and the sensors that generate the point clouds used throughout this work.

A.1 Basics of Lidar Imaging

Lidar sensors are based on the principle of time of flight, where the distance is estimated by actively emitting light and the measuring of the time delays for the response. This principle is visualized in Figure A.1. The optical signal is reflected from a target, traced back, and detected in order to measure the delay of the light waves traveling through the atmosphere. In the simplest case, a short light pulse and the time of arrival are used, while more complex methods based on the modulation of the amplitude and frequency are also available. Based on such pointwise estimation, lidar sensors are constructed to cover a large field of view. Therefore, in most cases, scanning systems are used with a so-called beam steering component. Here, some sort of optics together with rotating components are often used in order to jointly rotate the unit with light source and receptors. Furthermore, multiple light sources and receptors are used in parallel along the spinning axis to obtain measurements for different angular directions simultaneously. A more detailed overview can be found in [Royo and Ballesta-Garcia 2019]. Furthermore, [Lambert et al. 2020] gives a thorough assessment of several lidar sensors related to AD.

Given the characteristics of such systems, certain properties of the generated point clouds emerge:

- The resolution in point clouds depends on single components such as the number

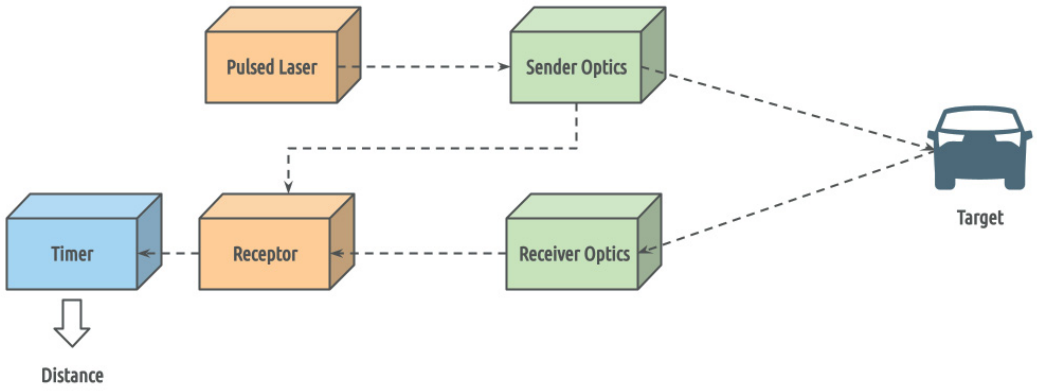


Figure A.1: *The principle of pulsed time of flight distance measurement.*

of light sources and detectors, their structure as well as the beam steering module, and decreases quadratically with distance.

- Point Clouds have a strongly varying density that is heavily influenced by spatial occlusion since they are measured by an optical process (see Figure A.2).
- The field of view is determined by the structure of light sources and detectors (vertical, see Figure A.2) and the beam steering module (horizontal) as well as the optics, respectively.
- The intensity of the reflected pulse varies according to the material (reflectivity) of the target. Therefore, extreme cases like highly reflective materials will introduce some noise in the measurement.

A.2 Velodyne HDL-64E

As one of the first high-definition scanning lidar sensors, the HDL-64E is based on a rotating head component, including 64 laser emitters mounted to specific vertical angles. The head spins at rates from $5Hz$ up to $20Hz$, generating approximately $1.3kk$ points per second of the full surrounding environment. Specifications can be found in Table A.1.

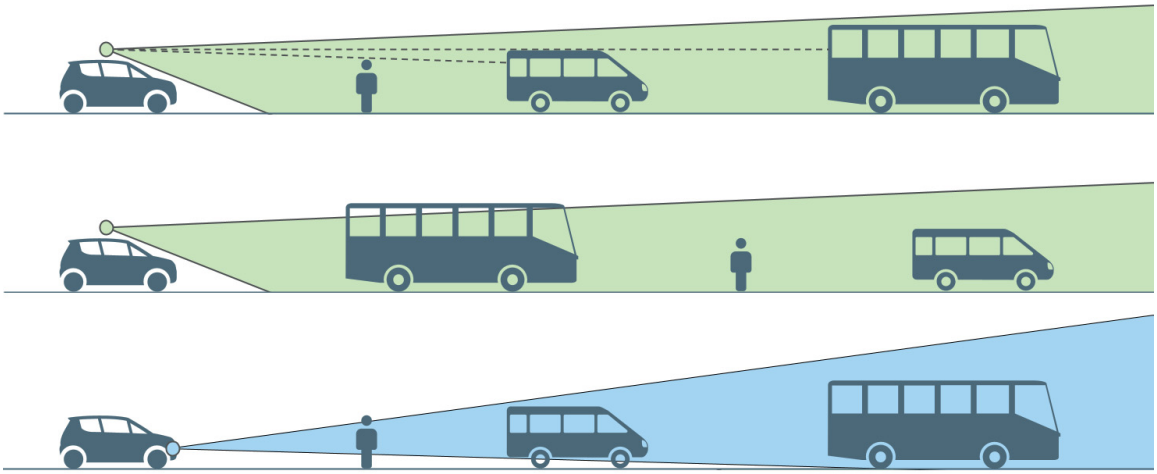


Figure A.2: *A schematic for the field of view of typical lidar setups for AD: Nearby objects will generate shadowed characteristics in the resulting point clouds, respectively. In contrast to the upper rows, the third image represents a lower mounting position related to Scala Gen2. Here, only a section of the person is visible, while the van is occluded by the person and the bus by both objects.*

A.3 Valeo Scala Laser Scanner

Both Scala Laser Scanners are automotive-grade scanning lidar sensors from Valeo. They were designed to be used for ADAS and AD applications mounted low for integration close to production. Based on rotating mirrors inside the housing, a limited area in front of the sensors is scanned. Specifications can be found in Table A.2 and Table A.3.

Category	Specifications	
Laser	Class	1 - eye safe
	Wavelength	905 nm
	Pulse	10 ns
Scan Pattern	Horizontal Field of View	360°
	Vertical Field of View	+2° up to -24.8° down
	Scan Rate	5 - 20 Hz
	Number of Layers	64
	Horizontal Resolution	0.08° - 0.35°
	Vertical Resolution	∼0.4°
	Distance Accuracy	<20 mm
Power	Operating Voltage	12 - 32 V
	Consumption	60 W

Table A.1: *Specifications of the Velodyne HDL-64E.*

Category	Specifications	
Laser	Class	1 - eye safe
	Wavelength	905 nm
	Pulse	<5 ns
Scan Pattern	Horizontal Field of View	145°
	Vertical Field of View	3.2°
	Scan Rate	25 Hz
	Number of Layers	3 per scan, 4 effective per 2 scans
	Horizontal Resolution	0.25°
	Vertical Resolution	0.8°
	Distance Resolution	<100 mm
Power	Operating Voltage	9 - 14 V
	Consumption	<10 W

Table A.2: *Specifications of the Scala Gen1.*

Category	Specifications	
Laser	Class	1 - eye safe
	Wavelength	905 nm
	Pulse	<5 ns
Scan Pattern	Horizontal Field of View	133°
	Vertical Field of View	10°
	Scan Rate	25 Hz
	Number of Layers	16
	Horizontal Resolution	0.125° for +/-15°
		0.25° for +/-15° to 66.5°
	Vertical Resolution	0.6°
	Distance Resolution	<100 mm
Power	Operating Voltage	9 - 14 V
	Consumption	<10 W

Table A.3: *Specifications of the Scala Gen2.*

Appendix B

Public Datasets

This annex provides additional information about public benchmark datasets with a focus on KITTI as used in both 3D object detection and MOT. Further observation of state of the art after the work in this thesis has revealed a growing number and size of point cloud data for AD.

B.1 Overview

In recent years many new datasets have been developed due to deep learning trends. All of the new datasets come with additional Python development kits and APIs. Some concentrate on individual routes, e.g. in the United States, motivated by the scenario of self-driving fleet services. Others cover data from large areas or several cities captured by multiple recording vehicles during a longer period of time. Such datasets are carefully selected and prepared with exact sensor calibrations and high-quality annotations usually done by humans. In Table B.1, the most widely used datasets in the domain of AD are presented. Some of the recent ones are still under development or planned extensions while being partially influenced by General Data Protection Regulation (GDPR). Almost all setups are based on experimental constructions mounted on the roof of vehicles, which is a significant contrast to current passenger cars. Except for PandaSet[1] and A2D2 [Geyer et al. 2020][2], all records are for non commercial academic use only.

KITTI. The pioneering Kitti Vision Benchmark Suite[3] [Geiger et al. 2012] has grown into a large collection of benchmarks since 2012. Therefore, it is the most frequently

[1]PandaSet available at https://scale.com/resources/download/pandaset, accessed: 2021-10-09
[2]A2D2 available at https://www.a2d2.audi/a2d2/en.html, accessed: 2021-10-09
[3]KITTI available at http://www.cvlibs.net/datasets/kitti/, accessed: 2021-10-09

Name	Year	Sensors	Sample Size	Framerate
Kitti	2012	Velodyne HDL-64E	$\varnothing 120k$ points	10 Hz
		2× Stereo Cam. 1.4MP	1382×512 pix(raw)	10 Hz
		GPS/IMU		N/A
Cityscapes	2016	Stereo Camera 2MP	2048×1024 pix	17 Hz
		Vehicle Odometry		N/A
		GPS		N/A
ApolloScape	2018	2× Riegl VUX-1HA	$\varnothing 100k$ points	10 Hz
		2× Camera	3130×960 pix	30 Hz
		GNSS/IMU		N/A
BDD100K	2018	Camera > 0.9MP	$\geq 1280 \times 720$ pix	30 Hz
		GPS/IMU		N/A
nuScenes	2019	Velodyne HDL-32E	$\varnothing 35k$ points	20 Hz
		6× Camera 1.4MP	1600×900 pix	12 Hz
		5× Long Range Radar	$\varnothing 625$ points	13 Hz
		GNSS/IMU		N/A
Argoverse	2019	2× Velodyne Ultra Puck	$\varnothing 107k$ points	10 Hz
		7× Camera	1920×1200 pix	30 Hz
		2× Stereo Camera	2056×2464 pix	5 Hz
		GPS		N/A
PandaSet	2019	Hesai Pandar64	$\varnothing 106k$ points	10 Hz
		Hesai PandarGT	$\varnothing 60k$ points	10 Hz
		6× Camera	1920×1080 pix	10 Hz
		GPS/IMU		N/A
Lyft L5	2019	3× Lidar	$\varnothing 216k$ points	10 Hz
		6× Camera	1224×1024 pix	10 Hz
		Camera	2048×864 pix	10 Hz
A2D2	2020	5× Velodyne VLP-16	$\varnothing 47k$ points	10 Hz
		6× Camera 2.3MP	1920×1208 pix	30 Hz
		GNSS/IMU		N/A
		Vehicle CAN		N/A
Waymo O.D.	2020	5× Lidar	$\varnothing 177k$ points	10 Hz
		5× Camera	1920×1280 pix	10 Hz
nuImages	2020	6× Camera 1.4MP	1600×900 pix	2 Hz
		GNSS/IMU		N/A
Ford AV	2020	4× Velodyne HDL-32E	$\varnothing 139k$ points	10 Hz
		6× Camera 1.3MP	1656×860 pix	15 Hz
		Camera 5MP	2464×1726 pix	6 Hz
		GPS/IMU		200 Hz

Name	Tasks	Training Samples
Kitti	Stereo, Optical/Scene Flow	≈ 200
	Depth Completion/Prediction	$93k$
	Odometry Estimation/Localization	11 sequences
	2D/3D/BEV Object Detection	$7,481$
	Multi Object Tracking	21 sequences
	Road/Lane Detection	289
	Semantic Segmentation (pixel wise)	200
Cityscapes	Semantic Segmentation (pixel wise, instance level, panoptic)/3D Object Detection	$5k$ $+20k$ (coarse)
ApolloScape	2D Object Detection	$5k$
	3D Object Detection/Multi Object Tracking	$\approx 6.4k$
	Stereo	$\approx 4.2k$
	Trajectory Prediction/Motion Forecasting	53 seq. @$60s$
	Lane Segmentation	$110k$
	Semantic Segmentation (pixel wise)	$147k$
BDD100K	2D Classification/2D Object Detection/Lane Detection	$70k$
	Semantic Segmentation (pixel wise, instance level)	$7k$
	Multi Object Tracking	$\approx 280k$
nuScenes	3D Object Detection/Multi Object Tracking/ Semantic Segmentation (point wise)/Odometry	$34k$
	Trajectory Prediction/Motion Forecasting	850 seq. @$20s$
Argoverse	3D Multi Object Tracking	$\approx 25k$
	Trajectory Prediction/Motion Forecasting	205,842 seq. @$5s$
PandaSet	3D Object Detection	$8,240$
	Semantic Segmentation (point wise)	$6,080$
Lyft L5	Trajectory Prediction/Motion Forecasting	$134k$ seq. @$25s$
	3D Object Detection	$55k$
A2D2	Semantic Segmentation (pixel wise, point wise)	$41,277$
	3D Object Detection	$12,497$
Waymo O.D.	3D Object Detection/3D Multi Object Tracking	$240k$
	2D Object Detection/2D Multi Object Tracking	$200k$
nuImages	2D Object Detection/Semantic Segmentation (pixel wise masks)	$67k$
Ford AV	Odometry Estimation/Localization/Mapping	N/A

Table B.1: *Overview with key figures of public datasets for AD, ordered by year.*

cited dataset. Included data was captured around the mid-size city of Karlsruhe, covering rural areas and highways. After 2012, there were many extensions for different tasks in computer vision, such as [Geiger et al. 2013; Fritsch et al. 2013; Menze and Geiger 2015; Behley et al. 2019].

Cityscapes. Another large scale dataset for computer vision originally released in 2016 is called Cityscapes[4] [Cordts et al. 2016]. This dataset contains stereo video sequences from 50 different cities in Germany and neighboring countries with high-quality pixel-level annotations and additional weakly annotated data. Originally, Cityscapes was intended for the assessment of computer vision algorithms for major tasks of semantic urban scene understanding. In 2020, Cityscapes was extended by the task of 3D object detection from camera for all types of vehicles [Gählert et al. 2020].

ApolloScape. As part of the Apollo open-source project for AD, the ApolloScape[5] [Huang, Cheng, et al. 2018] dataset was developed in several iterations since 2018. It contains a collection of camera and lidar data with street view scenes from four regions of China with varying weather and daytime conditions, enhanced with diverse annotations. ApolloScape is intended for the development of multi-sensor fusion and multi-task learning, widely used for research and academics.

BDD100K. This dataset[6] [Yu et al. 2020] is a collection of $100k$ high-resolution video sequences recorded by dash cameras from Nexar[7] behind the windshield of public cars in the United States. Additionally, BDD100K contains rough trajectories created from GPS/IMU information recorded by cell phones. The diversity is increased by different weather conditions, day times, and recordings from diverse cities.

nuScenes. A very popular dataset with 360 deg coverage using the entire sensor suite of a real self-driving platform approved for public roads was published named nuScenes[8] [Caesar et al. 2020]. Alongside cameras, it includes lidar and radars plus localization as well as semantic maps to be used as additional priors. nuScenes covers

[4]Cityscapes available at https://www.cityscapes-dataset.com/, accessed: 2021-10-09
[5]ApolloScape available at http://apolloscape.auto/, accessed: 2021-10-09
[6]BDD100K available at https://bdd-data.berkeley.edu/, accessed: 2021-10-09
[7]see https://us.getnexar.com/
[8]nuScenes available at https://www.nuscenes.org/nuscenes, accessed: 2021-10-09

urban driving situations from Boston and Singapore with careful data curation and high-quality annotations. This dataset set a new order of magnitude of available public datasets in 2019/2020 and is well suited for a broad range of research topics related to AD.

nuImages. As part of nuScenes, nuImages[9] complements the dataset with additional camera annotations sampled from the overall database for diversity reasons.

Argoverse. Composed of two datasets, one for temporal tracking of objects in more than 100 sequences, and one with a strong focus on motion forecasting, Argoverse [10] [Chang et al. 2019] includes high definition maps with 290km of geometric and semantic metadata. It was collected by a fleet of AD vehicles in Pittsburgh and Miami spanning different seasons, weather conditions, and times of the day.

PandaSet. PandaSet[11] is constructed on complex driving scenarios in urban environments with more than 100 scenes of 8 seconds each, selected from 2 routes in Silicon Valley in the United States: 1) San Francisco and 2) El Camino Real from Palo Alto to San Mateo. It comes with the cuboid and pointwise semantic segmentation annotations and is available for commercial use (CC BY 4.0 license).

Lyft L5. The first part of the Lyft Level 5[12] [Kesten et al. 2019] dataset was released in 2019 and deals with real-world environmental perception by providing human-labeled cuboids of traffic agents. In 2020, it was extended by the largest collection of motion prediction data released to date [Houston et al. 2020]. The data was collected from another autonomous fleet on Palo Alto routes for more than half a year. This dataset provides high-definition spatial and semantic maps with context about traffic agents and their motion.

A2D2. The Audi Autonomous Driving Dataset[13] [Geyer et al. 2020] was captured in the cities of Gaimersheim, Munich, and Ingolstadt, covering highways, country roads, and inner-city with closed loops suitable for the tasks of relocalization or loop closure. A2D2 combines lidar with camera with non-sequential annotations for 3D

[9] nuImages available at https://www.nuscenes.org/nuimages, accessed: 2021-10-09
[10] Argoverse available at https://www.argoverse.org/, accessed: 2021-10-09
[11] PandaSet available at https://scale.com/resources/download/pandaset, accessed: 2021-10-09
[12] Lyft L5 available at https://self-driving.lyft.com/level5/data/, accessed: 2021-10-09
[13] A2D2 available at https://www.a2d2.audi/a2d2/en.html, accessed: 2021-10-09

bounding boxes restricted to the area visible to the front center camera. Commercial use is permitted by CC BY-ND 4.0 license.

Waymo Open Dataset. As an excerpt of the sensor suite from the Waymo AD platform, one of the largest and most diverse datasets was published named Waymo Open Dataset[14] [Sun et al. 2020]. It contains nearly $2k$ sequences at 20 seconds length with bounding box labels for lidar as well as camera images from San Francisco, Phoenix, and Mountain View, California. In addition, object tracking approaches are supported by unique instance identifiers throughout all annotations. Overall, the setup is also capable of investigating sensor fusion or domain adaptation research.

Ford AV Dataset. Lastly, a multi-agent dataset collected by a fleet of AD vehicles in Detroit and Michigan during 2017 to 2018 was announced in 2020[15] [Agarwal et al. 2020]. It includes driving around airports, freeways, city centers, university campuses, and suburban areas with recordings from multiple cameras and lidar point clouds augmented with 3D map information. In order to support collaborative tasks, this dataset contains data from multiple vehicles driving through the same environment simultaneously.

B.2 Synthetic Datasets

In addition to the numerous datasets from actual recordings, simulated synthetic ones also exist. However, the simulation of specific point cloud generating sensors is still uncommon. A major advantage is the ability to control and scale even for rare corner cases, as well as the possibility to generate highly balanced datasets with sufficient diversity. In contrast, there are often domain shifts between natural and synthetic data, as these are often far from reality. Therefore, modern synthetic datasets are often produced by game engines like Unity [Unity Technologies 2021] or Unreal [Epic Games, Inc. 2021], with advanced physical rendering techniques and sensor models. Furthermore, [Dosovitskiy et al. 2017] developed an open-source simulator supporting flexible specification of sensor suites and environmental conditions.

[14]Waymo Open Dataset is available at https://waymo.com/open/, accessed: 2021-10-09
[15]Ford AV dataset is available at https://avdata.ford.com/, accessed: 2021-10-09

GTA V. Playing for Data [Richter et al. 2016] and Playing for Benchmarks [Richter et al. 2017] called GTA V datasets[16] are large generated image datasets. Both are created from the commercial video game Grand Theft Auto V using the existing highly realistic synthetic world. They include $25k$ and $250k$ high-resolution image samples with annotations for most vision tasks extracted during rendering.

Synscapes. Another photo-realistic synthetic dataset for street scene parsing is called Synscapes[17] [Wrenninge and Unger 2018]. This dataset is produced on a procedural, physical rendering engine with a wide variety of environmental factors and configurations for high diversity and details. Altogether, there are $25k$ generated images at 1440×720 or 2048×1024 pixels, as well as object annotations, dense depth, and pixel-wise semantic labels.

SYNTHIA. Following the other synthetic datasets, SYNTHIA[18] [Ros et al. 2016] is a collection of imagery and dense depth as well as pixel-wise semantic labels. With more than $200k$ realistic high definition images from driving scenarios, a wide diversity was modeled with towns, modern cities, highways, or green areas. Moreover, different weather, lighting conditions, and day-time variations are available.

B.3 KITTI

The KITTI dataset has been released and expanded in several steps since 2012. Raw data was captured taking advantage of the AD platform Annieway. A modified Volkswagen Passat B6 was equipped with recording hardware, two high-resolution color and grayscale video cameras, a Velodyne lidar as well as a GPS localization system, as visualized in Figure B.1. In total, roughly 3 TB of data, i.e. 6 hours of traffic scenarios around Karlsruhe were originally recorded and post-processed to extract representative subsets for mobile robotics and computer vision tasks [Geiger et al. 2013]. According to the spin rate of the lidar sensor, all cameras are triggered at 10 frames per second with shutter time dynamically adjusted. In addition, data from the GPS/IMU system is synchronized based on the closest timestamp. Hence, in the worst case, the resulting time difference is $5ms$.

[16] GTA V datasets are available at https://download.visinf.tu-darmstadt.de/data/from_games/ and https://playing-for-benchmarks.org/, accessed: 2021-10-09

[17] Synscapes is available at https://synscapes.on.liu.se/, accessed: 2021-10-09

[18] SYNTHIA is available at https://synthia-dataset.net/, accessed: 2021-10-09

Figure B.1: *The KITTI Annieway AD platform, reproduced from* ***http://www.cvlibs.net/datasets/kitti/index.php*** *.*

Moreover, 3D object ground truth was generated by a set of human annotators in the form of 3D bounding boxes to objects such as *Cars*, *Vans*, *Trucks*, *Trams*, *Pedestrians* and *Cyclists*. Here, an object is only annotated if it is visible in the reference front camera. Each bounding box contains further information as either visible, partially occluded, fully occluded, or truncated. Thus, there are regions marked as *Dontcare*, e.g. if they are too far away or groups of small objects like *Pedestrians*.

KITTI Object Detection. From all raw recordings, 7,481 samples were chosen for training and 7,518 samples for testing, according to the number of non-occluded objects in the scene as well as the entropy of the object orientation distribution to ensure diversity. Figure B.2 presents a few samples. At the same time, it was ensured that no samples from individual sequences were contained in both splits. Due to the class distribution, only *Cars*, *Pedestrian* and *Cyclists* are evaluated. Furthermore, during evaluation, ground truth objects are filtered by truncation, occlusion, and *Dontcare* labels. Additionally, only detections larger than a certain amount of pixels in the images are considered. Here, 3D predictions must also be projected back into 2D. The object detection benchmark is subdivided into 2D with additional orientation similarity, 3D, and birdview comparison according to the function used to calculate the similarity between ground truth and prediction.

KITTI Multi Object Tracking. In contrast to the KITTI object detection dataset, this dataset contains 21 full sequences for training and 29 sequences for testing. Only the classes *Car* and *Pedestrian* are evaluated, as of the required numbers of labeled objects for comprehensive evaluation. Again, ground truth objects are filtered by truncation, occlusion, and *Dontcare* regions, while predictions are filtered according to their height in 2D. Moreover, the benchmark uses 2D bounding box overlap to compute the evaluation metrics.

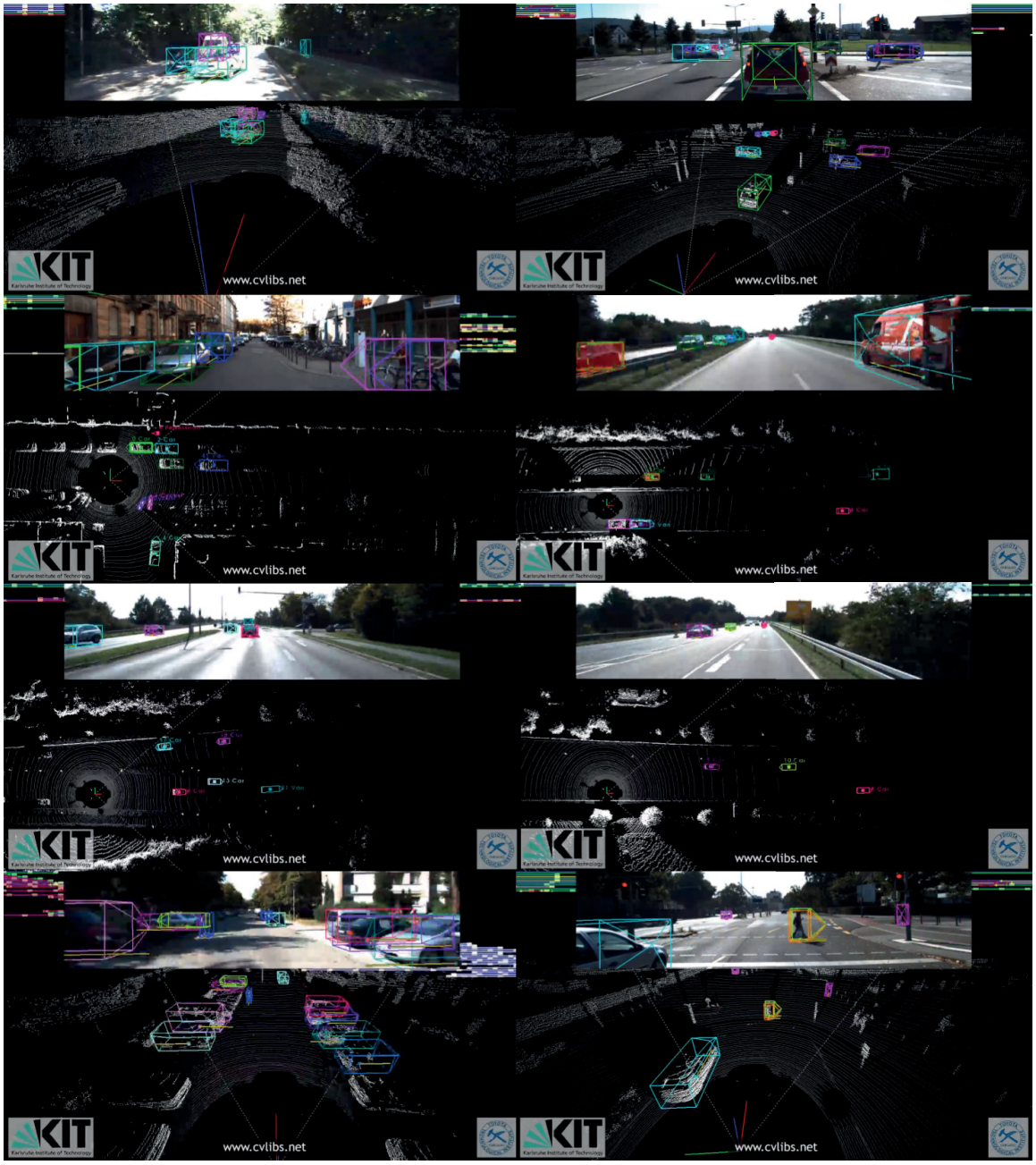


Figure B.2: *Examples from the KITTI object detection dataset, reproduced from* `http://www.cvlibs.net/datasets/kitti/index.php`.

Figure B.3: *Example sequence from the KITTI object tracking dataset, reproduced from* `http://www.cvlibs.net/datasets/kitti/index.php`.

Appendix C

Error Measures

This annex describes all evaluation metrics used in this thesis for both object detection and MOT in more detail.

C.1 Object Detection

The object detection task is judged by precision $P = \frac{TP}{TP+FP}$ over recall $R = \frac{TP}{TP+FN}$ curves with the AP as the principal quantitative measure. Hence, the evaluation follows the PASCAL metrics [Everingham et al. 2010] originally proposed by [Salton and McGill 1983]. Predictions are distinguished true or false based on the IOU with ground truth bounding boxes denoted as:

$$\text{IOU} = \frac{area(B_p \cap B_t)}{area(B_p \cup B_t)} \tag{C.1}$$

where B_p is the predicted bounding box, and B_t is the ground truth target. Predictions are classified as correct as soon as the IOU is above a certain threshold. Duplicate detections from the same object are considered false.

Average Precision (AP). Following [Simonelli et al. 2019], the AP is denoted as:

$$\text{AP}|_R = \frac{1}{|R|} \sum_{r \in R} \rho_{interp}(r) \tag{C.2}$$

where precision values provided by $\rho_{interp}(r)$ are averaged over equally spaced recall levels $R_{40} = \{1/40, 2/40, ..., 1\}$. For each interval, the maximum precision at recall value greater or equal than r is taken according to the interpolation function $\rho_{interp}(r) = \max_{r':r' \geq r} \rho(r')$.

Threshold	Easy	Mod.	Hard
minimum height	40	25	25
maximum truncation	0.15	0.30	0.5
maximum occlusion	0	1	2

Table C.1: *KITTI object detection evaluation parameters.*

KITTI. The protocol for the evaluation on the KITTI dataset first divides all objects according to the classes *Car*, *Pedestrian*, and *Cyclist*. The minimum overlap for detections to be considered as correct is defined as 0.7 for *Cars* and 0.5 for *Pedestrians*, and *Cyclists* respectively. Furthermore, three subcategories, namely easy, moderate and hard, are introduced within the ground truth, based on the number of pixels visible in the image, occlusion, and truncation labels. All definitions can be found in Table C.1. In addition, detections that match *Dontcare* annotations are ignored.

Moreover, the evaluation is split into 2D detection, birdview detection, and 3D detection. Here, the computation of the IOU from Equation C.1 is modified with respect to the complexity, where 2D means bounding box overlap in image space (pixel coordinates) with vertical bounding boxes only, birdview means 2D bounding box overlap with rotated bounding boxes on the ground plane (world coordinates) and 3D considers yaw rotated bounding box overlap in 3D space with 7 degrees of freedom.

C.2 Multi Object Tracking

To measure the performance of the proposed methods for MOT, the common CLEAR MOT [Bernardin and Stiefelhagen 2008; Li et al. 2009] and newer HOTA [Luiten et al. 2020] metrics are used. These metrics are presented below.

Multiple Object Tracking Accuracy (MOTA) is a combination of the three most important measures for describing tracking performance, calculated as:

$$\text{MOTA} = 1 - \frac{\sum_t (FP_t + FN_t + IDs_t)}{\sum_t GT_t} \tag{C.3}$$

where for frame t, GT_t is the number of objects present in the ground truth, FP_t are the false positives as well as FN_t are the false negatives in t and IDs_t is the number of identity switches.

Multiple Object Tracking Precision (MOTP) measures the misalignment between the predicted bounding boxes and the ground truth, defined as:

$$\text{MOTP} = \frac{\sum_{i,t} d_t^i}{\sum_t c_t} \tag{C.4}$$

where d_t^i is the IOU of track i with the assigned ground truth target t and c_t is the number of matches found in frame t. Hence, it heavily relies on the detections without taking any tracking performance such as consistent trajectories into account.

Identity Switches (IDs) referes to the number of times an object is assigned a new identifier in its track.

False Positive (FP) represents the total number of occurrences where an object is detected although no object exists.

False Negative (FN) represents the total number of occurrences where an existing object is not detected.

Mostly Tracked (MT) targets represents the number of ground truth tracks that are assigned the same label for at least 80% of the sequence.

Mostly Lost (ML) targets represents the number of ground truth tracks that are assigned the same label for at most 20% of the sequence.

Fragmentation (Frag). A counter for the number of times an object is lost in a frame but then re-detected in a future frame is called Fragmentation.

Based on these metrics, HOTA was designed to provide a single metric for all three components of tracking (localization, detection, and association). It can be seen as the geometric mean of the detection and association score. Therefore, the concepts of True Positive Association (TPA), False Negative Association (FNA), and False Positive Association (FPA) were defined as:

$$TPA(c) = \{k\}, \quad k \in \{TP \,|\, prID(k) = prID(c) \wedge gtID(k) = gtID(c)\} \tag{C.5}$$

$$\begin{aligned} FNA(c) = \{k\}, \quad & k \in \{TP \,|\, prID(k) \neq prID(c) \wedge gtID(k) = gtID(c)\} \\ & \cup \{FN \,|\, gtID(k) = gtID(c)\} \end{aligned} \tag{C.6}$$

$$\begin{aligned} FPA(c) = \{k\}, \quad & k \in \{TP \,|\, prID(k) = prID(c) \wedge gtID(k) \neq gtID(c)\} \\ & \cup \{FP \,|\, prID(k) = prID(c)\} \end{aligned} \tag{C.7}$$

where c is the set of correctly associated true positives, the set of ground truth detections assigned with different or no identifier if they were missed, and the set of predictions assigned with different or no identifier if they did not correspond to an object, respectively, furthermore, HOTA can be decomposed into sub metrics which allow an individual analysis of the tracking performance. These metrics are defined as follows.

Detection Accuracy (DetA).

$$\mathrm{DetA}_\alpha = \frac{|TP|}{|TP| + |FN| + |FP|} \tag{C.8}$$

Association Accuracy (AssA).

$$\mathrm{AssA}_\alpha = \frac{1}{|TP|} \sum_{c \in \{TP\}} \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)| + |FPA(c)|} \tag{C.9}$$

Higher Order Tracking Accuracy (HOTA).

$$\mathrm{HOTA}_\alpha = \sqrt{\mathrm{DetA}_\alpha \cdot \mathrm{AssA}_\alpha} \tag{C.10}$$

Detection Recall (DetRe).

$$\mathrm{DetRe}_\alpha = \frac{|TP|}{|TP| + |FN|} \tag{C.11}$$

Detection Precision (DetPr).

$$\mathrm{DetPr}_\alpha = \frac{|TP|}{|TP| + |FP|} \tag{C.12}$$

Association Recall (AssRe).

$$\mathrm{AssRe}_\alpha = \frac{1}{|TP|} \sum_{c \in \{TP\}} \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)|} \tag{C.13}$$

Association Precision (AssPr).

$$\mathrm{AssPr}_\alpha = \frac{1}{|TP|} \sum_{c \in \{TP\}} \frac{|TPA(c)|}{|TPA(c)| + |FPA(c)|} \tag{C.14}$$

Localization Accuracy (LocA).

$$\text{LocA}_\alpha = \frac{1}{|TP|} \sum_{c \in \{TP\}} S(c) \tag{C.15}$$

where $S(c)$ is the spatial similarity score function used to generate the set of true positives TP, such as IOU.

These scores are evaluated over a range of different IOU thresholds α and the major score is approximated by:

$$\text{HOTA} = \frac{1}{19} \sum_{\alpha \in \{0.05, 0.1, \ldots, 0.95\}} \text{HOTA}_\alpha \tag{C.16}$$

KITTI. Similar to the object detection metrics, objects considered for evaluation are filtered based on predefined parameters. Here, the maximum truncation is set to zero, the minimum height is set to 25 pixels, and the maximum occlusion is equal to 2. For comparison of detections and ground truth, 2D bounding box overlap in image space (pixel coordinates) is used with a minimum overlap for the CLEAR MOT metrics set to 0.5.

Appendix D

Comparative Measures for Bounding Boxes

This annex very briefly reviews the commonly used Intersection Over Union (IOU) for bounding box comparison and describes the properties of the proposed Scale Rotation Translation Score S_{srt} in more detail.

D.1 Intersection Over Union

The IOU, also known as Jaccard similarity coefficient or Jaccard index, is a statistic to measure the similarity and diversity of finite sample sets. This also includes objects that are often referred to as bounding boxes. In general, the IOU can be defined as:

$$IOU(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{D.1}$$

where the similarity of two bounding boxes (or sample sets) A and B is defined as the size of the intersection divided by the size of the union. Therefore, results are by design $0 \leq IOU(A, B) \leq 1$. The IOU is widely used for the comparison of 2D bounding boxes in computer vision, among other fields. In contrast, the increased number of 7 to 9 degrees of freedom for 3D bounding boxes leads to significantly more complex calculations because there are many possibilities for the different poses of the objects in relation to each other. Another downside is the disregard for rotations with multiples of π. This might cause ambiguities that can have a negative impact on the training of Deep Neural Network (DNN). Nevertheless, the IOU is often the method of choice for 3D detectors as well.

D.2 Scale Rotation Translation Score

In order to justify the properties of the developed scoring together with its underlying parameters, Figure D.1 presents an exhaustive assessment in comparison to Intersection Over Union (IOU). First, a set of $n = 1k$ sample object instances are created with linear interpolation, given a realistic lower and upper bound for each single parameter (x, y, z, length, width, height, yaw). Thus, objects centers are limited to $x \in [-2, 2], y \in [-2, 2], z \in [-1, 1]$ with step size:

$$\frac{v_{max} - v_{min}}{n}, v \in \{x, y, z, length, width, height, yaw\} \tag{D.2}$$

and so on. Then, n^2 scores are calculated for all samples compared with each other. This is repeated twice, once for S_{srt} and once for IOU, and plotted as: a) and d) 3D surface with x and y sample indices and z corresponds to the score; b) and e) filled contours from the scores over the sample indices; c) and f) histogram of the number of times a score occurs. Furthermore, to show the effects of differences in individual parameters and the interaction with combined ones, except i), plots g) to o) compare samples in which all variations of two elements are generated, with one fixed target object as well as all other parameters of the generated samples fixed. The changing elements are plotted on the axes x and y. In i), only the yaw entries of the samples have different values, whereas the yaw of the target object is set to 0.0 to center the plot. As can be seen in a) to f), the peak reaches a value of 1.0 along the diagonal, i.e. whenever equal samples are compared. Also, the score decreases perpendicularly to the diagonal but less steeply. In general, the selected S_{srt} setup is more liberal, see a) and d), or b) and e), but respects the heading of the orientation, see i) and l). This results in almost equal distribution, instead of an imbalance towards zero, as indicated by c) and f), respectively. The peak in f) around 0.1 comes from the outer regions in d), where the object samples do not overlap while having similar sizes and orientations. It can be ignored as long as all subscores S_s, S_r and S_t have similar weights α, β, γ, because only predictions with the best scores are used as $\mathbb{1}_{ij}^{obj}$ during training. An optional penalty can be used to approximate IOU, as presented in [Simon et al. 2019]. While plots g) and h), or j) and k) are very similar for both scores, a major difference can be seen in terms of orientation in i), l) and o) as well as m) and n). This yields a considerable deviation of the scores since IOU cannot distinguish rotations by π. Additionally, IOU equals one at $\pm\frac{\pi}{2}$ because all samples have a squared shape. As a result, a kind of periodic parabolic curve can be seen

in i). In contrast, S_{srt} assigns 1.0 if the samples are equal with linearly decreasing scores the more the yaw angles deviate from each other.

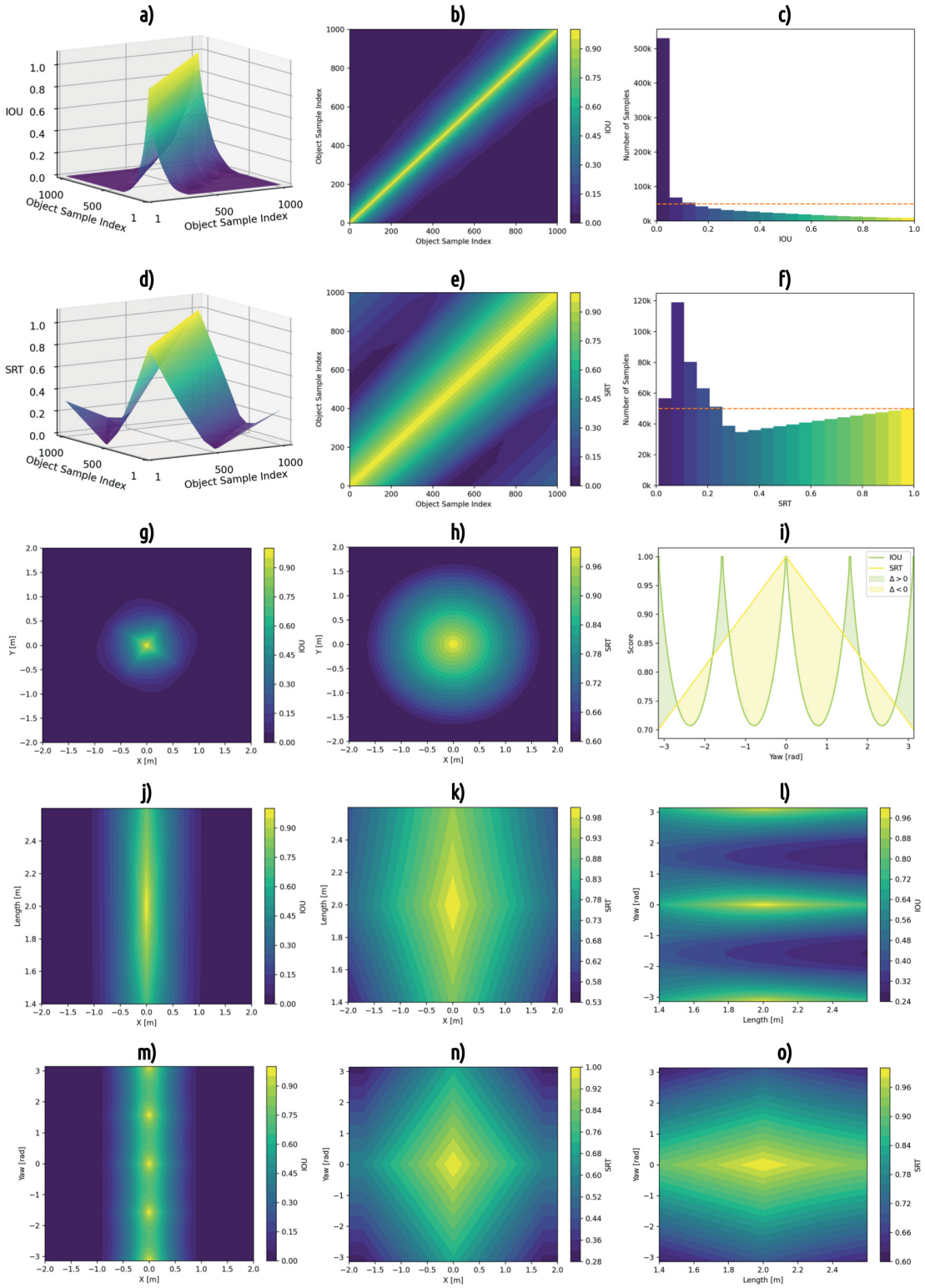


Figure D.1: *Quantitative analysis of S_{srt} compared to IOU using generated objects with realistic linearly interpolated object centers, dimensions and yaw orientations.*

Bibliography

Achlioptas, P., O. Diamanti, I. Mitliagkas, and L. Guibas (2018). "Learning representations and generative models for 3d point clouds". In: *International Conference on Machine Learning (ICML)*. Vol. 1, pp. 40–49.

Agarwal, S., A. Vora, G. Pandey, W. Williams, H. Kourous, and J. McBride (2020). "Ford Multi-AV Seasonal Dataset". In: *arXiv preprint 2003.07969*.

Alhaija, H., S. Mustikovela, L. Mescheder, A. Geiger, and C. Rother (2018). "Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes". In: *International Journal of Computer Vision (IJCV)*.

Almahairi, A., S. Rajeshwar, A. Sordoni, P. Bachman, and A. Courville (2018). "Augmented cyclegan: Learning many-to-many mappings from unpaired data". In: *International Conference on Machine Learning*. PMLR, pp. 195–204.

Alotaibi, A. (2020). "Deep Generative Adversarial Networks for Image-to-Image Translation: A Review". In: *Symmetry* 12.10, p. 1705.

Atienza, R. (2019). "A conditional generative adversarial network for rendering point clouds". In: *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, pp. 10–17.

Aurora (2021). *Source*. `https://aurora.tech/press`. Accessed: 2021-07-16.

Baser, E., V. Balasubramanian, P. Bhattacharyya, and K. Czarnecki (2019). "Fantrack: 3d multi-object tracking with feature association network". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, pp. 1426–1433.

Barker, A. L., D. E. Brown, and W. N. Martin (1995). "Bayesian estimation and the Kalman filter". In: *Computers & Mathematics with Applications* 30.10, pp. 55–77.

Behley, J., M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall (2019). "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 9297–9307.

Beltran, J., C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. de la Escalera (2018). "BirdNet: a 3D Object Detection Framework from LiDAR information". In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3517–3523.

Berclaz, J., F. Fleuret, E. Turetken, and P. Fua (2011). "Multiple object tracking using k-shortest paths optimization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.9, pp. 1806–1819.

Beyer, L., A. Hermans, and B. Leibe (2015). "Biternion nets: Continuous head pose regression from discrete training labels". In: *German Conference on Pattern Recognition (GCPR)*, pp. 157–168.

Borji, A. (2019). "Pros and cons of GAN evaluation measures". In: *Computer Vision and Image Understanding* 179, pp. 41–65.

Blackman, S. S. and R. Popoli (1999). *Design and analysis of modern tracking systems*. Artech House Publishers.

Bryant, D. S., B. T. Vo, B. N. Vo, and B. A. Jones (2018). "A Generalized Labeled Multi-Bernoulli Filter with Object Spawning". In: *IEEE Transactions on Signal Processing* 66, pp. 6177–6189.

Bernardin, K. and R. Stiefelhagen (2008). "Evaluating multiple object tracking performance: the clear mot metrics". In: *EURASIP Journal on Image and Video Processing* 2008, pp. 1–10.

Bay, H., T. Tuytelaars, and L. Van Gool (2006). "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer, pp. 404–417.

Bar-Shalom, Y., P. K. Willett, and X. Tian (2011). *Tracking and data fusion: A handbook of Algorithms*. Vol. 11. YBS publishing Storrs, CT, USA:

Caccia, L., H. van Hoof, A. Courville, and J. Pineau (2019). "Deep Generative Modeling of LiDAR Data". In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*.

Caesar, H., V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom (2020). "nuScenes: A multimodal dataset for autonomous driving". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11621–11631.

Cai, Z., Q. Fan, R. S. Feris, and N. Vasconcelos (2016). "A unified multi-scale deep convolutional neural network for fast object detection". In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 354–370.

Carion, N., F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko (2020). "End-to-end object detection with transformers". In: *European Conference on Computer Vision*. Springer, pp. 213–229.

Chang, M. F., J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays (2019). "Argoverse: 3D tracking and forecasting with rich maps". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8740–8749.

Chaabane, M., P. Zhang, R. Beveridge, and S. O'Hara (2021). "DEFT: Detection Embeddings for Tracking". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Chen, X., K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun (2015). "3d object proposals for accurate object class detection". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 424–432.

Chen, X., K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun (2016). "Monocular 3d object detection for autonomous driving". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2147–2156.

Chen, X., H. Ma, J. Wan, B. Li, and T. Xia (2017). "Multi-view 3D object detection network for autonomous driving". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1907–1915.

Chen, Y., W. Li, C. Sakaridis, D. Dai, and L. Van Gool (2018). "Domain Adaptive Faster R-CNN for Object Detection in the Wild". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3339–3348.

Chen, Q., Y. Wang, T. Yang, X. Zhang, J. Cheng, and J. Sun (2021). "You only look one-level feature". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13039–13048.

Chiu, H.-k., J. Li, R. Ambruş, and J. Bohg (2021). "Probabilistic 3D multi-modal, multi-object tracking for autonomous driving". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 14227–14233.

Choi, Y., M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo (2018). "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8789–8797.

Chung, J., C. Gulcehre, K. Cho, and Y. Bengio (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint 1412.3555*.

Cortinhal, T., F. Kurnaz, and E. E. Aksoy (2021). "Semantics-aware multi-modal domain translation: From LiDAR point clouds to panoramic color images". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3032–3048.

Choi, J., T. Kim, and C. Kim (2019). "Self-ensembling with gan-based data augmentation for domain adaptation in semantic segmentation". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 6830–6840.

Cordts, M., M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele (2016). "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3213–3223.

Cruise (2021). *Source*. `https://www.getcruise.com/news`. Accessed: 2021-07-16.

Cai, Z. and N. Vasconcelos (2018). "Cascade r-cnn: Delving into high quality object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6154–6162.

— (2019). "Cascade r-cnn: High quality object detection and instance segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). "Imagenet: A large-scale hierarchical image database". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 248–255.

Destatis (2021). *Federal Statistical Office - Verkehrsunfaelle*. `https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Verkehrsunfaelle/_inhalt.html`. Accessed: 2021-07-15.

Doucet, A., S. Godsill, and C. Andrieu (2000). "On sequential Monte Carlo sampling methods for Bayesian filtering". In: *Statistics and computing* 10.3, pp. 197–208.

Dong, Z., G. Li, Y. Liao, F. Wang, P. Ren, and C. Qian (2020). "Centripetalnet: Pursuing high-quality keypoint pairs for object detection". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10519–10528.

Dosovitskiy, A., G. Ros, F. Codevilla, A. López, and V. Koltun (2017). "CARLA: An open urban driving simulator". In: *Conference on Robot Learning (CoRL)*.

dSPACE GmbH (2021). *AutoBox*. `https://www.dspace.com/en/inc/home/products/hw/accessories/autobox.cfm`. Accessed: 2021-09-09.

Dalal, N. and B. Triggs (2005). "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee, pp. 886–893.

Duan, K., S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian (2019). "Centernet: Keypoint triplets for object detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6569–6578.

Engelcke, M., D. Rao, D. Z. Wang, C. H. Tong, and I. Posner (2017). "Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1355–1361.

Epic Games, Inc. (2021). *Unreal Engine*. `https://www.unrealengine.com/en-US/`. Accessed: 2021-09-09.

Everingham, M., L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman (2010). "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2, pp. 303–338.

Fischer, K., M. Simon, S. Milz, H.-M. Groß, and P. Maeder (2021). "Stickypillars: Robust and Efficient Feature Matching on Point Clouds using Graph Neural Networks". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 313–323.

Fischer, K., M. Simon, S. Milz, and P. Mäder (2022). "StickyLocalization: Robust End-to-End Relocalization on Point Clouds Using Graph Neural Networks". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 2962–2971.

Fritsch, J., T. Kuhnl, and A. Geiger (2013). "A new performance measure and evaluation benchmark for road detection algorithms". In: *IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 1693–1700.

Frossard, D. and R. Urtasun (2018). "End-to-end Learning of Multi-sensor 3D Tracking by Detection". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 635–642.

Gählert, N., N. Jourdan, M. Cordts, U. Franke, and J. Denzler (2020). "Cityscapes 3D: Dataset and Benchmark for 9 DoF Vehicle Detection". In: *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*.

Gao, S., M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. H. Torr (2019). "Res2net: A new multi-scale backbone architecture". In: *IEEE transactions on pattern analysis and machine intelligence*.

Granström, K., M. Baum, and S. Reuter (2017). "Extended Object Tracking: Introduction, Overview, and Applications". In: *Journal of Advances in Information Fusion* 12.2.

Geiger, A., P. Lenz, C. Stiller, and R. Urtasun (2013). "Vision meets robotics: The KITTI dataset". In: *The International Journal of Robotics Research* 32, pp. 1231–1237.

Geyer, J., Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, and P. Schuberth (2020). "A2D2: Audi Autonomous Driving Dataset". In: arXiv: `2004.06320 [cs.CV]`.

Girshick, R., J. Donahue, T. Darrell, and J. Malik (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587.

Girshick, R. (2015). "Fast R-CNN". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448.

Geiger, A., P. Lenz, and R. Urtasun (2012). "Are we ready for autonomous driving? the KITTI vision benchmark suite". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361.

Gokaslan, A., V. Ramanujan, D. Ritchie, K. I. Kim, and J. Tompkin (2018). "Improving shape deformation in unsupervised image-to-image translation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 649–665.

Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). "Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 2672–2680.

Gonzalez-Garcia, A., J. Van De Weijer, and Y. Bengio (2018). "Image-to-image translation for cross-domain disentanglement". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 1287–1298.

Haiderbhai, M., S. Ledesma, N. Navab, and P. Fallavollita (2020). "Generating X-ray Images from Point Clouds Using Conditional Generative Adversarial Networks". In: *42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, pp. 1588–1591.

He, K., X. Zhang, S. Ren, and J. Sun (2015a). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

— (2015b). “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1026–1034.

— (2016). “Deep Residual Learning for Image Recognition”. In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

He, K., G. Gkioxari, P. Dollar, and R. Girshick (2017). “Mask R-CNN”. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2961–2969.

He, C., H. Zeng, J. Huang, X.-s. Hua, and L. Zhang (2020). “Structure Aware Single-stage 3D Object Detection from Point Cloud”. In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11873–11882.

Huang, K. and Q. Hao (2021). “Joint Multi-Object Detection and Tracking with Camera-LiDAR Fusion for Autonomous Driving”. In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*.

Hoffman, J., E. Tzeng, T. Park, J.-y. Z. Phillip, I. Kate, S. Alexei, and T. Darrell (2018). “CyCADA : Cycle-Consistent Adversarial Domain Adaptation”. In: *International Conference on Machine Learning (ICML)*, pp. 1989–1998.

Houston, J., G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska (2020). *One Thousand and One Hours: Self-driving Motion Prediction Dataset*. `https://level5.lyft.com/dataset/`.

Howard, A. G., M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam (2017). “MobileNets: Efficient convolutional Neural Networks for Mobile Vision Applications”. In: *arXiv preprint 1704.04861*.

Hochreiter, S. and J. Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.

Hu, H.-N., Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krahenbuhl, T. Darrell, and F. Yu (2019). “Joint monocular 3D vehicle detection and tracking”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5390–5399.

Huang, G., Z. Liu, L. Van Der Maaten, and K. Q. Weinberger (2017). “Densely connected convolutional networks”. In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4700–4708.

Huang, X., X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang (2018). “The apolloscape dataset for autonomous driving”. In: *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, pp. 1067–1073.

Huang, X., M.-Y. Liu, S. Belongie, and J. Kautz (2018). "Multimodal unsupervised image-to-image translation". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 172–189.

He, Y., X. Zhang, and J. Sun (2017). "Channel Pruning for Accelerating Very Deep Neural Networks". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1389–1397.

Inoue, N., R. Furuta, T. Yamasaki, and K. Aizawa (2018). "Cross-Domain Weakly-Supervised Object Detection through Progressive Domain Adaptation". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5001–5009.

Ioffe, S. and C. Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR, pp. 448–456.

Isola, P., J.-Y. Zhu, T. Zhou, and A. A. Efros (2017). "Image-to-Image Translation with Conditional Adversarial Networks". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1125–1134.

Julier, S. J. and J. K. Uhlmann (1997). "New extension of the Kalman filter to nonlinear systems". In: *Signal processing, sensor fusion, and target recognition VI*. Vol. 3068. International Society for Optics and Photonics, pp. 182–193.

Kahou, S. E., V. Michalski, R. Memisevic, C. Pal, and P. Vincent (2017). "RATM: recurrent attentive tracking model". In: *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*. IEEE, pp. 1613–1622.

Kalman, R. E. (1960). "A new approach to linear filtering and prediction problems". In: *Journal of Basic Engineering*.

Kingma, D. P. and J. Ba (2015). "Adam: A Method for Stochastic Optimization". In: *International Conference for Learning Representations (ICLR)*.

Kesten, R., M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet (2019). *Lyft Level 5 Perception Dataset 2020*. `https://level5.lyft.com/dataset/`.

Kim, T., M. Cha, H. Kim, J. K. Lee, and J. Kim (2017). "Learning to discover cross-domain relations with generative adversarial networks". In: *International Conference on Machine Learning*. PMLR, pp. 1857–1865.

Kim, S., J. Choi, T. Kim, and C. Kim (2019). "Self-training and adversarial background regularization for unsupervised domain adaptive one-stage object detec-

tion". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 6092–6101.

Krishnan, A. and J. Larsson (2016). "Vehicle detection and road scene segmentation using deep learning". MA thesis.

Kim, A., A. Ošep, and L. Leal-Taix'e (2021). "EagerMOT: 3D Multi-Object Tracking via Sensor Fusion". In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 1097–1105.

Ku, J., M. Mozifian, J. Lee, A. Harakeh, and S. Waslander (2018). "Joint 3D Proposal Generation and Object Detection from View Aggregation". In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8.

Kuang, H., B. Wang, J. An, M. Zhang, and Z. Zhang (2020). "Voxel-FPN: Multi-scale voxel feature aggregation for 3D object detection from LIDAR point clouds". In: *Sensors* 20.3, p. 704.

Kumar, V. R., S. Milz, C. Witt, M. Simon, K. Amende, and J. Petzold (2018). "Near-field Depth Estimation using Monocular Fisheye Camera: A Semi-supervised learning approach using Sparse LIDAR Data". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Deep Vision: Beyond Supervised learning*.

Kumar, V. R., S. Milz, C. Witt, M. Simon, K. Amende, J. Petzold, S. Yogamani, and T. Pech (2018). "Monocular fisheye camera depth estimation using sparse lidar supervision". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 2853–2858.

Kingma, D. P. and M. Welling (2013). "Auto-encoding variational bayes". In: *arXiv preprint 1312.6114*.

Kim, H.-K., K.-Y. Yoo, and H.-Y. Jung (2020). "Color Image Generation from LiDAR Reflection Data by Using Selected Connection UNET". In: *Sensors* 20.12, p. 3387.

Lambert, J., A. Carballo, A. M. Cano, P. Narksri, D. Wong, E. Takeuchi, and K. Takeda (2020). "Performance analysis of 10 models of 3D LiDARs for automated driving". In: *IEEE Access* 8, pp. 131699–131722.

Lang, A. H., S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom (2019). "Pointpillars: Fast encoders for object detection from point clouds". In: *IEEE International*

Conference on Computer Vision and Pattern Recognition (CVPR). Vol. 2019-June, pp. 12689–12697.

Liu, M.-Y., T. Breuel, and J. Kautz (2017). "Unsupervised image-to-image translation networks". In: *Advances in neural information processing systems*, pp. 700–708.

Law, H. and J. Deng (2018). "Cornernet: Detecting objects as paired keypoints". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 734–750.

Lee, B., E. Erdenee, S. Jin, and P. K. Rhee (2016). "Multi-Class Multi-Object Tracking using Changing Point Detection". In: *European Conference on Computer Vision (ECCV)*, pp. 68–83.

Lee, H.-Y., H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang (2018). "Diverse image-to-image translation via disentangled representations". In: *European Conference on Computer Vision (ECCV)*, pp. 35–51.

Level 5 (2021). *Source*. `https://level-5.global/`. Accessed: 2021-07-16.

Lenz, P., A. Geiger, and R. Urtasun (2015). "FollowMe: Efficient Online Min-Cost Flow Tracking with Bounded Memory and Computation". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 4364–4372.

Li, Y., C. Huang, and R. Nevatia (2009). "Learning to associate: Hybridboosted multi-target tracker for crowded scene". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 2953–2960.

Li, C.-L., M. Zaheer, Y. Zhang, B. Poczos, and R. Salakhutdinov (2019). "Point cloud gan". In: *IEEE International Conference on Learning Representations Workshops (ICLR)*.

Li, X., W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang (2020). "Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection". In: *Advances in Neural Information Processing Systems* 33, pp. 21002–21012.

Li, B. (2017). "3D Fully Convolutional Network for Vehicle Detection in Point Cloud". In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 1513–1518.

Liang, X., H. Zhang, L. Lin, and E. Xing (2018). "Generative semantic manipulation with mask-contrasting gan". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 558–573.

Liang, M., B. Yang, Y. Chen, R. Hu, and R. Urtasun (2019). "Multi-Task Multi-Sensor Fusion for 3D Object Detection". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7345–7353.

Lin, T.-Y., M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick (2014). "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer, pp. 740–755.

Lin, T.-Y., P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie (2017). "Feature Pyramid Networks for Object Detection". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2117–2125.

Lin, T.-Y., P. Goyal, R. Girshick, K. He, and P. Dollár (2017). "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988.

Liu, W., D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg (2016). "SSD: Single shot multibox detector". In: *European Conference on Computer Vision (ECCV)*, pp. 21–37.

Liu, M.-Y., X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz (2019). "Few-shot unsupervised image-to-image translation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10551–10560.

Liu, Z., X. Zhao, T. Huang, R. Hu, Y. Zhou, and X. Bai (2020). "TANet : Robust 3D Object Detection from Point Clouds with Triple Attention". In: *Conference on Artificial Intelligence (AAAI)*.

Liu, Z., Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo (2021). "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *International Conference on Computer Vision (ICCV)*.

Li, X. R. and V. P. Jilkov (2003). "Survey of maneuvering target tracking. Part I. Dynamic models". In: *IEEE Transactions on aerospace and electronic systems* 39.4, pp. 1333–1364.

Lin, Y.-P. and T.-P. Jung (2017). "Improving EEG-based emotion classification using conditional transfer learning". In: *Frontiers in human neuroscience* 11, p. 334.

Lowe, D. G. (1999). "Object recognition from local scale-invariant features". In: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. Ieee, pp. 1150–1157.

Luiten, J., A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixe, and B. Leibe (2020). "HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking". In: *International Journal of Computer Vision (IJCV)*.

Li, C. and M. Wand (2016). "Precomputed real-time texture synthesis with markovian generative adversarial networks". In: *European Conference on Computer Vision (ECCV)*, pp. 702–716.

Luo, W., B. Yang, and R. Urtasun (2018). "Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3569–3577.

Li, B., T. Zhang, and T. Xia (2016). "Vehicle Detection from 3D Lidar Using Fully Convolutional Network". In: *Robotics: Science and Systems*.

Mahler, R. P. (2007). *Statistical Multisource-Multitarget Information Fusion*. Artech House, Inc.

— (2014). *Advances in statistical multisource-multitarget information fusion*. Artech House.

Menze, M. and A. Geiger (2015). "Object scene flow for autonomous vehicles". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3061–3070.

Maas, A. L., A. Y. Hannun, A. Y. Ng, et al. (2013). "Rectifier nonlinearities improve neural network acoustic models". In: *Proc. icml*. Vol. 30. 1. Citeseer, p. 3.

Milz, S., M. Simon, K. Fischer, M. Pöpperl, and H.-M. Groß (2019). "Points2Pix: 3D Point-Cloud to Image Translation Using Conditional GANs". In: *German Conference on Pattern Recognition (GCPR)*. Vol. 3. 1, pp. 387–400.

Mirza, M. and S. Osindero (2014). "Conditional generative adversarial nets". In: *arXiv preprint 1411.1784*.

Munkres, J. (1957). "Algorithms for the assignment and transportation problems". In: *Journal of the society for industrial and applied mathematics* 5.1, pp. 32–38.

Murthy, J. K., G. S. Krishna, F. Chhaya, and K. M. Krishna (2017). "Reconstructing vehicles from a single image: Shape priors for road scene understanding". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 724–731.

National Instruments Corporation (2021). *NI PXI*. `https://www.ni.com/de-de/shop/pxi.html`. Accessed: 2021-09-09.

Ning, G., Z. Zhang, C. Huang, X. Ren, H. Wang, C. Cai, and Z. He (2017). "Spatially supervised recurrent convolutional neural networks for visual object tracking". In: *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4.

Neubeck, A. and L. Van Gool (2006). "Efficient non-maximum suppression". In: *International Conference on Pattern Recognition (ICPR)*. Vol. 3. IEEE, pp. 850–855.

NVIDIA Corporation (2017). *Jetson TX2*. `https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetson-tx2/`. Accessed: 2021-08-07.

— (2021). *NVIDIA DRIVE AGX Developer Kit*. `https://developer.nvidia.com/drive/drive-agx`. Accessed: 2021-09-09.

Paszke, A., A. Chaurasia, S. Kim, and E. Culurciello (2016). "ENet : A Deep Neural Network Architecture for Real-Time Semantic Segmentation". In: *arXiv preprint 1606.02147*.

Patel, V. M., R. Gopalan, R. Li, and R. Chellappa (2015). "Visual domain adaptation: A survey of recent advances". In: *IEEE signal processing magazine* 32.3, pp. 53–69.

Peters, T. and C. Brenner (2020). "Conditional adversarial networks for multimodal photo-realistic point cloud rendering". In: *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 88.3, pp. 257–269.

Poucin, F., A. Kraus, and M. Simon (2021). "Boosting Instance Segmentation With Synthetic Data: A Study To Overcome the Limits of Real World Data Sets". In: *IEEE International Conference on Computer Vision Workshops (ICCV)*, pp. 945–953.

Pang, S., D. Morris, and H. Radha (2020). "CLOCs : Camera-LiDAR Object Candidates Fusion for 3D Object Detection". In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*.

Pirsiavash, H., D. Ramanan, and C. C. Fowlkes (2011). "Globally-optimal greedy algorithms for tracking a variable number of objects". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 1201–1208.

Pan, S. J. and Q. Yang (2009). "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10, pp. 1345–1359.

Qi, C. R., H. Su, K. Mo, and L. J. Guibas (Dec. 2017). "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660.

Qi, C. R., L. Yi, H. Su, and L. J. Guibas (2017). "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 5099–5108.

Qi, C. R., W. Liu, C. Wu, H. Su, and L. J. Guibas (2018). "Frustum PointNets for 3D Object Detection from RGB-D Data". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 918–927.

Ristic, B., S. Arulampalam, and N. Gordon (2003). *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house.

Regmi, K. and A. Borji (2018). "Cross-View Image Synthesis Using Conditional GANs". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3501–3510.

Royo, S. and M. Ballesta-Garcia (2019). "An overview of lidar imaging systems for autonomous vehicles". In: *Applied sciences* 9.19, p. 4093.

Redmon, J., S. Divvala, R. Girshick, and A. Farhadi (2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788.

Ren, S., K. He, R. Girshick, and J. Sun (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 91–99.

Reuter, S., B.-T. Vo, B.-n. Vo, and K. Dietmayer (2014). "The labeled multi-bernoulli filter". In: *IEEE Transactions on Signal Processing* 62.12, pp. 3246–3260.

Redmon, J. and A. Farhadi (2017). "YOLO9000: Better, Faster, Stronger". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7263–7271.

— (2018). "YOLOv3: An Incremental Improvement". In: *arXiv preprint 1804.02767*.

Ronneberger, O., P. Fischer, and T. Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.

Roth, M., G. Hendeby, and F. Gustafsson (2014). "EKF/UKF maneuvering target tracking using coordinated turn models with polar/Cartesian velocity". In: *IEEE International Conference on Information Fusion (FUSION)*, pp. 1–8.

Richter, S. R., Z. Hayder, and V. Koltun (2017). "Playing for Benchmarks". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2213–2222.

Richter, S. R., V. Vineet, S. Roth, and V. Koltun (2016). "Playing for Data : Ground Truth from Computer Games". In: *European Conference on Computer Vision (ECCV)*, pp. 102–118.

Ros, G., L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez (2016). "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3234–3243.

Reuse, M., M. Simon, and B. Sick (2021). "About the Ambiguity of Data Augmentation for 3D Object Detection in Autonomous Driving". In: *IEEE International Conference on Computer Vision Workshops (ICCV)*, pp. 979–987.

Saito, K., Y. Ushiku, T. Harada, and K. Saenko (2019). "Strong-weak distribution alignment for adaptive object detection". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6956–6965.

Sämann, T., K. Amende, S. Milz, C. Witt, M. Simon, and J. Petzold (2018). "Efficient Semantic Segmentation for Visual Bird ' s-eye View Interpretation". In: *International Conference on Intelligent Autonomous Systems (IAS)*, pp. 679–688.

Sandler, M., A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen (2018). "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520.

Scheidegger, S., J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granstrom (2018). "Mono-Camera 3D Multi-Object Tracking Using Deep Learning Detections and PMBM Filtering". In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 433–440.

Sharma, S., J. A. Ansari, J. K. Murthy, and K. M. Krishna (2018). "Beyond Pixels: Leveraging Geometry and Shape Cues for Online Multi-Object Tracking". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3508–3515.

Shao, H., Y. Wang, Y. Fu, and Z. Yin (2020). "Generative image inpainting via edge structure and color aware fusion". In: *Signal Processing: Image Communication* 87, p. 115929.

Shi, S., C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li (2020). "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10529–10538.

Shi, S., Z. Wang, J. Shi, X. Wang, and H. Li (2020). "From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Simon, M., S. Milz, K. Amende, and H.-M. Groß (2018). "Complex-YOLO: Real-time 3D Object Detection on Point Clouds". In: *European Conference on Computer Vision (ECCV)*, pp. 1–14.

Simon, M., K. Amende, A. Kraus, J. Honer, T. Sämann, H. Kaulbersch, S. Milz, and H.-M. Groß (2019). "Complexer-YOLO: Real-Time 3D Object Detection and Tracking on Semantic Point Clouds". In: *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*.

Simonelli, A., S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder (2019). "Disentangling monocular 3d object detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1991–1999.

Song, S., S. P. Lichtenberg, and J. Xiao (2015). "SUN RGB-D: A RGB-D scene understanding benchmark suite". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 567–576.

Simon, M. and S. Milz (2018). "Echtzeit 3D Objekterkennung mit Punktwolken". In: *34. VDI/VW Gemeinschaftstagung Fahrerassistenzsysteme und automatisiertes Fahren 2018*. VDI Verlag GmbH, pp. 125–136.

Salton, G. and M. J. McGill (1983). *Introduction to modern information retrieval*. mcgraw-hill.

Song, Z., W. Chen, D. Campbell, and H. Li (2020). "Deep Novel View Synthesis from Colored 3D Point Clouds". In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 1–17.

Shi, W. and R. R. Rajkumar (2020). "Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1711–1719.

Sun, S., H. Shi, and Y. Wu (2015). "A survey of multi-source domain adaptation". In: *Information Fusion* 24, pp. 84–92.

Sun, P., H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov (2020). "Scalability in perception for autonomous driving: Waymo open dataset". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2446–2454.

Shi, S., X. Wang, and H. Li (2019). "PointRCNN: 3D object proposal generation and detection from point cloud". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–779.

Simonyan, K. and A. Zisserman (2015). "Very deep convolutional networks for large-scale image recognition". In: *IEEE International Conference on Learning Representations (ICLR)*.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2015). "Going Deeper with Convolutions". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9.

Tang, H., D. Xu, N. Sebe, and Y. Yan (2019). "Attention-guided generative adversarial networks for unsupervised image-to-image translation". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8.

Tibshirani, R. (1996). "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1, pp. 267–288.

Tripathy, S., J. Kannala, and E. Rahtu (2018). "Learning image-to-image translation using paired and unpaired training samples". In: *Asian Conference on Computer Vision*. Springer, pp. 51–66.

Tompson, J., R. Goroshin, A. Jain, Y. LeCun, and C. Bregler (2015). "Efficient object localization using convolutional networks". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 648–656.

Tan, M., R. Pang, and Q. V. Le (2020). "Efficientdet: Scalable and efficient object detection". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781–10790.

Unity Technologies (2021). *Unity*. `https://unity.com/de`. Accessed: 2021-09-09.

Valeo (2021). *Source*. `https://www.valeo.com/en/autonomous-driving-a-major-technological-challenge-for-cities/`. Accessed: 2021-07-16.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 5998–6008.

Viola, P. and M. Jones (2001). "Rapid object detection using a boosted cascade of simple features". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. IEEE, pp. I–I.

Viola, P. and M. J. Jones (2004). "Robust real-time face detection". In: *International journal of computer vision* 57.2, pp. 137–154.

Vo, B.-N., M. Mallick, Y. Bar-Shalom, S. Coraluppi, R. Osborne, R. Mahler, and B.-t. Vo (2015). "Multitarget tracking". In: *Wiley encyclopedia of electrical and electronics engineering* 2015.

Vu, T., H. Jang, T. X. Pham, and C. D. Yoo (2019). "Cascade RPN: Delving into High-Quality Region Proposal Network with Adaptive Convolution". In: *Conference on Neural Information Processing Systems (NeurIPS)*.

Vo, B.-t. and B.-n. Vo (2011). "A Random Finite Set Conjugate Prior and Application to Multi-Target Tracking". In: *International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ICISSNIP)*, pp. 431–436.

Wang, X., M. Yang, S. Zhu, and Y. Lin (2013). "Regionlets for generic object detection". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 17–24.

Wang, T.-C., M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro (2018). "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8798–8807.

Wang, T., X. Zhang, L. Yuan, and J. Feng (2019). "Few-shot adaptive faster r-cnn". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7173–7182.

Wang, S., Y. Sun, C. Liu, and M. Liu (2020). "PointTrackNet: An end-to-end network for 3-D object detection and tracking from point clouds". In: *IEEE Robotics and Automation Letters* 5.2, pp. 3206–3212.

Wang, W., E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao (2021). "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions". In: *IEEE ICCV*.

Waymo (2021). *Source*. `https://waymo.com/press`. Accessed: 2021-07-16.

Wayve (2022). *A New Approach to Self-Driving: AV2.0*. `https://wayve.ai/blog/a-new-approach-to-self-driving-av2-0`. Accessed: 2022-01-28.

Wang, C.-Y., A. Bochkovskiy, and H.-Y. M. Liao (2021). "Scaled-YOLOv4: Scaling Cross Stage Partial Network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13029–13038.

Wilson, G. and D. J. Cook (2018). "A survey of unsupervised deep domain adaptation". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, pp. 1–46.

Wen, W., C. Wu, Y. Wang, Y. Chen, and H. Li (2016). "Learning structured sparsity in deep neural networks". In: *Advances in neural information processing systems* 29, pp. 2074–2082.

Weng, X., J. Wang, D. Held, and K. Kitani (2020). "3d multi-object tracking: A baseline and new evaluation metrics". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 10359–10366.

Weng, X., Y. Wang, Y. Man, and K. M. Kitani (2020). "Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6499–6508.

Wang, X. and A. Gupta (2016). "Generative image modeling using style and structure adversarial networks". In: *European conference on computer vision*. Springer, pp. 318–335.

Wu, Y. and K. He (2018). "Group normalization". In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19.

Williams, J. L. (2015). "Marginal multi-Bernoulli filters: RFS derivation of MHT, JIPDA, and association-based MeMBer". In: *IEEE Transactions on Aerospace and Electronic Systems* 51.3, pp. 1664–1687.

Wu, J., T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman (2016). "Single image 3D interpreter network". In: *European Conference on Computer Vision (ECCV)*, pp. 365–382.

Wu, J., Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum (2017). "MarrNet: 3D shape reconstruction via 2.5D sketches". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 541–551.

Wu, W., K. Cao, C. Li, C. Qian, and C. C. Loy (2019). "Transgaga: Geometry-aware unsupervised image-to-image translation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8012–8021.

Wu, Z., S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip (2020). "A comprehensive survey on graph neural networks". In: *IEEE Transactions on neural networks and learning systems* 32.1, pp. 4–24.

Wu, H., W. Han, C. Wen, X. Li, and C. Wang (2021). "3D Multi-Object Tracking in Point Clouds Based on Prediction Confidence-Guided Data Association". In: *IEEE Transactions on Intelligent Transportation Systems*.

Wu, H., Q. Li, C. Wen, X. Li, X. Fan, and C. Wang (Aug. 2021). "Tracklet Proposal Network for Multi-Object Tracking on Point Clouds". In: *Proceedings of the Thir-*

tieth International Joint Conference on Artificial Intelligence, IJCAI-21. Ed. by Z.-H. Zhou. International Joint Conferences on Artificial Intelligence Organization, pp. 1165–1171.

Wrenninge, M. and J. Unger (2018). "Synscapes: A photorealistic synthetic dataset for street scene parsing". In: *arXiv preprint 1810.08705*.

Wan, E. A. and R. Van Der Merwe (2000). "The unscented Kalman filter for nonlinear estimation". In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee, pp. 153–158.

Weng, X., Y. Yuan, and K. Kitani (2021). "PTP: Parallelized Tracking and Prediction with Graph Neural Networks and Diversity Sampling". In: *IEEE Robotics and Automation Letters* 6.3, pp. 4640–4647.

Xu, D., D. Anguelov, and A. Jain (2018). "PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 244–253.

Xiang, Y., A. Alahi, and S. Savarese (2015). "Learning to track: Online multi-object tracking by decision making". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 4705–4713.

Xie, S., R. Girshick, P. Dollár, Z. Tu, and K. He (2017). "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500.

Yang, Z., Y. Sun, S. Liu, X. Shen, and J. Jia (2019). "STD: Sparse-to-Dense 3D Object Detector for Point Cloud". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1951–1960.

Yang, Z., Y. Sun, S. Liu, and J. Jia (2020). "3DSSD: Point-based 3D Single Stage Object Detector". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11040–11048.

Yi, Z., H. Zhang, P. Tan, and M. Gong (2017). "Dualgan: Unsupervised dual learning for image-to-image translation". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2849–2857.

Yu, F., V. Koltun, and T. Funkhouser (2017). "Dilated residual networks". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 472–480.

Yang, B., W. Luo, and R. Urtasun (2018). "PIXOR: Real-time 3D Object Detection from Point Clouds". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7652–7660.

Yan, Y., Y. Mao, and B. Li (2018). "SECOND: Sparsely Embedded Convolutional Detection". In: *Sensors* 18.10, p. 3337.

Yogamani, S., J. Horgan, G. Sistu, P. Varley, D. O. Dea, M. Uricar, S. Milz, M. Simon, K. Amende, C. Witt, H. Rashed, S. Chennupati, S. Nayak, S. Mansoor, X. Perroton, and P. Perez (2019). "WoodScape: A multi-task, multi-camera fisheye dataset for autonomous driving". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 9308–9318.

Yoo, J. H., Y. Kim, J. S. Kim, and J. W. Choi (2020). "3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-View Spatial Feature Fusion for 3D Object Detection". In: *European Conference on Computer Vision (ECCV)*.

Yu, F., H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell (2020). "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2633–2642.

Ye, M., S. Xu, and T. Cao (2020). "HVNet: Hybrid Voxel Network for LiDAR Based 3D Object Detection". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1631–1640.

Yin, T., X. Zhou, and P. Krahenbuhl (2021). "Center-based 3d object detection and tracking". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11784–11793.

Zhang, W., H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy (2019). "Robust multi-modality multi-object tracking". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2365–2374.

Zhang, X., F. Wan, C. Liu, R. Ji, and Q. Ye (2019). "FreeAnchor: Learning to Match Anchors for Visual Object Detection". In: *Advances in Neural Information Processing Systems* 32, pp. 147–155.

Zhao, S., B. Li, X. Yue, Y. Gu, P. Xu, R. Hu, H. Chai, and K. Keutzer (2019). "Multi-source Domain Adaptation for Semantic Segmentation", Advances in Neural Information Processing Systems". In.

Zhang, H., H. Chang, B. Ma, N. Wang, and X. Chen (2020). "Dynamic R-CNN: Towards High Quality Object Detection via Dynamic Training". In: *European Conference on Computer Vision (ECCV)*.

Zheng, W., W. Tang, S. Chen, L. Jiang, and C.-W. Fu (2021). "CIA-SSD: Confident IoU-Aware Single-Stage Object Detector From Point Cloud". In: *AAAI*.

Zheng, W., W. Tang, L. Jiang, and C.-W. Fu (2021). "SE-SSD: Self-Ensembling Single-Stage Object Detector From Point Cloud". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14494–14503.

Zhu, J. Y., T. Park, P. Isola, and A. A. Efros (2017). "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2223–2232.

Zhu, J. Y., R. Zhang, D. Pathak, T. Darrell, A. Efros, O. Wang, and E. Shechtman (2017). "Toward Multimodal Image-to-Image Translation". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 465–476.

Zhu, X., W. Su, L. Lu, B. Li, X. Wang, and J. Dai (2020). "Deformable DETR: Deformable Transformers for End-to-End Object Detection". In: *International Conference on Learning Representations*.

Zhang, L., Y. Li, and R. Nevatia (2008). "Global data association for multi-object tracking using network flows". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 1–8.

Zoox (2021). *Source*. `https://www.zoox.com/press/`. Accessed: 2021-07-16.

Zeng Wang, D. and I. Posner (2015). "Voting for Voting in Online Point Cloud Object Detection". In: *Robotics: Science and Systems (RSS)*, pp. 10–15607.

Zhang, R., T. Pfister, and J. Li (2019). "Harmonic unpaired image-to-image translation". In: *IEEE International Conference on Learning Representations (ICLR)*.

Zhou, Y. and O. Tuzel (2018). "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4490–4499.

Zhou, H., B. Ummenhofer, and T. Brox (2018). "DeepTAM: Deep Tracking and Mapping". In: *European Conference on Computer Vision (ECCV)*, pp. 822–838.