Technische Universität München

TUM School of Engineering and Design

LiDAR Domain Adaptation for Perception of Autonomous Vehicles

Hans Carl Sebastian Huch, M.Sc.

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften

genehmigten Dissertation.

Die Dissertation wurde am 24.07.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 05.02.2025 angenommen.

The best way to predict the future is to invent it.

— Alan Kay

Acknowledgments

This dissertation was written during my time as a Ph.D. candidate at the Institute of Automotive Technology, Technical University of Munich, Germany, from 2019 to 2023. I would like to express my sincere gratitude to the following people.

First and foremost, to Prof. Dr.-Ing. Markus Lienkamp, my supervisor, for the invaluable guidance and support. You provided me with the freedom to explore my own path while offering valuable feedback when needed, which was crucial in the completion of this work.

To Prof. Dr. Hyunchui Shim, whom I had the pleasure of meeting during the Indy Autonomous Challenge, and who graciously took on the role of the second examiner. Your insights and expertise significantly enriched this dissertation.

To Prof. Dr.-Ing. Johannes Betz, for taking over the chairmanship of the examination. Your commitment and support were crucial in the final stages of my Ph.D. journey.

To the National Research Foundation of Singapore, under its Campus for Research Excellence And Technological Enterprise program, and to the Federal Ministry of Education and Research of Germany (BMBF) within the Wies'n Shuttle project in the MCube cluster. Their financial support made this work possible.

To my colleagues in Singapore and Germany, I appreciate you greatly. Our countless technical discussions and your support not only advanced my research, but also led to enduring friendships that made my time at the institute both productive and enjoyable.

To the entire TUM Autonomous Motorsport team, as being part of this team was an honor. Without the collective effort and support of the entire team, the completion of this thesis would not have been possible.

To my group, AV Perception, for your regular and constructive feedback, especially during our weekly meetings, brightening my daily life and greatly contributing to this research.

To my proofreaders, Philipp Hafemann, Matthias Rowold, and Ilir Tahiraj, for your thorough feedback on the first draft of this thesis.

To the several students I supervised during my time at the institute. Your contributions to the research field through your theses enriched this project and significantly contributed to the success of this dissertation.

To my friends and the flexlab team, thank you for your support and encouragement.

Lastly, I want to express my deepest gratitude to my family, especially my parents, Patricia and Thomas, my sister, Laura, and my girlfriend, Sandra, for always having my back and believing in me throughout the ups and downs of this Ph.D. journey.

Munich, June 2024

Sebastian Huch

Contents

List of Abbreviations

AD	Autonomous Driving
AMCW	Amplitude Modulation Continuous-Wave
AP	Average Precision
API	Application Programming Interface
AV	Autonomous Vehicle
BEV	Bird's-Eye View
CARLA	Car Learning to Act
CD	Chamfer Distance
CNN	Convolutional Neural Network
DARPA	Defense Advanced Research Projects Agency
DL	Deep Learning
FMCW	Frequency Modulation Continuous-Wave
FN	False Negative
FoV	Field of View
FP	False Positive
FPS	Farthest Point Sampling
GAN	Generative Adversarial Network
GNSS	Global Navigation Satellite System
IAC	Indy Autonomous Challenge
IMU	Inertial Measurement Unit
IoU	Intersection over Union
KPI	Key Performance Indicator
LiDAR	Light Detection and Ranging
MEMS	Microelectromechanical Systems Microscanning
OEM	Original Equipment Manufacturer
OPA	Optical Phased Array
RaDAR	Radio Detection and Ranging
RTK	Real-Time Kinematic Positioning
SLAM	Simultaneous Localization And Mapping
t-SNE	t-Distributed Stochastic Neighbor Embedding
TP	True Positive
UMAP	Uniform Manifold Approximation and Projection
USA	United States of America
VAE	Variational Autoencoder
WHO	World Health Organization

Formula Symbols

Formula Symbols	Unit	Description
B_i	-	3D bouding box
c_0	$\mathrm{m\,s^{-1}}$	Speed of light
C	-	Channels
h	-	Index of vertical step
h_b	m	Height of a bounding box
H	-	Number of vertical steps
i	-	Index of point cloud
I_r	-	Normalized LiDAR intensity
j	-	Index of point within a point cloud
l_b	m	Length of a bounding box
n_epoch	-	Number of epochs
n_it	-	Number of iterations per epoch
n_patch	-	Number of patches
N_j	-	Number of points within a point cloud
$\mathbf{O}_{i,k}$	-	Object point cloud k of scene point cloud $\mathbf{X}_i$
p	-	Precision
P	-	Data distribution
r	-	Recall
r_update	-	Update ratio
r_{wh}	m	Radial distance from LiDAR to target point
S	-	Point set
$\mathcal{S}$	-	Source domain
t_of	s	Time of flight
t_{wh}	s	Point-wise time stamp
$\mathcal{T}$	-	Target domain

Formula Symbols	Unit	Description
v_{wh}	$\mathrm{m\,s^{-1}}$	Point-wise velocity
w	-	Index of horizontal step
w_{b}	m	Width of a bounding box
W	-	Number of horizontal steps
x_{c}	m	x coordinate of the center of a bounding box
x_j	m	x coordinate of a point j in a point cloud $\mathbf{X}$
$\mathbf{X}_i$	-	Set of points
y_{c}	m	y coordinate of the center of a bounding box
y_j	m	y coordinate of a point j in a point cloud $\mathbf{X}$
z_{c}	m	z coordinate of the center of a bounding box
z_j	m	z coordinate of a point j in a point cloud $\mathbf{X}$
α	-	Learning rate
δ	-	Downsampling factor
θ	°	Orientation angle of a bounding box
θ_h	°	Elevation angle
λ_{patch}	-	Patch size
ϕ_w	°	Azimuth angle

1 Introduction

1.1 Background and Motivation

Autonomous Vehicles (AVs) aim to revolutionize the transportation sector by reducing emissions and improving road safety [1]. The latter has a high impact on human life, as an estimated 1.19 million deaths from road traffic occurred in 2021 according to the latest road safety report published by World Health Organization (WHO) [2]. The long-term vision of replacing the human driver with a combination of intelligent software and hardware in the vehicle is to achieve accident-free traffic with zero fatal accidents. In addition to the emission and safety aspect of AVs, they promise not only to transform the mobility experience by freeing up previously dedicated time to driving, but also to make mobility accessible to groups of people previously excluded, such as the elderly [3], children [4], or people with disabilities [3].

Research and development in Autonomous Driving (AD) has a long history, mainly led by pioneering works and competitions, such as Prometheus in 1987 [5], the Defense Advanced Research Projects Agency (DARPA) challenges in the mid 2000s [6, 7], the autonomous drive on the iconic Bertha Benz Memorial Route by Mercedes in 2013 [8], and the recent Indy Autonomous Challenge (IAC) in 2021 [9]. This has paved the way for the commercial development of autonomous vehicles, such as the Google self-driving car [10] as a result of the DARPA challenges, which was later established under Alphabet as the independent self-driving technology company Waymo [11]. Several other companies such as Cruise [12], Uber [13], Zoox [14], and most Original Equipment Manufacturers (OEMs) research in the field of AD, as this promises to generate US$300 to US$400 billion in revenue by 2035, according to a recent report by McKinsey [15] from 2023. The economic impact on society is even greater [16], with an estimated impact of yearly US$1.2 trillion in total or US$3,800 per citizen in United States of America (USA) due to productivity gains and reduction of costs from crashes [17]. Despite the large efforts and funding invested in research and development of AVs, technology is far from mature, with several accidents of AVs by Uber in 2018 [18], Tesla in 2020 [19], and Waymo in 2024 [20], even when operating in restricted operational design domains. These recent accidents highlight the need for further development of AD technology, which is especially driven by software used to understand the environment, plan vehicle motion, and control the vehicle.

In general, the software stack for AVs follows the common modular approach, in which the task is divided into several sequential modules. These modules include perception, planning, and control [21]. Perception algorithms process sensor data from cameras, Light Detection and Ranging (LiDAR) sensors, Radio Detection and Ranging (RaDAR) sensors, ultrasonic sensors, Global Navigation Satellite System (GNSS) devices, and Inertial Measurement Units (IMUs) to extract relevant knowledge from the vehicle's environment. This information is used in the planning algorithm to make decisions about the route followed by the vehicle. Lastly, the control algorithms execute the planned trajectories.

The perception task is to perceive the environment using the sensor data and extract information on the AV's surrounding static and dynamic objects, but also to localize the ego vehicle with respect to the global and local world [21]. Specifically, object detection is the task of determining the location and velocity of the surrounding objects.

To this end, Deep Learning (DL) models are used to extract features from raw or preprocessed sensor data. These features are used in later stages of the DL models to estimate the 3D pose (position and orientation) of objects, their dimensions, and their semantics, i.e. the interpretation of the detected object.

The DL models are implemented in the form of neural networks, that are mainly trained in a supervised fashion, which means that they require an annotated training dataset. In the context of automotive perception, these datasets consist of frames of the sensor data, i.e., images or point clouds, and the labels for each frame, including the properties that should be estimated by the neural network during inference. Sensor data for the generation of these datasets are recorded using a vehicle equipped with the respective sensors. Annotation of the data is often a manual process that involves human effort or supervision and hence is time-consuming and costly. The performance of these networks is closely coupled with the quality and quantity of the training dataset [22–25], a phenomenon known as data hunger [26]. This means that many frames need to be annotated to obtain a dataset that can be used for training a safety-critical application, such as neural networks for automotive perception, which directly affect the vehicle's ability to understand the environment.

Furthermore, neural networks perform best when applied to data from the domain in which they were trained. A domain refers to the specific context or environment in which the data is collected and where the neural network operates. For example, AVs are required to operate in different geographical areas and at different times of day and seasons, that is, they encounter a diversity of domains. Training a neural network with data from one domain and using it with data from a different target domain can lead to a decrease in performance. This issue may arise, for instance, when data from simulations are utilized during the training phase, yet the network is deployed in the real world. This gap between the training data and testing data is termed **domain shift** and will be analyzed in-depth in this thesis.

To bridge the gap between domains, the field of **domain adaptation** has emerged. The idea of domain adaptation methods is to fully or partially reduce the domain shift without using annotations from the target domain. If these methods are successful, they enable scalable perception systems by following these steps: First, an annotated source dataset is generated in any domain. Second, this source dataset is adapted using the domain adaptation method to match the characteristics of the desired target domain. Finally, this adapted dataset can be used to train the perception system, e.g. the object detection algorithm, for usage in the target domain.

An exemplary application of domain shift and domain adaptation is the IAC, which serves as a further motivation for the research conducted in this thesis. The IAC represents the world's first head-to-head racing series, employing fully autonomous vehicles in its competitive framework. This competition between international universities is held from 2021 onward and fosters development and research in the field of AVs. The vehicles utilized in the IAC are Dallara AV-21, as depicted in Figure 1.1, which are adapted from the Dallara IL-15. These AV-21 are retrofitted and equipped with the sensors, computing hardware, and actuators required to enable AD. The task of the participating teams is to develop the entire AD software pipeline and compete in head-to-head races on different race tracks throughout the world.

At the IAC, the challenge on the part of perception is not only the development of robust algorithms but also the generation of training data for those algorithms. The perception pipeline of most teams is mainly based on LiDAR object detectors that make use of 3D point clouds [27]. Consequently, this thesis will concentrate on the LiDAR sensor. The generation of annotated real-world point clouds is especially challenging due to several reasons, e.g. because the race cars with the specific sensor setup are not built yet, a race occurs on a not-yet-visited race track, or the annotation process takes too long given the tight schedule of the competition. Hence, the usage of synthetic data generated in simulation environments is a viable alternative to real-world data, but this leads to a sim-to-real domain shift between simulated and real-world data. This raises the question of the source of the sim-to-real domain shift in LiDAR point clouds and how to bridge it using methods from the field of domain adaptation. Both questions should be answered as part of this thesis.

Figure 1.1: Indy Autonomous Challenge [28].

1.2 Dissertation Outline and Contributions

This thesis starts with a brief introduction of the background of LiDAR perception and DL for AD in Chapter 2 to lay the foundations for the rest of the thesis. Subsequently, this thesis reviews the state of the art in the field of domain shift and domain adaptation primarily for the sim-to-real application in Chapter 3. Derived from the state of the art, Chapter 4 formulates the research questions that serve as a guide for the rest of the thesis. Following this, the state of the art is extended with the following contributions:

- Chapter 5 introduces a method for the generation of a scenario- and distribution-aligned simulation and real-world dataset which aims to minimize the differences between the simulation and real-world data. This dataset is open-source, enabling the analysis of the sim-to-real domain shift.
- Furthermore, Chapter 5 proposes a method to quantify and analyze the sim-to-real domain shift, and quantification is carried out using 3D LiDAR object detectors.
- An adversarial domain adaptation approach is proposed to enable the adaptation of 3D point clouds, focused on the sim-to-real application. This approach is customizable to operate with distribution-aligned datasets at the object-level (Chapter 6) and at the scene-level (Chapter 7), i.e. adapting point clouds of single objects or entire scenes. Moreover, the generalization ability of this method is demonstrated by modifying the architecture to work with a non-distribution-aligned dataset at both object- and scene-level in Chapter 8.

Chapter 9 presents a detailed critical discussion of the methods and results and highlights the possibilities for future work in the research field presented. Finally, the thesis is summarized in Chapter 10.

2 Preliminaries in LiDAR Perception and Deep Learning for Autonomous Driving

In AD, perceiving the environment is crucial. Accurate perception is crucial for downstream modules of the AV software stack, as perception errors can have a high impact on prediction, planning, and control, and therefore the safety of the AV and its surroundings [24, 25, 29]. Perception consists of multiple tasks, such as object detection, semantic segmentation, and road and lane detection [29]. The object detection task incorporates the accurate detection of all relevant traffic participants, static or dynamic, such as vehicles, cyclists, and pedestrians. The detection of these traffic participants includes the three-dimensional measurement of the relative position but also the identification of the class, i.e. the semantics of the object. To this end, exteroceptive sensors onboard are required to provide the input data for object detection algorithms. The most commonly used modalities are camera, LiDAR, RaDAR, and ultrasonic sensors [29]. Each modality has advantages and disadvantages over the other modalities and is suitable for a specific task, but for robustness and reliability, high redundancy of sensors is needed [29].

LiDAR sensors capture 3D information around the AV by directly measuring the relative distance of each data point from the LiDAR sensor [30]. LiDAR sensors fulfill the requirements of perception systems by combining long-range, high spatial resolution, and real-time performance [30]. In the following, the basic working principle of the LiDAR sensors and the different scanning systems are explained briefly in Section 2.1 before introducing the different representations of the output of the LiDAR sensor in Section 2.2. The special features of point cloud processing in neural networks are explained in Section 2.3. Since these neural networks need large datasets for training and testing, datasets in the domain of AD are introduced in Section 2.4.

2.1 Working Principle of LiDAR Sensors

In this section, the measurement principle of LiDAR sensors is briefly described. An extensive review on LiDAR sensors, their working principle, and their applications can be found in [31]. LiDAR sensors are active sensors, as they actively emit light pulses, as depicted in Figure 2.1. These light pulses reflect on the surfaces of the targets. Depending on the distance from the target, the backscattered signal can be detected by the receiver of the LiDAR sensor [30]. This measurement principle is similar to the principle of RaDAR sensors, but instead of electromagnetic waves in the millimeter range, LiDAR sensors use waves with a higher frequency and a shorter wavelength. The radial distance r from the target to the sensor can be calculated using the time-of-flight formula of the round trip [30–32]

$$r = \frac{\mathrm{c}_0 t_{\mathrm{of}}}{2}, \tag{2.1}$$

with the speed of light $\mathrm{c}_0 = 3 \times 10^8\,\mathrm{m\,s^{-1}}$ and the measured time of flight t_{of}.

For robotic and automotive applications, the rays of LiDAR sensors are invisible and must adhere to the eye-safety protection Class 1 of IEC 60825-1:2014 [33]. The near-infrared spectrum with a wave length

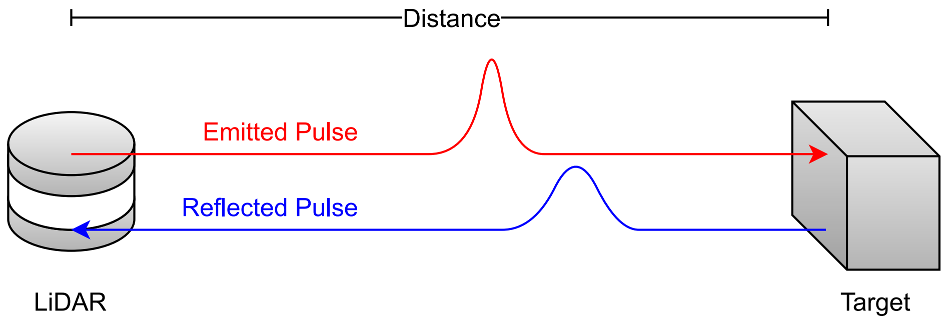


Figure 2.1: Simplified LiDAR sensor measurement principle.

of 900 to $1050\,\mathrm{nm}$ [34] or the short-wave infrared around $1550\,\mathrm{nm}$ is chosen [32], since they provide an effective transmission through the atmosphere. The advantage of short-wave infrared LiDAR sensors is that the maximum output power allowed is higher, and therefore a higher detection range is possible. However, electromagnetic absorption by water is stronger at $1550\,\mathrm{nm}$ [32], making these LiDAR sensors more susceptible to adverse weather conditions.

The beams have low divergence, which allows high measurement distances due to reduced power decay with distance [34]. In general, LiDAR sensors have a high distance measurement accuracy, with an error of a few millimeters on average [34]. Royo et al. [30] report a centimeter accuracy for pulsed or Amplitude Modulation Continuous-Wave (AMCW) LiDAR sensors, and a millimeter accuracy for Frequency Modulation Continuous-Wave (FMCW) LiDAR sensors. This accuracy can be degraded to $0.5\,\mathrm{m}$ in the worst case [34]. The range resolution is proportional to the resolution of the time counting; for example, a discretization of $0.1\,\mathrm{ns}$ equals a distance of $1.5\,\mathrm{cm}$. This discretization is limited by the jitter and noise of the electronics [30].

To guide the laser beam, different scanning systems exist, classified as mechanical spinning or solid-state systems [32]. In **mechanical spinning** LiDAR sensors, a rotating mirror or prism is controlled by a motor for beam steering. This mirror or prism rotates periodically with a fixed spinning rate, which is usually in the range of ten to $30\,\mathrm{Hz}$. They usually have a horizontal Field of View (FoV) of $360\,°$ and multiple laser beams stacked to achieve a vertical resolution [32]. This vertical resolution reaches four up to 128 lasers beams [34]. Due to their system design and wide FoV, these mechanical spinning LiDAR sensors are bulky and fragile due to vibrations of the mechanical components [32]. **Solid-state** LiDAR sensors can be further categorized into the Microelectromechanical Systems Microscanning (MEMS), flash, and Optical Phased Array (OPA) systems. MEMS LiDAR sensors are near-solid-state devices with a mirror on a chip. This mirror rotates 1D or 2D around an axis by the opposing forces from a torsion bar and the electromagnetic force applied during operation. This allows custom scan patterns and the ability to dynamically adjust the FoV [32]. Flash LiDAR systems have a working principle similar to that of a camera. A single laser is spread by an optical diffusor to illuminate the whole scene simultaneously, and a 2D photodiode array receives laser returns. The advantage of flash LiDAR sensors is that the entire scene is captured at once and therefore no motion distortion occurs due to capturing at different time steps. However, the power of a single laser is spread into multiple beams, leading to limited achievable ranges [32]. In OPA LiDAR sensors, the laser beams are steered using a phase modulator. This phase modulator controls the shape of the optical wavefront and hence the steering angles [32].

2.2 Representation of LiDAR Data

Despite differences in the scanning system, each ray returns the radial distance r_{wh} to the target using the time-of-flight formula in Equation 2.1. In addition, the sensors can return the intensity I_r of the reflected light pulses. The horizontal and vertical orientation of each ray can be described by the azimuth angle ϕ_w and the elevation angle θ_h, respectively (Figure 2.2). The indices $1 \leq w \leq W$ and $1 \leq h \leq H$ denote discrete horizontal and vertical orientations. In mechanical spinning LiDAR sensors, the number of horizontal steps W depends on the spinning rate, and the number of vertical steps H depends on the number of vertically stacked lasers.

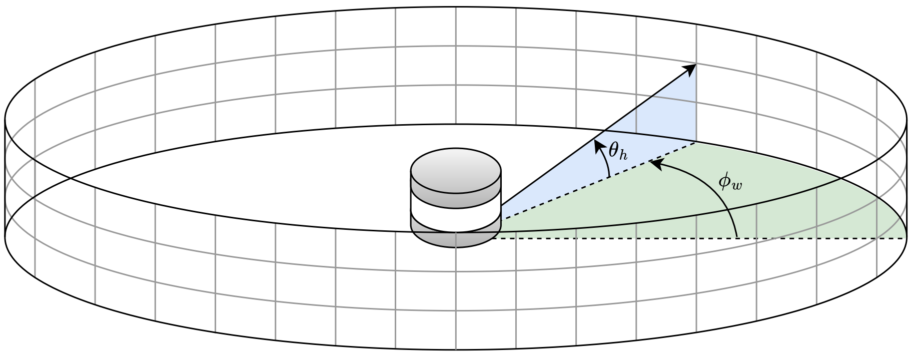


Figure 2.2: Representation of the generation of a sensor image using a mechanical spinning LiDAR sensor.

The resulting output of the LiDAR sensor is the **sensor image** (also known as depth image, range image, range view, or front view), as depicted in Figure 2.3c for a $360°$ sweep and the same sensor image enlarged to the central $120°$ in Figure 2.3d. The sensor image is defined in $\mathbb{R}^{W \times H \times C}$, whereas the channels C comprise the radial distance from the target r_{wh} and the reflected intensity I_r. C can further contain additional information, such as the point-wise velocity v_{wh} or the point-wise time stamp t_{wh} at which each laser pulse hits the target. The point-wise time stamp can be used to compensate for motion distortion, which occurs if the pose (position or orientation) of the LiDAR sensor changes during the finite time needed to capture a scan. This time is measured between capturing the first and last point of a point cloud. Motion distortion is a typical characteristic of point clouds in the automotive domain, as the LiDAR sensor is mounted on a vehicle, which constantly changes its pose. Using the change in pose measured with GNSS and IMU and the point-wise time stamps, the relative position of each point within the point cloud can be corrected.

The spherical sensor image can be transformed into a **3D point cloud** (Figure 2.3a). Formally, a point cloud is an unordered set of points $\mathbf{X}_i = \{p_j|_{j=1}^{N_j}\}$. Each i-th point cloud contains N_j points, and each point $p_j = \{x_j, y_j, z_j\} \in \mathbb{R}^3$ contains the 3D coordinates. The transformation between the spherical sensor image and the 3D point cloud using Cartesian coordinates xyz can be described by the mapping in Equation 2.2:

$$\begin{pmatrix} x_{wh} \\ y_{wh} \\ z_{wh} \end{pmatrix} = r_{wh} \cdot \begin{pmatrix} \cos\phi_w \cdot \sin\theta_h \\ \sin\phi_w \cdot \sin\theta_h \\ \cos\theta_h \end{pmatrix}. \tag{2.2}$$

The definition of the point cloud coordinate system is according to the vehicle axis system of SAE J670 [35]. In this thesis, a consistent definition of the coordinate system is used for all 3D point clouds. Specifically, the x axis is aligned with the longitudinal axis of the vehicle on which the LiDAR sensor is mounted and is positive toward the vehicle front, which is usually the direction of driving. The y axis is orthogonally facing the front of the vehicle and is positive towards the left side of the vehicle. Finally, the z axis is the vertical axis and is positive upward.

In this thesis, 3D point clouds are used in the form of *object point clouds* or *scene point clouds*. The latter is an entire point cloud recorded during a single scan of the automotive LiDAR sensor, as in Figure 2.3a. An object point cloud is a cropped section of a scene point cloud and only comprises points belonging to a single object. These object point clouds can be extracted from scene point clouds using the 3D bounding box of the object and usually only contain a fraction of the points of the corresponding scene point cloud.

This thesis examines object and scene point clouds at both the local and global levels. The global level describes the entire *global structure* of a point cloud, i.e. the semantics of a scene point cloud or the object itself in an object point cloud. In contrast, the local level describes the detailed *local structures* and inter-point relationships of neighboring points within an object or scene point cloud, e.g. specific noise patterns on different surfaces or caused by different sensors.

The 3D point cloud can also be converted into a **2D Bird's-Eye View (BEV) image** by projecting the z component of each point onto a common plane, i.e. $z = 0$, and converting the continuous x-y coordinates to discrete pixel positions on the BEV image (Figure 2.3b).

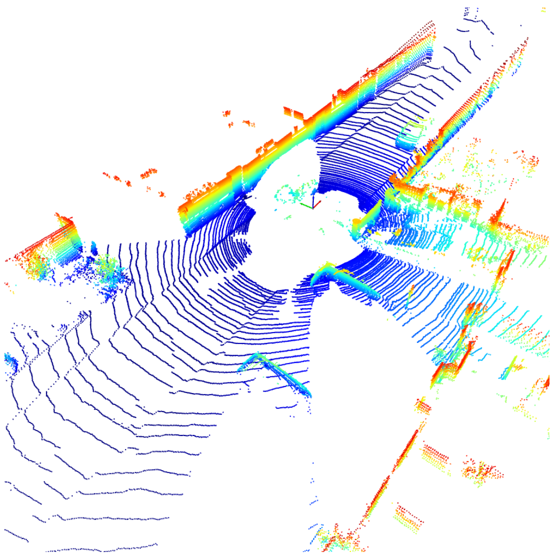
Another representation of LiDAR data is the transformation of 3D point clouds into a **voxel grid**. To create a voxel grid from a 3D point cloud, the point cloud is discretized using rectangular cuboids with a fixed edge length. The content of each cuboid can be multidimensional, from binary classification if the cuboid contains any point to multiple features such as the average coordinate of the points, the total number of points within the cuboid, or the average intensity of the points. The advantage of using the voxel grid representation is that the LiDAR data are stored in a structured and regular grid, allowing the use of algorithms and DL techniques developed for the camera modality, as the camera images are also stored in a structured and regular grid.

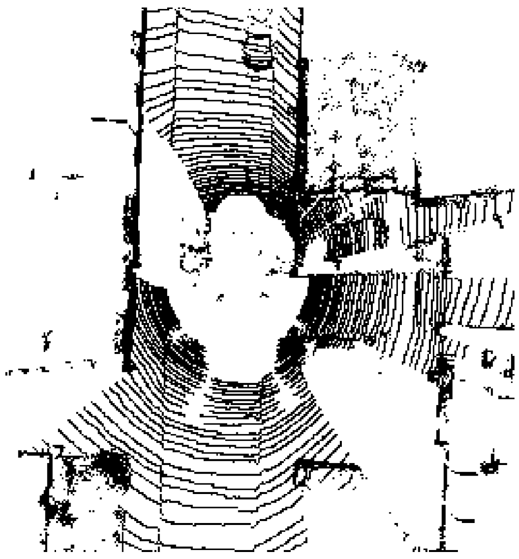
Projection onto a 2D BEV or voxel grid leads to discretization and therefore loss of information compared to the 3D point cloud representation, which preserves the rich geometric, shape, and scale information [38]. Therefore, the 3D point cloud representation is the preferred representation of the LiDAR data for many applications such as AD and robotics, which require intensive scene understanding [38].

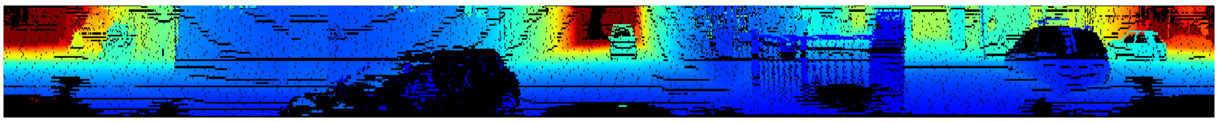
2.3 Fundamentals of Deep Learning with Point Clouds

DL techniques not only lead research and industry in many areas, such as computer vision and natural language processing but also found their way into everyday life in the form of applications like ChatGPT [39]. DL techniques are also used to perform a variety of tasks to understand 3D point clouds, which become increasingly available and affordable due to the rapid development of 3D acquisition technologies such as LiDAR sensors and RGB-D cameras [38]. According to Guo et al. [38], in the field of AD and robotics, research on 3D point cloud understanding includes tasks such as 3D shape classification, 3D object detection and tracking, 3D point cloud segmentation, 3D point cloud registration [40], 6-degree-of-freedom pose estimation [41], and 3D reconstruction [42].

3D shape classification methods have the objective of determining the semantic class of a point cloud, which is usually restricted to object point clouds. **3D object detection and tracking** methods aim to localize objects in scene point clouds and further classify and track the localized objects in multiple sequential point clouds. **3D point cloud segmentation** methods determine the semantic class of each point and can be categorized into semantic segmentation and instance segmentation. Both methods divide the point cloud into regions based on the category of objects, but the latter distinguishes further between instances of the same semantic class. According to Elbaz et al. [40], **3D point cloud registration** is described as the process of determining the transformation between two sets of point clouds that have distinct coordinate systems. This technique is the key enabler for applications such as Simultaneous Localization And Mapping (SLAM). **6-degree-of-freedom pose estimation** algorithms predict the 6D pose, i.e., position and orientation, of


(a) 3D point cloud. Points with the same color have the same height in the coordinate system.


(b) 2D bird's-eye view with $0.1\,\mathrm{m}$ resolution.


(c) Sensor image with distance encoding in the range $0\,\mathrm{m}$ to $30\,\mathrm{m}$ for the full $360°$ point cloud. The color encodes the distance of each measurement from $0\,\mathrm{m}$ (blue) to $30\,\mathrm{m}$ (red). The black pixels indicates that there are no measurements in this range, i.e. the LiDAR sensor receiver detects no reflected pulse or the intensity is below a defined threshold.

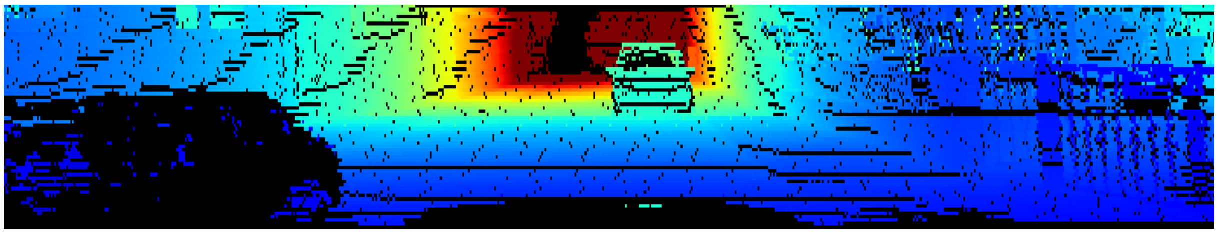

(d) Sensor image as in Figure 2.3c, enlarged to the central $120°$.

Figure 2.3: Different LiDAR representations of the same point cloud from the KITTI [36, 37] dataset.

objects in the 3D space, which is a crucial skill for applications such as industrial robotics and warehouse automation technologies [41]. Finally, the goal of **3D reconstruction** is to rebuild the three-dimensional geometry and structure of objects and scenes [42]. This process is crucial for various applications such as robot navigation, object recognition, scene understanding, 3D modeling and animation, industrial control, and medical diagnosis [42], and involves using 2D images or 3D point clouds as input data.

Despite the large variety of tasks in the field of 3D point cloud understanding, all face the same challenges of extracting meaningful information from the 3D point clouds. The principles of extracting features from 3D point clouds using neural networks are briefly discussed in Subsection 2.3.1. Using these basics, the specific task of object detection using LiDAR point cloud data is explained in Subsection 2.3.2.

2.3.1 Principles of Neural Networks for Feature Extraction from Point Clouds

Feature extraction is the process of transforming raw data into a set of meaningful and relevant features that can be utilized by subsequent layers of the neural network for tasks such as object detection. 3D point clouds have three main properties causing the application of feature extractors in neural networks to be challenging [43]. First, point clouds are **irregular**, meaning that points are not evenly sampled around the sensor; some regions have dense points, while others have sparse points. Second, the **unstructured** nature of point clouds means that the points are not on a fixed grid, the distance between neighboring points varies, compared to equidistant pixels in 2D images. Lastly, point clouds are **unordered**, meaning that the order of the points in a list does not influence the 3D point cloud, i.e., point clouds are permutation invariant. Figure 2.4 visualizes these three properties.

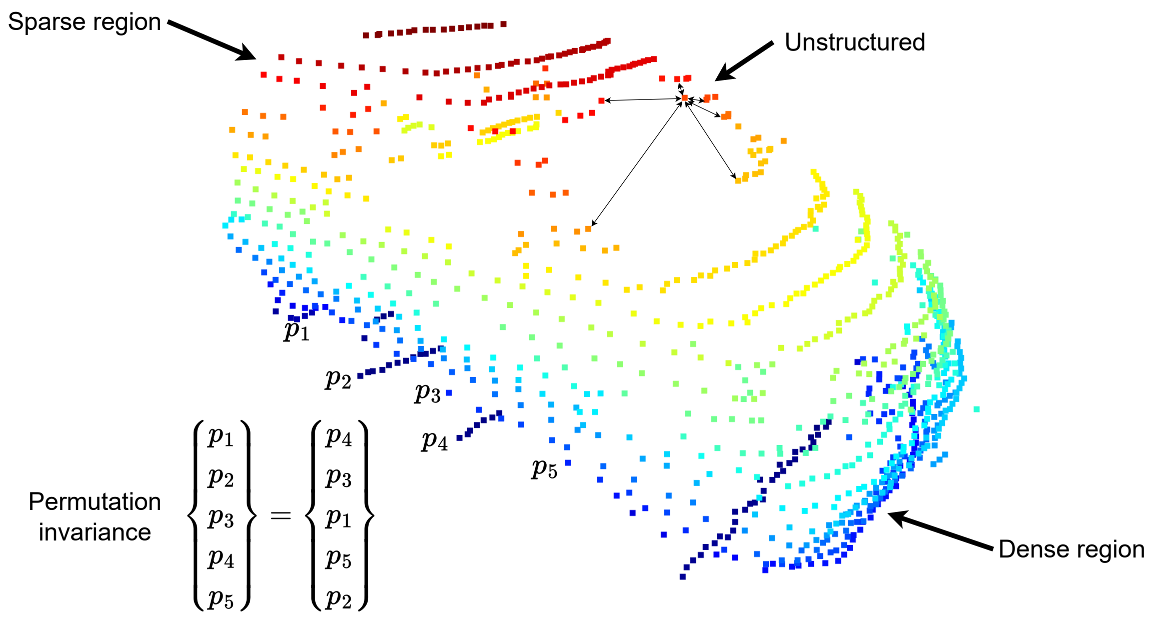

Sparse region
Unstructured
p_1
p_2
p_3
p_4
p_5
Permutation invariance
$\begin{Bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{Bmatrix} = \begin{Bmatrix} p_4 \\ p_3 \\ p_1 \\ p_5 \\ p_2 \end{Bmatrix}$
Dense region

Figure 2.4: Properties of point clouds visualized on an object point cloud extracted from a scene point cloud of the KITTI [36, 37] dataset.

These properties of point clouds do not allow the use of conventional Convolutional Neural Networks (CNNs) as used in image processing, since the convolution operation in CNNs requires data in an ordered, regular, and structured grid [43]. However, to address these challenges, point clouds can be converted to a structured grid format using a voxelization representation (Section 2.2), or neural networks designed specifically to handle point clouds without prior conversion can be employed. The former relies on discretization and therefore a loss of information, especially noticeable in the local structures, meaning the geometric relationships between adjacent points are not preserved. To avoid this information loss, methods and algorithms belonging to the second approach will be explained in the following.

A seminal work in this area is PointNet [44], introduced in 2016. In contrast to conventional 3D data processing techniques that depend on voxel grids or multiple views, PointNet directly handles unprocessed point clouds. PointNet's primary characteristic is its capacity to capture spatial hierarchies of features through a symmetric function, guaranteeing invariance to the point order in the input, thus addressing the issue of handling unordered point clouds. The architecture is efficient and suitable for tasks such as object classification, semantic segmentation, or part segmentation. The architecture comprises multiple layers that individually apply functions to each point, followed by an aggregation process to extract global features from the point cloud.

However, PointNet focuses mainly on capturing global features and may not effectively aggregate local features. This limitation can result in less than optimal performance on tasks that require attention to local structures or specific details [43]. Moreover, the efficiency of PointNet may be influenced by the density of the point cloud. Variations in point density in various parts of the point cloud can result in inconsistent feature extraction. To address the remaining two challenges, several follow-up works were proposed to improve PointNet, e.g., PointNet++ [45] or KPConv [46].

PointNet++ is a continuation of the original PointNet, aiming to tackle the challenge of aggregating local features. This network introduces a hierarchical model that recursively implements PointNet on subdivided segments of the input point cloud. By doing so, this method enables the model to comprehend both local patterns on a small scale and broader contexts on a larger scale, thus enhancing its effectiveness in tasks demanding fine-grained details.

KPConv introduces a point convolution operation that adapts to the density of the point cloud, enabling the flexible and efficient processing of point clouds with varying densities and effectively dealing with the issue of irregularity.

In addition to these methods, many other approaches exist, which according to Bello et al. [43] can be classified into approaches exploring the local neighborhood correlation among adjacent points [47–55], approaches that do not explore the local neighborhood correlation among adjacent points [56–60], and graph-based approaches [61–64]. For a detailed review of these approaches, the reader is referred to [43]. Although there exist many approaches, most of them are variations of the original PointNet and PointNet++ [65].

2.3.2 LiDAR Object Detection

3D object detection is a critical component for various applications, including AVs and robotics. It can be achieved using different sensor modalities such as a camera, LiDAR, RaDAR, or any combination of these. In this section, the focus is on methods and algorithms that use DL to extract relevant information from sensory input to perceive the environment of the AV.

Generally, the goal of 3D object detection is the prediction of the location, size, orientation, the classes of relevant and critical objects that potentially influence driving decisions and interact with the AV, such as other vehicles, pedestrians, and cyclists in the 3D space [66]. The input to these algorithms is the sensor data, that is, point clouds from LiDAR sensors or images from the camera. The output of these algorithms is usually the 3D bounding boxes of all objects predicted in the sensory input, and each bounding box B_i is represented as a 3D cuboid that encloses the object $B_i = [x_c, y_c, z_c, l_b, w_b, h_b, \theta, class]$ [66]. Here, $[x_c, y_c, z_c]$ are the center coordinates of the cuboid, $[l_b, w_b, h_b]$ are the dimensions length, width, and height of the object, θ is the orientation angle around the z axis (also yaw angle or heading), and $class$ is the category of the object, i.e. semantics. These 3D bounding boxes can be used in downstream tasks that, at a high level, include prediction, planning, and control of the AV.

In the following, the focus is on 3D object detection using point clouds captured by automotive LiDAR sensors. The detection of objects from outdoor automotive point clouds poses its own challenges, such as non-uniform point distribution leading to various point cloud densities depending on the object's distance. Furthermore, 3D object detectors for the automotive domain have the requirement of real-time processing to be able to react to the environment in a reasonable time to comply with the safety-critical nature of AD, which requires computationally efficient methods [66].

Similarly to the representation of point clouds introduced in Section 2.2, LiDAR 3D object detection methods can also be categorized into point-based, grid- or voxel-based, and range-based approaches, which are briefly discussed in the following. Despite the differences, all three approaches have in common that they are trained in a supervised fashion, meaning that annotated point clouds are used as network input and the network prediction is compared with the annotation during training.

Point-based approaches employ feature extractors based on point-based backbone networks as explained in Subsection 2.3.1, hence they use permutation-invariant operators as used in PointNet [67]. Based on downsampled points and their features, prediction heads predict the parameters of 3D bounding boxes. PointRCNN [68] and Pointformer [69] are prominent representatives of point-based approaches.

Grid-based and **voxel-based** approaches utilize a discretized BEV, pillar, or voxel representation of the point cloud and employ traditional 2D CNNs or 3D sparse neural networks to capture features of the grids [66]. The resulting BEV feature map is processed by 2D backbone networks and detection heads to predict objects. According to Qian et al. [67], voxel-based methods gain advantages over BEV-based methods due to reduced scale ambiguity and fewer occlusions. PIXOR [70], PointPillars [71], and VoxelNet [56] are prominent representatives of grid-based approaches and use BEV, pillar, or voxels, respectively.

Lastly, **range-based** approaches use the 2D sensor depth image where each pixel contains the depth information rather than RGB values [66]. These approaches use networks designed for conventional 2D object detection models as the representation is similar and the most prominent network of this approach is LaserNet [72].

The prediction of 3D objects can be guided by using anchors, which are predefined 3D cuboids placed in the 3D space and that use the prior knowledge of expected dimensions for the different object categories, i.e. objects of the same category typically have the same cuboid aspect ratio. These anchors are placed throughout the input space and the existence, position, size, and orientation offset of each anchor to best fit the detected objects are regressed by the network [73]. In contrast to anchor-based object detection, anchor-free approaches directly predict key points, such as the center or boundaries of detected objects. This is done by first segmenting the point cloud into foreground and background points, whereas the former are points potentially belonging to objects. For each foreground point, a prediction for an object is made, which can be refined in later stages by using the non-maximum suppression post-processing technique. This offers greater flexibility, as these algorithms are less biased towards the predefined objects' shapes and sizes.

The most common metric for evaluation of 3D object detection is Average Precision (AP) [67], which is calculated as the area under the precision-recall curve [38], and is adapted from 2D object detection. To apply AP on 3D object detection, [36] propose 3D AP and BEV AP, which differ in the matching of the predictions and the ground truths for recall and precision calculation [66]. In 3D AP, a True Positive (TP) is counted if a defined threshold of 3D Intersection over Union (IoU) of a predicted and a ground truth 3D cuboids is reached, whereas in BEV AP, the intersection is calculated between the 2D BEV-projected cuboids. The thresholds for IoU are usually 0.3, 0.5, and 0.7 for different difficulty levels [67]. For the calculation of AP, usually 40 equal-spaced recall levels $[1/40, 2/40, ..., 40/40]$ are evaluated.

For the official benchmark of the public large-scale dataset nuScenes, Caesar et al. [74] propose the center AP, which matches a predicted 3D cuboid with a ground truth 3D cuboid if the 2D center distance of the cuboids is below a threshold. Furthermore, they calculate additional metrics quantifying the translation error, the scaling error, the orientation error, the velocity error, and the attribute error (classification) and combine them with the center AP to obtain the overall nuScenes detection score. Unlike AP and its variations, Deng et al. [75] and Philion et al. [76] propose to measure the performance of object detection algorithms using motion planning as a downstream task. Although that removes some of the limitations of the AP, such as weighting the error of a misdetection the same for close and far objects, the use of downstream tasks incorporates new challenges, e.g. the need for pretrained motion planners [66].

More details on 3D object detection for different modalities, applications, and metrics can be found in the surveys [66, 67, 73].

2.4 Datasets for Autonomous Driving

All networks introduced in Subsection 2.3.2 rely on supervised training, that is, they require annotated datasets for training. These datasets not only play a crucial role in network training but can also be used to benchmark and compare the performance of different object detection algorithms. In this thesis, the focus is on outdoor datasets as they are relevant in the field of AD. As briefly explained in Subsection 2.3.2, large-scale outdoor point clouds are characterized by the fact that the relevant objects for the AD applications are spatially separated and the point clouds are sparse. In general, the datasets can be separated into datasets collected in the real world and synthetic datasets generated in a simulated environment. In the following, the focus is on datasets for the tasks 3D point cloud object detection and tracking, as compared to 3D point cloud segmentation, which requires point-wise labeling of every point in each point cloud, which ultimately leads to higher labeling effort and costs.

2.4.1 Real-World Datasets

Real-world datasets are collected by sensors-equipped vehicles that are driving in a certain area to capture a variety of scenes. These vehicles are equipped with multiple sensors, which usually cover multiple modalities. This can facilitate the subsequent manual labeling process, in which a 3D bounding box is assigned to each relevant object for all predefined classes. Real-world datasets have the advantage of high realism by capturing the unpredictability of real environments, such as unexpected driver or pedestrian behavior, and by including mixed weather conditions. Moreover, the sensors present in the real-world datasets demonstrate characteristics comparable to those of the similar sensors used in the AV, guaranteeing precision in replicating real driving conditions. As data collection and manual annotation are a time-intensive process, the creation of real-world datasets is costly, and, therefore, only a few publicly available datasets exist. Additionally, these datasets are limited in size, expressed by the number of labeled frames, and only cover a limited region and scenario diversity.

Pioneering work in the field of publicly available real-world datasets is the KITTI dataset [36, 37] that covers urban, rural, and highway areas in the city of Karlsruhe, Germany. It includes around 15,000 annotated point clouds from a single $360°$ rotating LiDAR sensor. Following the success of KITTI, several other real-world datasets have been released. They mainly address the scale of the data, the diversity of the data, provide more annotated categories, or provide data of more modalities [66]. Among those datasets are the Oxford RobotCar dataset [77], nuScenes [74], BLVD [78], Waymo Open Dataset [79], ApolloScape [80, 81], and Argoverse [82]. For example, the Waymo Open Dataset provides around 200,000 labeled frames; nuScenes includes not only daytime and sunny weather but also data captured at night or in rainy conditions. Moreover, nuScenes also introduces more object categories, such as traffic cones, and includes radar data in addition to LiDAR and camera data.

2.4.2 Synthetic Datasets

Synthetic datasets are recorded in a simulation environment by rendering synthetic scenes while driving a vehicle equipped with virtual sensors. Simulation environments are usually based on game engines such as Unreal Engine [83] or Unity [84]. During sensor data recording, the object labels of all surrounding objects, i.e. the ground truth, can be stored and associated with each frame. As time-consuming and costly annotation becomes no longer necessary, synthetic data are more scalable than real-world data. Furthermore, scenarios can be exactly reproduced for systematic testing and can also include rare or dangerous situations that can not be replicated in the real world. Despite the benefits of using synthetic data, the reduced realism compared to the real-world dataset opens up new challenges. In addition, only a few synthetic datasets are publicly available.

[85] and [86] suggest the use of the open-world video game Grand Theft Auto V [87] to create a dataset that contains more than 250,000 annotated images at the pixel level. This dataset is intended for various vision tasks, such as optical flow, semantic instance segmentation, object detection, and tracking. The synthetic datasets Virtual KITTI [88] and the successor Virtual KITTI 2 [89] target the same perception tasks and

mimic the real-world KITTI dataset. Car Learning to Act (CARLA) [90] is a simulation environment that is based on Unreal Engine and allows for the collection of perception data by providing a Python Application Programming Interface (API) for custom scenario generation. In addition, CARLA includes several virtual sensor models for the camera, LiDAR, and RaDAR.

The All-In-One Drive dataset [91] employs CARLA to generate a large-scale synthetic dataset that includes multiple sensor modalities, camera, LiDAR, RaDAR, IMU, GNSS, and the respective annotations for perception tasks. All-In-One Drive also incorporates out-of-distribution driving scenarios, such as traffic accidents, violations of traffic rules, and adverse weather conditions.

3 Related Work

This chapter presents the existing literature in the context of LiDAR domain shift and LiDAR domain adaptation for AVs. The primary objective of this chapter is to review the state of the art in these areas to understand the evolution and challenges in this field. This review serves as a basis for the definition of the research gap and the formulation of the research questions in Chapter 4. The scope of the related work chapter comprises the areas of LiDAR domain shift and LiDAR domain adaptation. In Section 3.1, the LiDAR domain shift is defined, and the causes and impacts of LiDAR domain shift are discussed, along with the methods used for its measurement and evaluation. Section 3.2 covers the problem definition of LiDAR domain adaptation. Furthermore, domain adaptation techniques, in particular for the application of 3D point clouds from the automotive domain, are discussed.

3.1 LiDAR Domain Shift

The first focus in this chapter is the concept of LiDAR domain shift, which emerges as a central aspect in today's LiDAR research. The phenomenon of LiDAR domain shift, which describes the change in LiDAR data under different conditions, poses unique challenges in the handling of LiDAR data and has great impacts on applications such as AD. This section aims to present the nature of LiDAR domain shift by introducing a definition and explaining the background, i.e., exploring the factors that contribute to the domain shift. In addition, the implications of the domain shift on perception algorithms for AVs are presented. Furthermore, the main contribution of this section is the discussion of state-of-the-art quantitative and qualitative metrics used to identify and precisely measure the shift between LiDAR point clouds originating from different domains. This discussion provides a foundation for comprehending the complexity of the LiDAR domain shift, paving the way for discussing methods to reduce the domain shift using domain adaptation techniques in Section 3.2.

3.1.1 Definition and Background

The goal of machine learning is for a model to learn an objective function through training data, enabling its application to unseen test data with minimal risk [92]. It is commonly assumed that the data used for training and testing models are drawn from a fixed distribution [93], have shared joint probability distributions [92], or share the same feature space [94]. This implies that the training and testing datasets are independent and identically distributed (i.i.d.) [95]. Using this assumption, the uniform convergence theory guarantees a training error close to the true error [93].

However, in real-world applications, this assumption is often violated and the distributions of training and testing data differ. This difference between training and testing data is called the domain shift. In the literature, the domain shift can also be found as *domain gap* [96], *distribution shift* [97], *dataset bias* [98], or *reality gap* in the case of simulated training data and real-world testing for AD applications [99]. The annotated training and testing data are termed source and target domains, denoted as $\mathcal{S}$ and $\mathcal{T}$, respectively. Therefore, in the challenge of domain shift, it is assumed that the data distribution P differs, that is, $\mathrm{P}_{\mathcal{S}} \neq \mathrm{P}_{\mathcal{T}}$ [97]. Typically, a domain refers to the scope of application of an algorithm [94]. In AD, this encompasses data attributes like lighting conditions, weather, sensor perspectives, object types (e.g., pedestrians, vehicles, traffic signs), and geographical settings. Domains can differ significantly; for example, urban driving domains may include heavy traffic, high-rise buildings, and intricate intersections, while rural driving domains might consist of light traffic, expansive fields, and straightforward road configurations. This domain shift can appear in various applications in which models are trained using a specific dataset and applied in the real world, and is mainly studied for image processing applications [100] with tasks such as semantic segmentation [101, 102] or visual recognition [103, 104].

Kouw et al. [105] define three special types of domain shift, that is, prior shift, concept shift, and covariate shift. **Prior shift** occurs when the class-conditional distributions, indicating the likelihood of data for a given class, stay consistent across various domains, while the prior probabilities, representing the relative proportion of each class, vary. In essence, the overall occurrence of each class shifts, but the relationship of each class with the data remains unchanged. For example, the proportion of different objects changes between datasets. **Concept shift** occurs when data distributions remain consistent between domains, while posterior distributions vary. This means that the data appear unchanged, but the way classes are assigned to the data is altered. For example, the appearance or characteristics of objects change between datasets. **Covariate shift** occurs when there is a discrepancy in the data distributions between domains, while the posterior distributions, which represent the likelihood of a class given the data, remain the same. Essentially, this implies that although the data distribution may vary, the association between the data and the classes remains consistent. For example, the environmental conditions such as weather or lighting change between datasets.

Schwonberg et at. [96] note that the overall domain shift is usually a mix between the three special forms of domain shift. However, Kouw et al. [105] note that the covariate shift is the most studied form of domain shift. Also, most related work presented in the following addressing the domain shift using domain adaptation methods focuses on the covariate shift to align the data distributions of source and target datasets.

There are a variety of reasons for the domain shift. For example, the statistical properties of a domain can change over time or space [92]. Another reason could be that it is preferable to use a publicly available and annotated source dataset for model training, as the collection and generation of a custom dataset is a time-consuming and costly process [92]. In the area of LiDAR domain shift, the reasons for the domain shift can be attributed to environmental factors and hardware variability. In the following, several types of domain shift are briefly explained using the notation of "*source*-to-*target*", whereas *source* and *target* represent the source and target datasets, respectively.

The **country-to-country** domain shift occurs when the dataset is collected in a country different from where the model trained on this dataset is utilized. Countries differ in terms of the shapes of traffic signs and road layouts, or vehicle shapes and sizes. For example, the best-selling vehicle in USA is a full-sized pickup truck with a length of over $5.8\,\mathrm{m}$, whereas the best-selling vehicle in Germany is a compact car with a length of around $4.3\,\mathrm{m}$ [106].

The **sensor-to-sensor** domain shift occurs when the sensor of the source dataset and the target application differ. This can be caused by a difference in resolution, mounting position, noise characteristics, intensity measurement, range, or scan pattern.

The **dataset-to-dataset** domain shift is a combination of the country-to-country and sensor-to-sensor domain shift. This can occur when a model is trained, for example, using the KITTI dataset, and applied in the USA using a vehicle with a different sensor setup.

The **weather-to-weather** domain shift occurs when the weather between source and target differs. Most datasets are captured during good and sunny weather, and hence the domain shift occurs when models trained with these datasets are tested during rainy or foggy conditions.

Similarly, the **season-to-season** domains shift describes the changes of seasons between the source and the target. For example, the scenery in winter may not only include snow, but also trees contain fewer leaves in winter, and convertibles are rarely seen during winter, i.e. the intra-class distributions differ.

The **day-to-night** domain shift mainly affects the camera, as LiDAR is an active sensor.

Finally, the **sim-to-real** can be a combination of multiple domain shifts, depending on the simulation settings. It occurs when training data is gathered in simulation, but the application of the model is in the real world. Here, the differences between simulation and the real world itself have multiple sources, from limited agent and environment selection, simplified virtual sensor models without noise characteristics of the real world, simplified physics, lighting, textures, and material properties to idealized behavior of agents. Especially, the noise characteristics of the LiDAR sensors are complex to model in simulation, as they are a combination of multiple independent noise sources, such as thermal noise, shot noise, background noise, readout noise, and speckle noise [30].

Domain shift affects data quality and application efficacy, regardless of the source of the domain shift. Typically, the efficacy of a dataset is determined not solely based on the volume of data, but also on the quality of the data in terms of its variety, distribution, and realism [107]. The domain shift specifically affects the aspect of the quality of a dataset. Models may experience a decrease in performance when used with data from a related but distinct domain compared to the one on which they were originally trained [95]. Even small changes in, for example, camera or LiDAR parameters can have a huge impact on neural networks [108]. This reduced model performance leads to higher error rates, which is especially problematic for safety-critical applications such as AD. Furthermore, the interpretation and validation of the model becomes more difficult, as the characteristics of the models on the source dataset might not be transferable to the behavior on the target dataset. To assess the domain shift and compare the source and target datasets, several methods exist, which will be discussed in Subsection 3.1.2.

3.1.2 Analysis of the Domain Shift

To analyze and measure the domain shift, there are various methods that can be categorized into either qualitative or quantitative methods. Moreover, different datasets are used to benchmark the performance of methods that minimize the domain shift. In the following, these evaluation methods, metrics, and benchmark datasets are presented.

Qualitative Analysis

The qualitative assessment of the domain shift is usually performed by a visual inspection of the samples from both the source and target domains and by comparison of the global and local structural differences of the samples. This is usually done to assess the quality of generated samples for image-based Generative Adversarial Networks (GANs) as in [109–115], where the quality of a few generated image samples is assessed by humans. This subjective evaluation can also be transferred to LiDAR point clouds, although these require specific domain knowledge of human raters [116].

Another common method for qualitative domain shift evaluation is t-Distributed Stochastic Neighbor Embedding (t-SNE) [117]. According to [96], after quantitative measures, t-SNE is the second most widely used method for evaluating domain alignment. T-SNE is a statistical technique that uses non-linear dimensionality reduction to reduce high-dimensional data and visualize the resulting dimensional-reduced data in two-

or three-dimensional maps. Points that are close together in the low-dimensional t-SNE graph typically correspond to similar points in the original high-dimensional space. Hence, the hypothesis is that neighboring points in the low-dimensional space indicate a lower domain shift of the high-dimensional input data than distant points. This means that after domain adaptation, the centroids of the adapted data points should be closer to the centroids of the target data points than the source centroids are to the target centroids [118].

As an example for t-SNE, Figure 3.1 shows a t-SNE graph generated with data from the MNIST dataset [119]. This dataset contains 28x28 pixel black and white images of handwritten digits from zero to nine, hence each image contains 784 dimensions. T-SNE is applied to reduce the dimension of each image to two dimensions, resulting in the t-SNE graph in which each point presents a dimension-reduced image. The formation of separate colored clusters indicates that t-SNE successfully differentiates between the ten different digits.

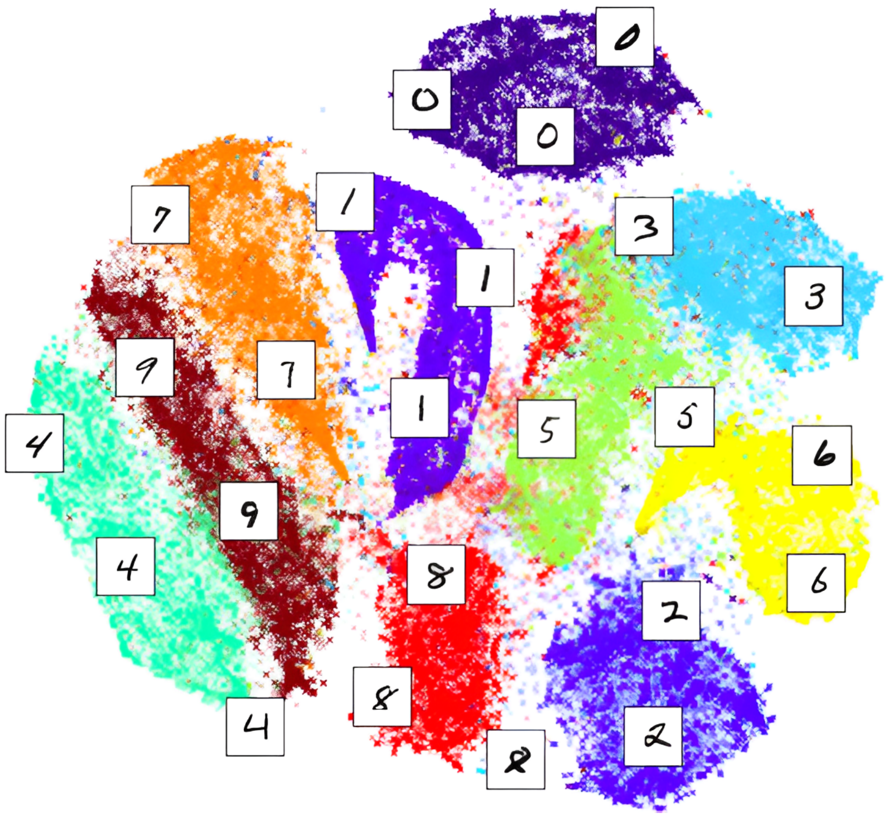


Figure 3.1: T-SNE graph for visualization of MNIST images (from [120]).

In the following, a few examples of the usage of t-SNE for domain shift analysis are described. [121] use t-SNE to visualize the feature embeddings generated by a feature extractor to demonstrate the efficacy of their LiDAR point cloud realism classifier. [44, 122, 123] use t-SNE to visualize the global shape features learned of the shapes in the ModelNet40 [124] or ShapeNet [125] datasets. [126–128] use t-SNE to visualize the distribution of features in the target domain to highlight the performance of their unsupervised domain adaptation approaches. Wang et al. [100] follow the same approach, but utilize t-SNE to generate a three-dimensional mapping instead of a two-dimensional mapping. Yan et al. [129] use t-SNE to compare their 3D shape reconstruction autoencoder with baselines and further analyze the size of the cluster radii in the t-SNE graphs. They concluded that based on the smaller cluster radii of their approach, it is less sensitive to sampling variations and able to learn more generalizable features.

It should also be mentioned that in addition to t-SNE for dimensionality reduction for visualization of high-dimensional network features, Uniform Manifold Approximation and Projection (UMAP) [130] provides another method similar to t-SNE. In contrast to t-SNE, UMAP preserves a more global structure of the input data. However, as of today, UMAP is not used in the context of qualitative domain shift evaluation.

Instead of directly applying t-SNE or UMAP on points clouds or latent feature vectors of task-specific networks, such as object detection or semantic segmentation, Hubschneider et al. [131] employ a Variational Autoencoder (VAE) to generate latent representations of real and synthetic point clouds, and compare and analyze these latent representations using t-SNE. The VAE is trained on real-world target data and can be used to generate latent representations of different source datasets, such as data from simulation or generated with GANs for domain adaptation.

Another qualitative domain shift analysis method is introduced by Triess et al. [116]. The authors conducted a human-centered survey to obtain a mean opinion score, which was first introduced by Ledig et al. [132] for RGB images. To obtain this mean opinion score, Triess et al. [116] asked human rating experts to assign a score to the generated point clouds, which should reflect the visual quality of the realism of these point clouds. In contrast to RGB images, LiDAR data requires specific domain knowledge, which necessitates the human raters to be familiar with LiDAR data rather than being laypersons. This process is time-consuming, potentially expensive, and does not scale to large or multiple datasets. Furthermore, the qualitative assessment done by humans is always subjective and makes a comparison of different methods across different works difficult. Therefore, this requires quantitative methods to assess the domain shift in point clouds.

Quantitative Analysis

Apart from qualitative methods to prove the presence of a domain shift, quantitative methods allow to assess the scale of the domain shift. This is especially important to assess the efficacy and compare methods that reduce domain shift, that is, domain adaptation techniques, which are presented in Section 3.2. The advantage of quantitative methods over qualitative methods is that the former allows for automatic evaluation of entire datasets, which makes these approaches more scalable than qualitatively assessing a few selected samples. Furthermore, quantitative evaluation is not susceptible to subjective assessment. In general, the quantitative methods can be categorized into statistical comparison methods directly applied to the data, and in methods using task-specific networks to evaluate the domain shift, e.g. using object detection or semantic segmentation networks.

Statistical comparison methods rely on analyzing the available data of all datasets relevant for the domain shift analysis, and can always be done as an initial step. Wang et al. [106] perform a comprehensive study of several publicly available real-world datasets using simple statistics to compare different characteristics and quantify the dataset-to-dataset domain shift. In detail, the authors compare the datasets KITTI [36, 37], Argoverse [82], nuScenes [74], Lyft [133], and Waymo Open Dataset [79], as seen in Figure 3.2. The dataset-to-dataset domain shift present between these public datasets is a combination of the country-to-country and sensor-to-sensor domain shift, as these datasets are captured in Germany (KITTI), USA (Argoverse, nuScenes, Lyft, Waymo Open Dataset), and Singapore (nuScenes), and use different LiDAR models, which have different properties, e.g. a different number of vertically stacked lasers. The authors compare the number of LiDAR points per car and per scene in the camera FoV up to a distance of $70\,\mathrm{m}$ and further analyze the ground truth size of the bounding boxes for cars. They conclude that on average the Waymo Open Dataset has more than ten times more points per car (1356) than nuScenes (86), and the average size in all three dimensions of cars in KITTI is lower than the average size of cars in the Waymo Open Dataset. The latter can be attributed to the fact that the average vehicles sold in each country differ, as explained in Subsection 3.1.1 [106]. Other statistical methods include the comparison of the object's spatial distribution and an average number of objects per point cloud, as in [134].

The second category of methods to quantify domain shifts is based on training and evaluating task-specific networks [135]. In general, the idea is to train semantic segmentation or object detection networks utilizing a source dataset for training the networks and assessing the performance of these trained networks on a target dataset. Standard metrics, such as mean IoU for semantic segmentation or mean AP for object detection, are used for quantitative evaluation. The mean IoU is calculated as

$$mean\ IoU = \frac{TP}{TP + FP + FN}, \tag{3.1}$$

with TP, False Positive (FP), and False Negative (FN) indicating the predicted pixels that are true positive, false positive, and false negative, respectively.

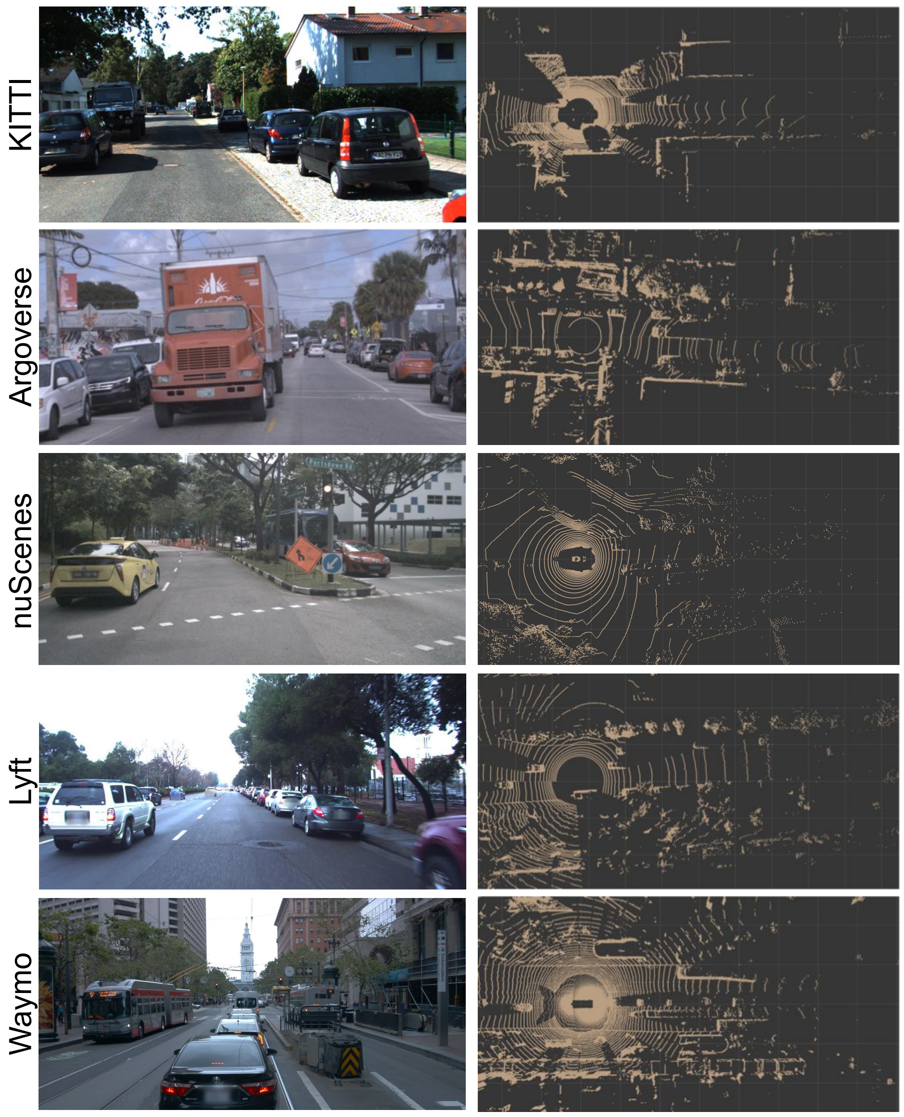


Figure 3.2: Front-view images (left) of five datasets and their corresponding LiDAR point clouds (right). The datasets are not only recorded in different geographic locations but also used different LiDAR sensor models to record the point clouds (from [106]).

Most of the work in this area of domain shift quantification originates from the camera modality. Nevertheless, few works apply similar metrics to the LiDAR modality and thereby prove their applicability to this domain. In the following, first works quantifying domain shift are presented using camera semantic segmentation and object detection, before describing works employing LiDAR semantic segmentation and object detection for quantification.

In order to measure the camera sim-to-real domain shift by utilizing semantic segmentation networks, the source domain is usually synthetic data from Grand Theft Auto V (GTA V) [85] or SYNTHIA [136], and the target domain is the Cityscapes dataset [137]. Several works use these datasets to test domain adaptation methods; however, only a few report the performance of models trained on both the source and target datasets, which is required to assess the domain shift. Furthermore, the choice of semantic segmentation network also varies, making a comparison of different methods difficult. For example, Dundar et al. [138] use the semantic segmentation networks Dilated Residual Network DRN-C-26 [139] and DeepLabV3 [140] and report the mean IoU for GTA V and SYNTHIA as simulated source datasets and Cityscapes as real-world target dataset. The results on DRN-C-26 show a mean IoU of $22.9\,\%$ (GTA V), $18.5\,\%$ (SYNTHIA), and $66.3\,\%$ (Cityscapes), and the results for DeepLabV3 a mean IoU of $40.0\,\%$ (GTA V), $31.8\,\%$ (SYNTHIA), and $75.6\,\%$ (Cityscapes). In conclusion, both networks show a pronounced sim-to-real domain shift, which can be specified at $35.6\,\%$ (GTA V-to-Cityscapes) or $43.8\,\%$ (SYNTHIA-to-Cityscapes) mean IoU for DeepLabV3.

Instead of using semantic segmentation networks to quantify camera domain shift, Adam et al. [141] and Nowruzi et al. [107] use the 2D Single Shot MultiBox Detector (SSD) [142] and compare the mean AP or the recall and precision achieved using different training datasets. Both works conclude that the metric values achieved by training with simulated data are in all experiments lower than the values when trained with real-world data, given that both are compared on the same test split of the respective real-world dataset. Nowruzi et al. [107] further analyze the performance of mixed data training, that is, combining simulated and real-world images in a single training dataset. In addition, they explored the possibility of initial training using simulated data and subsequent refinement with the target real-world data. Even though this reduces the domain shift to a larger extent than mixed data training, it still can not fully close the sim-to-real domain gap. Seib et al. [143] and Burdorf et al. [144] also investigate the possibilities of using mixed data to train 2D object detectors and hence quantify the sim-to-real domain shift. The findings suggest that synthetic data have the potential to substitute real-world data to some extent, but this requires a larger number of samples. Additionally, the use of synthetic data for initial network training can improve performance, surpassing the performance achieved by only relying on real-world data for training [88, 89]. Instead of comparing the network output, that is, calculating metrics using the network outputs and comparing these, Ljungqvist et al. [145] utilize both synthetic and real-world data to train 2D object detectors and analyze the outputs of the individual network layers. In detail, the authors compute the linear centered kernel alignment similarity index [146], which provides layer-wise information on the similarity of features. The comparison of networks trained on synthetic and real-world data reveals a strong resemblance in the initial layers of the network, while demonstrating more significant discrepancies in the detection head of the network.

In the field of domain shift quantification for LiDAR point clouds using neural networks, the approach involves the use of semantic segmentation or object detection networks, similar to the camera domain. Most of the work is focused on semantic segmentation networks, as these are the primary applications of domain adaptation approaches. This is justified in the fact that the labeling effort for pixel-wise and point-wise annotation of real-world images and point clouds exceeds the labeling effort of placing 2D or 3D bounding boxes in images or point clouds needed for object detection. For reference, annotating a single 2D image with semantic labels of Cityscapes requires seven minutes for coarse annotations and $1.5\,\mathrm{h}$ for fine annotations [137]. Yue et al. [147] and Spiegel et al. [148] train LiDAR semantic segmentation networks using synthetic, real, or mixed data, and evaluate trained networks on real-world data using the IoU metric. Yue et al. [147] conclude that a domain shift of $35.6\,\%$ IoU exists between the source GTA V synthetic dataset ($29.0\,\%$ IoU) and the target KITTI real-world dataset ($64.6\,\%$ IoU), and that the performance on the target dataset can be increased by $1.4\,\%$ IoU if using mixed data instead of data from only the real world for training. There exist also several works targeting LiDAR domain adaptation methods, and they implicitly evaluate the domain shift using LiDAR semantic segmentation networks to compare the performance of their domain adaptation approaches.

Examples include [116, 149–152] for the sensor-to-sensor domain shift, [153–156] for dataset-to-dataset domain shift, and [157, 158] for sim-to-real domain shift.

Alongside LiDAR semantic segmentation, the second task-specific method of using neural networks for domain shift quantification is to employ LiDAR 3D object detectors and analyze the 3D mean AP achieved when training these detectors with different datasets. Dworak et al. [159] train the 3D object detection networks VoxelNet [56], YOLO3D [160], and PointPillars [71] on synthetic data generated in CARLA and test the networks on real-world KITTI data. The best configuration achieves a mean AP of up to $89\,\%$ when trained and tested on CARLA data, but only achieves $19\,\%$ mean AP when tested on KITTI data. Dworak et al. [159] also analyze the performance of using mixed data for training or fine-tuning using real-world data. Although the results align with the results of [107] for camera object detection, these additional methods can not fully reduce the existing sim-to-real domain shift. Similarly to [159], Fang et al. [161] and Manivasagam et al. [162] employ 3D object detectors and train them using data from CARLA and KITTI. However, their goal is not the quantification of the sim-to-real domain shift, but rather to access the performance of their methods that generate synthetic LiDAR data. Tsai et al. [163] and Wang et al. [106] focus on the dataset-to-dataset domain shift in real-world datasets, and quantify it using the same method of training the 3D object detectors PV-RCNN [164], SECOND [165], or PointRCNN [68], respectively.

Unlike the works discussed previously, Triess et al. [121] develop an adversarial network capable of quantifying the realism of synthetically generated LiDAR point clouds. To validate the quantitative metric, they evaluated it using semantic segmentation networks.

3.2 LiDAR Domain Adaptation

The second major area of focus in this chapter is the concept of LiDAR domain adaptation, which tries to solve the challenge of domain shift introduced in the previous Section 3.1. LiDAR domain adaptation is solved using a variety of methods, which will be introduced in this section. First, the problem of domain adaptation independent of the sensor modality is introduced alongside the specific terminology used in this field. Afterward, domain adaptation approaches specifically designed to adapt 3D point clouds of LiDAR are discussed, with an emphasis on methods developed for AD applications. This review of the state of the art serves as a basis for deriving the research gap in Section 4.1.

3.2.1 Definition and Taxonomy

Building upon the insights of the definition and background of domain shift in Subsection 3.1.1, this section narrows the focus to the specific challenge of domain adaptation. Domain adaptation addresses the challenge of creating models to adapt data or to utilize other techniques to make task-specific models generalize well in the target domain, although these models are trained on a different source data distribution [93]. This can be achieved by creating approaches that learn the mapping between the source and target domain so that a task-specific model trained on the adapted source data can be applied to the target data at the test time [166]. The motivation for domain adaptation is the prevalence of deep neural networks that are often data-hungry, that is, the performance of DL models is limited due to insufficient training data in terms of size and diversity [26]. This requires researchers to exploit methods that allow the use of large-scale datasets that can be easily generated and annotated, but this poses the challenge of domain shift, as discussed in detail in Section 3.1. For the domain adaptation problem, the training data from the source domain are usually label-rich, whereas the dataset from the target domain is label-scarce or even has no labeled data [98].

The main focus of research in the domain adaptation area is on vision-based approaches, that is, adapting camera images for tasks such as classification, semantic segmentation, or object detection [94]. In general, the main application is classification (image-level predictions) or semantic segmentation (pixel-wise predictions), and more approaches and benchmarks can be found in this area. Nevertheless, some works address domain adaptation for camera object detection, as highlighted in a comprehensive survey on unsupervised domain adaptation of object detectors [97].

Domain adaptation can be assigned to the area of transfer learning [167], as seen in Figure 3.3. Within this field, domain adaptation is categorized as transductive transfer learning, which, together with inductive transfer learning and unsupervised transfer learning, is a special form of transfer learning. **Transductive transfer learning** is distinguished by the presence of annotated data solely in the source domain, and in the special case of domain adaptation, the source and target domains differ, but the task (e.g. object detection) remains the same. In **inductive transfer learning**, the source and target tasks differ. Unlike transductive transfer learning, this requires annotated data in the target domain [167]. Examples of inductive transfer learning are multi-task learning or self-taught learning, whereas the former makes use of annotated data from the source domain. The last subcategory of transfer learning is **unsupervised transfer learning**, in which the target and source domains are different but related. Related domains are those found in similar environments, such as outdoor driving scenarios, whereas dissimilar domains refer to specific attributes within these environments, such as sunny versus rainy weather conditions [94]. Furthermore, unsupervised transfer learning is characterized by the absence of annotated data in either domain during the training process. Examples are clustering or dimensionality reduction [167].

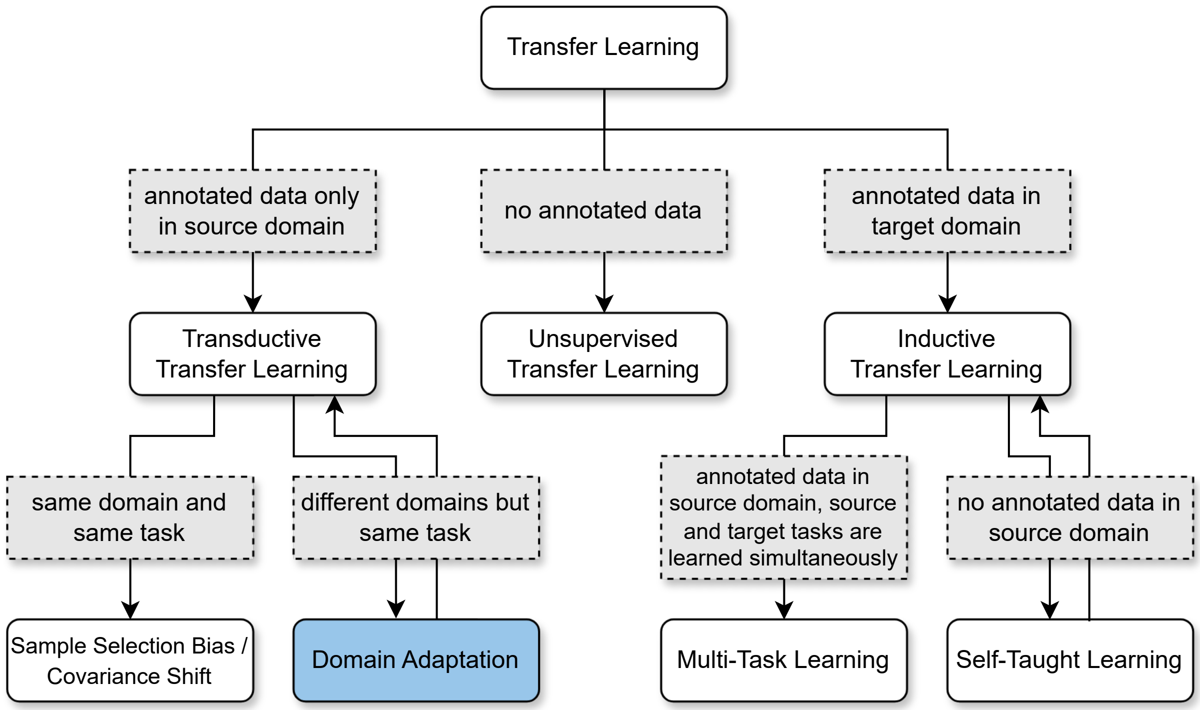

Transfer Learning
annotated data only in source domain
no annotated data
annotated data in target domain
Transductive Transfer Learning
Unsupervised Transfer Learning
Inductive Transfer Learning
same domain and same task
different domains but same task
annotated data in source domain, source and target tasks are learned simultaneously
no annotated data in source domain
Sample Selection Bias / Covariance Shift
Domain Adaptation
Multi-Task Learning
Self-Taught Learning

Figure 3.3: Integration of domain adaptation into the overarching research field of transfer learning and differentiation from related research fields. The domain adaptation is highlighted, as it will be further categorized in the following (adapted from [94]).

Domain adaptation itself can be further categorized into the three paradigms of semi-supervised, weakly supervised, and unsupervised methods [97] (Figure 3.4). In the context of object detection, **semi-supervised domain adaptation** requires the source dataset to be fully annotated, that is, the bounding boxes and classes for each sample need to be labeled. For the target dataset, only a subset is fully annotated. Examples of semi-supervised domain adaptation include [168–171]. **Weakly-supervised domain adaptation** methods are similar to semi-supervised methods in the sense that the source dataset is fully annotated with bounding boxes and classes for all samples, but the target dataset labels only contain information on the presence or absence of any classes in form of binary annotations, i.e. these are weak labels. In [172], these weak labels are bounding boxes, but the task is semantic or instance segmentation. Paul et al. [173] use image-level labels to predict the presence of classes. In the last category, **unsupervised domain adaptation** methods, the source domain dataset is fully labeled while the target domain dataset is not labeled at all. For the standard unsupervised domain adaptation, both the source and target domains can be used during the adaptation, while the source-free unsupervised domain adaptation describes the process of only using target data during adaptation without access to source data [96]. Oza et al. [97] highlight that the unsupervised domain adaptation formulation is the most challenging and the other formulations can be easily adapted from unsupervised domain adaptation. For this reason, most research in the area of domain adaptation is focused on the field of unsupervised domain adaptation.

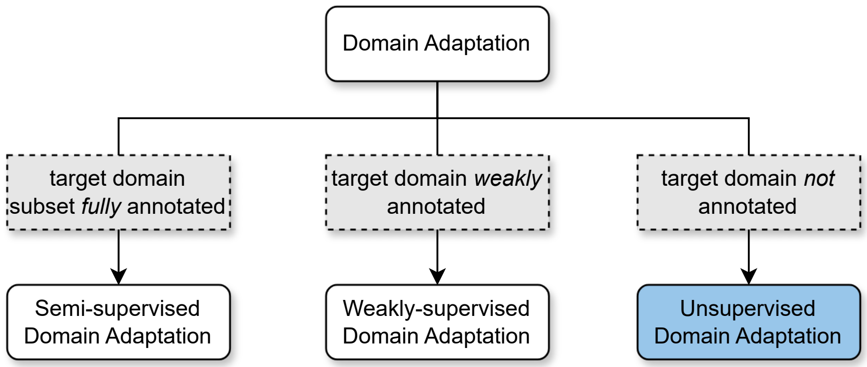

Domain Adaptation
target domain subset *fully* annotated
target domain *weakly* annotated
target domain *not* annotated
Semi-supervised Domain Adaptation
Weakly-supervised Domain Adaptation
Unsupervised Domain Adaptation

Figure 3.4: Categorization of different forms of domain adaptation according to [97]. Unsupervised domain adaptation is highlighted, as it will be further categorized in the following.

According to [96, 174, 175], unsupervised domain adaptation itself can be categorized into different adaptation spaces. For neural networks, these adaptation spaces can be at the input, feature, or output levels of the network (Figure 3.5).

The **input space domain adaptation** is in most cases a form of style transfer, in which the style of the frames is stylized before serving as network input during training or inference, such as converting daytime scenes into nighttime scenes. This style transfer can be applied in two ways. The style of the source domain can be transferred to match the style of the target domain before training the network, or the style of the target domain can be transferred during inference to the style of the source domain with which the network was initially trained. The latter requires style transfer during inference, which can increase latency or affect real-time capabilities for AD applications. Style transfer is usually done using methods based on GANs, for example, CycleGAN [176] for image-based style transfer, but it can also be achieved using histogram matching or normalization techniques [96]. Other forms of input space adaptation include data augmentation and frame mixing, which describe methods that increase the number of source domains or mix the source with target domains within the images or point clouds.

Feature space domain adaptation (also network-level domain adaptation [175]) tries to align the distributions of the source and target domains in the layers of the network. In this way, the network should map the same semantic inputs of both domains to represent the same things in the output space. Feature-space alignment can be achieved using methods minimizing the divergence measure by using adversarial training or by using self-supervised approaches.

Lastly, **output space domain adaptation** methods avoid the complexity of dealing with the latent space and try to achieve a distribution alignment in the output space [174], e.g. in the pixel-wise segmentation

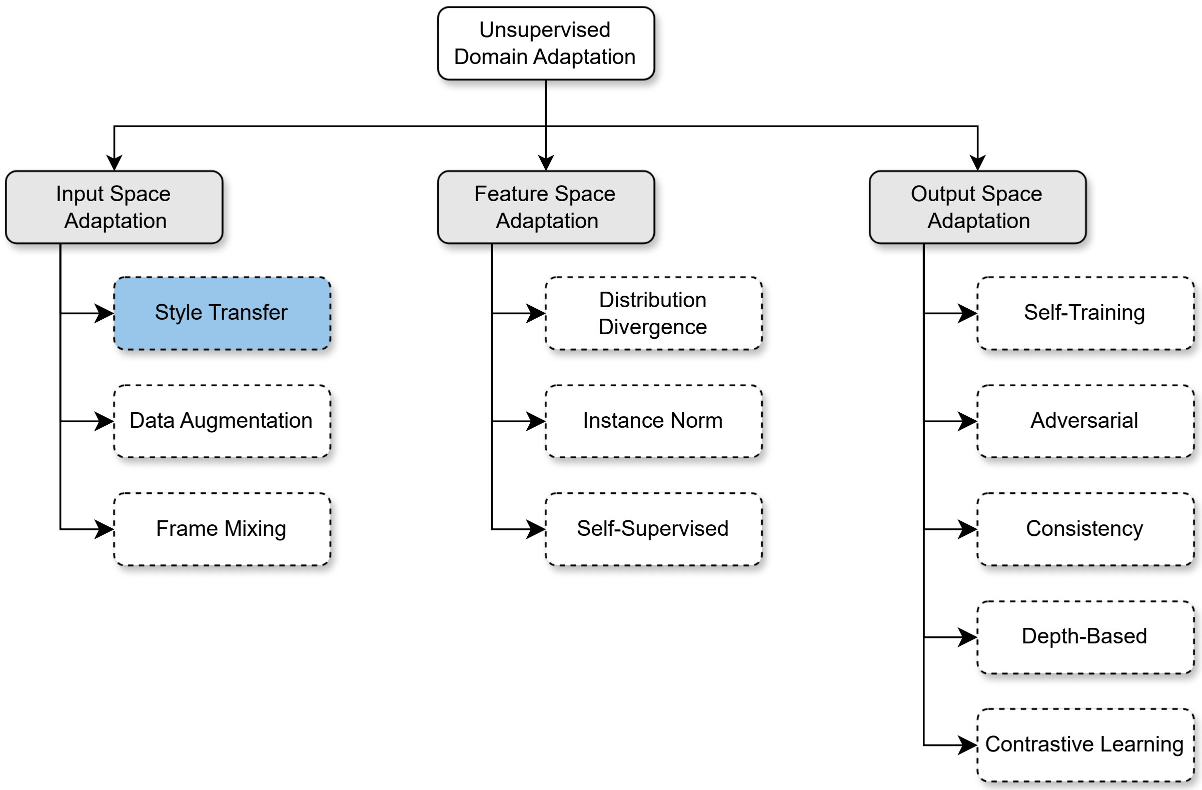


Figure 3.5: Adaptation spaces of unsupervised domain adaptation according to [96]. Style transfer is highlighted as it is the method of choice for LiDAR 3D domain adaptation in this thesis.

predictions. This is mainly done using adversarial learning strategies or iterative self-training using generated pseudo-labels [96].

A further distinction in the field of domain adaptation can be made concerning the semantic classes C in the source ($C_\mathcal{S}$) and the target dataset ($C_\mathcal{T}$) that are learned during the training process. Here, the main forms are closed-set domain adaptation and open-set domain adaptation, whereas closed-set is more common in research according to [96]. In **closed-set domain adaptation**, the classes within the source and target domains are exactly the same; e.g., both datasets contain annotated samples of cars, pedestrians, and bicycles ($C_\mathcal{S} = C_\mathcal{T}$). In **open-set domain adaptation**, however, the classes can differ; for example, the target dataset can additionally contain samples of trucks ($C_\mathcal{S} \subset C_\mathcal{T}$), and this class is not existing in the source dataset. Toldo et al. [174] further introduce partial domain adaptation, open-partial domain adaptation, and boundless domain adaptation. Partial domain adaptation describes the form in which all possible classes are part of the source dataset, but only a subset of these classes are relevant for the target dataset ($C_\mathcal{S} \supset C_\mathcal{T}$). Open-partial domain adaptation is a combination of open-set and partial domain adaptation, which means that some classes are only present in the source dataset and other classes are only present in the target dataset, and some classes are present in both datasets ($C_\mathcal{S} \neq C_\mathcal{T}$ and $C_\mathcal{S} \cap C_\mathcal{T} \neq \emptyset$). Boundless domain adaptation is similar to open-set domain adaptation, but all target domain classes are individually learned C_{learned} ($C_\mathcal{S} \subset C_\mathcal{T}$ and $C_{\text{learned}} = C_\mathcal{S} \cup C_\mathcal{T}$).

Other methods closely related to domain adaptation include domain randomization or data augmentation. The goal of domain randomization is to create multiple stylized versions of the source domain which is used during training, to make the task-specific model generalize better during inference on the target domain, as it is not biased towards a single source domain [97]. One way to achieve this for object detection is to insert a variety of objects with different styles into the original image or point cloud, as in [99, 177, 178] for image-based object detection. An advantage of domain randomization is that it does not need access to data from the target domain during training. Data augmentation for domain adaptation is similar to domain randomization in the sense that objects are inserted into existing frames, but in the case of data augmentation, data from other domains are inserted into frames of the target domain. This also increases the number of domains seen during training but still requires annotated target data. For example, Abu Alhaija et al. [179] developed a pipeline to render synthetic objects and augment real-world camera images using the synthetic objects rendered to create a robust object detection network for the application of AD.

3.2.2 LiDAR Domain Adaptation using 3D Point Clouds

The previous Subsection 3.2.1 provided an in-depth overview of the formulation of the domain adaptation problem and explained the specific taxonomy in this area. These formulations originate mainly in the area of image- or vision-based domain adaptation. However, these are still valid for different sensor modalities, such as LiDAR. In this subsection, the specifics for LiDAR-based domain adaptation are explained and the differences between the LiDAR-based methods compared to camera-based domain adaptation are highlighted. The goal is to give a brief overview of the general research landscape of LiDAR domain adaptation and to highlight research trends in this area. Furthermore, the specific field of domain mapping is explained in detail, which should serve as a baseline for the remainder of this thesis, in which domain mapping plays a crucial role. A focus will be on methods designed for the domain adaptation regarding the sim-to-real domain shift, as this type of domain shift is also the primary focus of this thesis. Ultimately, this subsection should cover state-of-the-art research in the area of LiDAR domain adaptation for AD to derive the research gap based on the state of the art in the subsequent Chapter 4.

Overview

All methods in the area of LiDAR domain adaptation can be classified as unsupervised domain adaptation methods [94], as introduced in Figure 3.5 in Subsection 3.2.1. This means the domain adaptation for LiDAR perception can be achieved by either adapting point clouds before passing them into the perception network (input space adaptation), by adapting the LiDAR perception network to align source and target distributions in the feature space (feature space adaptation), or by adapting the output space (output space adaptation), e.g. the bounding box labels. To specifically categorize LiDAR-based domain adaptation methods, Triess et al. [94] introduce four categories (Figure 3.6). These include domain-invariant feature learning, normalization statistics, domain-invariant data representation, and domain mapping, while the latter three categories originally were defined by Wilson et al. [135] and are data-driven, and the first method is model-driven and belongs to the feature space adaptation category. In the following, the approaches within these categories are presented. An overview of all approaches, their taxonomy, and application can be found in Table 3.1.

Domain-Invariant Feature Learning

Domain-invariant feature learning techniques are located in the category of feature space domain adaptation, characterized by being the only LiDAR-based unsupervised domain adaptation methods directly modifying perception networks. These approaches aim to align the features extracted by the perception network feature encoder between the different domains. Hence, the extracted features should be domain-invariant, and source and target domain features should follow the same distribution. As these features are part of the perception network, these are not hand-crafted but rather learned during the training process. The assumption is that if a perception network performs well using the source domain during training, the same trained network should also perform well using target data as input, as the intermediate features are domain-invariant and hence the origin of the data is indistinguishable for the detection or segmentation network head. In general, there exist

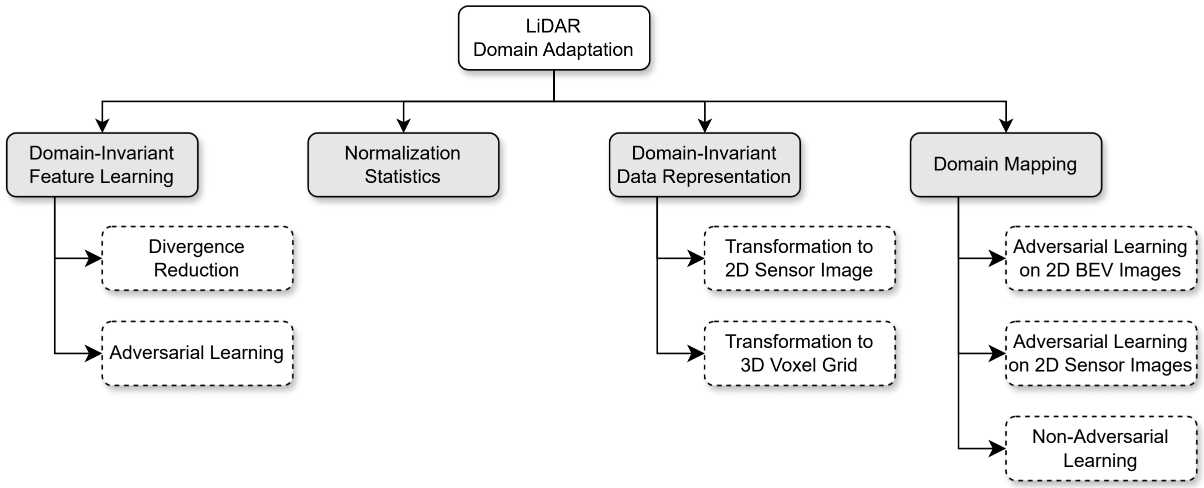

LiDAR Domain Adaptation
Domain-Invariant Feature Learning
Normalization Statistics
Domain-Invariant Data Representation
Domain Mapping
Divergence Reduction
Adversarial Learning
Transformation to 2D Sensor Image
Transformation to 3D Voxel Grid
Adversarial Learning on 2D BEV Images
Adversarial Learning on 2D Sensor Images
Non-Adversarial Learning

Figure 3.6: Categorization of LiDAR-based domain adaptation methods according to [94].

two methods to accomplish a feature alignment in the perception network. One is based on the reduction of a divergence measure, and the second is based on adversarial learning.

The first method aims to reduce the divergence measure of the features. Wu et al. [158] propose the network SqueezeSegV2, an improved version of SqueezeSeg [194], a DL model for point cloud segmentation. SqueezeSegV2 is resilient to dropout noises present in point clouds, a common challenge in the sim-to-real domain shift. SqueezeSegV2 achieves this by combining three major components in a domain adaptation training pipeline: progressive domain calibration, geodesic correlation alignment, and learned intensity rendering. Based on the geodesic correlation alignment proposed by Morerio et al. [195], the authors of SqueezeSegV2 synchronize the batch statistics of the source and target domain data using geodesic correlation alignment, which are synthetic and real-world data in their case, and constrain the network to learn a domain-invariant feature representation. In detail, they calculate the geodesic distance between the output distributions of a synthetic data batch and a real-world data batch and use this geodesic distance as a loss to penalize the discrepancy between the source and target data. The authors note that the Euclidean distance could also be used instead of the geodesic distance; however, they chose the geodesic distance as it takes into account the manifold curvature. Another similar approach is cross-model UDA (xMUDA) [156], which leverages multimodal data from 2D camera images and 3D LiDAR point clouds to reduce the day-to-night, country-to-country, and dataset-to-dataset domain shifts. This semantic segmentation architecture consists of two parallel branches that transfer knowledge at the prediction level. A 2D branch predicts the pixel-wise semantic classes using images as input, and a 3D branch predicts the point-wise semantic classes using point clouds as input. Knowledge transfer across modalities is done using the Kullback–Leibler divergence as part of the overall loss function, with the objective for the predictions of the 2D branch to estimate the predictions of the 3D branch and vice versa. Similarly to align features to reduce the weather-to-weather domain shift, 3D-CoCo [180] presents a contrastive instance alignment method to enforce domain-invariant features, which aligns the feature centroids of similar sample clusters independent of the source or target domain.

The second method for domain-invariant feature learning employs discriminators to enforce networks to learn domain-invariant features adversarially. LiDARNet [154] is a LiDAR domain adaptation model for semantic segmentation using a two-branch structure to extract domain-shared and domain-unique features using discriminators. Using these extracted features, the model is induced to learn a mapping between both domains and, hence, to reduce the domain shift. The authors state that the model not only increases the performance in the target domain but also keeps almost the same performance in the source domain after adaptation; hence, it can be applied in both the source and target domains. Wang et al. [181] explore

Table 3.1: Taxonomy of Domain Adaptation Methods for LiDAR Perception. DIFL: Domain-Invariant Feature Learning, NormStat: Normalization Statistics, DIDR: Domain-Invariant Data Representation, DM: Domain Mapping.

Method	**Paper**	**2D/3D**	**Domain Shift**	**Task**	**Technique**
DIFL	*SqueezeSegV2* [158]	3D	Sim-to-Real	Segmentation	Divergence Measure Reduction
	xMUDA [156]	3D	Day-to-Night Country-to-Country	Segmentation	Kullback–Leibler Divergence
	3D-CoCo [180]	3D	Sensor-to-Sensor	Detection	Contrastive Instance Alignment
	LiDARNet [154]	2D	Sim-to-Real	Segmentation	Adversarial Learning
	Wang et al. [181]	3D	Range-to-Range	Detection	Adversarial Learning
	DeBortoli et al. [182]	3D	Sim-to-Real	Detection	Adversarial Learning
	SRDAN [183]	3D	Sim-to-Real Day-to-Night	Detection	Adversarial Learning
NormStat	Wang et al. [106]	3D	Country-to-Country Dataset-to-Dataset	Detection	Statistical Normalization
DIDR	Triess et al. [116]	2D	Sensor-to-Sensor Dataset-to-Dataset	Segmentation	Up-Sampling 2D Sensor Image
	Shan et al. [184]	2D	Sim-to-Real Sensor-to-Sensor	Occupancy Mapping	Up-Sampling 2D Sensor Image
	Elhadidy et al. [152]	2D	Sensor-to-Sensor	Segmentation	Up-Sampling 2D Sensor Image
	Alonso et al. [155]	2D	Sensor-to-Sensor	Segmentation	Scan Line Dropping 2D Sensor Image
	PiLaNet [149]	3D	Sensor-to-Sensor	Segmentation	3D Voxelization
	Yi et al. [153]	3D	Sensor-to-Sensor	Segmentation	3D Voxel Surface Completion
	SPG [185]	3D	Dataset-to-Dataset Weather-to-Weather	Detection	3D Voxel Point Generation Module
DM	Saleh et al. [186]	2D	Sim-to-Real	Detection	CycleGAN 2D Bird's-Eye-View
	Sallab et al. [187]	2D	Sim-to-Real	Detection	CycleGAN 2D Bird's-Eye-View
	Sallab et al. [188]	2D	Sim-to-Real	Detection	CycleGAN 2D Bird's-Eye-View
	DUSty [189]	3D	Sim-to-Real	-	Adversarial Dropout Learning on 2D Sensor Images
	ePointDA [157]	3D	Sim-to-Real	Segmentation	Adversarial Dropout Learning on 2D Sensor Images
	Alonso et al. [155]	2D	Sensor-to-Sensor	Segmentation	Data Alignment using Geometric Shifts
	Langer et al. [150]	2D	Sensor-to-Sensor	Segmentation	Raycasting on SLAM-generated Map
Other	*SF-UDA3D* [190]	3D	Dataset-to-Dataset	Detection	Temporal-Consistent Self-Training
	ST3D [191]	3D	Dataset-to-Dataset	Detection	Self-Training
	FAST3D [192]	3D	Dataset-to-Dataset	Detection	Temporal-Consistent Self-Training
	You et al. [193]	3D	Dataset-to-Dataset	Detection	Temporal-Consistent Self-Training

a cross-range adaptation for 3D object detection in LiDAR point clouds, focusing on adapting far-range observations to near-range to optimize detection performance. This architecture also employs a discriminator for adversarial learning of global feature alignment to learn domain-invariant features, which implies that the features are range-invariant in the context of cross-range adaptation. In detail, a feature discriminator aligns the features in such a way that features from far-range objects have a distribution similar to that of features from near-range objects. Quantification of their method is done by training BEV object detectors, including feature alignment discriminators. It should be noted that this is a special case of domain adaptation, as it requires access to source (near-range) and target (far-range) annotations. Debortoli et al. [182] employ an adversarial discriminator during training to align the features of a 3D feature extractor for object detection on real-world point clouds. This application is targeted toward the sim-to-real domain shift found in robotic environments, such as search and rescue robots. A similar approach is SRDAN [183].

Normalization Statistics

Normalization techniques are typically used in vision-based neural networks to address the internal covariate shift to improve training convergence, stability, and speed. These techniques can also be applied to LiDAR-based networks to improve feature extraction in point clouds. In particular, these comprise batch normalization [196] and its adaptations such as adaptive batch normalization [197], group normalization [198], instance normalization [199], or batch-instance normalization [200] to align the data distributions in the network layers. In image-related tasks, these methods are also used for domain adaptation. Here, the idea is that normalization can separate domain and task knowledge in images, leading to style-invariant representations. However, Triess et al. [94] highlight that the findings of using normalization for vision-based domain adaptation can not be transferred to LiDAR-based domain adaptation. For example, Rist et al. [151] discuss that simply using adaptive batch normalization in LiDAR object detection does not improve detector performance when tested in a sensor-to-sensor domain adaptation setup. Another variant of using statistics to improve model generalization is presented by Wang et al. [106]. The authors use statistical normalization by calculating the mean car sizes of the source and target domains and adding the difference in mean size to the source bounding boxes. Different dataset-to-dataset experiments show that applying statistical normalization can reduce the domain shift. However, this approach needs access to target data to calculate the mean object sizes or other sources to obtain the data, such as local vehicle authorities of the respective target area or car-selling websites.

Domain-Invariant Data Representation

Domain-invariant data representation refers to the input data of the perception network; therefore, these methods belong to the input space adaptation approaches. Essentially, the goal is to find a representation of the input point clouds in which the perception network can not distinguish between the source and target data. These data preprocessing algorithms can transform 3D point clouds into a 2D sensor image or a 3D voxel grid, as introduced in Section 2.2.

In a sensor-to-sensor domain adaptation setup, which mainly focuses on the different sensor resolutions, 2D sensor images captured from different sensors can be easily aligned by upsampling or dropping scan lines. For example, a 2D sensor image of two LiDAR sensors with different vertical resolutions can be aligned by upsampling the horizontal rows of the 2D sensor image of the lower resolution LiDAR or by dropping certain horizontal rows of the 2D sensor image of the higher resolution LiDAR. Triess et al. [116] developed a CNN-based residual upsampling network to synthetically increase the vertical resolution of 2D sensor images by a factor of two. This network can be used to upsample LiDAR point clouds captured with a 32-layer sensor to a 64-layer equivalent and use the upsampled source data to train a perception network to reduce the sensor-to-sensor domain shift for a 64-layer sensor as the target application. Shan et al. [184] follow a similar approach of vertically upsampling of 2D sensor images up to four times, based on the widely used U-Net network architecture [201], and evaluate their architecture using simulated data from CARLA. Elhadidy et al. [152] perform a similar type of domain adaptation for the semantic segmentation of point clouds using 2D sensor images as a domain-invariant data representation. Their network is capable of upsampling either eight-, 16-, or 32-layer 2D sensor images to a 64-layer sensor image, which is, in turn, used to train the semantic segmentation network RangeNet++ [202] and compared to RangeNet++ trained with the original 64-layer 2D sensor images. The results of all three upsampling methods show an increase in the mean IoU for semantic segmentation [116, 152] compared to the baseline methods or a reduction in the $\mathcal{L}1$ error for occupancy mapping [184]. Instead of upsampling the vertical resolution of 2D sensor images, Alonso et al. [155] align the domains by reducing the number of vertical layers of the sensor with the higher number of layers to have the same number as the other sensor. They evaluated their approach using a semantic segmentation network and calculated the mean IoU for three different target datasets. The results show that intuitively the higher the initial deviation of the number of LiDAR layers between the source and target dataset, the higher the increase in mean IoU after aligning the number of layers. Furthermore, the results show that compared to other data alignment strategies analyzed by [155], this type of domain-invariant data representation has one of the highest impacts on segmentation performance.

The second form of domain-invariant data representation is the voxelization of 3D point clouds. Here, the point cloud is discretized using rectangular cuboids. Therefore, domain-specific characteristics can be removed while maintaining the semantics of the scene. Some work uses this representation to solve the domain adaptation challenge. For example, based on VoxelNet [56] and PointPillars [71], PiLaNet [149] is a pillar-based labeling network for semantic segmentation of point clouds using the regularity of voxel grid for cross-sensor portability. The authors analyze the sensor-to-sensor domain adaptation performance of their network by training and evaluating the network with datasets from a 32-layer and a 128-layer LiDAR sensor. The results show that their network achieves a better cross-sensor domain adaptation performance than an approach using a 2D sensor image (LiLaNet [203]) in a comparable setting. However, the results of segmentation are expressed in terms of mean IoU drops by more than 16 % when evaluating the opposite sensor dataset, showing that this domain-invariant data representation approach can not fully eliminate the sensor-to-sensor domain shift. The authors note that this performance drop might be related to their backbone CNN, which is not optimized to handle varying densities of the input data. The cross-sensor domain shift also is targeted using 3D geometric priors as a domain-invariant data representation in the work of [153]. The authors present a domain adaptation approach that leverages geometric priors to convert the problem of domain adaptation into a task of completing a 3D surface. Their approach consists of a sensor-specific sparse voxel completion network that transforms the point cloud into a domain-invariant voxel representation and a subsequent perception network to perform a perception task, which is semantic segmentation in their case. The sparse voxel completion network retrieves the underlying 3D surfaces from sparse LiDAR voxels originating from various sources, and the subsequent sensor-agnostic semantic segmentation network categorizes the restored 3D surfaces. The authors argue that moving the domain adaptation problem from the perception network to the surface completion network is beneficial, as the latter can be trained self-supervised. Another domain-invariant data representation method using 3D voxel grids is SPG [185], which integrates a point generation module into a point cloud object detector. The goal of the point generation module is to generate domain-invariant object point clouds, which substitute the object point clouds at their initial location in the scene point cloud. The downstream object detection task utilizes these augmented scene point clouds.

To summarize, the advantage of the domain-invariant data representation methods is that they do not require a modification of the perception network, and hence the domain-adapted point clouds can be used by multiple perception networks. However, these data preprocessing algorithms are mostly based on hand-crafted methods, which limits their generalization. As shown by Piewak et al. [149], the sensor-to-sensor domain gap can not be fully eliminated, even though their 3D voxel approach achieved better results than comparable 2D sensor image-based approaches.

Domain Mapping

In the area of domain mapping, the goal is to transfer the data from the source domain to the target domain before passing the adapted data into the perception network for training. The idea is that the source domain data is transferred to appear like the target domain while maintaining the semantic meaning, and hence the labels, which now serve as pseudo-labels, are still identical to the adapted source data. Once the perception network is trained using the adapted source data, it can be applied in the target domain. Triess et al. [94] mention that domain mapping is mainly used to reduce the dataset-to-dataset or sim-to-real domain shift. Most methods in the area of domain mapping rely on adversarial networks, as domain mapping has its origin in image-based domain adaptation. For image-based domain adaptation, image-to-image translation is usually done pixel-wise using conditional GANs [204–207]. Similarly to image-based approaches, LiDAR-based domain mapping approaches also employ GANs to adapt point clouds.

For example, Saleh et al. [186] use the unmodified cycle-consistent GAN CycleGAN [176] to adapt synthetic point clouds from simulation to real-world point clouds. To use the image-based network CycleGAN, they transform 3D point clouds into 2D BEV images, as explained in Section 2.2. Instead of image-to-image translation with conditional GANs that requires paired images of the source and target data, CycleGAN is trained unsupervised and does not require paired images, so the source and target datasets do not need to be aligned. Therefore, Saleh et al. [186] use synthetic data from the Motion-Distorted LiDAR Simulation (MDLS) dataset [208] and real-world data from the KITTI dataset. They evaluated their domain mapping method using YOLOv3 [209] for object detection, which detects objects on the adapted 2D BEV point cloud images and reported AP. The results show a successful decrease in the sim-to-real domain shift using their adapted data to train YOLOv3 from previously $27.33\,\%$ to $22.48\,\%$ AP. They further analyze the performance of mixing real-world training data with adapted data, resulting in a $7.03\,\%$ AP performance increase over only using real-world data. Note that the number of training images from synthetic, adapted, and mixed data is higher than the number of training images from real-world data, which could also be attributed to the reduction of domain shift. Nevertheless, as synthetic data is inexpensive to generate and adapt, this approach shows the potential to use domain adaptation for LiDAR object detection. This can be underlined by similar work of [187] and [188]. The authors follow the same approach of using CycleGAN to adapt BEV images of point clouds from a synthetic source dataset to a real-world target style. In their experiments, they use CARLA as a synthetic dataset and KITTI as a real-world dataset. The evaluation of [187] is mainly focused on qualitative visual inspection of the adapted point cloud images, and the degree of sim-to-real domain shift reduction is not quantified. However, the qualitative assessment shows promising results of their approach. Sallab et al. [188] conduct a quantitative evaluation similar to [186] by training the custom object detection algorithm oriented YOLO, based on YOLO3D [160], and report the mean AP for different configurations of the training dataset. They do not explicitly evaluate the performance of using adapted simulated data only for training the object detector but mix the adapted data with the real-world data to analyze the performance gain of adding adapted simulated data to real-world data. The mix of adapted and real-world data training is indeed increasing the performance compared to using real-world data only or using mixed simulated and real-world data.

Instead of using the lossy projection of point clouds on 2D BEV images to employ image-based CycleGAN, other works use the lossless projection of point clouds to 2D sensor images, which is also used by domain-invariant data representation domain adaptation methods. However, instead of using hand-crafted methods to align the 2D sensor images of the source and target domains, adversarial networks are utilized by domain mapping methods to learn the correspondence between the source and target 2D sensor images. Based on the work of [210] using a GAN architecture to generate 2D sensor image LiDAR data, DUSty [189] extends this framework and incorporates a learnable point dropout mask, which should mimic the point dropout present in the real-world target dataset. Similarly, ePointDA [157] learns the point dropout for the domain mapping of the simulated source data to the real-world target data. Their architecture is also based on a GAN structure that uses 2D sensor view images as input and output. Evaluation using a semantic segmentation perception network shows that learnable dropout has the greatest influence on mean IoU compared to other parts of the ePointDA framework targeting sim-to-real domain shift reduction, that is, feature alignment and a transferable segmentation model.

Alongside adversarial domain mapping methods, Alonso et al. [155] and Langer et al. [150] present non-adversarial approaches. Alonso et al. [155] present multiple data alignment strategies, such as removing scan lines as discussed in domain-invariant data representations. They also employ other simple strategies in the area of domain mapping, such as xyz shifts to align sensor locations, per-class augmentation, and using only relative rather than absolute distances of the input data. As already discussed in domain-invariant data representations, the dropping of scan lines has the greatest potential to reduce the domain shift, but the additional domain mapping strategies also reduce the domain shift to an extent. For non-adversarial domain mapping in a sensor-to-sensor setup, Langer et al. [150] fuse sequential LiDAR scans to build a 3D map using SLAM [211], in which they later generate semi-synthetic scans using raycasting. These semi-synthetic scans are projected into 2D sensor images for the task of semantic segmentation.

To summarize, the best approaches in domain mapping methods use adversarial learning by either adapting 2D BEV or 2D sensor image projections, whereas the former can not be re-projected into 3D space and the latter do not make use of the geometric correlations between points in 3D space.

Other Methods

Other methods not classified in the former categories are, for example, self-training approaches, such as source-free unsupervised domain adaptation for 3D object detection (SF-UDA3D) [190], an unsupervised domain adaptation method that does not require target and source annotations. This architecture leverages the temporal consistency of pseudo-labels generated by the 3D object detection network PointRCNN [68] pretrained on the unavailable source dataset. The idea is that the predictions made by the network in subsequent target point clouds can be tracked by a tracking algorithm and should be consistent. Using these predictions, the target dataset can be annotated in a self-training manner and fine-tuned using the pseudo-labeled target data to increase object detection performance in the target domain. Similar self-training approaches using pseudo-labels generated by source-trained networks include ST3D [191], FAST3D [192], and [193]. Lastly, MLC-Net [212] utilizes a teacher-student paradigm with multi-level consistency to generate reliable pseudo-targets.

4 Problem Statement

This chapter summarizes the findings of the literature review by deriving the research gap in Section 4.1 and further formulates the research questions in the area of LiDAR domain adaptation based on the open challenges in Section 4.2. In addition, the overview of the methodology of this thesis is presented in Section 4.3.

4.1 Conclusions and Research Gap Derived from Related Work

Chapter 3 discussed the related work in the area of LiDAR domain adaptation by breaking down this topic into two parts. First, the importance and motivation of LiDAR domain adaptation are highlighted by discussing the topic of LiDAR domain shift, its cause, impact, and quantitative and qualitative measures. Second, domain adaptation approaches are categorized independently of the sensor modality, as most of the works in this area are focused on camera domain adaptation. Then, the specific approaches in the limited research area of LiDAR domain adaptation are presented and detailed. Based on these discussions in these two areas and the conclusions drawn in this section, the research gap will be derived in the following. The purpose of this section is to highlight the shortcomings and open research topics in these areas and to pave the way for the formulation of the relevant guiding research questions for this thesis in the next Section 4.2.

In the area of LiDAR domain shift, the focus is on quantitative and qualitative measures to evaluate the different forms of domain shift presented. [94] emphasize that the dataset-to-dataset domain shift is the most considered form of domain shift, but it involves a combination of individual domain shifts, such as a combination of country-to-country and sensor-to-sensor domain shifts. Another example of combined domain shifts is the sim-to-real domain shift, which not only includes a simplified virtual sensor model and hence a sensor-to-sensor domain shift, but also different environments and agents from the real-world target domain, hence a superimposed country-to-country domain shift. This makes it difficult to compare different domain adaptation approaches, as they usually only target a specific form of domain shift. To isolate the influence of the sensor-to-sensor domain shift in a sim-to-real setting, a dataset created in a simulated environment with a minimal country-to-country domain shift is necessary. A dataset fulfilling these requirements is currently not available. Publicly available synthetic datasets are limited to Grand Theft Auto V, Virtual KITTI, and CARLA datasets and environments. Grand Theft Auto V and CARLA datasets are based on artificial landscapes and do not replicate existing real-world environments. Virtual KITTI mimics the real-world KITTI dataset. However, it does not include LiDAR point clouds and is focused on camera images only. Quantitative evaluation of domain shift is done mainly by training and evaluating perception networks on source and target data. However, a weakness in related work regarding domain shift evaluation is the qualitative measure. Here, in addition to t-SNE for high-dimensional feature space visualization, researchers focus on visual inspection of entire scene point clouds, without analyzing local structures, which is the crucial adaptation space, especially for the sim-to-real, sensor-to-sensor, or weather-to-weather domain shift.

In the research field of LiDAR domain adaptation, there are only a few approaches compared to the field of camera domain adaptation. The taxonomy of these approaches in Table 3.1 shows that most of these approaches are either approaches that adapt only a 2D representation of point clouds or approaches are directly integrated into the perception network, i.e. domain-invariant feature learning approaches. The former has the drawback that 3D point clouds can not be recovered or 3D perception networks can not be applied. The latter has the drawback that these approaches can only be applied to specific perception networks or perception tasks, which is not scalable for the usage and further development of these networks. Also, each new update of the perception network requires an adaptation and validation of the domain-invariant feature learning approach. Domain mapping approaches working with 3D data offer the greatest flexibility in terms of scalability for domain adaptation of perception networks, but the only two approaches that fulfill these requirements are DUSty [189] and ePointDA [157]. Both approaches are also focused on the specific topic of analyzing the learning of dropout in synthetic point clouds, which is a special case of sim-to-real domain shift. Furthermore, both approaches make use of 2D sensor images, which allow the lossless recovery of 3D point clouds, but do not make use of the 3D geometric local dependencies between neighboring points. Another shortcoming in the related work of sim-to-real domain adaptation is that the impact of simple methods potentially increasing the performance of simulation, such as integrated sensor noise models in simulation, are not studied in detail.

These conclusions from related work allow for a precise formulation of research gaps:

- First, an annotated dataset with both synthetic and real-world point clouds in the same environments with the same agents is not available to analyze the sim-to-real domain shift systematically.
- Second, no systematic analysis of the sim-to-real domain shift includes a study of the impact of simulation improvements.
- Third, as also [94] emphasizes, none of the adversarial domain mapping approaches is capable of adapting realistic 3D point clouds for LiDAR domain adaptation that bridges the sim-to-real domain shift directly in the 3D space.

4.2 Research Questions

Following the research gaps highlighted in the previous section, this section serves to formulate the guiding research questions. The literature does not provide a conclusive approach to solve the 3D LiDAR domain adaptation challenge for a sim-to-real domain shift. The goal of this thesis is to explore and develop a method capable of solving this open research topic by answering the primary research question (PRQ):

PRQ: How can domain adaptation approaches directly applied to 3D point clouds bridge the sim-to-real domain shift to enhance LiDAR perception in autonomous vehicles?

Here, domain adaptation refers to the general type of method and focus of this thesis. The term sim-to-real domain shift is used to describe the particular type of domain shift that this thesis is investigating. 3D point clouds refer to the aspect of using and adapting 3D point clouds directly instead of 2D projections. LiDAR perception refers to the specific application considered in this thesis, whereas the primary focus is 3D object detection, but the method should be applicable to other LiDAR perception tasks.

To answer the primary research question, it is further subdivided into detailed research questions Q1 and Q2 addressing the domain shift analysis in Chapter 5. Q3 to Q5 are addressing the domain adaptation in Chapter 6 to Chapter 8.

Q1: How should a sim-to-real LiDAR dataset be designed to effectively capture the nature and extent of the sim-to-real domain shift, and what methodologies can be employed to quantify and analyze this shift through various performance indicators?

Using the developed method for domain shift analysis and quantification to answer this research question, the domain shift from simulation to reality in the created dataset needs to be quantified, which serves as a baseline for further research questions.

Q2: What impact do simulation enhancements, such as the integration of sensor noise models, have on mitigating the sim-to-real domain shift in 3D LiDAR data?

Q2 addresses the research gap related to the influence of simulation improvements.

Q3: How can LiDAR domain adaptation techniques be applied at the object-level to mitigate the impact of sim-to-real domain shift, and what are the specific challenges and solutions associated with object-level domain adaptation?

Q3 should introduce a domain adaptation technique that utilizes adversarial learning and adapts 3D point clouds, aiming to fill the existing research gap in this field. In Q3, object-level refers to the perception scope, which are the labeled objects for the object detection task.

Q4: In what ways does the scene-level LiDAR domain adaptation address the broader challenges posed by the sim-to-real domain shift compared to object-level domain adaptation?

Q5: How can LiDAR domain adaptation techniques be developed and applied when source and target data distributions are not aligned?

Q4 and Q5 extend Q3 to apply the developed domain adaptation method at the scene-level or on point clouds from non-distribution-aligned data. The research questions formulated will be addressed throughout this work and revisited as part of the discussion in Section 9.1.

4.3 Methodology Overview

The formulation of the research questions paves the way for describing the methodology for how these questions are answered in the following chapters. First, Chapter 5 addresses Q1 by presenting a method to quantify the sim-to-real domain shift. This is achieved initially by introducing a novel dataset that combines synthetic and real-world data, which is utilized to evaluate the domain shift and to conduct a more detailed qualitative analysis of the domain shift at a local level. This chapter further answers the research question Q2 on how sensor model noise in simulation influences the sim-to-real domain shift. Chapter 6 addresses the research question Q3 by presenting a LiDAR domain adaptation method that can adapt synthetic 3D point clouds of objects to their real-world counterparts. The adapted point clouds, which closely resemble the real world, are employed to train 3D object detection algorithms and reduce the sim-to-real domain shift on target real-world point clouds. Chapter 7 and Chapter 8 extend this domain adaptation approach to answer the primary research question PRQ. In detail, Chapter 7 incorporates the adaptation of entire scene point clouds to answer Q4, and Chapter 8 extends the method to a publicly available, non-distribution-aligned real-world dataset, showing the generalization capabilities of the domain adaptation approach and answering Q5. The discussion in Chapter 9 critically reviews the method, the implementations, and the results, and concludes the thesis by answering the research questions and highlighting the directions for future work.

5 Analysis of the LiDAR Sim-to-Real Domain Shift

In this chapter, a detailed analysis of the LiDAR sim-to-real domain shift is presented. The methodology followed is described in Section 5.1, which further outlines the remainder of the chapter. The content of this chapter is based on and adapted from [213].

5.1 Method

This chapter presents an approach to systematically analyze the LiDAR sim-to-real domain shift by extending the related work in the field of LiDAR domain shift analysis discussed in Section 3.1. The goal of this chapter is to address and answer the research questions Q1 and Q2.

Q1: How should a sim-to-real LiDAR dataset be designed to effectively capture the nature and extent of the sim-to-real domain shift, and what methodologies can be employed to quantify and analyze this shift through various performance indicators?

Q2: What impact do simulation enhancements, such as the integration of sensor noise models, have on mitigating the sim-to-real domain shift in 3D LiDAR data?

These questions will be answered by first describing the general methodology in this section before delving into the specific details in the following sections.

The general methodology is as follows. The analysis of the domain shift is separated into quantitative and qualitative evaluation and extends the state of the art in both areas.

For quantitative evaluation, the method is based on the idea of using task-specific perception networks to directly measure the impact of using source datasets from different domains to train the perception network, similar to the methods presented in [116, 147–158] for semantic segmentation or in [159, 161, 162] for object detection. In contrast to these methods using non-matching datasets and hence combining multiple forms of LiDAR domain shift, this thesis presents a domain shift analysis method using a scenario- and distribution-aligned real-world and simulated dataset to isolate the sim-to-real domain shift. The process of generating this synthetic and real-world dataset that is used to analyze the sim-to-real domain shift in an isolated manner without superimposing other domain shift forms is explained in Section 5.2. In a second step, the specific perception networks used for domain shift quantification are introduced in Section 5.3. This is followed by proposing the specific Key Performance Indicators (KPIs) used for the quantitative and qualitative evaluation of the domain shift using the presented networks.

Qualitative evaluation is performed using t-SNE to visualize the high-dimensional feature space of the perception networks as in [44, 100, 121–123, 126–129]. Additionally, this thesis adds a contribution to the state of the art by presenting a qualitative evaluation based on visual inspection at the object-level, i.e. analyzing the relevant targets of the perception task.

Once the basics of the analysis approach are presented, the results and in-depth analysis at the local object-level are presented in Section 5.5.

5.2 Dataset Generation

The analysis of the sim-to-real domain shift necessitates a pair of datasets, one containing real-world data and the other containing simulated data. The goal of the dataset is to minimize the discrepancy between the real-world and simulated data and to isolate all possible forms of domain shift despite the sim-to-real domain shift. This means that the same environment, agents, scenarios, positional distributions, and sensor characteristics should be used in the real world and simulated counterpart environment.

To this end, data are recorded during the Vegas Autonomous Challenge, the inaugural head-to-head autonomous race involving international university teams, as part of the IAC series [9, 214]. The race track is the Las Vegas Motor Speedway, a $2.4\,\mathrm{km}$ tri-oval, depicted in Figure 5.1. The race cars are Dallara AV-21 (Figure 5.2a), which are Dallara IL-15 race cars that have been adapted and fitted with the necessary sensors for autonomous operation, such as cameras, LiDAR sensors, RaDAR sensors, GNSS devices, IMUs, compute units, and actuation. The LiDAR configuration consists of three sensors, each with $120\,°$ horizontal FoV, placed with $120\,°$ horizontal orientation spacing to cover a full $360\,°$ area around the race car. The sensors offer a variable frame rate between one and $30\,\mathrm{Hz}$ but are fixed at $20\,\mathrm{Hz}$ for the trade-off between frequency and resolution. The sensor range is up to $250\,\mathrm{m}$ at $10\,\%$ reflectivity. The sensor driver outputs the point-wise xyz and intensity values and directly fuses the three individual point clouds to produce a single $360\,°$ point cloud per time step.

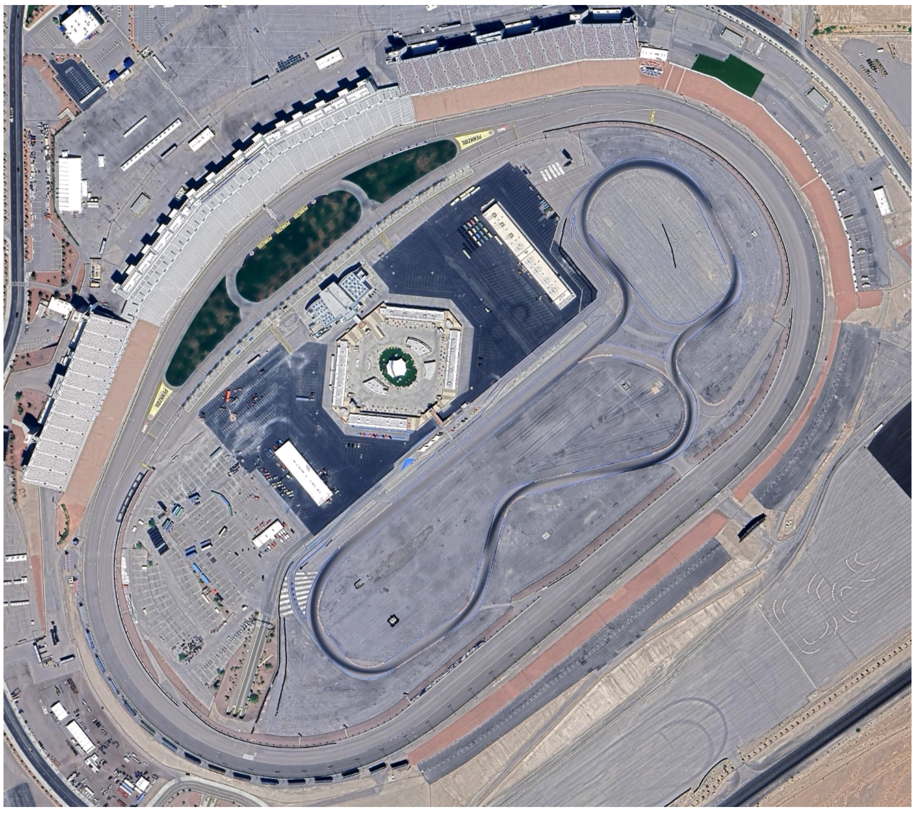
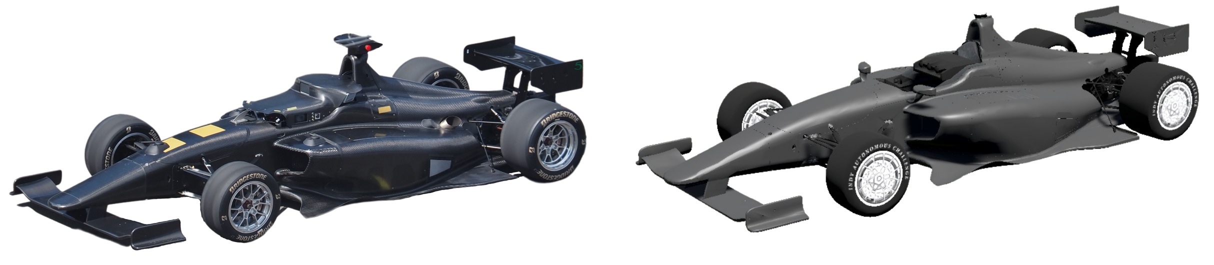


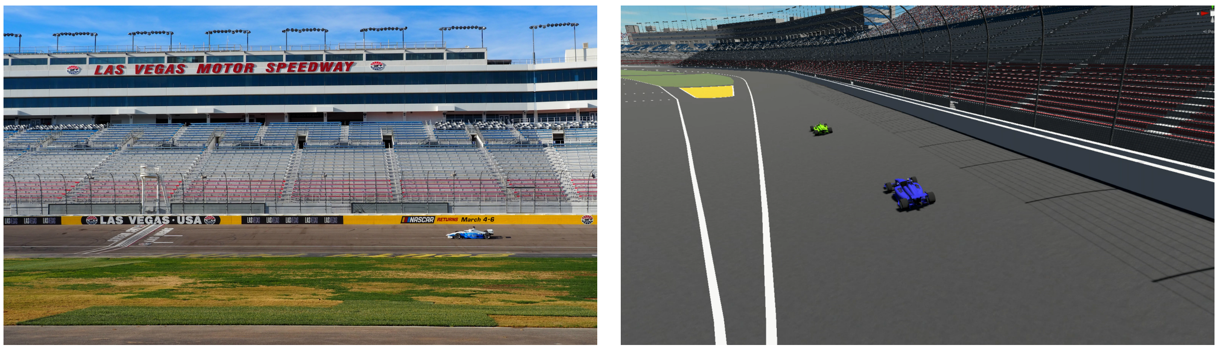
Figure 5.1: Tri-oval race track Las Vegas Motor Speedway used for real-world data collection (screenshot from [215]).



(a) Real-world race car. (b) Simulation model of the race car.

Figure 5.2: Race car AV-21 (from [213]).

During each test run and actual competition (Figure 5.3a), LiDAR sensor data is recorded along with GNSS data from all vehicles on the track. Recording of GNSS data from both ego vehicle and other agents is crucial for both auto-labeling the real-world data and re-simulating the scenarios in the 3D simulation environment. GNSS data includes the vehicle's position in latitude and longitude and the orientation and velocity of each vehicle, each recorded with a frequency of $20\,\mathrm{Hz}$.



(a) Real-world picture of the autonomous race car AV-21 on the race track. (b) Simulation based on the Unity Engine [84]. The scenario was previously captured in the real world and then replicated in the 3D simulation based on the captured trajectories of all agents on the race track.

Figure 5.3: Comparison of the race track Las Vegas Motor Speedway in real world and simulation. Data for the scenario- and distribution-aligned IAC dataset are recorded in both environments.

An auto-labeling pipeline is developed to label the recorded real-world point clouds [216]. For this purpose, the relative position and orientation of each agent toward the ego vehicle are calculated based on the GNSS data of both vehicles. The GNSS data of both vehicles are temporally not aligned as the sensors are not triggered at the same time, but given the $20\,\mathrm{Hz}$ recording frequency, the maximum time deviation between the measurements of both GNSS devices is $25\,\mathrm{ms}$. This equals $1.75\,\mathrm{m}$ traveled at a speed of $70\,\mathrm{m\,s^{-1}}$, a typical speed achieved during the final race. This error in label generation is corrected with a two-step refinement pipeline. First, GNSS tracks are interpolated to a common time step of $1\,\mathrm{ms}$, which is a valid approach, since the velocity of both vehicles is quasi-constant during each time interval of $25\,\mathrm{ms}$. This reduces the error at a speed of $70\,\mathrm{m\,s^{-1}}$ to $0.07\,\mathrm{m}$, which is in the magnitude of the root mean squared error of the Real-Time Kinematic Positioning (RTK)-corrected GNSS device, which is specified at $0.01\,\mathrm{m}$ [217]. The second label-refinement step analyzes the point distribution in and around the proposal of the bounding box of the first step and can move the bounding box up to $1\,\mathrm{m}$ in the x or y direction if the majority of points are outside the initial box proposal. The limit of $1\,\mathrm{m}$ is derived from empirical observations. The size of the bounding box is always constant with length $l_b = 4.88\,\mathrm{m}$, width $w_b = 1.90\,\mathrm{m}$, and height $h_b = 1.18\,\mathrm{m}$ since the vehicle type is identical throughout the entire dataset recording.

The same GNSS data are used to generate the scenario-identical digital twin of the real-world dataset in simulation. To reduce the potential sources of domain shift to a minimum, the simulator, which is built on the Unity Engine [84], incorporates a 3D representation of the race track (Figure 5.3b), a 3D model of the race car (Figure 5.2b), and a virtual model of the LiDAR sensor. The virtual model of the LiDAR sensor is a custom implementation and replicates the features of the sensor used on the real-world vehicle in terms of resolution, frequency, scan pattern, and range specifications [218]. It contains a model to calculate the point-wise intensity using the ray incidence angle and target material, but as this model is not validated with real-world data, only the xyz values are used. To minimize the sim-to-real domain shift caused by this limitation, the intensity channel of the real-world data is excluded as well. The final real-world and simulated datasets are composed by selecting every fifth point cloud of the entire data available, as following point clouds show a high similarity. This results in 6,000 point clouds per domain, divided into 4,000 point clouds for training, 1,000 for validation, and 1,000 for testing covering multiple laps on the race track. In the following chapters, this scenario- and distribution-aligned dataset is referred to as IAC dataset.

5.3 Neural Network Selection and Configuration

The networks used to analyze the sim-to-real domain shift are state-of-the-art LiDAR 3D object detection algorithms. As explained in Subsection 2.3.2, LiDAR object detection networks are categorized into point- or voxel-based approaches. To analyze whether the LiDAR sim-to-real domain shift is equally pronounced for both approaches, a network is selected from each category. For the point-based approach, PointRCNN [68] is selected, which was at the top of the KITTI 3D object detection benchmark at the time of selection. PointRCNN is a two-stage approach that extracts features on a local point-level using PointNet++ [45] as a backbone and subsequently predicts objects in 3D space. As a voxel-based object detection approach, PointPillars [71] is chosen, which also ranked similarly to PointRCNN on the KITTI 3D object detection benchmark. PointPillars is a one-stage approach that uses vertical pillars, each in which PointNet [44] is applied to extract pillar-wise features from points within the pillars and subsequently uses a 2D CNN backbone to predict 3D bounding boxes.

Both networks are used in their original configuration, with a single modification on the final output layer, which is reduced to only predict a single class, as the IAC dataset also contains only a single object type. Furthermore, the mean sizes of the anchors are fixed to the ground truth dimensions of the objects, that is, a length $l_b = 4.88\,\mathrm{m}$, a width $w_b = 1.90\,\mathrm{m}$, and a height $h_b = 1.18\,\mathrm{m}$. As the intensity channel of the virtual sensor model of the LiDAR in simulation is not validated, only the positional xyz values are used as input features. Despite the range of the LiDAR sensor in the real world of up to $250\,\mathrm{m}$, the point clouds are restricted to a horizontal extent of $100\,\mathrm{m}$. The reason is that at a distance of more than $100\,\mathrm{m}$, the number of points reflected by the relevant objects is too low to extract meaningful features. The networks are trained for 75 epochs for each training dataset, motivated by no further decrease in validation loss. Network training and evaluation are done using the DL library PyTorch [219], which uses non-deterministic functions. Therefore, every network undergoes training five times, and the mean and standard deviations of the selected metrics are reported, similar to [144].

5.4 KPIs for Evaluation

Before analyzing the performance of the trained networks, a statistical comparison is conducted based on the raw point cloud data from the simulated and real-world datasets. Therefore, simple metrics are calculated, such as the maximum and minimum extent of the point cloud in all dimensions, the maximum, minimum, and mean number of points in either the entire scene point cloud or only within the target boxes. This upstream statistical comparison should enable a coarse estimation of the similarity of the simulated and real-world datasets, before using them to train the object detection networks.

Once selected networks are trained using simulated or real-world datasets, their performance needs to be assessed using a suitable metric. Therefore, a conventional metric for evaluating object detection is chosen, that is, AP. The calculation of AP is based on the predictions of the object detection network, i.e. the 3D bounding boxes, by comparing these predictions with the ground truth data, i.e. the 3D box annotations. Each predicted bounding box is categorized as either TP or FP, depending on whether the 3D box's overlap IoU with a ground truth bounding box exceeds a defined threshold, e.g. 0.5 or 0.7. In the remainder of this work, the more strict IoU threshold 0.7 is used as the base evaluation threshold. Nevertheless, the IoU threshold 0.5 is reported together for comparison, as in [192, 193]. In case a ground truth box is not covered by a predicted bounding box, it is counted as a FN. Based on the number of TP, FP, and FN for all point clouds in the test dataset, the metrics precision p and recall r can be calculated as follows [220]

$$p = \frac{TP}{TP + FP} \tag{5.1}$$

and

$$r = \frac{TP}{TP + FN}. \tag{5.2}$$

Using the point cloud-wise calculated precision and recall values, the precision-recall curve can be generated. This precision-recall-curve is the baseline to calculate the AP, which is essentially an approximation of the area under the curve by averaging the precision values p_{interp} at specific recall values r [221] and is calculated as

$$AP|_R = \frac{1}{|R|} \sum_{r \in R} p_{\text{interp}}(r), \tag{5.3}$$

whereas the interpolation of the precision values is defined as

$$p_{\text{interp}}(r) = \max_{r':r' \geq r} p(r'). \tag{5.4}$$

In the original KITTI benchmark [36], eleven equal-spaced recall levels are used $R_{11} = \{0, 0.1, 0.2, \ldots, 1\}$. As [222] remark, starting with $r = 0$ can result in $AP|_{R11} = 1/11 \approx 0.0909$ for the entire dataset even if only a single prediction is TP. The authors suggest using a 40-point interpolation $R_{40} = \{1/40, 2/40, 3/40, \ldots, 1\}$ starting at $r = 1/40$ instead of an 11-point interpolation starting at $r = 0$. This thesis also adopts this suggestion to use $AP|_{R40}$, and all the following notations of AP refer to $AP|_{R40}$.

Furthermore, IoU is based on the 3D box overlap, so the notation is denoted 3D AP. Alongside the 3D AP, there is the BEV AP, which uses the 2D BEV projections of the bounding boxes to calculate the IoU. In this thesis, only 3D AP is considered, as the networks used predict the boxes in the 3D space and also the datasets used contain scenarios with elevated objects. The average precision is in the range of $0 \leq$ 3D AP ≤ 1 and is usually expressed in percent, whereas higher values are better. In the remainder of this thesis, the 3D AP, measured in terms of IoU thresholds of $70\,\%$ and $50\,\%$, is referred to as 3D AP (0.7) and 3D AP (0.5), respectively.

Besides the quantitative evaluation of the sim-to-real domain shift as proposed in related work, this thesis further analyzes the domain shift qualitatively by inspecting the point cloud visually on an object-level. That is, 3D renderings of the extracted object point clouds are compared in different poses, and also aggregated point clouds from different poses and ranges to cover the entire object are compared. The idea of this visual inspection is to analyze the differences in simulated and real-world data on a local level, which can provide valuable insights to LiDAR domain adaptation approaches. As explained in Section 5.1, the visualization method t-SNE is also used to qualitatively analyze the domain shift present in the high-dimensional feature space of networks.

5.5 Results

Based on the order of the proposed KPIs for evaluation, the results of the statistical comparison are first presented. Following this, the results of the quantitative evaluation using neural networks are shown, which is completed by the qualitative analysis.

5.5.1 Statistical Comparison

In the first step, the locations of all objects in the IAC dataset are analyzed. Figure 5.4 shows the relative location of the annotated objects to the ego vehicle in BEV and their distribution on the x and y axis, with xyz convention as defined in Section 2.2. These locations were extracted from all 6,000 point clouds in the real-world dataset. The plot for the simulated data is not depicted, as it would show identical data because the scenarios are identical in simulation and the real world.

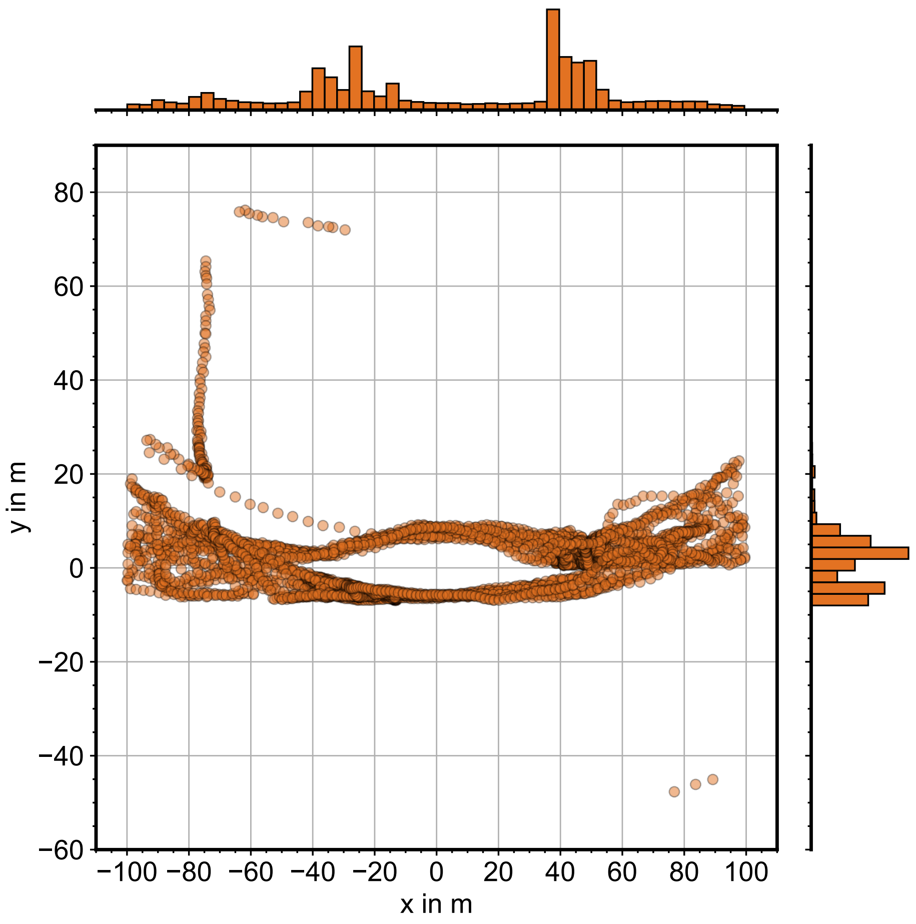


Figure 5.4: Object locations and distributions from real-world dataset relative to the ego vehicle (adapted from [213]). The outliers with large y values can be attributed to scenarios where the ego vehicle is driving on the track while the other vehicle is driving on the pit lane, or vice versa.

The plot shows that there exist two predominant positions, namely, the front left position at $(+35\,\mathrm{m}, +5\,\mathrm{m})$ and the back right position at $(-30\,\mathrm{m}, -5\,\mathrm{m})$. This can be explained by the fact that the competition format required the race cars to keep a minimum distance from opponents if no overtaking maneuver was taking place. Despite the existence of this unequal location distribution within the real-world dataset, it has no impact on the sim-to-real domain shift, since the positions of the objects in the simulation are based on the positions from the real-world dataset, and hence follow the same distribution.

Table 5.1 shows a comparison of key attributes of the simulated and real-world datasets, such as the point cloud range and the number of points for the scene or objects individually. Comparison of the point cloud range reveals that both the real-world data and the simulation data cover a similar area, cropped at a distance of $100\,\mathrm{m}$. The average number of points of the real-world point clouds is lower compared to the simulated point clouds, for both the scene and object point clouds. For the object point clouds, this reduction is more pronounced, which might be attributed to the materials and hence, reflectivity properties of the objects. Note that in simulation, point intensity does not influence the likelihood of existence of points, whereas, in the real world, reflections with an intensity lower than a threshold are not included in the output point cloud.

Table 5.1: Comparison of key statistical characteristics between the real-world and simulated datasets. Every value is computed from the complete training set of each dataset, utilizing the whole point cloud range up to $100\,\mathrm{m}$ (adapted from [213]).

Attribute			Real-world dataset	Simulated dataset
Point cloud range	mean	x in m	2.2	2.1
		y in m	−1.3	−0.5
		z in m	1.2	1.9
	min	x in m	−100.0	−100.0
		y in m	−100.0	−87.0
		z in m	−12.0	−1.5
	max	x in m	100.0	100.0
		y in m	100.0	100.0
		z in m	25.1	23.3
Number of points per scene point cloud	mean		73,123	78,776
	min		52,762	52,695
	max		79,690	81,538
Number of points per object point cloud	mean		219	251
	min		0	5
	max		4,959	6,465

Despite the differences in key attributes between real-world and simulated datasets, they show an overall similarity when comparing high-level statistical measures. This alignment of the datasets enables a reduction of the domain shift to a minimum, and the remainder of the domain shift is mainly due to the characteristics of the virtual sensor model in simulation and will be quantified in the following.

5.5.2 Quantitative Evaluation

For the quantitative evaluation using object detection networks, the following notation is used for the remainder of this chapter. An experiment in which a network is trained with simulated data and tested on real-world data is referred to as *sim-to-real*. Hence, the sim-to-real domain shift is the difference between the sim-to-real performance and the oracle performance, which is real-to-real in this case. Figure 5.5 displays the results for the 3D AP (0.7) of PointRCNN trained and tested with simulated or real-world data, each combination showing the five training runs with individual points and the mean of these runs with a horizontal line. This diagram shows a notable sim-to-real domain shift of approximately $14\,\%$ AP when comparing real-to-real $52\,\%$ with sim-to-real $38\,\%$, despite using a scenario- and distribution-aligned dataset. Furthermore, a domain shift in the opposite direction, i.e. a real-to-sim domain shift, is also notable. The domain shift from real to simulated is even more pronounced than the shift from simulated to real, which can be attributed to generally superior results when performing tests with simulated data. Especially using simulated data for training and testing, i.e. sim-to-sim, achieves a high AP of over $90\,\%$ due to the absence of noisy characteristic of simulated data. These results align with the domain shift quantification results of [107, 159].

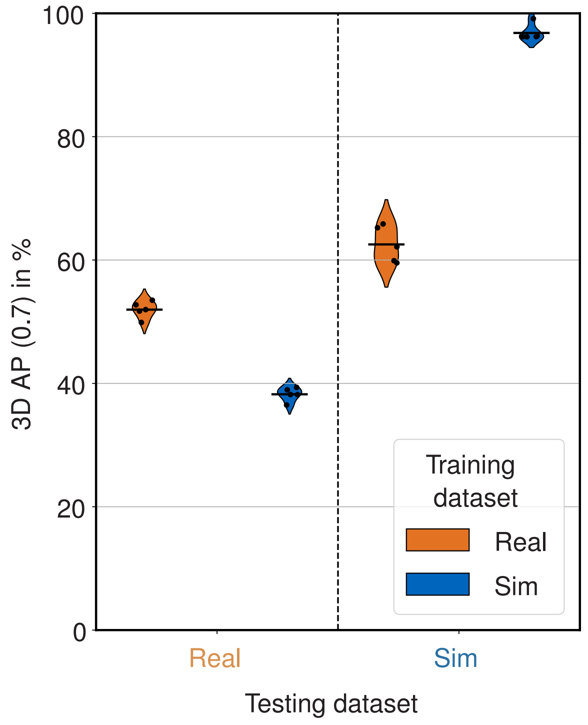
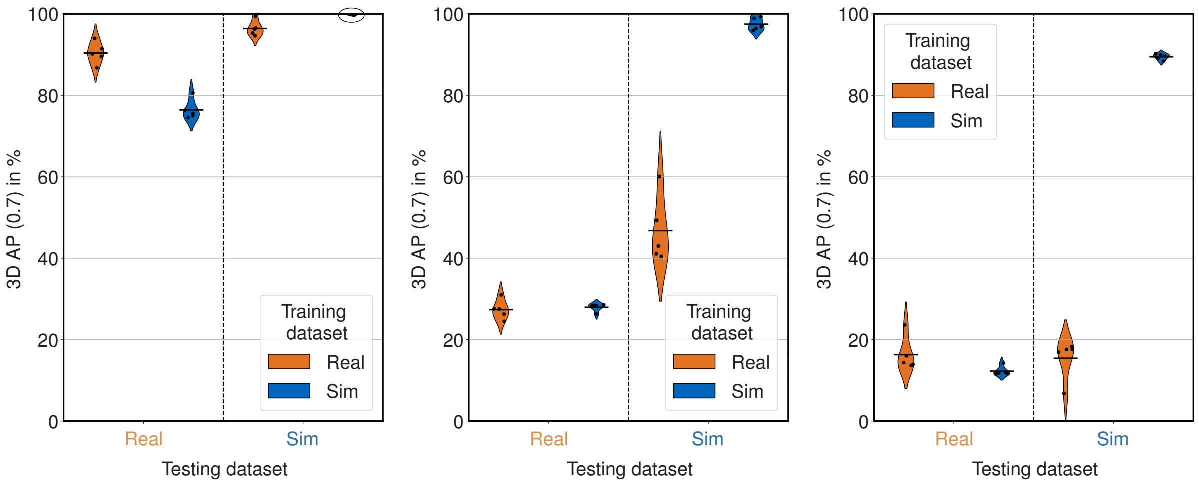


Figure 5.5: The 3D AP (0.7) for PointRCNN, which was trained and evaluated using either *real* or *sim* data. The horizontal lines represent the mean AP and the five points represent the five individual training cycles for each training and testing combination (adapted from [213]).

Full results, including the performance of the second object detection network PointPillars, can be found in Table A.1 in Section A.1. It also includes the 3D AP (0.5) and the recall for both IoU thresholds, all of which show similar trends as the 3D AP (0.7). As expected, the performance of 3D AP (0.5) is higher than the performance of 3D AP (0.7) due to the lower IoU threshold, but the relative magnitude of the domain shift in both directions is similar. The general performance of PointPillars is lower compared to the performance of PointRCNN, which aligns with their ranking order in the KITTI benchmark. However, the domain shift has a similar extent for both PointRCNN and PointPillars networks, although the domain shift is somewhat more pronounced with PointPillars, which indicates that this voxel-based architecture is more susceptible to domain shift than the point-based PointRCNN.

Figure 5.6 shows the same training runs as in Figure 5.5, but is evaluated only in close-range $[0.0\,\mathrm{m}, 33.3\,\mathrm{m})$, mid-range $[33.3\,\mathrm{m}, 66.6\,\mathrm{m})$, or long-range $[66.6\,\mathrm{m}, 100.0\,\mathrm{m}]$. This range-individual evaluation provides information on which area is most susceptible to domain shift. In general, with increasing evaluation range, the AP drops as expected. Furthermore, the sim-to-sim performance is less affected by range than the real-to-real performance, indicating that there are fewer noisy objects in the simulated data compared to the real-world data. For the sim-to-real domain shift, mid- and long-range are barely affected by domain shift, whereas for the real-to-sim domain shift, these ranges are affected more than the close-range. Both can be attributed to the fact that real-world mid- and long-range objects are more noisy than close-range objects. This leads to challenges during testing on real-world data for both the real-world and simulated trained networks. In contrast, simulated mid- and long-range objects still have clear structures, which are hard to detect for networks trained on real-world data. This will be analyzed in the following qualitative assessment of the domain shift.

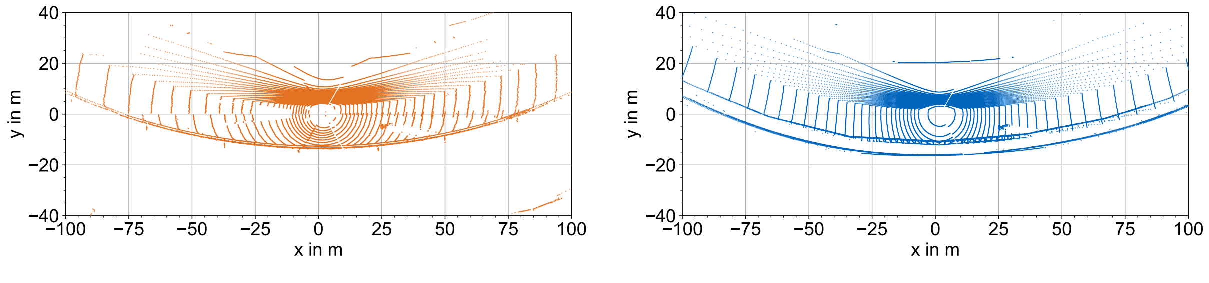


(a) Close-Range $[0.0\,\text{m}, 33.3\,\text{m})$ (b) Mid-Range $[33.3\,\text{m}, 66.6\,\text{m})$ (c) Long-Range $[66.6\,\text{m}, 100.0\,\text{m}]$

Figure 5.6: The 3D AP (0.7) for PointRCNN, which was trained and evaluated using either *real* or *sim* data. The horizontal lines represent the mean AP and the five points represent the five individual training cycles for each training and testing combination. Note that in (a), the mean AP of sim-to-sim is close to $100\,\%$ and, therefore, marked with an ellipse (adapted from [213]).

5.5.3 Qualitative Evaluation

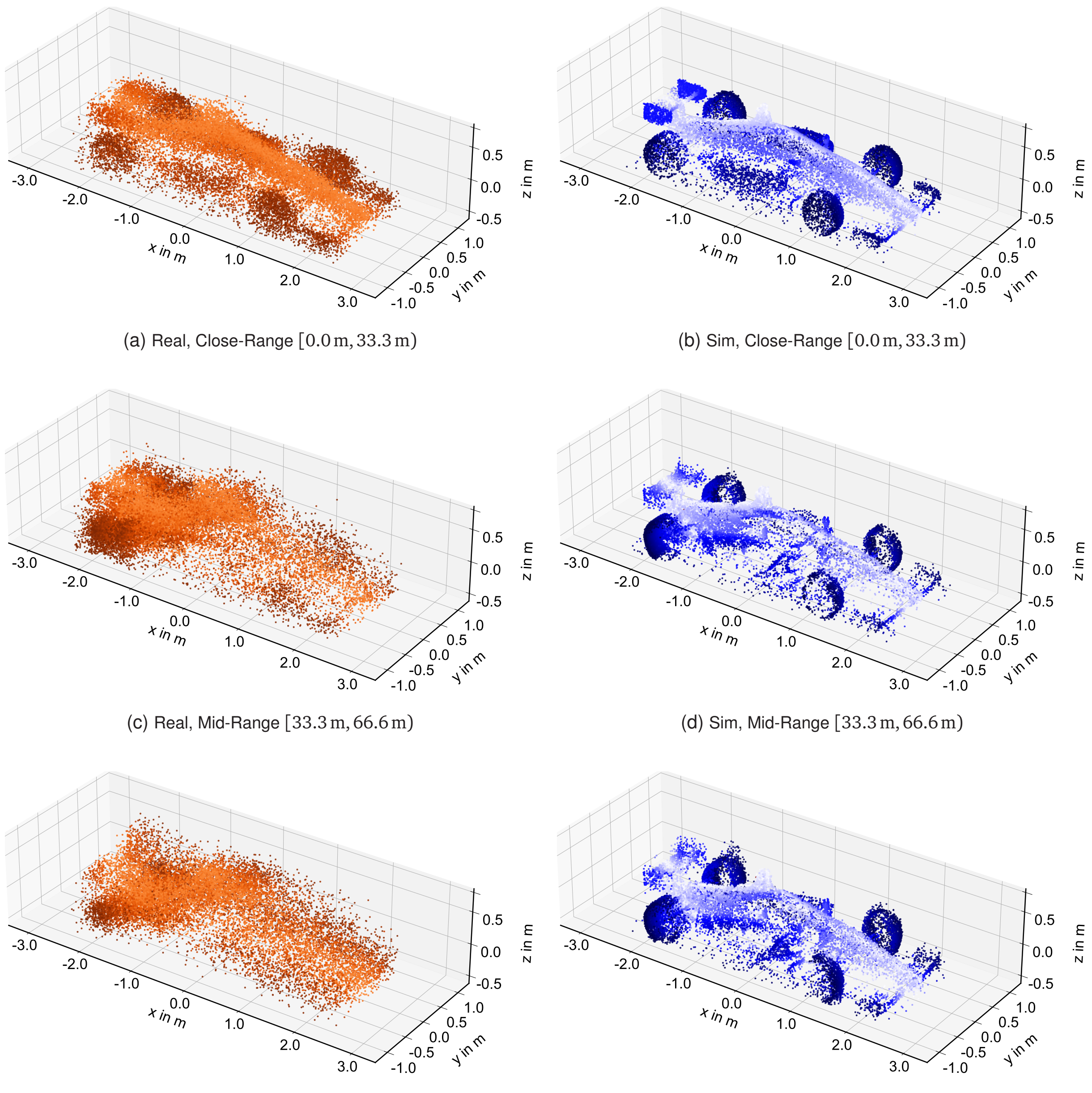
Before qualitatively analyzing the datasets at the object-level, Figure 5.7 shows a scene point cloud from both the real-world dataset (Figure 5.7a) and the simulated dataset (Figure 5.7b) of the same scenario in BEV projection. This high-level comparison shows similarities, but also some differences between the datasets. First, the general structure of the scenery made up of the race track, grass, walls, and one object at approximately $(+25\,\text{m}, -5\,\text{m})$ in the ring-shaped point cloud can be identified in both point clouds. Differences include the out-of-track reflections present in the real-world point cloud, and a LiDAR shadow behind the ego vehicle due to blockage of the ego vehicle's rear wing. Furthermore, the local structure of the point cloud rings is more noisy in the real-world point cloud, as opposed to the smooth rings in the simulated counterpart.



(a) Point cloud captured in the real world. (b) Point cloud generated in the 3D simulation environment.

Figure 5.7: Point clouds visualized in bird's-eye-view of the real-world (left) or simulated (right) dataset. In both plots, the ego vehicle recording the point clouds is driving to the right (adapted from [213]).

The IAC dataset used only contains a single object class, which is a non-deformable race car. This enables the comparison of point clouds of this object within a dataset domain across different ranges, but more importantly, the comparison of the object point clouds across the simulated and real-world domains. To this end, Figure 5.8 shows the aggregated and zero-centered 3D object point clouds of real-world or simulated data for the three different distant ranges. Comparing the close-range of real-world and simulated data, the distinct shape of the object is sharper and less noisy in the simulated point cloud, leading to a domain shift. With increasing distance, the shape in both domains becomes more noisy; however, even at long range, the shape of the object in the simulated data is still recognizable. This further explains the observation of quantitative analysis, in which increasing distance leads to a larger performance decrease for real-world data than for the simulated data.



(a) Real, Close-Range $[0.0\,\mathrm{m}, 33.3\,\mathrm{m})$

(b) Sim, Close-Range $[0.0\,\mathrm{m}, 33.3\,\mathrm{m})$

(c) Real, Mid-Range $[33.3\,\mathrm{m}, 66.6\,\mathrm{m})$

(d) Sim, Mid-Range $[33.3\,\mathrm{m}, 66.6\,\mathrm{m})$

(e) Real, Long-Range $[66.6\,\mathrm{m}, 100.0\,\mathrm{m}]$

(f) Sim, Long-Range $[66.6\,\mathrm{m}, 100.0\,\mathrm{m}]$

Figure 5.8: Normalized and aggregated point clouds from the real dataset (left column) and the sim dataset (right column). The visualization is split into the three distance ranges, in which the object point clouds were captured or generated before normalization: close-range $0.0\,\mathrm{m}$ to $33.3\,\mathrm{m}$ (a)(b), mid-range $33.3\,\mathrm{m}$ to $66.6\,\mathrm{m}$ (c)(d), and long-range $66.6\,\mathrm{m}$ to $100\,\mathrm{m}$ (e)(f). The aggregation is done by randomly selecting $20{,}000$ points from all available object points per distance range. The total number of points within the long-range of the real-world dataset only contains $15{,}679$ points and hence, is limited to this number (adapted from [213]).

For further details on the qualitative domain shift analysis by means of visualizing point clouds, the reader is referred to the corresponding publication [213], in which in addition to the point cloud noise, the effect of point cloud dropout is explained in detail. However, in this thesis, the primary focus is on the LiDAR sim-to-real domain shift due to point cloud noise.

The results of t-SNE dimension reduction to compare the high-dimensional latent feature space are depicted in Figure 5.9 for feature vectors extracted from PointRCNN, generated by passing real-world point clouds through the network, which is trained with real-world (orange) or simulated data (blue). Each point represents a high-dimensional latent feature vector reduced to two dimensions, generated by inference with a single real-world point cloud. The five distinct clusters per training dataset are due to the non-deterministic training, which produces different network weights, and hence different latent representations per training. The core statement of this t-SNE graph is that even when passing the same point clouds through the same network trained with different datasets, the network differentiates between the data domains with which it was trained, emphasizing that the domain shift exists within the network.

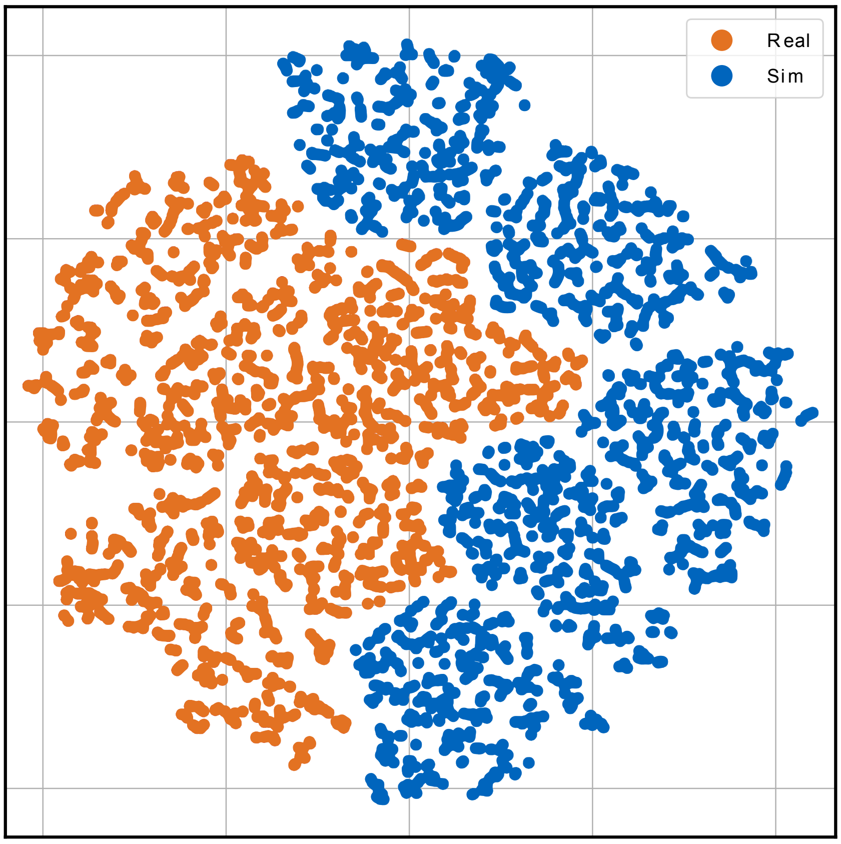


Figure 5.9: A visualization using t-SNE technique of the latent feature space generated by PointRCNN, trained separately on real and simulated data for five training sessions each. Each point represents a feature vector produced by the network's inference when analyzing a single point cloud (adapted from [213]).

5.5.4 Parameter Sensitivity Analysis

The quantitative analysis of the domain shift in Subsection 5.5.2 shows the presence of a sim-to-real domain shift even in the scenario- and distribution-aligned dataset, and the qualitative analysis in Subsection 5.5.3 identifies point cloud noise and dropout as possible causes of the domain shift. In this parameter sensitivity analysis, the impact on the domain shift of simple models that integrate these sensor effects should be analyzed to answer the research question Q2. To this end, two additional datasets based on the original simulated dataset are generated. The first dataset adds Gaussian noise in the longitudinal ray direction to every sensor measurement. Similar to [148], a standard deviation of $\sigma = 2\,\mathrm{cm}$ is used for the Gaussian noise. In the second dataset, $20\,\%$ of the points per point cloud are randomly omitted, simulating a synthetic point dropout. The datasets are denoted as *sim noise* and *sim dropout*, respectively. PointRCNN and PointPillars are both trained with the additional simulated datasets and evaluated on the real-world dataset to measure the sim-to-real domain shift.

All results can be found in Table A.1 in Section A.1. Figure 5.10 shows the 3D AP (0.7) of PointRCNN trained with the additional datasets compared to the real-to-real and sim-to-real results. The plot reveals that the *sim noise* dataset can reduce the sim-to-real domain shift from approximately $14\,\%$ using the original simulated dataset to $10.35\,\%$ using *sim noise*, a relative reduction of approximately $25\,\%$. However, the second dataset, *sim dropout*, results in a lower 3D AP than the original simulated dataset and therefore even increases the sim-to-real domain shift, hence having a negative impact.

To summarize the findings of the parameter sensitivity analysis, the simple noise model in simulation can reduce the sim-to-real domain shift to an extent but not fully eliminate it. Simple dropout modeling by random dropping points, however, is increasing the sim-to-real domain shift, and more sophisticated dropout methods, e.g. based on the point intensity, have to be studied in future work. Further details, such as t-SNE graphs and 3D aggregated point clouds for these two additional datasets, can be found in the corresponding publication [213].

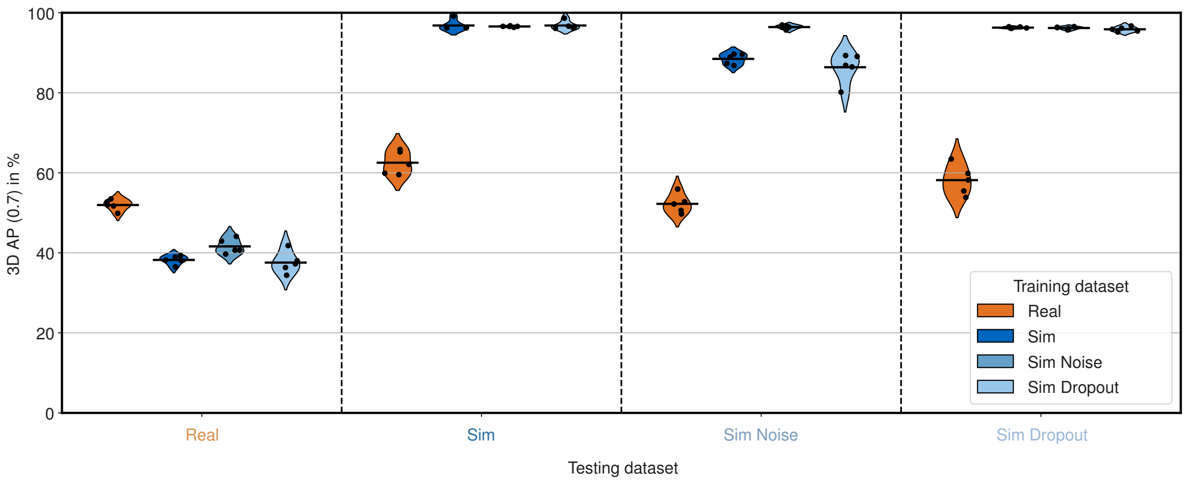


Figure 5.10: The 3D AP (0.7) for PointRCNN, which was trained and evaluated using either *real*, *sim*, *sim noise*, or *sim dropout*, data. The horizontal lines represent the mean AP and the five points represent the five individual training cycles for each training and testing combination (adapted from [213]).

6 LiDAR Domain Adaptation of 3D Point Clouds at the Object-Level

In the previous Chapter 5, the domain shift was analyzed in detail by quantification using a simulated and real-world scenario- and distribution-aligned dataset, and non-accurate LiDAR sensor noise modeling is identified as a possible cause of the resulting domain shift. The qualitative analysis of the LiDAR sim-to-real domain shift further shows that, especially at the object-level, the point clouds between simulation and the real world differ. In this chapter, a LiDAR domain adaptation method is presented to adapt simulated point clouds at the object-level to real-world point clouds, to minimize the prevalent domain shift analyzed in the previous chapter. The goal of this chapter is to address and answer the research question Q3.

Q3: How can LiDAR domain adaptation techniques be applied at the object-level to mitigate the impact of sim-to-real domain shift, and what are the specific challenges and solutions associated with object-level domain adaptation?

To this end, the problem is first formulated and the notation is explained in Section 6.1. The method, including the choice of architecture and loss function, is discussed in Section 6.2. Section 6.3 explains the specifics of the dataset, the preprocessing, and the training details. This chapter is concluded by presenting the results in Section 6.4. This chapter is based on and adapted from [223].

6.1 Notation and Problem Formulation

Formally, a point cloud can be denoted as an unordered point set $\mathbf{X}_i = \{p_j|_{j=1}^{N_j}\}$. Each of the i point sets contains N_j points p_j, and each point is characterized by 3D coordinates $p_j = \{x_j, y_j, z_j\} \in \mathbb{R}^3$. In annotated datasets, each point cloud $\mathbf{X}_i$ has a set of annotations $\mathbf{Y}_i = \{y_k|_{k=1}^{N_k}\}$, containing k 3D bounding boxes y_k, each characterized by the seven degrees of freedom $y_k = \{c_x, c_y, c_z, l, w, h, \theta_z\}_k$ for the center, dimension, and rotation around the z axis [163]. As the rotations around the x and y axes are usually approximately zero in autonomous driving applications, these two degrees of freedom are neglected. In this chapter, rather than adapting entire scene point clouds $\mathbf{X}_i$ between data domains, the focus is on adapting single object point clouds $\mathbf{O}_{i,k}$, referring to the point cloud of the k-th object within the i-th scene point cloud and containing all points enclosed by the 3D bounding box y_k.

This chapter presents a method for the unsupervised domain adaptation of LiDAR point clouds. As described in Subsection 3.2.1, in unsupervised domain adaptation, the source dataset is fully annotated, while there is no access to the annotations of the target dataset. Formally, the source domain $\mathbf{S} = \{(\mathbf{X}_i^{\mathrm{S}}, \mathbf{Y}_i^{\mathrm{S}})|_{i=1}^{N_{\mathrm{S}}}\}$ contains N_{S} point clouds $\mathbf{X}_i^{\mathrm{S}}$ with their corresponding annotations $\mathbf{Y}_i^{\mathrm{S}}$. However, the target domain $\mathbf{T} = \{\mathbf{X}_i^{\mathrm{T}}|_{i=1}^{N_{\mathrm{T}}}\}$ only contains N_{T} point clouds $\mathbf{X}_i^{\mathrm{T}}$ without their corresponding annotations. This leads to the objective of unsupervised domain adaptation, aiming to comprehend a mapping function Ψ that connects point clouds from the source domain $\mathbf{X}_i^{\mathrm{S}}$ to those in the target domain $\mathbf{X}_i^{\mathrm{T}}$ without access to the target annotations.

6.2 Method

As discussed in related work in Subsection 3.2.2, domain adaptation for LiDAR point clouds can be achieved using approaches belonging to domain-invariant feature learning methods, normalization statistics methods, domain-invariant data representation methods, or domain mapping methods. Domain-invariant feature learning methods are model-driven and, therefore, require a modification of each specific perception network, which is not scalable for the application of AD, in which multiple perception networks can be applied for different perception tasks. Nowadays, normalization techniques are applied to neural networks in the form of e.g. batch normalization, hence they are usually part of domain mapping methods or learning-based domain-invariant data representation methods. Domain-invariant data representation methods transform the 3D point cloud data to a 2D sensor image or a 3D voxel grid to achieve domain invariance. Despite not requiring modifications to the target perception network due to their operation in the input space, these methods restrict the choice of perception networks to those that utilize these specific representations. Moreover, the transformation to the domain-invariant data representation mostly relies on hand-crafted techniques, which limits the generalization.

For these reasons, domain mapping is chosen as an approach in this thesis to adapt point clouds. Domain mapping works on the input space, and hence modifications to perception networks are not necessary. Furthermore, these methods are learning-based and therefore can leverage the information of the target data to assist or improve the adaptation process. Domain mapping originates from image-based domain adaptation and is typically based on adversarial networks in this context. The adversarial domain adaptation approaches for LiDAR point clouds are limited to methods that work with the 2D BEV representation of LiDAR point clouds. The 3D LiDAR point clouds can not be recovered from adapted 2D BEV images and therefore can not be used by perception networks using these 3D point clouds as input. In addition, [94] emphasizes the absence of any adversarial LiDAR domain mapping technique that can directly adapt point clouds in the 3D space. As a consequence, this thesis presents the first LiDAR domain adaptation approach based on adversarial learning that adapts 3D point clouds.

The remainder of this section explains the approach of the proposed object-level LiDAR domain adaptation method by first providing a brief overview in Subsection 6.2.1, then explaining the network in detail in Subsection 6.2.2, and finally describing the specific loss function designed for network training.

6.2.1 Conceptual Framework and Network Overview

Only annotated objects in the source dataset are adapted, as the qualitative evaluation of the domain shift in the previous chapter showed a mismatch of simulated and real-world point clouds at the object-level. This focus on adapting object point clouds instead of scene point clouds brings the following two advantages. First, object point clouds are less sparse than entire scene point clouds and strive for more uniform density and improved retention of global structure within the object point cloud. Second, networks used for adaptation are initially intended for point cloud completion of single objects, and hence the transferability to other applications is easier when the focus is on single objects. Nevertheless, the transferability to adapting entire scene point clouds using the developed approach will be analyzed in Chapter 7.

Figure 6.1 depicts a general overview of the method presented in this chapter. In the first data preprocessing step, the relevant source and target object point clouds, $\mathbf{O}_{i,k}^{\mathrm{S}}$ and $\mathbf{O}_{i,k}^{\mathrm{T}}$, respectively, are extracted from the source and target scene point clouds, $\mathbf{X}_{i}^{\mathrm{S}}$ and $\mathbf{X}_{i}^{\mathrm{T}}$, respectively, based on their 3D bounding boxes. After extraction, the point clouds are zero-centered and axis-aligned, i.e., the absolute coordinates of the center of the 3D bounding box are subtracted from every point coordinate, and the point cloud is rotated to align with the global coordinate system defined in Section 2.2. Zero-centering is performed to eliminate any possible bias in the dataset. These objects can include all classes for the object detection task, such as cars, vans, trucks, motorcycles, pedestrians, cyclists, etc. Target object point clouds are used for training and source object point clouds are used for inference. Both steps will be briefly explained in the following.

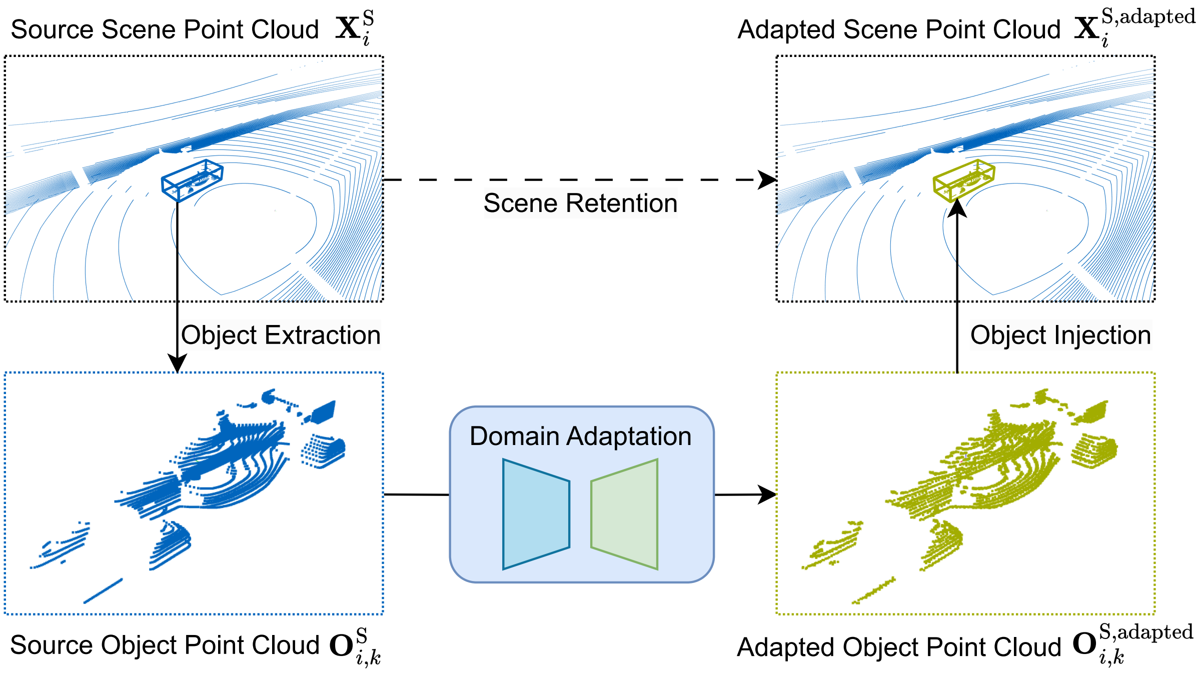

Source Scene Point Cloud $\mathbf{X}_i^{\mathrm{S}}$
Adapted Scene Point Cloud $\mathbf{X}_i^{\mathrm{S,adapted}}$
Scene Retention
Object Extraction
Object Injection
Domain Adaptation
Source Object Point Cloud $\mathbf{O}_{i,k}^{\mathrm{S}}$
Adapted Object Point Cloud $\mathbf{O}_{i,k}^{\mathrm{S,adapted}}$

Figure 6.1: High-level overview of the object-based domain adaptation method. The first step comprises the extraction of the object point cloud $\mathbf{O}_{i,k}^{S}$ from a source scene point cloud $\mathbf{X}_i^{S}$ (here: simulated data, blue). The trained domain adaptation network subsequently adapts the object point cloud to generate a target-style object point cloud (here: real-world, green) $\mathbf{O}_{i,k}^{S,\mathrm{adapted}}$, which is re-injected in their initial locations within the source scene point cloud. The resulting output $\mathbf{X}_i^{S,\mathrm{adapted}}$ combines the original source scene point cloud with the adapted object point cloud (adapted from [223]).

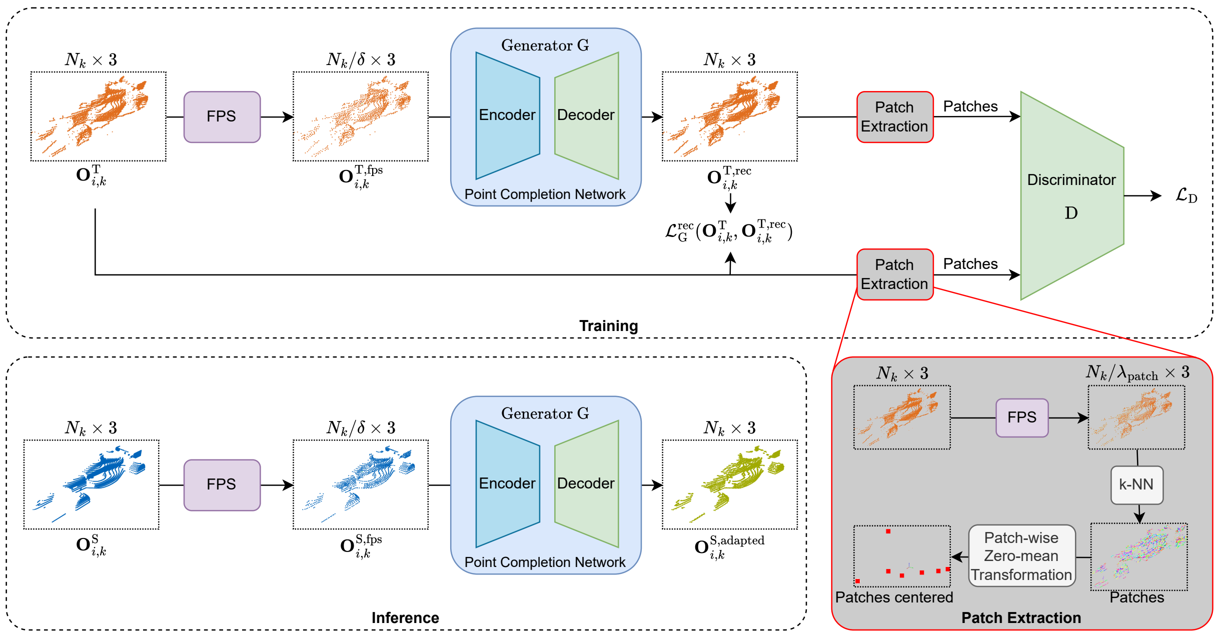
The domain adaptation network training is carried out solely using target object point clouds. These target object point clouds are downsampled before being passed to the network. The goal of the downsampling is to create a representation of these point clouds that is invariant across domains, and the objective of the point completion network being used is to reconstruct the original non-downsampled version, given the downsampled version as input. This forces the network during training to create target-style point clouds using domain-invariant object point clouds as input. At the same time, the global structure of the object point cloud must be maintained throughout the downsampling process, and the point completion network should only apply the target style locally. Keeping the global structure is a hard requirement, as during inference, the domain-adapted objects are injected back into their original location within the corresponding scene point cloud and should neither change the semantic meaning nor the viewpoint.

Once the network is trained to reconstruct target-style object point clouds, it can be used during inference to adapt source object point clouds, as depicted in Figure 6.1. To this end, the source object point clouds $\mathbf{O}_{i,k}^{\mathrm{S}}$ that are extracted undergo downsampling in the same manner as the target object point clouds during training, and the point completion network reconstructs these downsampled source object point clouds by applying the learned target style. The output of the network is the adapted object point cloud $\mathbf{O}_{i,k}^{\mathrm{S,adapted}}$. These adapted object point clouds are re-injected into their original location within the source scene point cloud $\mathbf{X}_i^{\mathrm{S}}$ where they have been extracted, with the same location and orientation. This results in the adapted scene point cloud $\mathbf{X}_i^{\mathrm{S,adapted}}$, which is the final output of the presented domain adaptation method and can be used for downstream perception tasks, for example, for training object detection networks. The hypothesis is that employing these adapted scene point clouds for training will lead to a decrease in domain shift compared to training with unaltered source scene point clouds.

As explained, the training and inference stages use different data domains as input. Training is based only on target data, and source data are not needed during training. This approach has two implications: First, some target data need to be annotated. However, the amount of annotated target data is low compared to the theoretically unlimited number of source data that can be adapted using the trained domain adaptation network. Second, once the domain adaptation network is trained, the trained network is capable of adapting data from various source domains, as it has not seen any source data during training and, hence, is not trained to only adapt data from a certain source domain. This approach is particularly beneficial for sim-to-real domain adaptation. It enables the generation of synthetic data in various simulation environments, while each environment can have different virtual LiDAR sensor models. After data generation in simulation, the data can be adapted to a common real-world target style that was learned during training.

6.2.2 Architecture and Network Design

The network architecture is founded on a GAN and consists of a generator network G and a discriminator network D which are trained adversarially. The goal of the generator is to produce target-style object point clouds given the downsampled target object point clouds, whereas the discriminator tries to detect and distinguish the generated samples from the original non-downsampled point clouds. The specifics of both networks will be explained in the following, after briefly discussing the preprocessing of the input data. Figure 6.2 illustrates the detailed processing steps within the architecture for both the training phase in the upper part and the inference phase in the lower part. Patch extraction is a preprocessing step of the discriminator and will also be explained as part of this subsection.



$N_k \times 3$
FPS
$N_k/\delta \times 3$
Generator G
Encoder
Decoder
Point Completion Network
$N_k \times 3$
$\mathbf{O}^{\mathrm{T}}_{i,k}$
$\mathbf{O}^{\mathrm{T,fps}}_{i,k}$
$\mathbf{O}^{\mathrm{T,rec}}_{i,k}$
$\mathcal{L}^{\mathrm{rec}}_{\mathrm{G}}(\mathbf{O}^{\mathrm{T}}_{i,k}, \mathbf{O}^{\mathrm{T,rec}}_{i,k})$
Patch Extraction
Patches
Discriminator D
$\mathcal{L}_{\mathrm{D}}$
Training
$\mathbf{O}^{\mathrm{S}}_{i,k}$
$\mathbf{O}^{\mathrm{S,fps}}_{i,k}$
$\mathbf{O}^{\mathrm{S,adapted}}_{i,k}$
Inference
$N_k/\lambda_{\mathrm{patch}} \times 3$
k-NN
Patch-wise Zero-mean Transformation
Patches centered
Patches
Patch Extraction

Figure 6.2: Visualization of the detailed network architecture of the domain adaptation network to adapt point clouds at the object-level. During the training process, object-level point clouds $\mathbf{O}^{T}_{i,k}$ are first downsampled using Farthest Point Sampling (FPS) and then reconstructed with a point completion network that serves as the generator G. To further assist in domain adaptation, a discriminator D is discriminating between point cloud patches with λ_{patch} points each from the reconstructed (fake) or original (real) object point cloud. In the inference step, FPS-downsampled source point clouds $\mathbf{O}^{S}_{i,k}$ are processed by the generator G for local adaptation to generate output point clouds in the target style $\mathbf{O}^{S,\text{adapted}}_{i,k}$ (adapted from [223]).

In general, the input for both training and inference is a zero-centered and axis-aligned object point cloud. In the first preprocessing step, the number of points in each point cloud is analyzed. The number of points per object point cloud varies strongly with the distance to the LiDAR sensor since the density of points generally decreases with increasing distance to the sensor. This requires the generator and discriminator to manage a varying quantity of points and further introduces the challenge of missing the local structure for point clouds consisting of only a few points. As the architecture is intended to adapt the local structure of point clouds, a minimum number of points is defined. Point clouds that contain less points than the predetermined minimum threshold are not considered in the training or inference steps.

Before passing the object point clouds to the generator, they are first downsampled in a second preprocessing step to create a domain-invariant representation. This downsampling step is introduced to reduce the discrepancy between the target object point clouds during training and the source object point clouds during inference. During downsampling, the number of points in each point cloud is reduced by a fixed downsampling factor δ. Farthest Point Sampling (FPS) algorithm [224] is used for downsampling, as FPS is capable of maintaining the global structure of the object point clouds, while eliminating local structures, thereby ensuring their domain-invariance. FPS iteratively calculates the distance from each point to the rest of all points and selects the point with the largest distance to the current point, resulting in a computational complexity that approaches $O(n^2)$ [225]. The choice of the downsampling factor δ defines the degree of reduction of the local structures, where a higher δ removes more points and, hence, removes more local structures. Empirical experiments show that a reasonable choice for the downsampling factor is $3 \leq \delta \leq 10$. Finally, the output of FPS is a downsampled object point cloud $\mathbf{O}_{i,k}^{\mathrm{T,fps}}$ or $\mathbf{O}_{i,k}^{\mathrm{S,fps}}$ for training or inference, respectively.

Generator

The primary goal of the generator G is to upsample the previously downsampled object point cloud and apply the learned local structure of the target domain during the upsampling step. The upsampling factor is fixed and set to be identical to the downsampling factor since the output number of points should be equal to the input number of points. During network training, the generator G reconstructs the target object point cloud, therefore, G acquires the ability to complete the global structure of the point cloud by incorporating the local structure. During inference, the identical trained G is employed to reconstruct the domain-invariant and downsampled source object point cloud, and it upsamples the point cloud by incorporating points based on the acquired local structure of the target. In this way, domain adaptation only occurs during inference, whereas training is based only on the reconstruction of target domain point clouds.

The requirements for the generator network are that it has to be able to upsample point clouds with a varying number of points. As a baseline, SeedFormer [226] is chosen, which is a state-of-the-art point completion network. SeedFormer employs upsample transformers to recover local structures and completes point clouds in a coarse to fine manner [226].

Several modifications are made to adapt the network to serve as a generator in the domain adaptation task and to meet the requirements. First, the original implementation of SeedFormer generates seeds that are derived from the global and patch features extracted. These seeds serve as support points to complete the point cloud in the decoder of SeedFormer. Instead of using the seeds generated by the encoder, the decoder in the domain adaptation setting is directly fed with the coarse, downsampled input point cloud. The reason for this is that the downsampled point cloud already includes all the areas needed to be covered by the output point cloud, which is not the case with the original point completion task, which uses partial point clouds as input. The second modification to SeedFormer concerns the handling of a variable number of points. To achieve this, the number of query points in the PointNet++ set abstraction layers in the encoder of SeedFormer is computed dynamically, taking into account the quantity of input points.

Discriminator

During training, the generator upsamples the previously downsampled object point cloud and produces the reconstructed object point cloud $\mathbf{O}_{i,k}^{\mathrm{T,rec}}$. This reconstructed point cloud maintains the global structure of the input point cloud while simultaneously containing the local structure of the target domain. This is always a trade-off and can lead to non-optimal solutions, as explained in the following example visualized in Figure 6.3.

For example, a point p_i in the input object point cloud $\mathbf{O}_{i,k}^{\mathrm{T}}$ (Figure 6.3a) has no direct neighboring points; hence, during downsampling using FPS (Figure 6.3b), it will be selected to serve as an input point for the generator (Figure 6.3c). The generator's task is to use this point p_i as a seed point and upsample it by adding $\delta - 1$ points in the vicinity of p_i (Figure 6.3d). This patch of $\delta - 1$ points can potentially collapse at the same coordinate as p_i, since there exists only a single point in the input object point cloud $\mathbf{O}_{i,k}^{\mathrm{T}}$, which serves as a comparison to $\mathbf{O}_{i,k}^{\mathrm{T,rec}}$ during training (Figure 6.3e). To avoid these cases of collapsing point cloud patches, a discriminator focusing on the local structures is employed during training. The goal of the patch discriminator is to distinguish between point cloud patches generated by the generator and point cloud patches extracted from the original input point cloud.

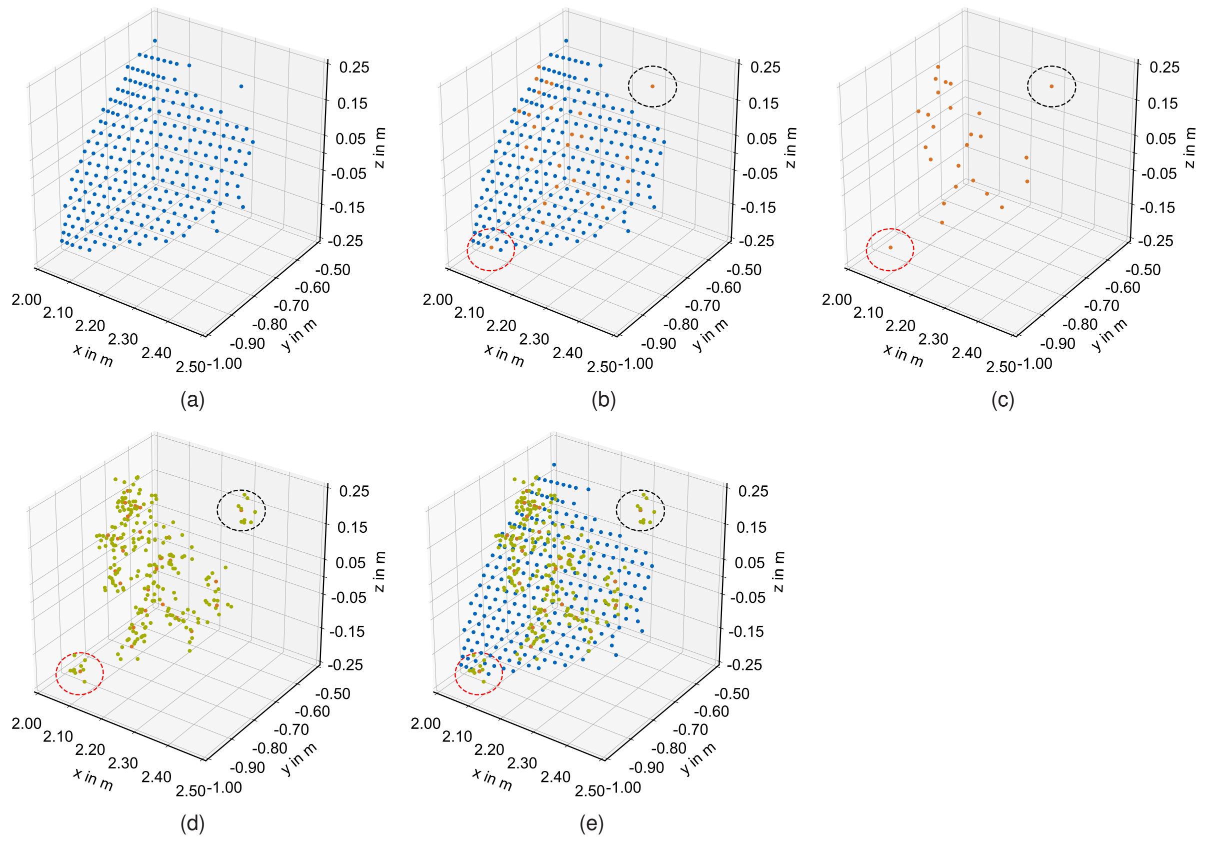


Figure 6.3: Example showing the challenge of collapsing point clouds patches. (**a**) Crop of the original point cloud (blue) serving as an input to the generator. (**b**) The FPS algorithm downsamples the original point cloud by factor δ (here $\delta = 10$). The selected points are visualized in orange. Furthermore, two points are highlighted by a black or red circle, respectively. In contrast to the red-highlighted point, the black-highlighted point has no direct neighboring points in the input cloud. (**c**) The input to the generator network, maintaining the global structure of the original point cloud. (**d**) The upsampled output of the generator network, which consists of the input points (orange) and the added $\delta - 1$ points (green) in the vicinity of each input point. (**e**) Each output point (orange and green) is compared to the closed point in the original point cloud (blue). Because there are no neighboring points within the black-highlighted circle, all points inside this circle are compared to the orange point located here, and consequently, they may collapse into this orange point.

To this end, the n_{patch} patches are extracted from both $\mathbf{O}_{i,k}^{\text{T,rec}}$ and $\mathbf{O}_{i,k}^{\text{T}}$ before passing them to the discriminator D, as depicted in lower right box in Figure 6.2. First, the seed points of the patches are determined using FPS in the same manner that the original input object point cloud is downsampled. The size of each patch λ_{patch} is set to be identical to the downsampling factor δ, since this is the size of a potentially collapsing patch, as explained before. Once the n_{patch} seeds are extracted, for each seed, the $\lambda_{\text{patch}} - 1$ neighbors are determined using the k-nearest neighbors algorithm. In the last step, the coordinates of each point cloud patch are transformed individually to have a mean of zero.

The extracted patches are passed to the discriminator D, which discriminates between *real* $\mathbf{O}_{i,k}^{\text{T}}$ and *fake* $\mathbf{O}_{i,k}^{\text{T,rec}}$ patches. The generator G and discriminator D are trained adversarially, i.e. the generator receives the negative loss of the discriminator, which gives an incentive to the generator to produce reconstructed point clouds $\mathbf{O}_{i,k}^{\text{T,rec}}$ less distinguishable from the original point clouds $\mathbf{O}_{i,k}^{\text{T}}$. As the discriminator becomes stronger during training, it can provide better feedback to the generator. The architecture of the discriminator comprises two set abstraction layers from PointNet++ with three subsequent fully connected layers. More details on the architecture can be found in the corresponding publication [223].

6.2.3 Loss Function

The objective of adversarial training is to reduce the individual loss functions of both generator $\mathcal{L}_{\text{G}}$ and discriminator $\mathcal{L}_{\text{D}}$. The linear combination of both loss functions results in the overall loss $\mathcal{L}_{\text{total}}$ and is defined as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{G}} + \mathcal{L}_{\text{D}}. \tag{6.1}$$

The generator loss function, denoted $\mathcal{L}_{\text{G}}$, is composed of two distinct components: reconstruction loss $\mathcal{L}_{\text{G}}^{\text{rec}}$ and adversarial loss $\mathcal{L}_{\text{G}}^{\text{adv}}$. For reconstruction loss $\mathcal{L}_{\text{G}}^{\text{rec}}$, Chamfer Distance (CD) is used as in the SeedFormer implementation [226]. The CD serves as a metric to evaluate the similarity between two point sets S_1, S_2 in a three-dimensional space $\mathbb{R}^3$, by aggregating the minimum squared Euclidean distances from each point in one set to its nearest neighbor in the other set, and vice versa. Mathematically, this distance is expressed as follows:

$$d_{\text{CD}}(S_1, S_2) = \frac{1}{2}\left(\frac{1}{|S_1|}\sum_{x \in S_1} \min_{y \in S_2} ||x - y||_2^2 + \frac{1}{|S_2|}\sum_{y \in S_2} \min_{x \in S_1} ||y - x||_2^2\right). \tag{6.2}$$

In this formulation, each component is normalized using the length of the point set $|S_1|$ or $|S_2|$, respectively. There exists also the non-normalized formulation of the CD, which will be used in the following. This metric is employed to define the reconstruction loss for object-level point cloud comparison as

$$\mathcal{L}_{\text{G}}^{\text{rec}}\left(O_{i,k}^{\text{T,rec}}, O_{i,k}^{\text{T}}\right) = d_{\text{CD}}\left(O_{i,k}^{\text{T,rec}}, O_{i,k}^{\text{T}}\right). \tag{6.3}$$

Experiments using the Earth Mover's Distance, a common metric for point set comparison, show worse results compared to the CD.

The adversarial component of the generator's loss, denoted as $\mathcal{L}_{\text{G}}^{\text{adv}}$, is determined as the negative logarithm of the discriminator's prediction on the generator's reconstructed point cloud, formulated as

$$\mathcal{L}_{\text{G}}^{\text{adv}}\left(O_{i,k}^{\text{T}}\right) = -\log\left(\text{D}\left(\text{G}\left(O_{i,k}^{\text{T}}\right)\right)\right), \tag{6.4}$$

where the notation $\text{G}\left(O_{i,k}^{\text{T}}\right)$ represents the generated reconstruction $O_{i,k}^{\text{T,rec}}$. A value of $\text{D}\left(\text{G}\left(O_{i,k}^{\text{T}}\right)\right)$ approaching one indicates the generator G has effectively fooled the discriminator.

The discriminator's loss function, represented as $\mathcal{L}_{\mathrm{D}}$, is calculated using the following equation

$$\mathcal{L}_{\mathrm{D}}\left(O_{i,k}^{\mathrm{T}}\right) = -\log\left(\mathrm{D}\left(O_{i,k}^{\mathrm{T}}\right)\right) - \log\left(1-\mathrm{D}\left(\mathrm{G}\left(O_{i,k}^{\mathrm{T}}\right)\right)\right). \tag{6.5}$$

In this context, the term $\mathrm{D}\left(O_{i,k}^{\mathrm{T}}\right)$ refers to the discriminator's result when analyzing an original point cloud, with an expected value close to one. On the other hand, $\mathrm{D}\left(\mathrm{G}\left(O_{i,k}^{\mathrm{T}}\right)\right)$ indicates the discriminator's evaluation of a point cloud reconstructed by the generator G, which ideally should approach zero, indicating the generator's contribution to generation.

Incorporated into the equation for the overall loss $\mathcal{L}_{\mathrm{total}}$, as specified in Equation 6.1, the calculation is expressed as follows

$$\begin{aligned}\mathcal{L}_{\mathrm{total}} &= \mathcal{L}_{\mathrm{G}}^{\mathrm{rec}}\left(O_{i,k}^{\mathrm{T,rec}}, O_{i,k}^{\mathrm{T}}\right) + \mathcal{L}_{\mathrm{G}}^{\mathrm{adv}}\left(O_{i,k}^{\mathrm{T}}\right) + \mathcal{L}_{\mathrm{D}}\left(O_{i,k}^{\mathrm{T}}\right)\\ &= d_{\mathrm{CD}}\left(O_{i,k}^{\mathrm{T,rec}}, O_{i,k}^{\mathrm{T}}\right)\\ &\quad -\log\left(\mathrm{D}\left(\mathrm{G}\left(O_{i,k}^{\mathrm{T}}\right)\right)\right)\\ &\quad -\log\left(\mathrm{D}\left(O_{i,k}^{\mathrm{T}}\right)\right) - \log\left(1-\mathrm{D}\left(\mathrm{G}\left(O_{i,k}^{\mathrm{T}}\right)\right)\right).\end{aligned} \tag{6.6}$$

With the CD from Equation 6.2, the final total loss function is defined as

$$\begin{aligned}\mathcal{L}_{\mathrm{total}} = \frac{1}{2}&\left(\frac{1}{|O_{i,k}^{\mathrm{T,rec}}|}\sum_{x\in O_{i,k}^{\mathrm{T,rec}}}\min_{y\in O_{i,k}^{\mathrm{T}}}||x-y||_2^2 + \frac{1}{|O_{i,k}^{\mathrm{T}}|}\sum_{y\in O_{i,k}^{\mathrm{T}}}\min_{x\in O_{i,k}^{\mathrm{T,rec}}}||y-x||_2^2\right)\\ &-\log\left(\mathrm{D}\left(\mathrm{G}\left(O_{i,k}^{\mathrm{T}}\right)\right)\right)\\ &-\log\left(\mathrm{D}\left(O_{i,k}^{\mathrm{T}}\right)\right) - \log\left(1-\mathrm{D}\left(\mathrm{G}\left(O_{i,k}^{\mathrm{T}}\right)\right)\right).\end{aligned} \tag{6.7}$$

6.3 Experimental Setup

This section describes the experimental setup used to evaluate the proposed domain adaptation method. To this end, the details of the dataset and model setup are introduced in Subsection 6.3.1 and Subsection 6.3.2, respectively.

6.3.1 Dataset

To train and test the developed domain adaptation method for sim-to-real domain shift reduction, an annotated dataset consisting of real-world and simulated point clouds is required. To minimize other sources of potential domain shift, such as scenario discrepancy, and to isolate the remaining domain shift, a dataset pairing is required that contains the same environment, agents, and scenarios in both domains. Therefore, the IAC dataset generated for the domain shift analysis is employed, as it fulfills the requirements. The details of the IAC dataset are described in Section 5.2, and only the specifics and differences are outlined in the following.

In general, the IAC dataset contains 3D annotated point clouds from both real-world and simulated environments, which are essential for assessing the domain adaptation via the downstream perception task, i.e. for training and evaluating using object detection networks. For the upstream domain adaptation approach, which adapts the point clouds on the object-level, only object point clouds of the source and target domains are needed. To this end, the object point clouds are extracted as explained in Subsection 6.2.1 and depicted in Figure 6.1. As defined in Subsection 6.2.2, a minimum number of points per object point cloud must be adhered to. This number is empirically set at 512 points per object point cloud, resulting in a total of 556 object point clouds per domain. The 556 real-world samples are divided into 528 for training, 14 for validation, and 14 for testing, and this target dataset is referred to as *real* dataset. The number of validation and testing samples is intentionally low compared to the number of training samples, since validation and testing only evaluate the performance of reconstructing the target domain, but do not evaluate the domain adaptation. Splits and correlation of scene and object point clouds are visualized in Figure 6.4. The 556 simulated samples are used entirely for domain adaptation inference with the trained model and are re-injected into the scene point cloud once adapted at the object-level. This source dataset is denoted as *sim* dataset, and the adapted dataset is denoted as *sim-to-real object full* dataset, i.e. showing the correlation of *source-to-target*.

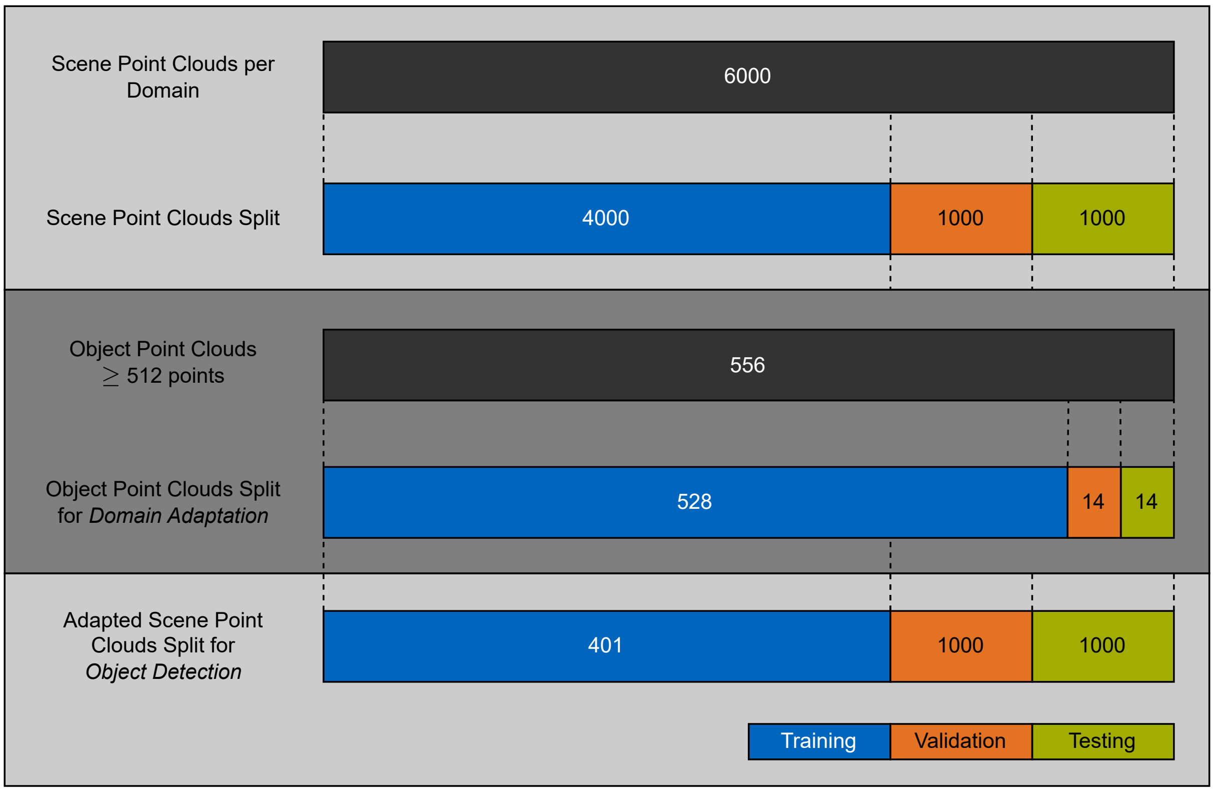

Scene Point Clouds per Domain
6000
Scene Point Clouds Split
4000
1000
1000
Object Point Clouds ≥ 512 points
556
Object Point Clouds Split for Domain Adaptation
528
14
14
Adapted Scene Point Clouds Split for Object Detection
401
1000
1000
Training
Validation
Testing

Figure 6.4: Dataset splits for scene and object point clouds of the sim-to-real scenario- and distribution-aligned IAC dataset.

6.3.2 Model Training and Evaluation

The domain adaptation model is trained in an adversarial manner in which both the generator and the discriminator are trained simultaneously. The generator aims to deceive the discriminator, while the discriminator tries to distinguish between the real and generated fake samples. The training process is schematically illustrated in Algorithm 1. Following the common practice of GAN training, the discriminator is updated more frequently than the generator during training, because the discriminator typically converges faster and requires more updates to provide meaningful gradients to the generator. In detail, the ratio between the discriminator and the generator updates is set to 5:1, i.e. the discriminator is updated five times for every generator update. The downsampling size δ is empirically set to seven, which will be analyzed in a parameter sensitivity analysis. The patch size λ_{patch} used as input for the discriminator is also set to seven, as both values should be equal, as explained before. The number of patches n_{patch} passed to the discriminator is based on the size of the variable input point cloud and is chosen to be equal to $\frac{1}{4}$ of the number of points of the object point cloud $\mathbf{O}_{i,k}^{\text{T,rec}}$ or $\mathbf{O}_{i,k}^{\text{T}}$. The batch size is limited to one, as the input number of points varies between each object point cloud and hence, can not be stacked. Both generator and discriminator networks are trained using the AdamW optimizer [227] with a learning rate of 10^{-4} using the PyTorch framework [219]. Every setup undergoes training for a total of 100 epochs, beyond which there is no noticeable decrease in validation loss.

Algorithm 1 Training Procedure for LiDAR Domain Adaptation

```
1:  Input: Target dataset $\mathbf{T}$, target validation dataset $\mathbf{T}_{\mathrm{val}}$, number of epochs $n_{\mathrm{epoch}}$, iterations/epoch $n_{\mathrm{it}}$,
    learning rate $\alpha$, downsampling size $\delta$, patch size $\lambda_{\mathrm{patch}}$, number of patches $n_{\mathrm{patch}}$, update ratio $r_{\mathrm{update}}$.
2:  Output: Trained generator $G_\Theta$ and discriminator $D_\Theta$.
3:  Initialize $G_\Theta$, $D_\Theta$.
4:  for $epoch = 1$ to $n_{\mathrm{epoch}}$ do
5:    for $iteration = 1$ to $n_{\mathrm{it}}$ do
6:      Sample a batch $\mathbf{O}^{\mathrm{T}}_{i,k}$ from $\mathbf{T}$.
7:      if ($iteration$ MOD $r_{\mathrm{update}}$) = 0 then
8:        Downsample: $\mathbf{O}^{\mathrm{T,fps}}_{i,k} \leftarrow \mathrm{FPS}(\mathbf{O}^{\mathrm{T}}_{i,k}, \delta)$
9:        Reconstruct: $\mathbf{O}^{\mathrm{T,rec}}_{i,k} \leftarrow G_\Theta(\mathbf{O}^{\mathrm{T,fps}}_{i,k})$
10:       Calculate $G_\Theta$ reconstruction loss: $\mathcal{L}^{\mathrm{rec}}_{\mathrm{G}_\Theta} \leftarrow d_{\mathrm{CD}}\left(\mathbf{O}^{\mathrm{T,rec}}_{i,k}, \mathbf{O}^{\mathrm{T}}_{i,k}\right)$
11:       Update $G_\Theta$ with $\mathcal{L}^{\mathrm{rec}}_{\mathrm{G}_\Theta}$, $\alpha$.
12:     end if
13:     $patches_{\mathrm{fake}} \leftarrow \mathrm{PATCH\ EXTRACTION}(\mathbf{O}^{\mathrm{T,rec}}_{i,k}, \lambda_{\mathrm{patch}}, n_{\mathrm{patch}})$
14:     $patches_{\mathrm{real}} \leftarrow \mathrm{PATCH\ EXTRACTION}(\mathbf{O}^{\mathrm{T}}_{i,k}, \lambda_{\mathrm{patch}}, n_{\mathrm{patch}})$
15:     Train $G_\Theta$ adversarially: $p_{\mathrm{fake}} \leftarrow D_\Theta(patches_{\mathrm{fake}})$
16:     Calculate $G_\Theta$ adversarial loss: $\mathcal{L}^{\mathrm{adv}}_{\mathrm{G}} \leftarrow -\log(p_{\mathrm{fake}})$
17:     Update $G_\Theta$ with $\mathcal{L}^{\mathrm{adv}}_{\mathrm{G}}$, $\alpha$.
18:     Train $D_\Theta$: $p_{\mathrm{real}} \leftarrow D_\Theta(patches_{\mathrm{real}})$
19:     Calculate $D_\Theta$ loss: $\mathcal{L}_{\mathrm{D}} \leftarrow -\log(p_{\mathrm{real}}) - \log(1 - p_{\mathrm{fake}})$
20:     Update $D_\Theta$ with $\mathcal{L}_{\mathrm{D}}$, $\alpha$.
21:     Update $\alpha$ according to scheduler policy (cosine annealing).
22:   end for
23:   Perform validation on $\mathbf{T}_{\mathrm{val}}$.
24: end for
```

The evaluation of the domain adaptation network is similar to the method presented in Chapter 5. That is, employing the object detection networks PointRCNN and PointPillars and reporting AP and recall evaluated at different ranges and different IoU thresholds. Unlike the evaluation in Chapter 5, the scene point clouds utilized for training object detection algorithms consist only of the subset that includes objects with more than 512 points, as these are the point clouds that are used for adaptation. Specifically, of the originally 4000 scene points clouds used for training object detection algorithms in Chapter 5, 401 scene points clouds of these are adapted and hence used for training the object detection algorithms. Using the remaining 3599 of the scene point clouds would make an isolated evaluation of the domain adaptation difficult, as the *sim-to-real object full* dataset would consist of a mix of adapted (401) and non-adapted (3599) point clouds.

To benchmark the proposed domain adaptation method, a simulated dataset is generated that includes a noise model that adds longitudinal Gaussian noise, having a standard deviation of $2\,\mathrm{cm}$, to every sensor ray, identical to the noise model presented in Subsection 5.5.4. For an equivalent comparison, only the rays that intersect with the objects are subjected to the noise model, similar to the presented domain adaptation method, which also only adapts the object point cloud, not the entire scene point clouds. This benchmark noise dataset is referred to as *sim noise* in the following. Alongside the quantitative evaluation, the adapted point clouds are compared qualitatively by visual inspection at the local level with the source and target domains.

In a further parameter sensitivity analysis, the influence of the discriminator is analyzed by training the domain adaptation network using the generator only and removing the discriminator. For this experiment, the loss function is reduced to only include the generator reconstruction loss, as expressed by

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{G}}^{\text{rec}}\left(O_{i,k}^{\text{T,rec}}, O_{i,k}^{\text{T}}\right). \tag{6.8}$$

The adapted dataset generated by this experiment is referred to as *sim-to-real no-GAN*. In two more experiments, the choice of the downsampling factor δ is analyzed. For this purpose, the domain adaptation network is trained with $\delta = 5$ or $\delta = 3$ instead of $\delta = 7$ to generate the two adapted datasets. The three datasets for the parameter sensitivity analysis are evaluated using the same method as described before.

6.4 Results

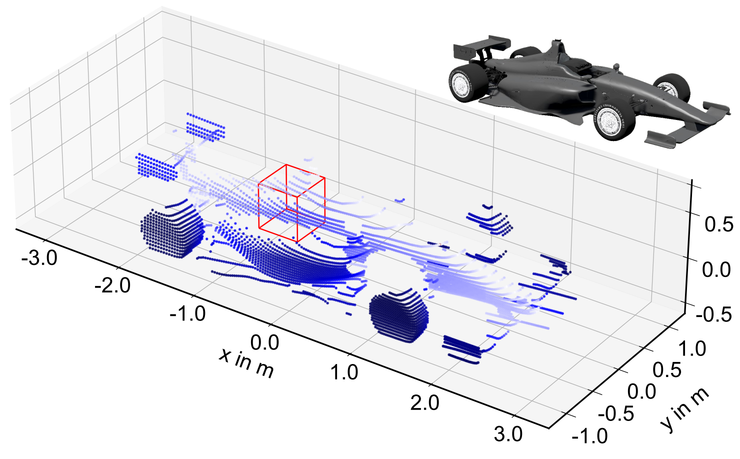
Before quantitative analysis of the results in Subsection 6.4.2, a visual inspection of the adapted object point clouds is carried out in Subsection 6.4.1. The results of the parameter sensitivity analysis in Subsection 6.4.3 conclude this chapter.

6.4.1 Qualitative Evaluation

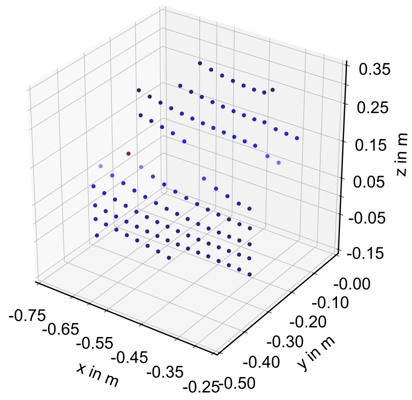
Figure 6.5 shows object points clouds from the source and target domains in Figure 6.5a and Figure 6.5e, respectively, and the adapted version of the source domain in Figure 6.5c. Alongside the object point clouds, the 3D plots in Figure 6.5b, Figure 6.5d, Figure 6.5f show the same cropped part extracted from the red boxes in the object point clouds. The point clouds highlight the disparities in sensor noise characteristics between the simulated and real-world point clouds, and that the adapted sim-to-real point cloud shows characteristics more similar to the target point cloud compared to the source point cloud, especially when analyzing the cropped areas. In detail, the simulated point cloud shows clear scan lines along the objects surface, whereas the points in the scan lines of the adapted and real-world point clouds show a more irregular pattern. For a more detailed qualitative comparison showing aggregated point clouds and comparing the latent feature space of the networks using t-SNE, the reader is referred to the corresponding publication [223]. To summarize the findings of the qualitative analysis, it indicates that the proposed domain adaptation is successful to an extent in reducing the prevalent sim-to-real domain shift.

6.4.2 Quantitative Evaluation

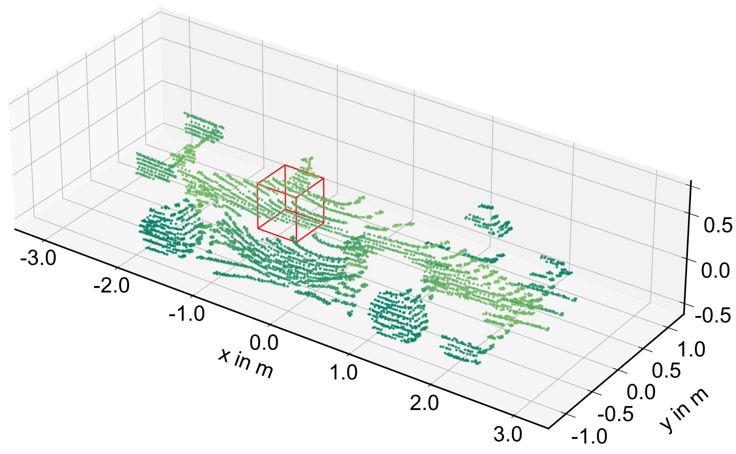
The object detection results of PointRCNN, which was trained using source, adapted, and target data, and evaluated on target data for the close- (Figure 6.6a) or full (Figure 6.6b) range, are depicted in Figure 6.6. Note that even though the same baseline dataset and network configuration are used, the results for source and target are different from the results in Chapter 5. The reason is that the datasets used in these experiments are composed of a subset of the original dataset, as explained in Subsection 6.3.1. Extensive results for both networks of average precision can be found in Table 6.1, and the recall results can be found in Table A.2 in Section A.2. To summarize the results, it can be stated that the sim-to-real domain shift of $8.63\,\%$ 3D AP (0.7) is reduced by almost $50\,\%$ to $4.36\,\%$ for PointRCNN in close-range. The results of



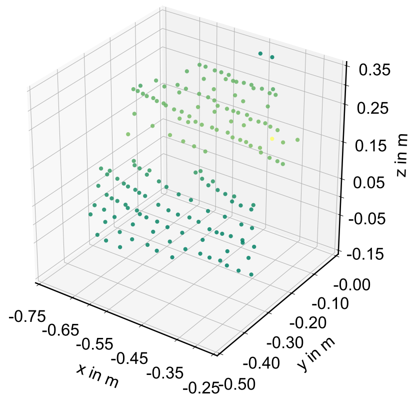
(a) *Sim*



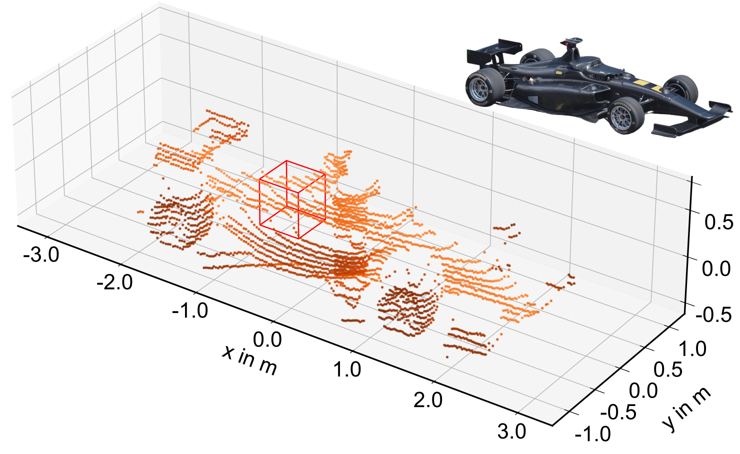
(b) *Sim* crop



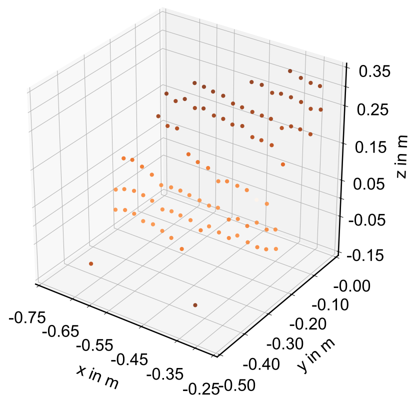
(c) *Sim-to-Real Object Full*



(d) *Sim-to-Real Object Full* crop



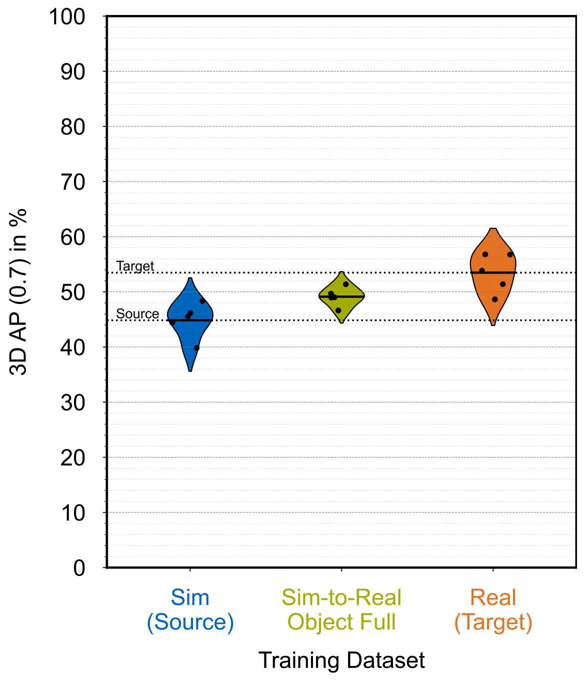
(e) *Real*



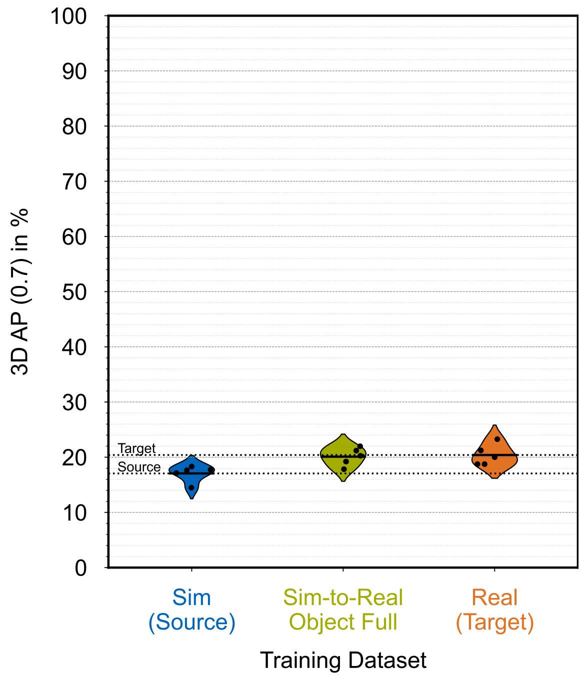
(f) *Real* crop

Figure 6.5: Individual object point clouds of the *sim*, *sim-to-real object full*, and *real* datasets are visualized for the comparative analysis of the domain adaptation technique. A detailed perspective of the local structure is given by the crops of the red boxes shown in (**b**,**d**,**f**) corresponding to the ones in (**a**,**c**,**e**), respectively. Simulated point clouds are depicted in blue shades, adapted point clouds are shown in green shades, and real-world point clouds are represented in orange shades. As a reference, the 3D model and the image of the object are included in (**a**,**e**), respectively (adapted from [223]).

PointPillars are similar, also showing a successful reduction of the sim-to-real domain shift using the proposed domain adaptation method. Compared with the baseline *sim noise* dataset, the proposed learning-based method consistently outperforms the integration of Gaussian noise in simulation, except in one case, that is, PointPillars 3D AP (0.5) evaluated in close-range.



(a) Close-Range $[0.0\,\mathrm{m}, 33.3\,\mathrm{m})$



(b) Full Range $[0.0\,\mathrm{m}, 100.0\,\mathrm{m})$

Figure 6.6: The 3D AP (0.7) for PointRCNN, which was trained and evaluated using either *sim*, *sim-to-real object full*, or *real* data and evaluated on *real* data (target). The horizontal lines represent the mean AP and the five points represent the five individual training cycles for each training and testing combination (adapted from [223]).

Table 6.1: The 3D AP (0.7) and 3D AP (0.5) for IoU thresholds of 70 % and 50 %, respectively. The PointRCNN and PointPillars networks are trained five times with the specified training dataset and then assessed on the test split of the *real* dataset. ↑: Higher values are better, with the highest value in each category highlighted in bold. All values in % (adapted from [223]).

Network	**Training Dataset**	**3D AP (0.7) ↑**		**3D AP (0.5) ↑**	
		Close-Range	Full Range	Close-Range	Full Range
PointRCNN	**Sim (Source)**	44.85±2.82	17.06±1.34	52.48±1.42	24.11±2.31
	Sim-Noise Object	41.34±3.89	15.58±2.83	49.14±5.07	22.17±4.47
	Sim-to-Real Object No-GAN	48.09±2.05	20.03±3.29	**57.94±0.81**	**30.58±1.75**
	Sim-to-Real Object $\delta = 3$	48.10±2.72	**20.39±3.44**	55.44±1.68	28.67±3.93
	Sim-to-Real Object $\delta = 5$	48.00±3.03	20.19±2.65	56.36±1.14	28.89±2.43
	Sim-to-Real Object Full	**49.12±1.53**	20.10±1.46	56.51±1.31	29.82±1.99
	Real (Target)	53.48±3.15	20.40±1.71	59.17±1.88	28.46±4.09
PointPillars	**Sim (Source)**	26.39±0.00	9.85±0.00	63.51±0.00	23.75±0.00
	Sim-Noise Object	31.46±0.00	11.80±0.00	**67.16±0.00**	28.45±0.00
	Sim-to-Real Object No-GAN	26.33±0.00	10.90±0.00	63.77±0.00	29.20±0.00
	Sim-to-Real Object $\delta = 3$	28.03±0.00	10.86±0.00	61.47±0.00	**30.01±0.00**
	Sim-to-Real Object $\delta = 5$	30.69±0.00	11.91±0.00	61.10±0.00	28.79±0.00
	Sim-to-Real Object Full	**38.03±0.00**	**14.41±0.00**	64.11±0.00	29.31±0.00
	Real (Target)	51.32±0.00	18.33±0.00	81.52±0.00	32.62±0.00

6.4.3 Parameter Sensitivity Analysis

The average precision results for the three experiments of the parameter sensitivity analysis are presented in Table 6.1, and the recall results can be found in Table A.2 in Section A.2.

The results for the analysis of the discriminator's impact on the domain adaptation are twofold. On the one hand, *sim-to-real no-GAN* achieves the best results for 3D AP (0.5) for both close- and full-range of PointRCNN. However, in all other cases, it is outperformed by the default method *sim-to-real object full* using the discriminator.

The parameter sensitivity analysis using a lower downsampling factor of $\delta = 3$ or $\delta = 5$ shows a tendency to decrease performance with a lower δ.

The corresponding publication [223] further analyzes switching source and target datasets, that is, training a real-to-sim domain adaptation using real as source and sim as target. In this setting, the domain adaptation method is trained to reconstruct simulated object point clouds and uses real-world object point clouds during inference to adapt them to the target style. This experiment shows that the domain adaptation method is also applicable to different source and target datasets, as it successfully reduces the real-to-sim domain shift. The complete results of this experiment can be found in [223].

7 LiDAR Domain Adaptation of 3D Point Clouds at the Scene-Level

In Chapter 5, the LiDAR sim-to-real domain shift is analyzed. This quantitative and qualitative analysis reveals a difference between the real-world and simulated data, especially notable when inspecting the point clouds at the object-level. As objects are especially relevant for the perception task, i.e., object detection, a method is developed and presented in Chapter 6 to adapt the domains of these object point clouds. The evaluation of this object-level domain adaptation method shows a successful reduction of the sim-to-real domain shift, paving the way for the use of synthetic data for perception network training. Nevertheless, a domain shift remains even when using the scenario- and distribution-aligned IAC dataset, as the adapted scene point clouds still contain points in the style of the source domain, i.e. all non-object points are based on the source domain. This leaves the open question of whether a domain adaptation at the scene-level can further reduce the remaining domain shift, which is expressed in Q4.

Q4: In what ways does the scene-level LiDAR domain adaptation address the broader challenges posed by the sim-to-real domain shift compared to object-level domain adaptation?

The goal of this chapter is to answer this question by extending the domain adaptation method presented in Chapter 6 to adapt point clouds at the scene-level, and the modifications needed for this will be explained in Section 7.1. To compare the scene-level domain adaptation with the object-level adaptation, the same dataset is used for the adaptation and evaluation of the experiments presented in Section 7.2, with the results presented in Section 7.3.

7.1 Method

The domain adaptation method to adapt point clouds at the scene-level is based on the object-level domain adaptation approach presented in Chapter 6. Instead of extracting object point clouds from scene point clouds, adapting them, and re-injecting them into their original locations within the scene point clouds, the entire scene point clouds are directly adapted by the network, as visualized in Figure 7.1. This approach is extended to handle scene point clouds, which substantially differ from object point clouds.

First, the average number of points per scene point cloud $\mathbf{X}_i$ is considerably higher than the average number of points per object point cloud $\mathbf{O}_{i,k}$. To tackle this challenge, this requires modifications of the training setup and parameters as described in Subsection 7.2.2.

Second, compared to object point clouds, scene point clouds are sparse and more irregular, i.e. the difference in point cloud density within a point cloud is higher, as the scene point clouds cover a larger area. This requires an extension of point cloud preprocessing, which will be explained in the following with the help of Figure 7.2. This figure shows a scene point cloud in the upper part in a BEV representation and two enlarged sections in the lower part. In both enlarged parts, a 2D circle with a diameter of $1\,\mathrm{m}$ is depicted

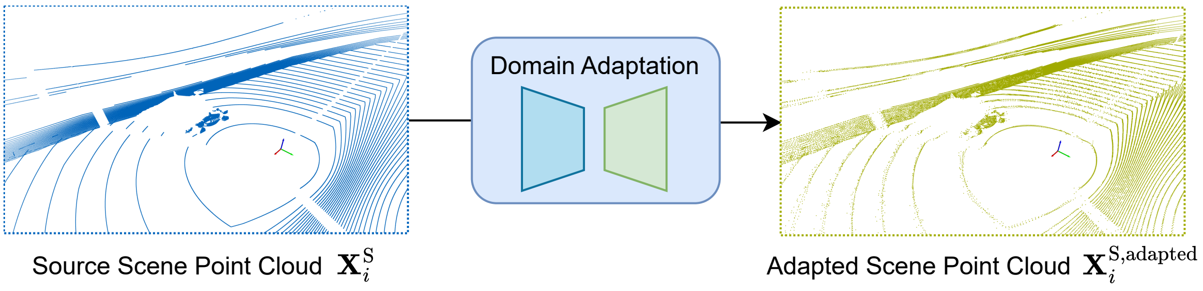

Domain Adaptation
Source Scene Point Cloud $\mathbf{X}_i^{\mathrm{S}}$
Adapted Scene Point Cloud $\mathbf{X}_i^{\mathrm{S,adapted}}$

Figure 7.1: High-level overview of the scene-based domain adaptation method. The domain adaptation network that has been trained adapts the source scene point clouds $\mathbf{X}_i^S$ (here: simulated data, blue) to generate the target-style scene point clouds $\mathbf{X}_i^{S,\text{adapted}}$ (here: real-world, green). The general architecture of the domain adaptation network is identical to the architecture of the domain adaptation network that operates at the object-level.

for visualization purposes. The enlarged right section covers an area close to the LiDAR sensor, so the density of the point cloud is high and many points fall within the projected circle. However, the left enlarged section covers a far range area, and thus the circle only includes two points due to low density. This means that low-density areas in the far range do not contain local structures and therefore can not be adapted by the domain adaptation method proposed in Chapter 6, which focuses on the adaptation of local structural differences of the source and target domains.

For this reason, a radius outlier filter is implemented as an upstream preprocessing algorithm to remove these points before processing the point cloud with the domain adaptation algorithm. This radius outlier filter is both used during training and inference of the domain adaptation network. The result of the radius outlier filter is also depicted in Figure 7.2. Here, black points are kept, and red points are removed before domain adaptation. To still keep the global structure during inference of the perception network, i.e. the object detector, the unaltered red points are appended to the adapted black points after domain adaptation. This allows for the application of the domain adaptation model to scene point clouds.

Apart from this modification, the rest of the domain adaptation neural network is identical to the network presented in Chapter 6.

7.2 Experimental Setup

This section describes the difference in the point cloud dataset used for the adaptation of the domain at the scene-level compared to the object-level in Subsection 7.2.1. Furthermore, the training specifics due to the higher average point number are explained in Subsection 7.2.2.

7.2.1 Dataset

The scenario- and distribution-aligned IAC dataset introduced in Section 5.2 is used for both training and testing of the domain adaptation algorithm, as well as for training and evaluating the object detection algorithms with the source, target, and domain adapted datasets. This is similar to the object-level domain adaptation; however, entire scene point clouds are used for training and inference of the domain adaptation algorithm. Thus, in addition to extracting 556 object point clouds that meet the minimum point requirement from the target (real-world) dataset for training, the complete set of 6000 scene point clouds from the real-world dataset is utilized. These 6000 scene point clouds are used for training, validation, and testing with

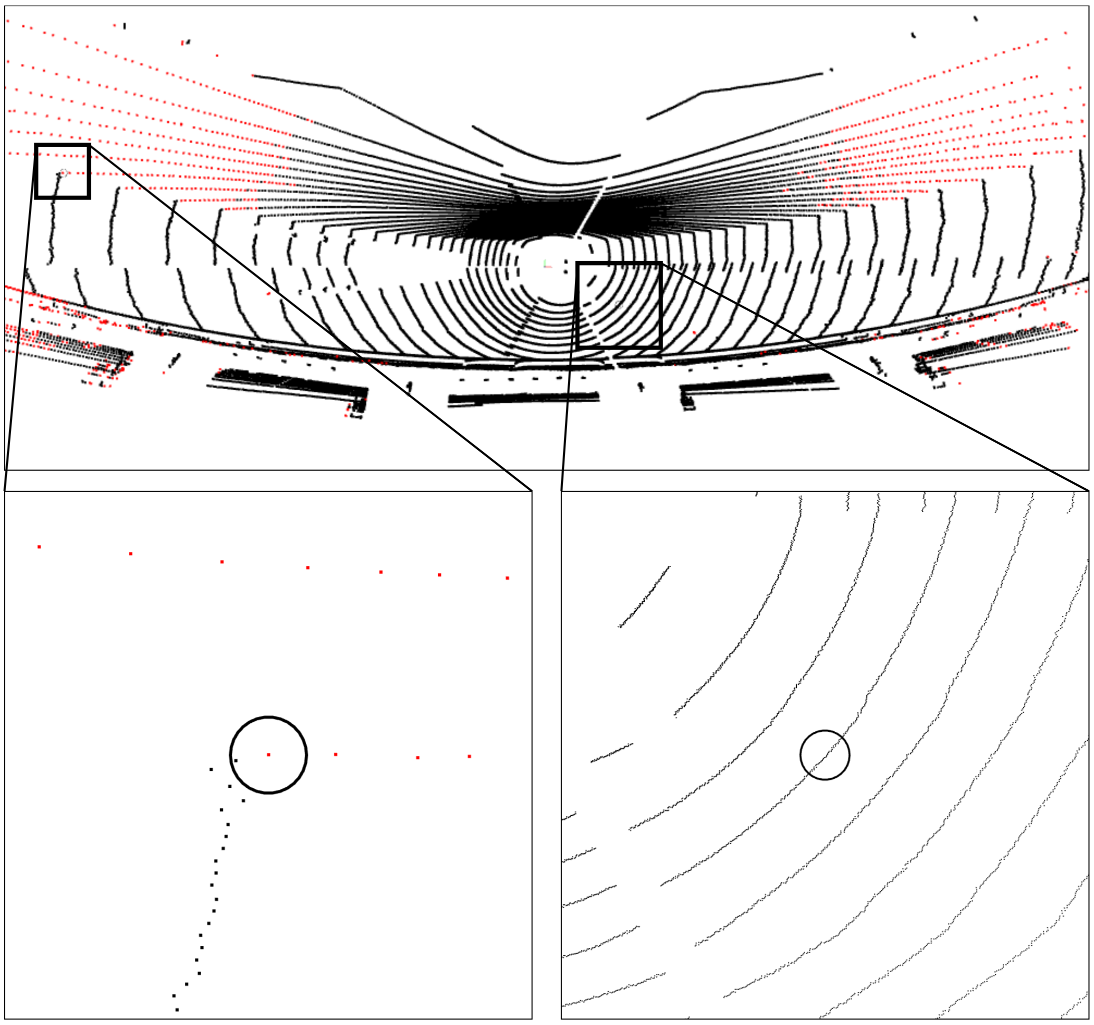


Figure 7.2: Example application of the radius outlier filter. The upper part shows a scene point cloud in BEV representation with two enlarged parts in the lower section. The black points indicate that they are passing through the radius outlier filter, whereas the red points are dropped by the filter and are not processed by the domain adaptation algorithm.

a split of 4000, 1000, and 1000, respectively. Once the domain adaptation algorithm is trained using the 4000 target point clouds, the inference is done using the entire source dataset, i.e. adapting 6000 source points.

A subset of the 6000 adapted point clouds is used for the qualitative evaluation in Subsection 7.3.1, as explained in the following and visualized in Figure 6.4. For an objective comparison with the object-level domain adaptation method of Chapter 6, of the 6000 adapted source scene point clouds, only 401 are selected for object detection training, and these are point clouds with the same index as those from the object-level domain adaptation evaluation. In summary, this means that Chapter 6 and Chapter 7 use the same scene point clouds for a comparable evaluation, with the difference that they are adapted either at the object-level or the scene-level, respectively.

7.2.2 Model Training and Evaluation

In general, the training process is similar to the training process when adapting the point clouds at the object-level. The adversarial training of the generator and discriminator is conducted as outlined in Algorithm 1, with the target dataset used for training consisting of scene point clouds instead of object point clouds. Despite the similarities in training, there exist a few differences which are explained in the following:

- First, the number of point clouds used for scene-level domain adaptation is approximately ten times higher than the number of point clouds for object-level domain adaptation. Therefore, the number of epochs for training is reduced from 100 to ten epochs to achieve a comparable number of total iterations. This further reduces the computation time during training.
- Second, the reconstruction loss $\mathcal{L}_{\mathrm{G}}^{\mathrm{rec}}$ of the overall generator loss $\mathcal{L}_{\mathrm{G}}$ is calculated using the chamfer distance d_{CD}. Instead of using the non-normalized formulation of the chamfer distance as used in Chapter 6, the normalized formulation defined in Equation 6.2 is used for scene-level domain adaptation. The reason is that the higher number of points per point cloud leads to a high value of $\mathcal{L}_{\mathrm{G}}^{\mathrm{rec}}$, which could vanish the influence of adversarial generator loss $\mathcal{L}_{\mathrm{G}}^{\mathrm{adv}}$, as the latter is always normalized and therefore does not scale with the number of points.
- Compared to the update ratio of 5:1 between the discriminator and the generator updates during training as in Subsection 6.3.2, a ratio of 2:1 is used for the scene-level domain adaptation. This decision is driven by the increased difficulty in generating target scene point clouds compared to object point clouds, and hence the necessity to provide the generator with more learning opportunities. To this end, a lower update ratio ensures balanced adversarial training.
- The number of patches n_{patch} is altered from $\frac{1}{4}$ to $\frac{1}{100}$ of the number of points of the input point cloud due to the higher number of points in scene point clouds.
- As the scene point clouds span a larger range than the object point clouds, they are clipped at a range of $100\,\mathrm{m}$ horizontally, instead of $3.3\,\mathrm{m}$ for object point clouds.
- Lastly, the radius outlier filter introduced in Section 7.1 is empirically set to contain at least two points within a radius of $0.5\,\mathrm{m}$.

7.3 Results

Before presenting the results of the evaluation using the object detection algorithms in Subsection 7.3.2, the adapted scene point clouds are qualitatively assessed in Subsection 7.3.1.

7.3.1 Qualitative Evaluation

Figure 7.3 contains simulated (source), sim-to-real scene full (adapted), and real-world (target) points clouds in BEV representation. The figure displays four scenarios of the dataset in the rows, and each subfigure contains an enlarged part to highlight the differences in the local structure. This visualization in the middle column shows the results of domain adaptation of entire scene point clouds. The results are twofold. First, the full view indicates that the adapted point clouds contain an overall noisy structure, similar to the characteristics of the target point clouds. Analyzing the local view, it shows that the local structure is visible in the point cloud rings, but compared to the real-world target, it shows more outlier points. However, the overall local structure is visually more similar to the local structure of the simulated source point clouds, indicating a form of domain adaptation.

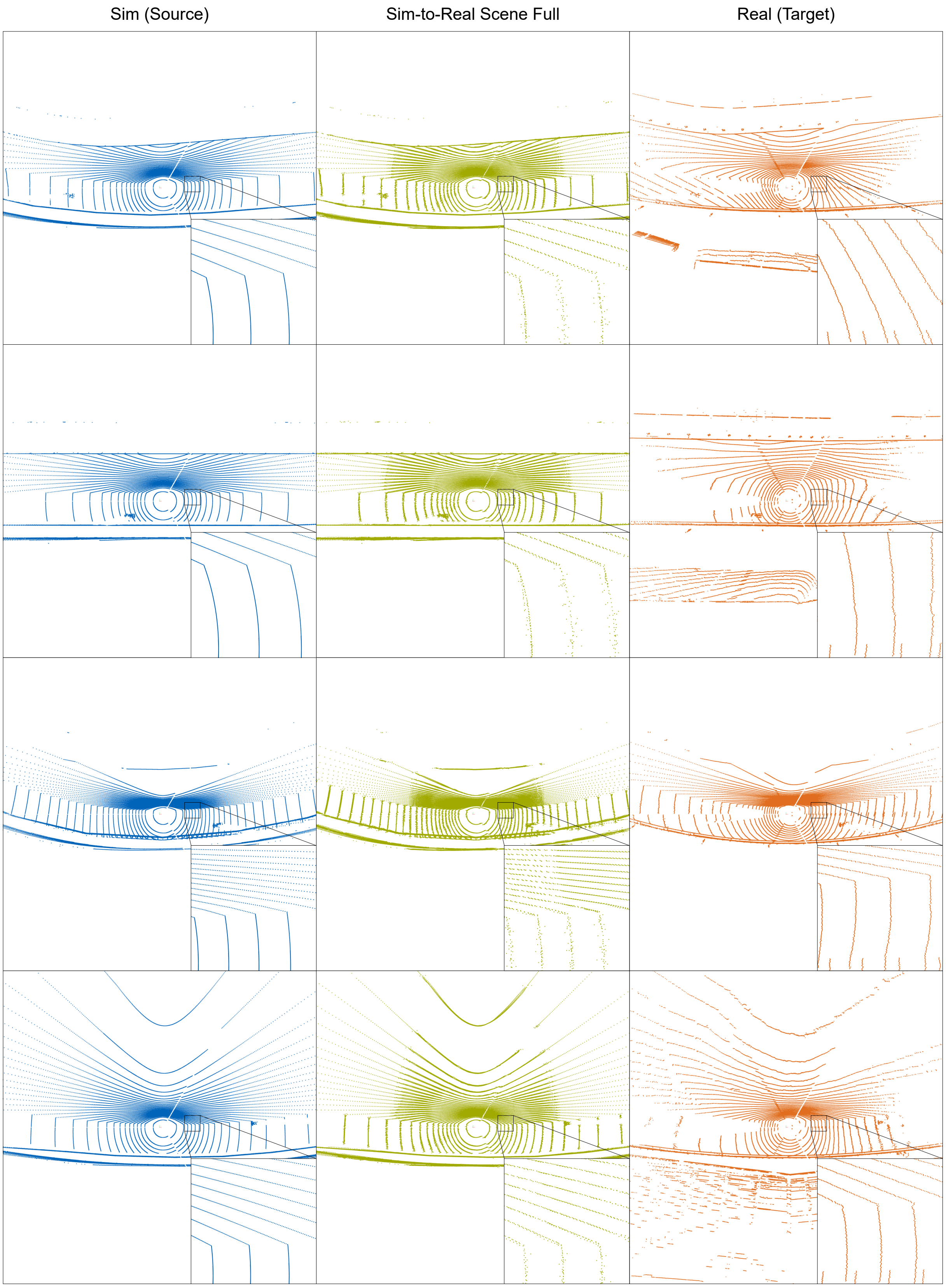


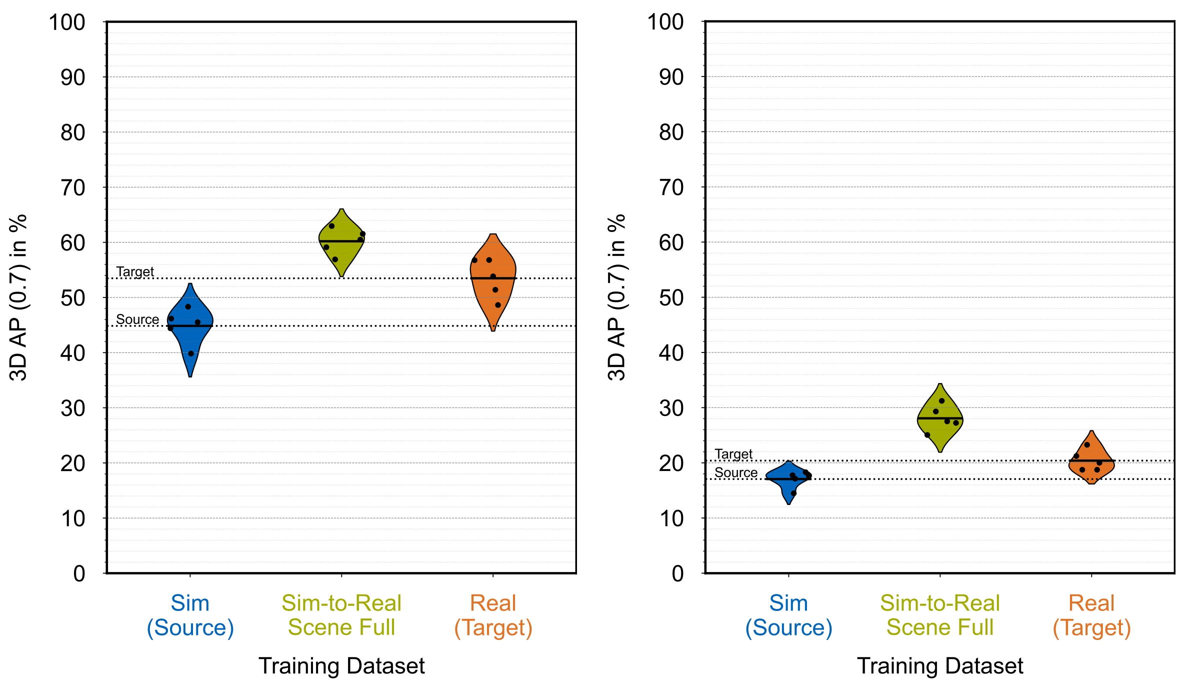
Figure 7.3: Qualitative results of the sim-to-real domain adaptation at the scene-level. Each row represents a different scenario, and the columns within each row represent the sim (source), sim-to-real scene full (adapted), and real-world (target) domains.

7.3.2 Quantitative Evaluation

The quantitative results of the object detection evaluation for the average precision are in Table 7.1, and the recall results can be found in Table A.3 in Section A.3. The 3D AP (0.7) for PointRCNN is further depicted in Figure 7.4 for close- and full range. The plots show that in both ranges, the sim-to-real scene full (adapted) dataset not only achieves a higher average precision than the source dataset but also outperforms the target dataset. This will be discussed in Section 9.1 and further compared with the results of the object-level adaptation from Chapter 6. The results for PointPillars show a mix of out-performance of the target dataset and performance close to the target dataset, but still outperforming the source dataset by a large margin.

Table 7.1: The 3D AP (0.7) and 3D AP (0.5) for IoU thresholds of 70 % and 50 %, respectively. The PointRCNN and PointPillars networks are trained five times with the specified training dataset and then assessed on the test split of the *real* dataset. ↑: Higher values are better, with the highest value in each category highlighted in bold. All values in %.

Network	**Train Dataset**	**3D AP (0.7) ↑**		**3D AP (0.5) ↑**	
		Close-Range	Full Range	Close-Range	Full Range
PointRCNN	**Sim (Source)**	44.85±2.82	17.06±1.34	52.48±1.42	24.11±2.31
	Sim-to-Real Scene $\delta = 7$	47.60±1.13	20.51±2.84	52.75±0.73	31.91±2.37
	Sim-to-Real Scene $\delta = 5$	46.41±6.36	20.94±5.93	52.59±4.36	29.58±7.06
	Sim-to-Real Scene Full	**60.20±2.07**	**28.07±2.08**	**69.70±1.07**	**42.54±1.81**
	Real (Target)	53.48±3.15	20.40±1.71	59.17±1.88	28.46±4.09
PointPillars	**Sim (Source)**	26.39±0.00	9.85±0.00	63.51±0.00	23.75±0.00
	Sim-to-Real Scene $\delta = 7$	**50.62±0.00**	**20.48±0.00**	**80.57±0.00**	37.36±0.00
	Sim-to-Real Scene $\delta = 5$	43.49±0.00	18.48±0.00	78.36±0.00	36.56±0.00
	Sim-to-Real Scene Full	44.20±0.00	20.37±0.00	79.80±0.00	**37.45±0.00**
	Real (Target)	51.32±0.00	18.33±0.00	81.52±0.00	32.62±0.00



(a) Close-Range $[0.0\,\mathrm{m}, 33.3\,\mathrm{m})$ (b) Full Range $[0.0\,\mathrm{m}, 100.0\,\mathrm{m})$

Figure 7.4: The 3D AP (0.7) for PointRCNN, which was trained and evaluated using either *sim*, *sim-to-real scene full*, or *real* data and evaluated on *real* data (target). Horizontal lines represent the mean AP and the five points represent the five individual training cycles per training and testing combination.

7.3.3 Parameter Sensitivity Analysis

Similarly to the parameter sensitivity analysis of object-level adaptation in Subsection 6.4.3, the influence of the downsampling factor δ on the scene-level domain adaptation is analyzed.

Figure 7.5 shows the qualitative evaluation of the two additional adapted datasets using $\delta = 7$ and $\delta = 5$. The datasets of the parameter sensitivity analysis are compared with the sim-to-real scene full dataset using $\delta = 3$ and with the target real-world dataset. The four rows show cropped examples from different point clouds, and within each row, the same scene is depicted. The visual comparison of the three downsampling factors shows that for scene domain adaptation, a smaller downsampling factor aligns better with the target domain, as the point cloud rings exhibit fewer outliers and are overall more consistent.

Quantitative results for $\delta = 7$ and $\delta = 5$ are shown in Table 7.1, along with the results of the original sim-to-real scene full dataset with $\delta = 3$. Both additional datasets outperform the source simulated dataset, hence the domain adaptation is successful in both cases. For PointPillars, the average precision of sim-to-real scene $\delta = 7$ is on par with the performance of sim-to-real scene full. The comparison of these two datasets in the case of PointRCNN, however, shows a noteworthy better performance of sim-to-real scene full with the lower $\delta = 3$.

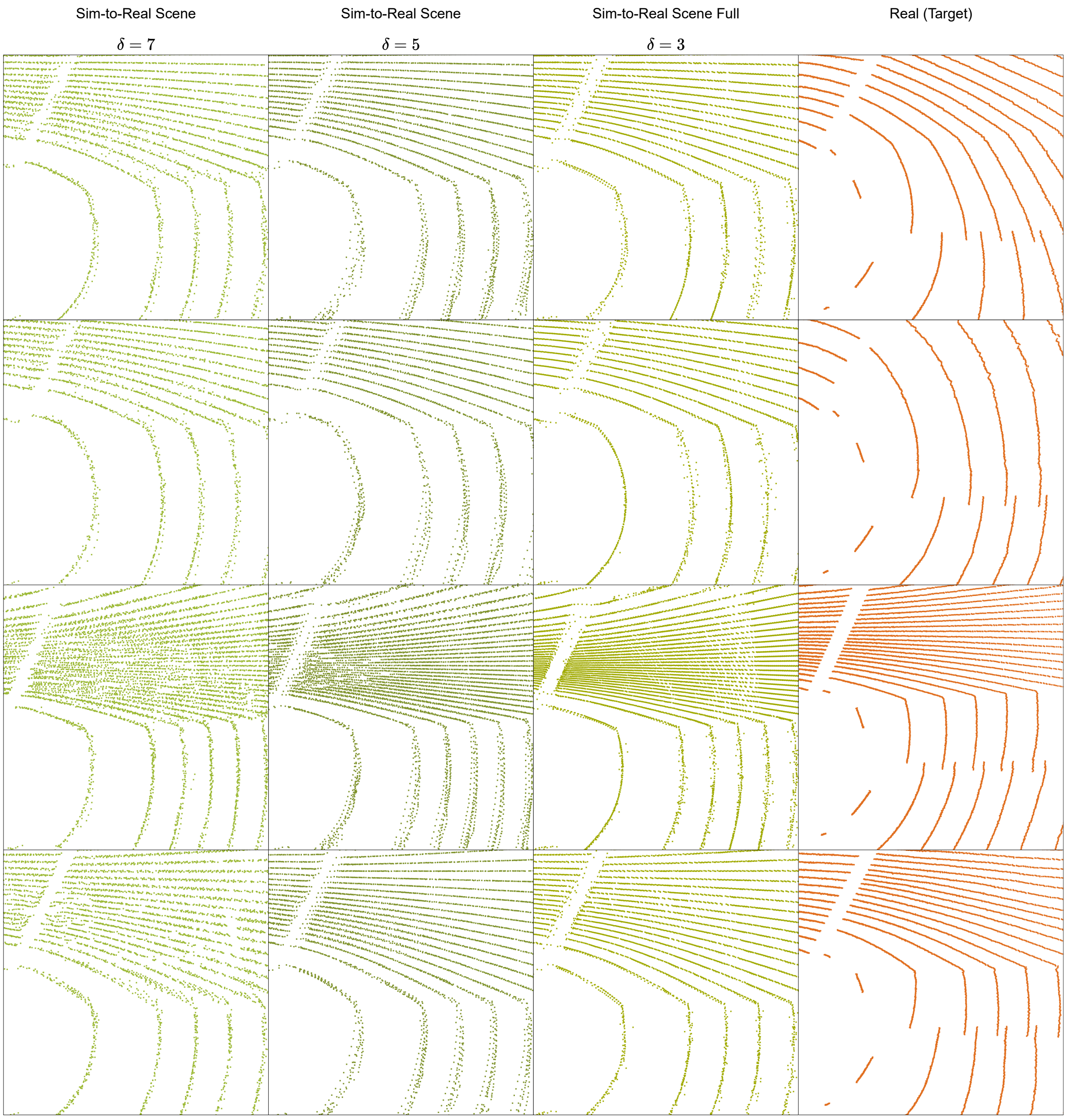


Figure 7.5: Qualitative results of the parameter sensitivity analysis of the sim-to-real domain adaptation at the scene-level. Each row represents a different scenario, and the columns within each row represent the four domains sim-to-real scene ($\delta = 7$), sim-to-real scene ($\delta = 5$), sim-to-real scene full ($\delta = 3$), and real-world (target).

8 LiDAR Domain Adaptation of 3D Point Clouds from Non-Distribution-Aligned Data

The previous chapters consider the topic of domain shift and domain adaptation using the scenario- and distribution-aligned IAC dataset. This has the advantage that the sim-to-real domain shift is isolated to focus on the local characteristics of the point clouds and neglects the global differences between synthetic and real-world data, such as different scenarios, agents, environments, and sensor characteristics. The developed domain adaptation method has been shown to reduce domain shift using either an object-level adaptation as in Chapter 6 or adapting entire scene point clouds as in Chapter 7 for the previously introduced sim-to-real distribution-aligned IAC dataset. As this distribution-aligned IAC dataset is a special case and usually not available or feasible to generate for application of AVs on public roads, this chapter delves into the sim-to-real domain adaptation of non-distribution-aligned data. The goal is to prove that the developed domain adaptation method is capable of generalization to other datasets, hence, increasing the potential applicability of the domain adaptation algorithm by answering the research question Q5.

Q5: How can LiDAR domain adaptation techniques be developed and applied when source and target data distributions are not aligned?

To this end, the method of domain adaptation of 3D point clouds from non-distribution-aligned data is explained in Section 8.1, before introducing the datasets and highlighting their differences in Subsection 8.2.1. Similarly to the previous chapters, the results in Section 8.3 are structured into qualitative and quantitative evaluations.

8.1 Method

In this chapter, both object- and scene-level domain adaptation are analyzed and evaluated. The goal is to analyze whether the findings from the object- and scene-level domain adaptation using distribution-aligned IAC dataset are transferable to non-distribution-aligned data. Similarly to the scene-level domain adaptation introduced in Chapter 7, the domain adaptation for non-distribution-aligned data is also based on the domain adaptation algorithm originally presented in Chapter 6. To this end, the object-level domain adaptation algorithm presented in Figure 6.2 is used with a few modifications, which are explained in the following, broken down into object- and scene-level domain adaptation.

During inference of object-level domain adaptation, the object point clouds $\mathbf{O}_{i,k}^{\mathrm{S}}$ are first extracted from the scene point clouds $\mathbf{X}_{i}^{\mathrm{S}}$. In the case of the non-distribution-aligned dataset, the number of objects k per scene point cloud is on average higher than one, compared to only one object per scene point cloud in the distribution-aligned IAC dataset. The consequence of this is that multiple object point clouds need to be adapted before re-injecting them into their original location within the scene point clouds, as depicted in Figure 8.1. As the domain adaptation pipeline is executed offline, the adaptation of the objects can be performed sequentially, and this does not influence the network design of the domain adaptation algorithm.

The number of objects per scene does not impact the domain adaptation at the scene-level, and therefore the method is identical to the method presented in Chapter 7 for the distribution-aligned IAC dataset, as depicted in Figure 8.2.

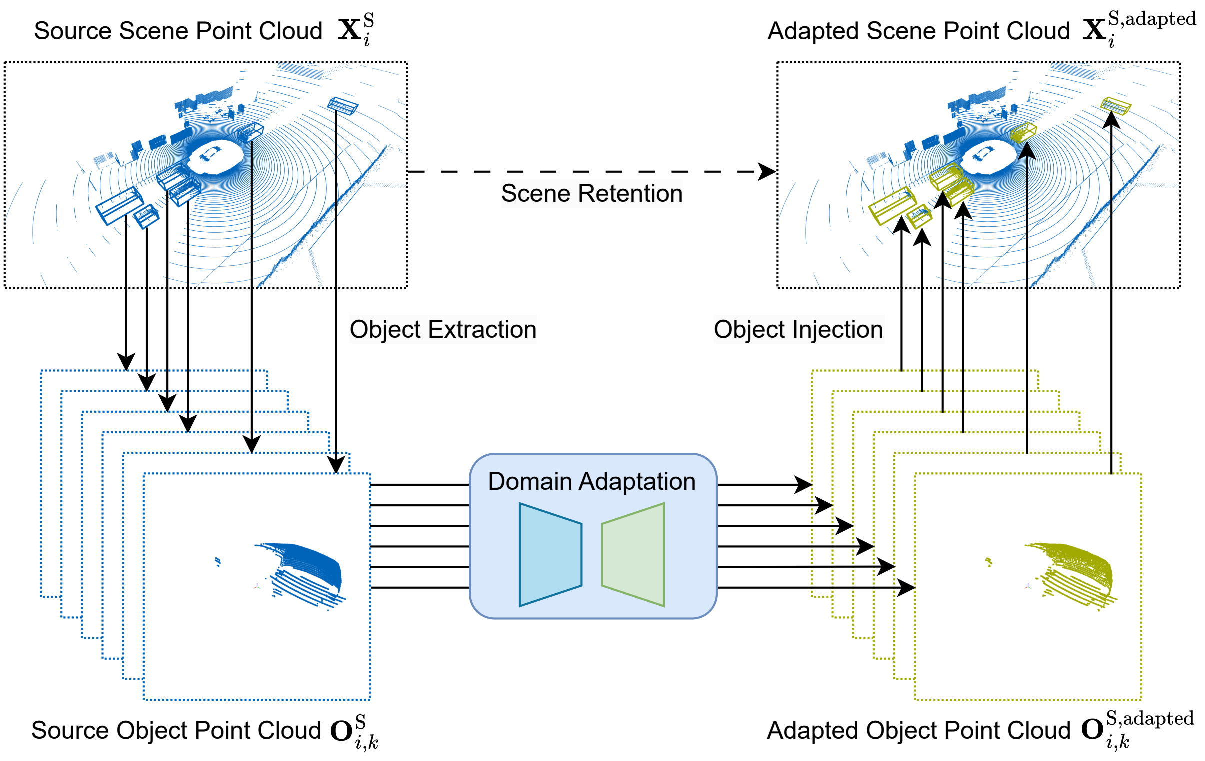

Source Scene Point Cloud $\mathbf{X}_i^{\mathrm{S}}$
Adapted Scene Point Cloud $\mathbf{X}_i^{\mathrm{S,adapted}}$
Scene Retention
Object Extraction
Object Injection
Domain Adaptation
Source Object Point Cloud $\mathbf{O}_{i,k}^{\mathrm{S}}$
Adapted Object Point Cloud $\mathbf{O}_{i,k}^{\mathrm{S,adapted}}$

Figure 8.1: High-level overview of the object-based domain adaptation method for non-distribution-aligned data. The first step comprises the extraction of the object point clouds $\mathbf{O}_{i,k}^{S}$ from a source scene point cloud $\mathbf{X}_i^{S}$ (here: simulated data, blue). The trained domain adaptation network subsequently adapts the object point clouds to generate target-style object point clouds (here: real-world, green) $\mathbf{O}_{i,k}^{S,\mathrm{adapted}}$, which are re-injected in their initial locations within the source scene point cloud. The resulting output $\mathbf{X}_i^{S,\mathrm{adapted}}$ combines the original source scene point cloud with the adapted object point clouds.

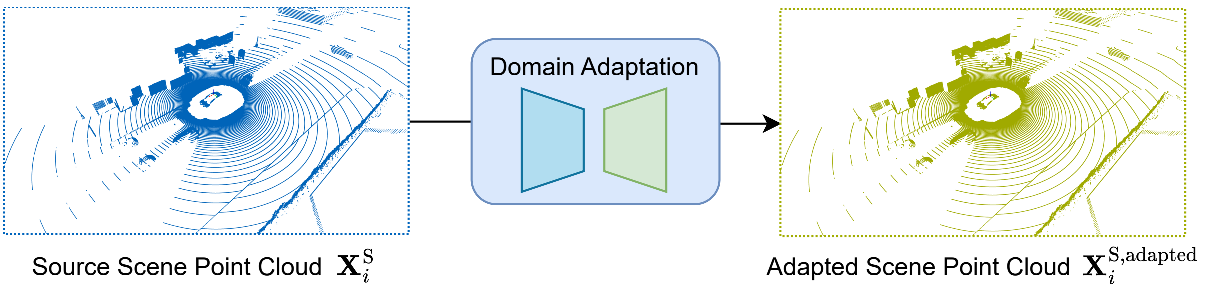

Domain Adaptation
Source Scene Point Cloud $\mathbf{X}_i^{\mathrm{S}}$
Adapted Scene Point Cloud $\mathbf{X}_i^{\mathrm{S,adapted}}$

Figure 8.2: High-level overview of the scene-based domain adaptation method for non-distribution-aligned data.The domain adaptation network that has been trained adapts source scene point clouds $\mathbf{X}_i^{S}$ (here: simulated data, blue) to generate target-style scene point clouds $\mathbf{X}_i^{S,\mathrm{adapted}}$ (here: real-world, green).

8.2 Experimental Setup

The experimental setup for the domain adaptation using the non-distribution-aligned datasets is similar to the experimental setup for domain adaptation using the distribution-aligned IAC dataset. Therefore, only the major differences will be explained in the following, which is mainly focused on the dataset being used.

8.2.1 Dataset

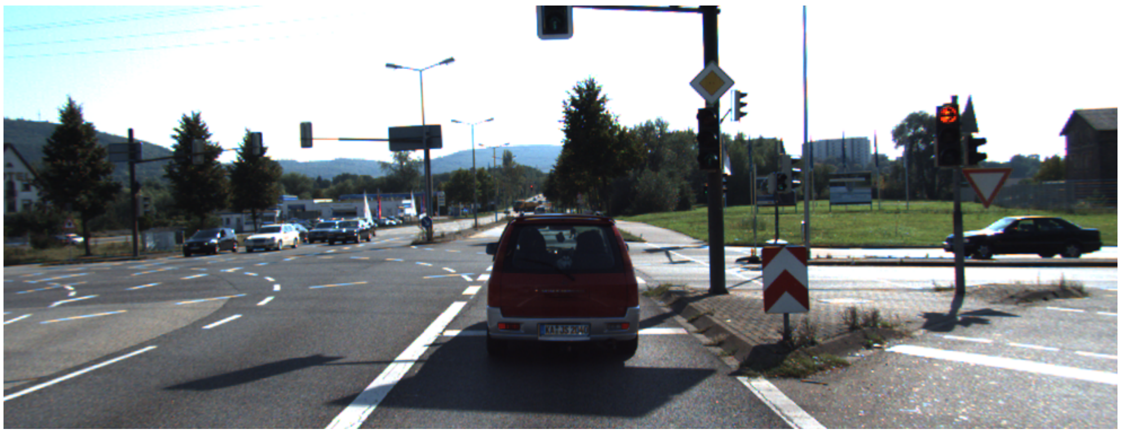
The selection of the synthetic and real-world datasets is conducted independently, as their distributions in terms of scenario or distribution alignment do not need to correspond.

Consequently, KITTI [37] is chosen as a real-world dataset, which contains around 15,000 annotated point clouds covering urban, rural, and highway areas in Germany. The main reasons for choosing KITTI are that first, point clouds are recorded using a single $360°$ rotating LiDAR, which coarse characteristics can be replicated in a simulation. Second, the object detection networks PointRCNN and PointPillar used for the evaluation are developed and optimized for the KITTI benchmark [36] and thus can be used without additional manual fine-tuning.

The synthetic dataset is generated using the simulation environment CARLA [90]. Although the All-In-One Drive dataset [91] is a publicly available dataset that is also based on CARLA, its LiDAR configuration differs strongly from the configuration of the real-world KITTI dataset. Thus, a new synthetic dataset is generated in CARLA with a sensor setup similar to KITTI. The ego vehicle is equipped with a $360°$ rotating LiDAR sensor. The sensor position on the ego vehicle is identical to the sensor position on the vehicle of the KITTI dataset. The LiDAR sensor spins with a frequency of $10\,\mathrm{Hz}$. To avoid collecting similarly looking point clouds from subsequent time steps, only every 20th point cloud is saved, i.e. every 2 seconds. The ego vehicle is autonomously driven and controlled by the integrated traffic manager in CARLA. The maps are populated with other agents, which also drive autonomously. The agents are limited to vehicles from a selection of around 25 actors and do not contain pedestrians or cyclists. 5,000 annotated point clouds each are collected on the three CARLA maps "Town1", "Town4", and "Town5", which cover a village, a highway, and an urban environment, respectively. As the intensity channel of the LiDAR sensor model is not validated, it is not recorded. This non-distribution-aligned dataset pair is referred to as CARLA-KITTI dataset in the following.

Figure 8.3 shows examples of scenes that are commonly found in the KITTI or CARLA datasets. This highlights the differences in datasets, with different environments, agents, scenery, weather, and road layouts, For example, KITTI is recorded in Germany, whereas CARLA replicates the style of cities found in the USA. Figure 8.4 shows the positions and their distributions of the vehicles of the entire datasets in BEV representation. Only vehicles in the FoV of the front camera are annotated in the KITTI dataset, which explains the limited triangular area found in the KITTI plot (Figure 8.4a). Apart from this difference, the distributions of the positions also show major differences. For example, KITTI contains many vehicles parked on both sides of the road, which explains the peaks of the distribution along the y axis. Furthermore, the distribution in CARLA along the x axis is less consistent, with many vehicles located between $x = 15\,\mathrm{m}$ and $x = 30\,\mathrm{m}$. This can be explained by situations in which the ego vehicle stops at red traffic lights and recording crossing traffic, which often occurs due to the regular road layout of the simulation.

Table 8.1 shows a quantitative statistical comparison of both datasets. The attributes belonging to the scene point clouds are more similar than the attributes regarding the objects, with an increased mean count of objects for each scene point cloud and a greater mean count of points for each object point cloud identified in the synthetic dataset. In addition, the object dimensions differ, since the vehicles in the synthetic dataset are on average $17\,\%$ longer and $20\,\%$ wider than the vehicles in the real-world dataset, indicating a country-to-country domain shift.


(a) KITTI [37]

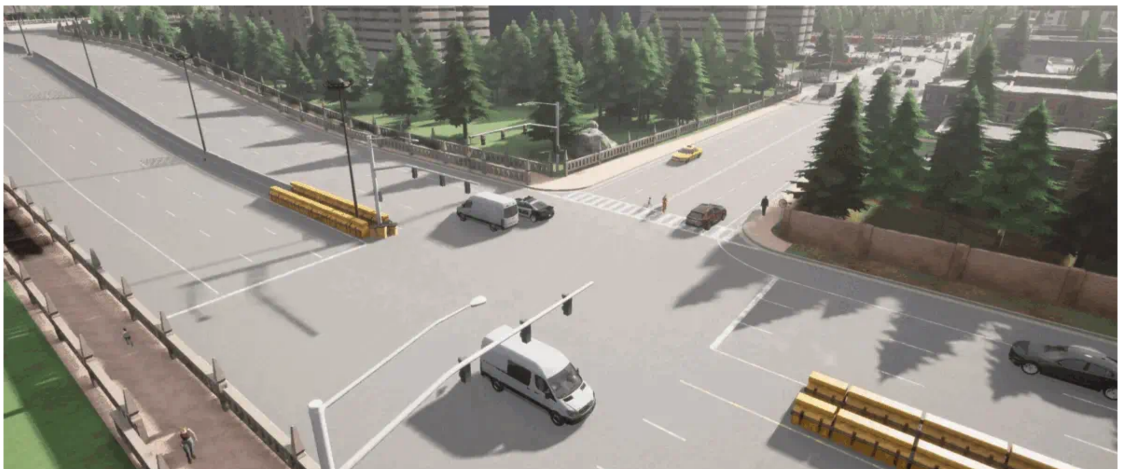
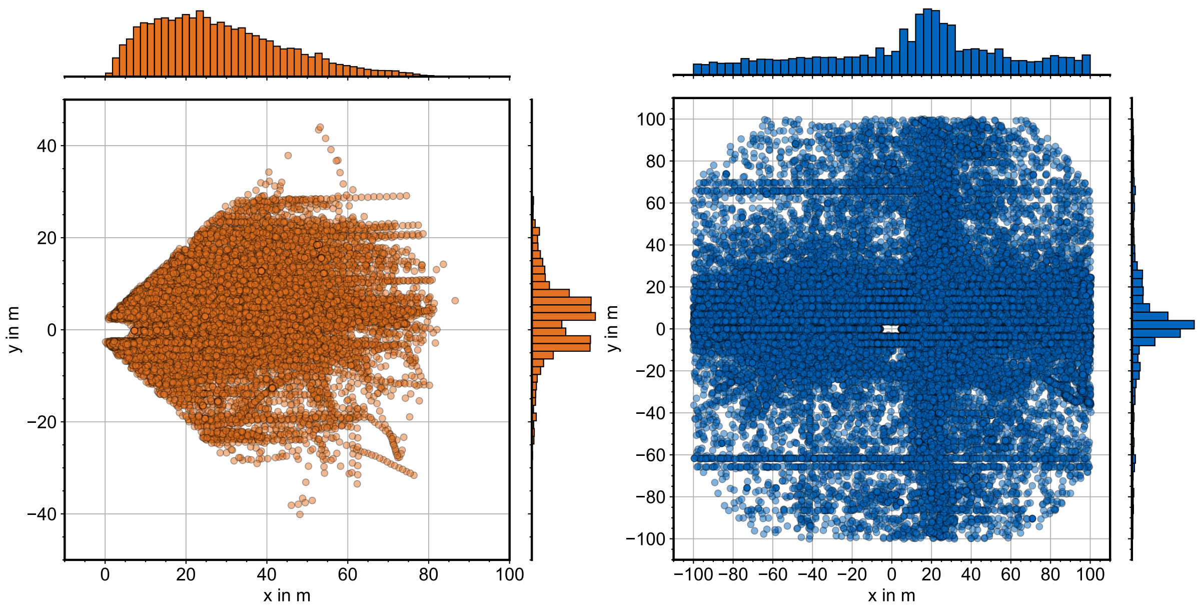

(b) CARLA [228]

Figure 8.3: Examples of environments, agents, and scenarios found in the KITTI or CARLA datasets.



(a) KITTI (b) CARLA

Figure 8.4: Bird's-eye-view plot of the object (car) locations extracted from the annotations of the KITTI or CARLA datasets.

Table 8.1: Comparison of key statistical characteristics between the KITTI and CARLA datasets. Every value is computed from the complete training set of each dataset, utilizing the whole point cloud range up to $100\,\mathrm{m}$.

Attribute			KITTI	CARLA
Scene point cloud range	min	x in m	-80.00	-100.00
		y in m	-79.98	-100.00
		z in m	-31.68	-26.88
	max	x in m	79.99	100.00
		y in m	79.99	100.00
		z in m	2.91	4.19
	mean	x in m	-0.07	0.28
		y in m	0.65	0.31
		z in m	-1.15	-1.23
Number of points per scene point cloud	min		78596	104445
	max		128467	129958
	mean		119225	125821
Number of objects per scene point cloud	min		0	0
	max		19	24
	mean		3.84	7.23
Number of points per object point cloud	min		0	0
	max		14466	20807
	mean		394	618
Object dimensions	min	h in m	1.14	1.30
		w in m	1.14	1.48
		l in m	2.19	2.21
	max	h in m	2.48	2.05
		w in m	2.04	2.16
		l in m	6.67	5.57
	mean	h in m	1.53	1.56
		w in m	1.63	1.95
		l in m	3.88	4.55

Overall, this analysis shows a notable difference between the datasets with a combination of sim-to-real, country-to-country, weather-to-weather, and dataset-to-dataset domain shifts.

The splits into training, validation, and testing for the scene and object point clouds are depicted in Figure 8.5. Following the common split of the KITTI dataset, 3,712 point clouds are used for training and the remaining 3,769 point clouds for testing, and no validation point clouds are used. The split for the object point clouds is skewed towards a high number of training samples compared to validation and testing, as the latter sets only evaluate the reconstruction, but not adaptation performance. This is done similarly to the split of the distribution-aligned IAC dataset.

8.2.2 Model Training and Evaluation

The adversarial training of the domain adaptation network is performed with a set of parameters similar to the training using the distribution-aligned IAC dataset in Chapter 6 for object-level and Chapter 7 for scene-level and follows the training process of Algorithm 1. The detailed parameters can be found in Table 8.2. The main differences are the number of point clouds used per epoch, and hence, the adapted number of epochs. As explained in Subsection 8.2.2, the reconstruction loss $\mathcal{L}_{\mathrm{G}}^{\mathrm{rec}}$ is calculated using the normalized formulation of the chamfer distance d_{CD} as defined in Equation 6.2. Here, this formulation is used not only for the scene-level domain adaptation but also for the object-level domain adaptation. The reason is that for the non-distribution-aligned CARLA-KITTI dataset, the absolute difference between the maximum (14466)

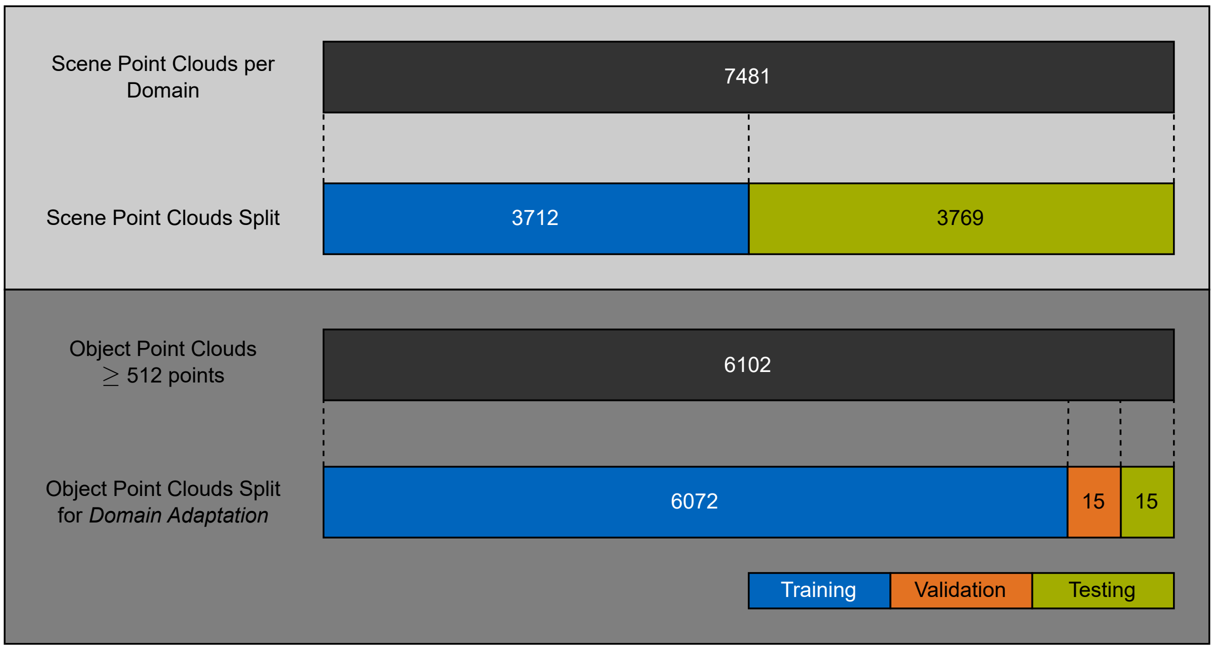

Scene Point Clouds per Domain
7481
Scene Point Clouds Split
3712
3769
Object Point Clouds ≥ 512 points
6102
Object Point Clouds Split for *Domain Adaptation*
6072
15
15
Training
Validation
Testing

Figure 8.5: Dataset splits for scene and object point clouds of the CARLA-KITTI dataset.

and minimum (set to 512) number of points per object point cloud is larger than the maximum (4959) and minimum (set to 512) number of points per object point cloud for the distribution-aligned IAC dataset (compare Table 8.1 and Table 5.1).

For evaluation with the object detection algorithms PointPillars and PointRCNN, the training and evaluation range of the source, adapted, and target datasets are limited to the range of the annotated KITTI dataset.

Table 8.2: Training parameters for non-distribution-aligned object-level and scene-level domain adaptation.

	Object-level	**Scene-level**
Number of point clouds	6072	3712
Epochs	20	10
Update ratio discriminator:generator	5:1	2:1
Point cloud range	$3.3\,\mathrm{m}$	$100.0\,\mathrm{m}$
Downsampling size δ	5	7
Patch size λ_{patch}	5	7
Number of patches n_{patch}	$\frac{1}{4}$	$\frac{1}{100}$
Radius outlier filter: number of points	3	2
Radius outlier filter: radius threshold	$0.2\,\mathrm{m}$	$0.5\,\mathrm{m}$

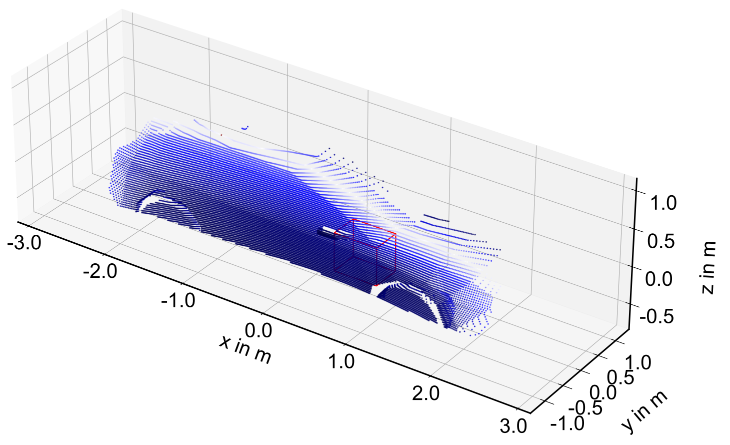
8.3 Results

In the same manner as in the previous chapters, the results are structured in a first visualization of qualitative examples of the adaptation and t-SNE graphs before presenting quantitative measures generated by evaluating object detectors.

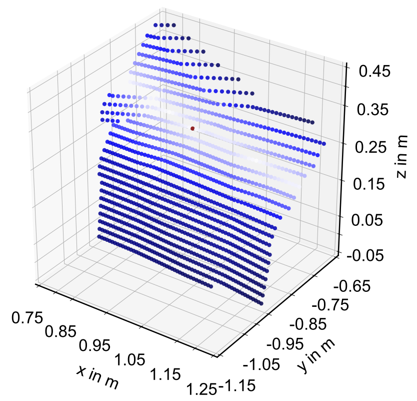
8.3.1 Qualitative Evaluation

Figure 8.6 shows three object point clouds from the *CARLA*, *CARLA-to-KITTI Object*, and *KITTI* dataset, along with a cropped section of each point cloud for visualization of the local structure. The *CARLA-to-*

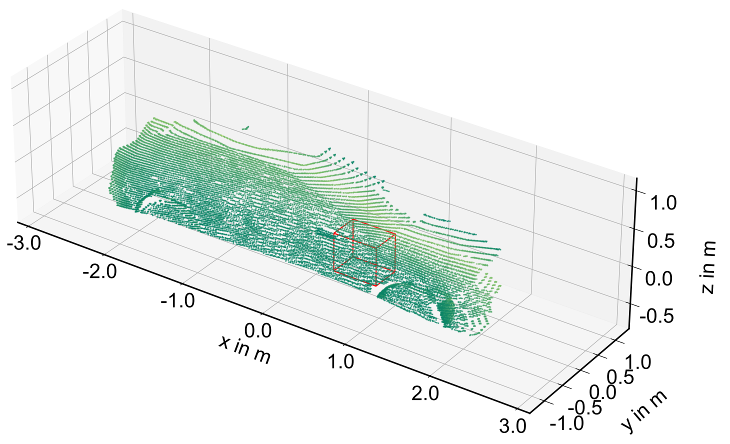
KITTI Object point cloud is the adapted version of the *CARLA* point cloud, while the *KITTI* point cloud is a random but similarly selected object in the dataset since a aligned object does not exist. The cropped sections in Figure 8.6b and Figure 8.6c highlight the output of the domain adaptation, in which local noise is injected onto the surface of the object, similar to the noise found in the real-world point cloud in Figure 8.6f.



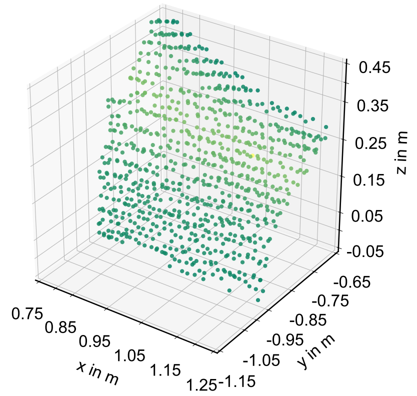
(a) *CARLA* (source)



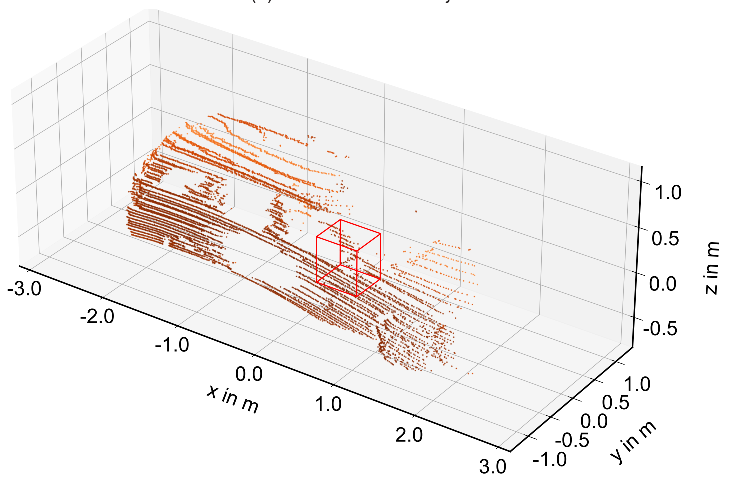
(b) *CARLA* crop



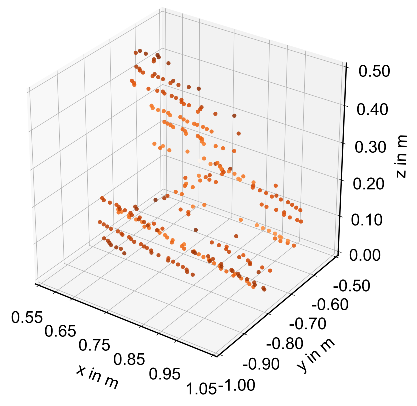
(c) *CARLA-to-KITTI Object*



(d) *CARLA-to-KITTI Object* crop



(e) *KITTI* (target)



(f) *KITTI* crop

Figure 8.6: Individual object point clouds of the *CARLA*, *CARLA-to-KITTI Object*, and *KITTI* datasets are visualized for the comparative analysis of the domain adaptation technique. A detailed perspective of the local structure is given by the crops of the red boxes shown in (**b**,**d**,**f**) corresponding to the ones in (**a**,**c**,**e**), respectively. Simulated point clouds are depicted in blue shades, adapted point clouds are shown in green shades, and real-world point clouds are represented in orange shades.

Qualitative examples of the scene-level domain adaptation are depicted in Figure 8.7. The three *CARLA-to-KITTI Scene* point clouds in the middle column are the adapted versions of the *CARLA* point clouds in the respective row on the left side. The BEV shows that the point clouds do not change on a global level after adaptation and only differ on a local level, as seen in the scaled-up sections in each plot.

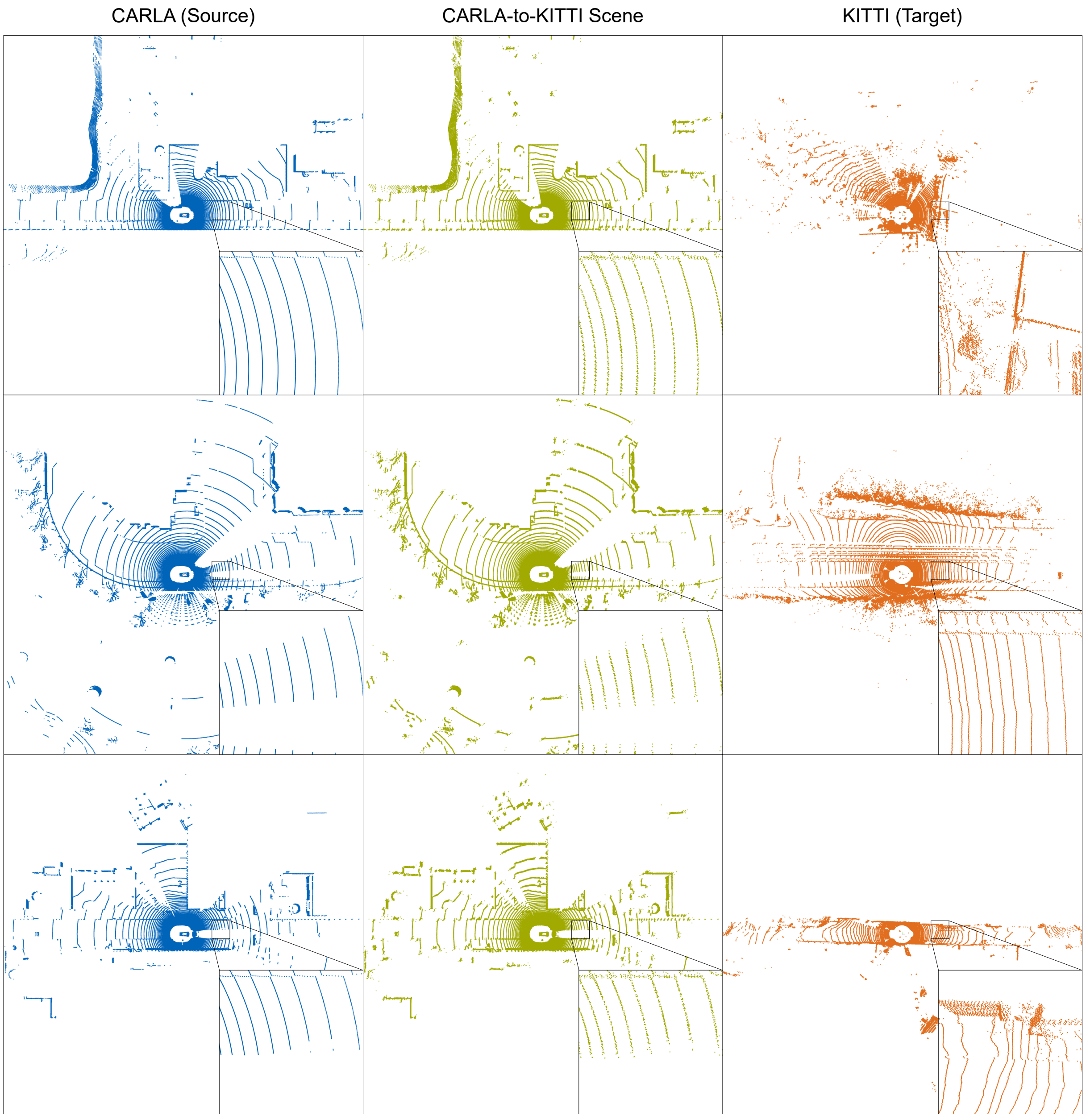
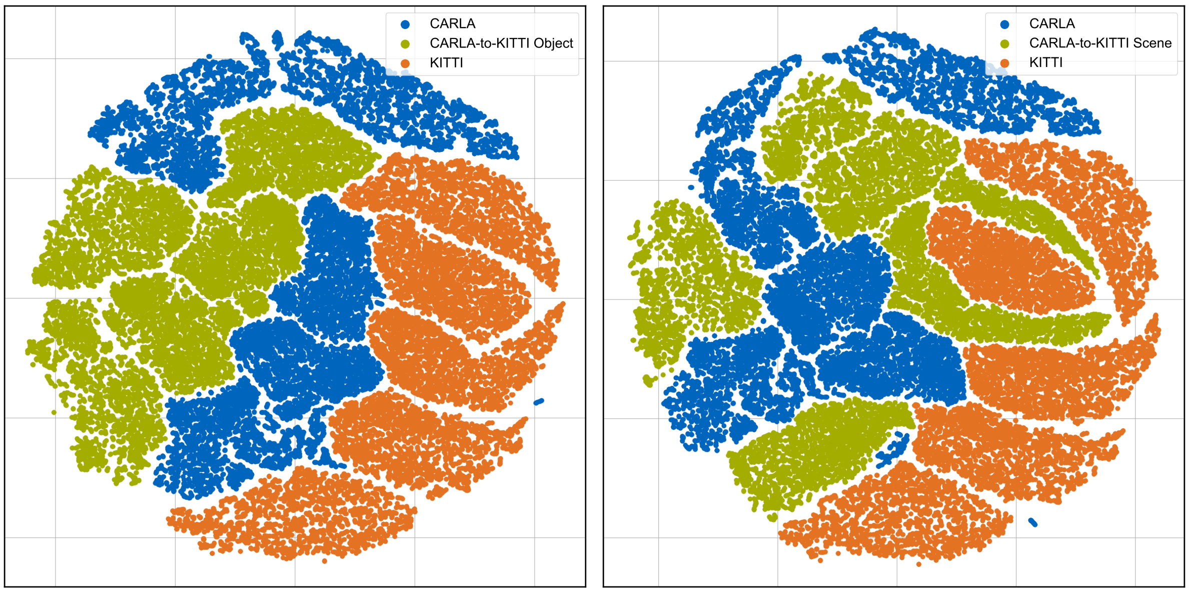


Figure 8.7: BEV plots of point clouds from *CARLA* (source), *CARLA-to-KITTI Scene* (adapted), and *KITTI* (target) datasets.

Figure 8.8 shows two t-SNE plots, in which either *CARLA-to-KITTI Object* (Figure 8.8a) or *CARLA-to-KITTI Scene* (Figure 8.8b) are compared with the source and target datasets. This side-by-side comparison shows that the scene-level domain adaptation is closer to the target than the object-level domain adaptation is to the target, indicating a better domain adaptation of the scene-level domain adaptation.

8.3.2 Quantitative Evaluation

Quantitative results for both object-level and scene-level domain adaptation can be found in Table 8.3 for both PointRCNN and PointPillars and are visualized for PointRCNN in Figure 8.9. Further results for the



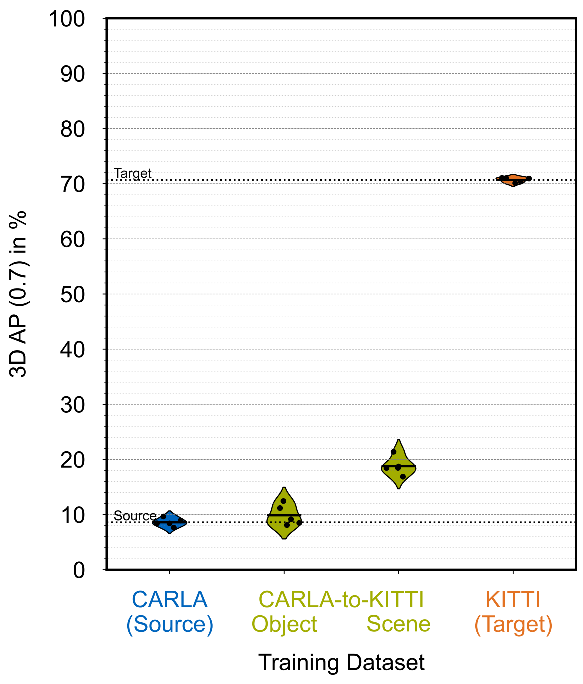
(a) *CARLA-to-KITTI Object* (b) *CARLA-to-KITTI Scene*

Figure 8.8: A visualization using t-SNE technique of the latent feature space generated by PointRCNN, trained separately on *CARLA* (source), *CARLA-to-KITTI Object*, *CARLA-to-KITTI Scene*, or *KITTI* (target) data for five training sessions each. Each point represents a feature vector produced by the network's inference when analyzing a single point cloud of the *KITTI* test set.

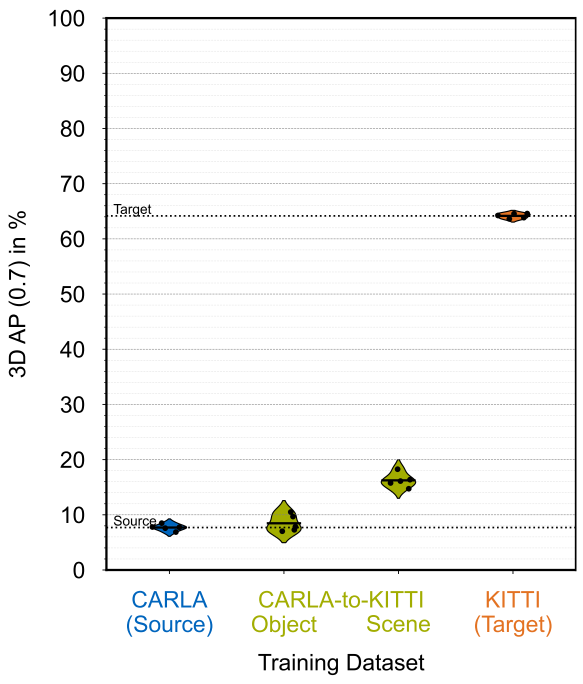
recall are located in Table A.4 in Section A.4. As expected, the scene-level domain adaptation outperforms the object-level domain adaptation in all categories. In detail, the object-level domain adaptation reduces the sim-to-real domain shift by only single-digit percentages, and in some cases (PointPillars) increases the sim-to-real domain shift. In general, domain adaptation is more successful for PointRCNN than for PointPillars. The scene-level domain adaptation successfully reduces the sim-to-real domain shift, e.g. by doubling the 3D AP (0.7) of the source dataset from $8.61\,\%$ to $18.78\,\%$ in close-range. Although domain adaptation is successful in most cases, the distance from the performance of the target dataset is still large and can not be fully reduced by the presented domain adaptation method.

Table 8.3: CARLA-to-KITTI: The 3D AP (0.7) and 3D AP (0.5) for IoU thresholds of $70\,\%$ and $50\,\%$, respectively. The PointRCNN and PointPillars networks are trained five times with the specified training dataset and then assessed on the test split of the *KITTI* dataset. ↑: Higher values are better, with the highest value in each category highlighted in bold. All values in %.

Network	**Train Dataset**	**3D AP (0.7) ↑**		**3D AP (0.5) ↑**	
		Close-Range	Full Range	Close-Range	Full Range
PointRCNN	**CARLA (Source)**	8.61±0.67	7.69±0.52	45.96±1.52	38.25±1.33
	CARLA-to-KITTI Object	9.88±1.67	8.46±1.38	46.18±2.57	38.94±2.46
	CARLA-to-KITTI Scene	**18.78±1.46**	**16.24±1.15**	**55.14±1.28**	**47.20±1.12**
	KITTI (Target)	70.69±0.39	64.16±0.38	86.97±0.54	84.03±0.80
PointPillars	**Sim (Source)**	3.71±0.00	3.02±0.00	33.11±0.00	27.44±0.00
	CARLA-to-KITTI Object	3.61±0.00	3.04±0.00	30.86±0.00	26.81±0.00
	CARLA-to-KITTI Scene	**5.43±0.00**	**4.51±0.00**	**38.66±0.00**	**33.56±0.00**
	Real (Target)	48.76±0.00	44.58±0.00	82.06±0.00	82.47±0.00



(a) Close-Range $[0.0\,\text{m}, 33.3\,\text{m})$



(b) Full Range $[0.0\,\text{m}, 100.0\,\text{m})$

Figure 8.9: The 3D AP (0.7) for PointRCNN, which was trained and evaluated using either *CARLA* (source), *CARLA-to-KITTI Object* (adapted), *CARLA-to-KITTI Scene* (adapted), or *KITTI* (target) data and evaluated on *KITTI* data. The horizontal lines represent the mean AP and the five points represent the five individual training cycles for each training and testing combination.

9 Discussion and Future Work

This chapter discusses the method and results presented in the previous chapters of this thesis and answers the research questions. Furthermore, the open challenges that can be addressed in future work are presented in Section 9.2.

9.1 Discussion

This section answers the research questions, critically discusses the results by comparing them with the state of the art, and highlights the limitations of the approach. The discussion is split into five parts. The first part discusses the analysis of the sim-to-real domain shift addressed in Chapter 5. The second part combines the discussion of the domain adaptation for object- and scene-level adaptation using the distribution-aligned IAC dataset presented in Chapter 6 and Chapter 7, respectively. The third part deals with the discussion of the domain adaptation approach applied to point clouds from non-distribution-aligned data, which was presented in Chapter 8. The real-world application of perception algorithms and the significance of LiDAR domain adaptation for this purpose are discussed in Subsection 9.1.4. Lastly, Subsection 9.1.5 highlights the practical relevance of this thesis in terms of economic, scientific, and societal impact.

9.1.1 Analysis of the LiDAR Domain Shift

Chapter 5 on the analysis of the domain shift serves to answer the research questions Q1 and Q2.

> **Q1: How should a sim-to-real LiDAR dataset be designed to effectively capture the nature and extent of the sim-to-real domain shift, and what methodologies can be employed to quantify and analyze this shift through various performance indicators?**

The first part of Q1 is answered by introducing a method that consists of three steps. The first step involves the capturing of real-world LiDAR and precise RTK-corrected GNSS data from all agents, while the latter is used to automatically annotate the LiDAR point clouds. In the second step, a 3D simulation environment is generated that matches the real-world scenery, including a virtual LiDAR sensor model to generate synthetic LiDAR point clouds in simulation. The last step is to replay the real-world GNSS trajectories in the simulated environment and simultaneously generate the synthetic point clouds.

The second part of the research question Q1 is answered by presenting a method that utilizes object detection algorithms trained using the datasets of both domains and compares their performance when evaluating in the opposite domain. Similar methods exist in the state-of-the-art literature for domain shift quantification [107, 159]. However, the method presented in this thesis expands the state of the art for domain shift analysis in two ways. First, the method presented in this thesis quantifies the domain shift using the distribution-aligned dataset and hence narrows the remaining domain shift down to the local differences between the simulation and the real world. Second, the analysis is extended not only to a quantitative evaluation using the object detectors but also introduces a qualitative comparison by comparing the datasets using t-SNE plots and comparing visual examples at the object-level, which is possible due to the usage of the scenario- and distribution-aligned IAC dataset. Using the method developed to answer Q1, the sim-to-real domain shift for the specific dataset is approximately $14\,\%$ 3D AP (0.7).

Q2: What impact do simulation enhancements, such as the integration of sensor noise models, have on mitigating the sim-to-real domain shift in 3D LiDAR data?

To answer Q2, the virtual LiDAR sensor model is extended to include either sensor noise in longitudinal ray direction or integrates a point dropout by randomly removing $20\,\%$ of the points. The evaluation shows that sensor noise successfully minimizes the sim-to-real domain shift, whereas dropout increases the sim-to-real domain shift. These experiments were previously not found in state-of-the-art domain shift analysis and hence contribute to an extension of state-of-the-art knowledge in this field.

Although the results of the domain shift analysis are a valuable contribution to the state-of-the-art, this discussion will highlight the limitations of the analysis, which will be elaborated on in the following.

Even though the 3D simulation environment to record the synthetic data is derived from the real world, still there exist discrepancies between the datasets. For instance, the 3D simulation environment has missing static objects in the scenery outside of the driveable area, and therefore a difference in global structure is visible in the point clouds. In addition, the real-world dataset is semi-automatically annotated using the GNSS recordings of the ego vehicle and the other agents. The GNSS units offer a high precision using RTK-correction, but are neither synchronized nor triggered to capture the location of the vehicles at identical time stamps. Interpolation and a high recording frequency as proposed in this thesis can mitigate potential offsets between labels derived from the relative GNSS positions and the actual position of vehicles in the point clouds. Furthermore, the subsequent refinement step can improve the auto-labeling. However, there might still be an error that is more pronounced in higher ranges as the density of points drops and the accuracy of the refinement steps decreases. These inaccuracies are present only in the real-world dataset and not in the simulated dataset, as the latter is automatically annotated by the simulation engine itself.

Nevertheless, the statistical comparison shows that the overall similarity of both datasets in terms of point distribution is high, which justifies the utilization of the dataset pair in the following analysis and for domain adaptation experiments. Although these inaccuracies can have an impact on the sim-to-real domain shift, the generated sim-to-real dataset is still the dataset with the most similar distribution between the two domains found in the literature. This makes it suitable for the analysis of the domain shift as well as for the domain adaptation focusing on the local structures.

The auto-labeling pipeline to annotate the real-world point clouds only defines seven degrees of freedom per bounding box and neglects the roll and pitch angles. This is done because the object detection networks PointRCNN and PointPillars also disregard these angles. However, the dataset contains scenarios in which either the ego vehicle or another agent is on the banked turns of the race track and a relative roll and pitch angle exists between both vehicles. This can further lead to inaccuracies in the labeling as ground reflections can be included in the 3D bounding box. Nonetheless, this is present in both simulated and real-world data and does not influence the analysis of the sim-to-real shift.

Moreover, the scenario- and distribution-aligned IAC dataset only contains one object class, i.e. the class *race car*, and within this class, all objects are identical in size and shape. This reduces the task complexity for object detectors, as they only have to learn the characteristics of this single object. Nevertheless, as the results show, even with reduced complexity, the sim-to-real domain shift is still pronounced for this simple object detection task.

To summarize, the findings underscored the presence of a sim-to-real domain shift in LiDAR data, impacted by factors like sensor noise and point cloud dropout. The addition of noise in the virtual LiDAR sensor model showed a reduction in domain shift, indicating an important impact of sensor noise on the sim-to-real domain shift. The approaches adopted for analyzing domain shift expand upon existing literature by providing a detailed quantitative and qualitative analysis. Despite advances in improving the virtual LiDAR sensor model, the complete elimination of the sim-to-real domain shift is not achieved, indicating the complexity and multifaceted nature of the problem, hence motivating research in learning-based domain adaptation approaches.

9.1.2 LiDAR Domain Adaptation at the Object- and Scene-Level

In Chapter 6 and Chapter 7 on the domain adaptation at the object- and scene-level using the distribution-aligned IAC dataset, the research questions Q3 and Q4 are addressed.

Q3: How can LiDAR domain adaptation techniques be applied at the object-level to mitigate the impact of sim-to-real domain shift, and what are the specific challenges and solutions associated with object-level domain adaptation?

Q3 is answered by presenting a method adapting object-level point clouds based on adversarial training. The key to this method is the following three characteristics. First, the training is performed using the target data only, and the inference is performed using the source data only. This makes this method capable of scaling to an infinite quantity of source data without requiring retraining for each new source domain. Second, domain-invariant representation within the network is achieved by using FPS downsampling, which keeps the global structure and solely removes the local characteristics of point clouds. Lastly, adversarial training is based on comparing small point cloud patches instead of comparing the entire object point clouds, which further improves the adaptation process within the reconstruction network to focus on the local structures. Evaluation using the same method as introduced in Chapter 5 shows a successful reduction of the sim-to-real domain shift on the same IAC dataset, emphasizing the efficiency of the domain adaptation technique.

Q4: In what ways does the scene-level LiDAR domain adaptation address the broader challenges posed by the sim-to-real domain shift compared to object-level domain adaptation?

To answer Q4, the method from Q3 is extended to incorporate adaptation of scene point clouds. This is achieved by modifying the training setup and parameters, and by adding a point cloud preprocessing before the actual domain adaptation is executed in the reconstruction network. The upstream preprocessing consists of a radius outlier filter, which removes areas with low density from the adaptation process. The results of the scene-level domain adaptation show an improvement compared to the object-level adaptation. Moreover, the adapted dataset even outperforms the target dataset, which can have several reasons elaborated in the following:

1. A possible cause could be that the model trained on the target dataset is overfitting on the training split and does not generalize well to the testing samples of the target dataset. In contrast, the model trained on the adapted data might introduce more variation in the training samples to prevent overfitting. However, the validation loss during training does not show signs of overfitting.
2. Another cause for the outperforming of the adapted dataset could be that the domain adaptation process enhances certain features or introduces beneficial regularization effects, e.g. by highlighting critical features in the point cloud more effectively.
3. Another explanation could be that objects in higher ranges contain more points in the adapted dataset compared to the target dataset. This higher number of points makes the detection for the object detectors easier, potentially leading to higher metric scores.

4. The last possible cause could be that the choice of hyperparameters and optimization techniques for object detection training is in favor of the adapted dataset compared to the target dataset. These hyperparameters were adopted from the original implementations of PointRCNN and PointPillars, and their fine-tuning is an open topic for future work.

The final cause can be analyzed in future work but does not diminish the significance of the presented domain adaptation approach that reduces the sim-to-real domain shift.

The object- and scene-level domain adaptation methods both have limitations, which are elaborated on in the following. Training of the object-level domain adaptation is carried out using target object point clouds. Hence, the target dataset needs to be annotated to extract the objects from the scene point clouds. This is counter-intuitive at first, as the goal of domain adaptation is to remove the necessity of annotated target data. This limits the approach; however, only a small subset of the target data needs to be annotated. The advantage that arises from choosing this approach is that once the network is trained using the target data, it has the capability to adapt to an infinite amount of source data from various source domains, eliminating the requirement for retraining for each individual source domain.

Both object-level and scene-level domain adaptation can adapt point clouds with a variable point number, but necessitate a certain quantity of points as sparse point clouds lack substantial information about the structure on a local level. This limits especially the object-level domain adaptation in a way that objects with a lower number than the defined minimum number of points are not adapted, which are usually objects at higher distances.

9.1.3 LiDAR Domain Adaptation for Non-Distribution-Aligned Data

The last research question Q5 is addressed in Chapter 8.

Q5: How can LiDAR domain adaptation techniques be developed and applied when source and target data distributions are not aligned?

To this end, the developed domain adaptation method for both object- and scene-level adaptation is modified for usage with non-distribution-aligned datasets, in this case, the CARLA-KITTI dataset. Due to the higher number of objects in scene point clouds in the CARLA-KITTI dataset compared to the IAC dataset, the domain adaptation pipeline is modified to extract, adapt, and re-inject all objects per scene point cloud. Except for tuning the training parameters for the CARLA-KITTI dataset, the domain adaptation algorithm is the same as for use with the IAC dataset. The results show that a transfer of the domain adaptation approach from the distribution-aligned IAC dataset to the non-distribution-aligned CARLA-KITTI dataset is successful. The object-level domain adaptation achieves only a minor improvement over the source domain, whereas the scene-level domain adaptation manages to reduce the sim-to-real domain shift to a greater extent.

Nevertheless, it should be mentioned that despite the scene-level domain adaptation reducing the sim-to-real domain shift, a large sim-to-real domain shift is still present in the CARLA-KITTI dataset after adaptation. The reason for this is that the sim-to-real domain shift is a superposition of several individual domain shifts, such as country-to-country, weather-to-weather, sensor-to-sensor, and dataset-to-dataset. The domain adaptation method presented in this thesis only tackles the sensor-to-sensor domain shift which arises from the simplified virtual sensor model in simulation compared to the real-world sensor, and hence, can not fully reduce the sim-to-real domain shift in a non-distribution-aligned dataset.

After answering the derived research questions Q1 to Q5, the primary research question can be answered.

PRQ: How can domain adaptation approaches directly applied to 3D point clouds bridge the sim-to-real domain shift to enhance LiDAR perception in autonomous vehicles?

An effective domain adaptation algorithm that minimizes the sim-to-real domain shift can leverage an adversarial network that concentrates on modifying the local structure of point clouds. This algorithm involves the preprocessing of point clouds to a domain-invariant representation, a reconstruction network based on operators for 3D point cloud handling, such as PointNet++, and a patch-based discriminator encouraging the generator to produce target-style local structures. This domain adaptation algorithm can be used for either object- or scene-level adaptation, whereas the latter reduces the domain shift to a greater extent. Furthermore, it can be applied to either distribution-aligned or non-distribution-aligned datasets. While the reduction of domain shift is more pronounced for distribution-aligned datasets, the method's applicability to non-distribution-aligned datasets enhances its relevance for practical applications, as explained in the next Subsection 9.1.5.

9.1.4 Real-World Application

The AP obtained in this thesis does not attain 100 % and remains under 60 % for the majority of configurations. However, neural networks are still applied to solve perception challenges in the real world. An example is the IAC, during which the TUM autonomous motorsport team deployed PointRCNN with a configuration similar to this thesis. For real-world integration, this network was trained with annotated real-world data collected during test runs on the race track. The performance of this network was similar to the performance of PointRCNN as employed in this thesis when trained and tested on real-world data, i.e. achieving a 3D AP (0.7) of approximately 50 %. Despite these limitations, this network has still been used successfully in competitive head-to-head races to detect opponent vehicles on the race track. Several factors contribute to why networks with an AP significantly less than 100 % can still be used effectively in real-world autonomous vehicles, as discussed in the following.

First, alongside a neural network that extracts information to detect objects within LiDAR point clouds, a parallel clustering pipeline complements the neural network. This pipeline uses a two-stage Euclidean clustering algorithm and detects all objects whose reflected points match the criteria regarding cluster size and point density. As clustering is an unsupervised machine learning technique, it does not require training and can detect vehicle components that the neural network fails to detect. Second, in addition to the LiDAR sensor, the race car is equipped with cameras and RaDAR sensors. These can be leveraged for sensor fusion to enhance the robustness of perception. For the IAC, the RaDAR sensor detections are utilized, providing both velocity and spatial data. Finally, the perception pipelines are followed by a subsequent tracking algorithm. This tracking algorithm not only fuses the individual perception pipelines, but also keeps track of the detected objects in the event they are not detected at each individual time step. This allows the perception systems to occasionally miss detections at some time steps without adversely affecting the prediction and planning algorithms.

In general, this combination of factors enables a robust perception system, even if each pipeline does not achieve the highest metric scores. Therefore, it is not necessary to attain 100 % AP. Nevertheless, the development goal should be to achieve the highest possible AP. Research in LiDAR domain adaptation algorithms, as presented in this thesis, can pave the way toward this goal, particularly by improving the performance of networks trained with synthetic simulation data.

9.1.5 Practical Relevance

The last part of the discussion highlights the practical relevance of the presented domain adaptation method in terms of economic, scientific, and societal impact.

Economic Relevance

The traditional data acquisition for the generation of an annotated training dataset involves significant financial and time investments, which are valuable resources for companies, startups, and researchers. Costs are associated with sensor-equipped vehicle operations, data recording, and manual annotation of LiDAR point clouds. In particular, manual annotation requires extensive human effort. Depending on the complexity and volume of the data, this can cost several minutes and dollars per 3D point cloud. The proposed domain adaptation method has the potential to drastically reduce the costs related to datasets, by reducing the amount of required annotated real-world data. Datasets in simulation can be generated with a fraction of the costs and time of real-world datasets and can include safety-critical scenarios difficult to capture in the real world. These simulated datasets can be adapted using the proposed sim-to-real domain adaptation method and hence used for training of object detection algorithms. Moreover, the adapted data can be used for network pretraining, and fine-tuning can be done using only a small subset of the real-world data. Similar studies are shown in [159], where pretraining with simulated data and subsequent fine-tuning using only $20\,\%$ of the available real-world data yields performance comparable to training with $100\,\%$ of the real-world data.

Scientific Relevance

The proposed solutions in this thesis are open source to accelerate research and development in AD. The distribution-aligned IAC dataset [229, 230] is publicly available for further studies of the sim-to-real domain shift. The source code of the domain adaptation algorithm is available on GitHub [231]. In addition to these contributions, the previously explained method of using adapted simulated data for pretraining of object detection algorithms can further accelerate the research in AD, as researchers can focus on collecting small and task-tailored real-world datasets used for fine-tuning instead of collecting large foundation datasets.

Societal Relevance

The research presented in this thesis enables the development of a more robust perception software, which is a crucial part of the AD software stack. Therefore, this research contributes to the overall development and diffusion of AVs, which can enable a revolution in the field of transportation. For example, with a dissemination of AVs, the number of vehicle crashes could be reduced as much as by $90\,\%$, according to [232]. This also has the positive impact of reducing the costs of healthcare and vehicle repairs, which are expected to save together up to US$180 billion yearly, according to a report by McKinsey Global Institute [233]. Furthermore, the number of parking spaces is expected to be reduced due to fewer shared vehicles, which are used more efficiently, with parking spaces currently taking up to $24\,\%$ of the area in cities in the USA [232]. Moreover, the costs of using shared vehicles are expected to be lower than those of owning vehicles [232], which leads to monetary relief for mobility spending. Lastly, the proposed domain adaptation approach reduces the need or extension of real-world datasets and hence the collection of these, which often involves large fleets contributing to environmental pollution.

9.2 Future Work

This thesis proposes a first approach to 3D domain adaptation targeting the sim-to-real domain shift in LiDAR object detection. The approach serves as a baseline for this research field and can be extended in several ways, which are elaborated in the following.

The perception task used for evaluating domain adaptation is object detection. However, extension to different tasks using LiDAR point clouds as input, such as semantic segmentation [234], learning-based LiDAR localization [235], or approaches combining object detection and localization tasks [236] can give insight on the generalizability of the domain adaptation method. This would provide a more comprehensive understanding of the effectiveness of the domain adaptation method for different perception tasks. In addition, the impact of different domain adaptation techniques on the performance of these tasks could also be investigated. This could potentially lead to the development of more robust and versatile domain adaptation methods for LiDAR point clouds.

In the current implementation, the domain adaptation method is tested with datasets containing only a single object class, that is, *race car* or *car* for the IAC or CARLA-KITTI datasets, respectively. Future work can extend the method by using datasets containing additional classes, such as pedestrians, cyclists, and trucks. This opens up new questions. For example, if the object-level domain adaptation can be trained on multiple classes simultaneously or if the network has to be trained for each class individually.

The domain adaptation approach presented is aimed at minimizing the sim-to-real domain shift. In future work, it can be analyzed if the same domain adaptation approach is also capable of reducing other forms of domain shift, e.g. the weather-to-weather domain shift within a single public dataset.

Also, the domain adaptation method only uses the xyz features of the point clouds and neglects the intensity channel, although some object detection algorithms support the inclusion of this additional feature channel. Including the intensity channel within the domain adaptation algorithm requires only minor algorithmic changes, but the simulated dataset must provide an initial value for the point-wise intensity. In future work, the incorporation of the intensity channel into the domain adaptation method can be analyzed. This will involve modifying the domain adaptation algorithm to accommodate the additional feature channel and updating the simulation environment to include initial values for point-wise intensity, which requires validation of the intensity values with the collected data in the real world. This enhancement can improve the performance of the object detection algorithms that utilize the domain adaptation method, as the intensity value is a distinct feature, especially for high-reflectivity objects such as license plates on vehicles.

Another interesting direction for future work could be the online application of the domain adaptation algorithm in the context of the inverted real-to-sim use case. The idea behind this application is that an object detection algorithm is trained using a large simulated dataset, and during inference, it is fed by real-world data recorded from a LiDAR sensor but adapted online to follow the characteristics of the simulation using a real-to-sim domain adaptation algorithm. This adds another algorithm to the AD software stack and hence can increase computational requirements and latency. However, this combination of real-to-sim domain adaptation and object detection pipeline has the advantage of being universally applicable to different real-world data inputs without retraining any of the two algorithms, as both of them are only trained using the simulated data.

The last area of future work concerns the definition or standardization of evaluating the realism of simulated data. To date, and also in this thesis, the realism of point clouds is assessed using metrics of object detection algorithms, such as the mean AP. However, the mean AP itself is not easy to interpret. This leads to the need for more comprehensive and interpretable metrics to assess the realism of simulated data. Developing such metrics could involve considering various aspects of the data, such as the accuracy of object shapes and positions, the consistency of lighting and shading, and the realism of textures. Furthermore, it would be beneficial to establish benchmarks for these metrics to provide clear goals for the development of simulation techniques. These benchmarks could be based on the performance of human perception or the requirements of specific applications of point cloud data. This leads to the final question in automotive perception that needs to be addressed: How good is good enough?

10 Summary and Conclusion

This thesis presents a method to bridge the sim-to-real domain shift in LiDAR point clouds in the context of AD. The method consists of analyzing the key characteristics of point clouds and introducing an algorithm based on adversarial learning for domain adaptation of point clouds in 3D space. This method can be applied to distribution-aligned as well as to non-distribution-aligned simulation and real-world data and operates on the object- or scene-level of point clouds. The method was developed based on findings from related work and evaluated using object detection algorithms, which are a crucial element in the perception task of AD. Extensive experiments on different datasets show the effectiveness of the presented method in reducing the sim-to-real domain shift.

This thesis is motivated by the challenge of data hunger in the training of safety-critical neural networks for automotive perception. This requires large annotated real-world datasets, which are costly to obtain, and hence new solutions should be analyzed. Another motivation for this thesis is the participation at the IAC, where training data is scarce and simulation data offers an alternative. However, the impact of the use of simulation data can not be assessed and needs to be analyzed.

In the related work section, the current state of the art in the field of domain shift and domain adaptation for the sim-to-real applications are presented. Based on the conclusions of the related work, the research gaps are highlighted and the research questions are derived, which serve as a guideline for the structure of the thesis.

The first part of the method deals with the analysis of the sim-to-real domain shift. To this end, an annotated distribution-aligned simulated and real-world LiDAR dataset is generated. Real-world data is recorded using autonomous race cars on a race track during the IAC and annotated using the GNSS positions of the agents. These positions are further used to generate a simulated counterpart of the dataset in a 3D simulation environment. Using this IAC dataset pair, the domain shift is quantified using the LiDAR perception algorithms PointRCNN and PointPillars and the object detection metric 3D AP. The specific sim-to-real domain shift is evaluated to be almost $14\,\%$ 3D AP. Qualitative analysis of the 3D point clouds shows that the noise pattern and LiDAR dropout differ between simulation and the real world, which is especially noticeable in the reflections of the objects.

Once the major contributing factor to the sim-to-real domain shift is identified, a domain adaptation method is developed to reduce this domain shift. This domain adaptation algorithm is based on adversarial learning by combining a point completion network as a generator and a discriminator that operates on patches of point clouds. The approach is the first LiDAR domain adaptation network that uses adversarial learning to adapt point clouds directly in the 3D space and leverage information found in the spatial relationship between points. One key characteristic of the method is that it is utilizing downsampling to produce representations of point clouds that are invariant to the domain before learning the local structure of the target domain in the network. This approach allows to use of target data only during training, and hence, is scalable to adapt a theoretically unlimited number of source domains during inference, eliminating the requirement for each new source domain to undergo retraining. As the qualitative analysis of the sim-to-real domain shift showed a difference, especially in the object point clouds, the approach focuses on the adaptation on the object-level.

Using the same evaluation method as introduced in domain shift quantification, the domain shift could be successfully reduced from $8.63\,\%$ to $4.36\,\%$ 3D AP on the distribution-aligned IAC dataset.

The same algorithm is modified to adapt scene-level point clouds with the idea of further reducing the remaining domain shift. The modifications mainly comprise the preprocessing of the point clouds, as the scene-level point clouds are more sparse than object-level point clouds. The evaluation shows that for the distribution-aligned IAC dataset, the scene-level domain adaptation is capable of fully eliminating the sim-to-real domain shift in various settings.

To further analyze the performance and generalization ability of the network on a non-distribution-aligned dataset, a synthetic dataset is generated using the simulation environment CARLA, and real-world LiDAR point clouds from the public KITTI dataset are used. This CARLA-KITTI dataset pair is tested for adaptation at the object- and the scene-level. The domain adaptation architecture is modified to handle the adaptation of multiple object point clouds per scene point cloud. The results of the CARLA-KITTI dataset show that adaptation at the scene-level is especially capable of reducing the domain shift from $62.08\,\%$ to $51.91\,\%$ 3D AP, although a large shift still exists after the adaptation. This demonstrates the success of the method. However, it's important to note that the overall domain shift is a combination of multiple domain shifts. The domain adaptation approach presented in this thesis specifically targets the domain shift that occurs due to differences in sensor modeling.

The discussion critically reviews the method and results, and further answers the research questions. The practical relevance of the method is highlighted, which is reflected mainly in saving costs and time due to the ability to use simulation data instead of expensive real-world data, which contributes to the overall faster development of reliable and robust perception systems for AVs.

Future work can include the extension of the proposed method to different perception tasks, such as segmentation, or extend the method by using datasets containing additional object classes. Furthermore, the inverse real-to-sim domain adaptation for online applications is an open challenge and can be addressed in follow-up works.

Finally, to support forthcoming research in the realm of 3D LiDAR domain adaptation, the code developed as part of this thesis is open source on GitHub.

List of Figures

List of Tables

Bibliography

[1] A. Costley, C. Kunz, R. Sharma and R. Gerdes, „Low Cost, Open-Source Platform to Enable Full-Sized Automated Vehicle Research," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 1, pp. 3–13, 2021, DOI: 10.1109/TIV.2020.3029771.

[2] World Health Organization, „Global status report on road safety 2023," 12/2023, ISBN: 978-92-4-008651-7. Available: https://www.who.int/publications/i/item/9789240086517.

[3] K. Faber and D. van Lierop, „How will older adults use automated vehicles? Assessing the role of AVs in overcoming perceived mobility barriers," *Transportation Research Part A: Policy and Practice*, vol. 133, pp. 353–363, 2020, DOI: 10.1016/J.TRA.2020.01.022.

[4] Y. C. Lee and J. H. Mirman, „Parents' perspectives on using autonomous vehicles to enhance children's mobility," *Transportation Research Part C: Emerging Technologies*, vol. 96, pp. 415–431, 2018, DOI: 10.1016/J.TRC.2018.10.001.

[5] E. D. Dickmanns, „The development of machine vision for road vehicles in the last decade," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 1, pp. 268–281, 2003, ISBN: 0780373464. DOI: 10.1109/IVS.2002.1187962.

[6] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L. E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian and P. Mahoney, „Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006, DOI: 10.1002/ROB.20147. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/rob.20147.

[7] C. Urmson, C. Baker, J. Dolan, P. Rybski, B. Salesky, W. R. Whittaker, D. Ferguson and M. Darms, „Autonomous Driving in Traffic: Boss and the Urban Challenge," *AI Magazine*, vol. 30, no. 2, pp. 17–28, 2009, DOI: 10.1609/AIMAG.V30I2.2238. Available: https://onlinelibrary.wiley.com/doi/full/10.1609/aimag.v30i2.2238.

[8] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knoppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein and E. Zeeb, „Making bertha drive-an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014, DOI: 10.1109/MITS.2014.2306552.

[9] J. Betz, T. Betz, F. Fent, M. Geisslinger, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, M. Lienkamp, B. Lohmann, F. Nobis, L. Ögretmen, M. Rowold, F. Sauerbeck, T. Stahl, R. Trauth, F. Werner and A. Wischnewski, „TUM autonomous motorsport: An autonomous racing software for the Indy Autonomous Challenge," *Journal of Field Robotics*, vol. 40, no. 4, pp. 783–809, 2023, DOI: 10.1002/ROB.22153. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/rob.22153.

[10] E. Guizzo. „*How Google's Self-Driving Car Works - IEEE Spectrum*," Oct. 2011. Available: https://spectrum.ieee.org/how-google-self-driving-car-works.

[11] Waymo. „*History of Waymo*,“ 2024. Available: https://waymo.com/about/#story.

[12] Cruise. „*About | Cruise Self Driving Car Company Mission | Cruise*,“ 2024. Available: https://www.getcruise.com/about/.

[13] Uber. „*Uber AV: Autonomous Mobility and Delivery*,“ 2024. Available: https://www.uber.com/us/en/autonomous/.

[14] Zoox. „*Zoox*,“ 2024. Available: https://zoox.com/.

[15] J. Deichmann, E. Ebel, K. Heineke, R. Heuss, M. Kellner and F. Steiner. „*Autonomous driving's future: Convenient and connected*,“ Jan. 2023. Available: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/autonomous-drivings-future-convenient-and-connected#/.

[16] M. Ostrovsky and M. Schwarz, „Carpooling and the economics of self-driving cars,“ *ACM EC 2019 - Proceedings of the 2019 ACM Conference on Economics and Computation*, pp. 581–582, 2019, ISBN: 9781450367929. DOI: 10.1145/3328526.3329625. Available: https://dl.acm.org/doi/10.1145/3328526.3329625.

[17] L. M. Clements and K. M. Kockelman, „Economic Effects of Automated Vehicles,“ *Journal of the Transportation Research Board*, vol. 2606, no. 1, pp. 106–114, 2017, DOI: 10.3141/2606-14. Available: https://journals.sagepub.com/doi/10.3141/2606-14.

[18] B. H. Frank. „*Uber parks its self-driving cars after fatal pedestrian crash in Tempe | VentureBeat*,“ Mar. 2018. Available: https://venturebeat.com/ai/uber-parks-its-self-driving-cars-after-fatal-pedestrian-crash-in-tempe/.

[19] R. Stumpf. „*Autopilot Blamed for Tesla's Crash Into Overturned Truck | The Drive*,“ June 2020. Available: https://www.thedrive.com/news/33789/autopilot-blamed-for-teslas-crash-into-overturned-truck.

[20] B. Templeton. „*Waymo's Double-Crash With Pickup Truck, Cyclist & Arson Examined*,“ Mar. 2024. Available: https://www.forbes.com/sites/bradtempleton/2024/03/05/waymos-double-crash-with-pickup-trucks-and-more-examined/?sh=78b9fcea170e.

[21] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus and M. H. Ang, „Perception, Planning, Control, and Coordination for Autonomous Vehicles,“ *Machines 2017, Vol. 5, Page 6*, vol. 5, no. 1, p. 6, 2017, DOI: 10.3390/MACHINES5010006. Available: https://www.mdpi.com/2075-1702/5/1/6/htm%20https://www.mdpi.com/2075-1702/5/1/6.

[22] C. A. Goodfellow Ian Bengio Yoshua. „*Deep Learning - Ian Goodfellow, Yoshua Bengio, Aaron Courville - Google Books*,“ 2016.

[23] X. Wu, D. Sahoo and S. C. Hoi, „Recent advances in deep learning for object detection,“ *Neurocomputing*, vol. 396, 2020, DOI: 10.1016/j.neucom.2020.01.085.

[24] D. Feng, C. Haase-Schutz, L. Rosenbaum, H. Hertlein, C. Glaser, F. Timm, W. Wiesbeck and K. Dietmayer, „Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges,“ *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, 2021, DOI: 10.1109/TITS.2020.2972974.

[25] D. Fernandes, A. Silva, R. Névoa, C. Simões, D. Gonzalez, M. Guevara, P. Novais, J. Monteiro and P. Melo-Pinto, „Point-cloud based 3D object detection and classification methods for self-driving applications: A survey and taxonomy,“ *Information Fusion*, vol. 68, 2021, DOI: 10.1016/j.inffus.2020.11.002.

[26] B. Gao, Y. Pan, C. Li, S. Geng and H. Zhao, „Are We Hungry for 3D LiDAR Data for Semantic Segmentation? A Survey of Datasets and Methods,“ *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, 2022, DOI: 10.1109/TITS.2021.3076844.

[27] F. Sauerbeck, S. Huch, F. Fent, P. Karle, D. Kulmer and J. Betz, „Learn to See Fast: Lessons Learned From Autonomous Racing on How to Develop Perception Systems,“ *IEEE Access*, vol. 11, 2023, DOI: 10.1109/ACCESS.2023.3272750.

[28] IAC. „*Indy Autonomous Challenge*,“ 2024. Available: https://www.indyautonomouschallenge.com/.

[29] E. Yurtsever, J. Lambert, A. Carballo and K. Takeda, „A Survey of Autonomous Driving: Common Practices and Emerging Technologies,“ *IEEE Access*, vol. 8, 2020, DOI: 10.1109/ACCESS.2020.2983149.

[30] S. Royo and M. Ballesta-Garcia, „An Overview of Lidar Imaging Systems for Autonomous Vehicles,“ *Applied Sciences*, vol. 9, no. 19, p. 4093, 2019, DOI: 10.3390/app9194093.

[31] H. Gotzig and G. O. Geduld, „LIDAR-Sensorik,“ in *Handbuch Fahrerassistenzsysteme* 2015, DOI: 10.1007/978-3-658-05734-3{_}18.

[32] Y. Li and J. Ibanez-Guzman, „Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems,“ *IEEE Signal Processing Magazine*, vol. 37, no. 4, 2020, DOI: 10.1109/MSP.2020.2973615.

[33] I. Standard, „Safety of laser products - Part 1: Equipment classification and requirements,“ *Iec 60825-1:2014*, no. May, 2014.

[34] E. Marti, M. Á. De Miguel, F. Garcia and J. Perez, „A Review of Sensor Technologies for Perception in Automated Driving,“ *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 4, 2019, DOI: 10.1109/MITS.2019.2907630.

[35] SAE, „Vehicle dynamics terminology,“ *SAE International*, no. July, 2008.

[36] A. Geiger, P. Lenz and R. Urtasun, „Are we ready for autonomous driving? the KITTI vision benchmark suite,“ in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2012, DOI: 10.1109/CVPR.2012.6248074.

[37] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, „Vision meets robotics: The KITTI dataset,“ *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013, DOI: 10.1177/0278364913491297. Available: https://journals.sagepub.com/doi/full/10.1177/0278364913491297.

[38] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu and M. Bennamoun. „*Deep Learning for 3D Point Clouds: A Survey*,“ 2021. DOI: 10.1109/TPAMI.2020.3005434.

[39] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, „Language models are few-shot learners,“ in *Advances in Neural Information Processing Systems*, 2020.

[40] G. Elbaz, T. Avraham and A. Fischer, „3D point cloud registration for localization using a deep neural network auto-encoder,“ in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, DOI: 10.1109/CVPR.2017.265.

[41] A. Zeng, K. T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez and J. Xiao, „Multi-view self-supervised deep learning for 6D pose estimation in the Amazon Picking Challenge,“ in *Proceedings - IEEE International Conference on Robotics and Automation*, 2017, DOI: 10.1109/ICRA.2017.7989165.

[42] X. F. Han, H. Laga and M. Bennamoun, „Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 5, 2021, DOI: 10.1109/TPAMI.2019.2954885.

[43] S. A. Bello, S. Yu, C. Wang, J. M. Adam and J. Li. „*Review: Deep learning on 3D point clouds*,“ 2020. DOI: 10.3390/rs12111729.

[44] C. R. Qi, H. Su, K. Mo and L. J. Guibas, „PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660, ISBN: 9781538604571. DOI: 10.48550/arxiv.1612.00593. Available: https://arxiv.org/abs/1612.00593v2.

[45] C. R. Qi, L. Yi, H. Su and L. J. Guibas, „PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 5105–5114. Available: http://arxiv.org/abs/1706.02413.

[46] H. Thomas, C. R. Qi, J. E. Deschaud, B. Marcotegui, F. Goulette and L. Guibas, „KPConv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, DOI: 10.1109/ICCV.2019.00651.

[47] Y. Li, R. Bu, M. Sun, W. Wu, X. Di and B. Chen, „PointCNN: Convolution on X-transformed points," in *Advances in Neural Information Processing Systems*, 2018.

[48] H. Zhao, L. Jiang, C. W. Fu and J. Jia, „Pointweb: Enhancing local neighborhood features for point cloud processing," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 5560–5568, 2019, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00571.

[49] W. Wu, Z. Qi and L. Fuxin, „PointCONV: Deep convolutional networks on 3D point clouds," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 9613–9622, 2019, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00985.

[50] Y. Liu, B. Fan, S. Xiang and C. Pan, „Relation-shape convolutional neural network for point cloud analysis," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 8887–8896, 2019, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00910.

[51] S. Lan, R. Yu, G. Yu and L. S. Davis, „Modeling local geometric structure of 3D point clouds using geo-cnn," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 998–1008, 2019, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00109.

[52] A. Komarichev, Z. Zhong and J. Hua, „A-CNN: Annularly convolutional neural networks on point clouds," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 7413–7422, 2019, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00760.

[53] Y. Xu, T. Fan, M. Xu, L. Zeng and Y. Qiao, „SpiderCNN: Deep learning on point sets with parameterized convolutional filters," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11212 LNCS, pp. 90–105, 2018, ISBN: 9783030012366. DOI: 10.1007/978-3-030-01237-3{_}6. Available: https://link.springer.com/chapter/10.1007/978-3-030-01237-3_6.

[54] J. Liu, B. Ni, C. Li, J. Yang and Q. Tian, „Dynamic points agglomeration for hierarchical point sets learning," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-October, pp. 7545–7554, 2019, ISBN: 9781728148038. DOI: 10.1109/ICCV.2019.00764.

[55] J. Yang, Q. Zhang, B. Ni, L. Li, J. Liu, M. Zhou and Q. Tian, „Modeling point clouds with self-attention and gumbel subset sampling," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 3318–3327, 2019, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00344.

[56] Y. Zhou and O. Tuzel, „VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499, 2018, ISBN: 9781538664209. DOI: 10.1109/CVPR.2018.00472.

[57] J. Li, B. M. Chen and G. H. Lee, „SO-Net: Self-Organizing Network for Point Cloud Analysis," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 9397–9406, 2018, ISBN: 9781538664209. DOI: 10.1109/CVPR.2018.00979.

[58] B. S. Hua, M. K. Tran and S. K. Yeung, „Pointwise Convolutional Neural Networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 984–993, 2018, ISBN: 9781538664209. DOI: 10.1109/CVPR.2018.00109.

[59] Y. Zhao, T. Birdal, H. Deng and F. Tombari, „3D point capsule networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 1009–1018, 2019, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00110.

[60] S. Sabour, N. Frosst and G. E. Hinton, „Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, 2017.

[61] R. Klokov and V. Lempitsky, „Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, pp. 863–872, 2017, ISBN: 9781538610329. DOI: 10.1109/ICCV.2017.99.

[62] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein and J. M. Solomon, „Dynamic Graph CNN for Learning on Point Clouds," *ACM Transactions on Graphics*, vol. 38, no. 5, p. 13, 2018, DOI: 10.1145/3326362. Available: https://arxiv.org/abs/1801.07829v2.

[63] C. Wang, B. Samari and K. Siddiqi, „Local Spectral Graph Convolution for Point Set Feature Learning," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11208 LNCS, pp. 56–71, 2018, ISBN: 9783030012243. DOI: 10.1007/978-3-030-01225-0{_}4. Available: https://link.springer.com/chapter/10.1007/978-3-030-01225-0_4.

[64] W. Han, C. Wen, C. Wang, X. Li and Q. Li, „Point2node: Correlation learning of dynamic-node for point cloud feature modeling," in *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, 2020, DOI: 10.1609/aaai.v34i07.6725.

[65] E. Camuffo, D. Mari and S. Milani, „Recent Advancements in Learning Algorithms for Point Clouds: An Updated Overview," *Sensors 2022, Vol. 22, Page 1357*, vol. 22, no. 4, p. 1357, 2022, DOI: 10.3390/S22041357. Available: https://www.mdpi.com/1424-8220/22/4/1357/htm.

[66] J. Mao, S. Shi, X. Wang and H. Li, „3D Object Detection for Autonomous Driving: A Comprehensive Survey," *International Journal of Computer Vision*, vol. 131, no. 8, pp. 1909–1963, 2023, DOI: 10.1007/S11263-023-01790-1/TABLES/18. Available: https://link.springer.com/article/10.1007/s11263-023-01790-1.

[67] R. Qian, X. Lai and X. Li, „3D Object Detection for Autonomous Driving: A Survey," *Pattern Recognition*, vol. 130, p. 108796, 2022, DOI: 10.1016/J.PATCOG.2022.108796.

[68] S. Shi, X. Wang and H. Li, „PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. Available: https://openaccess.thecvf.com/content_CVPR_2019/papers/Shi_PointRCNN_3D_Object_Proposal_Generation_and_Detection_From_Point_Cloud_CVPR_2019_paper.pdf.

[69] X. Pan, Z. Xia, S. Song, L. E. Li and G. Huang, „3D Object Detection with Pointformer," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021, DOI: 10.1109/CVPR46437.2021.00738.

[70] B. Yang, W. Luo and R. Urtasun, „PIXOR: Real-time 3D Object Detection from Point Clouds," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, DOI: 10.1109/CVPR.2018.00798.

[71] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang and O. Beijbom, „PointPillars: Fast Encoders for Object Detection from Point Clouds," 2019. Available: https://github.com/nutonomy/second.pytorch.

[72] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez and C. K. Wellington, „Lasernet: An efficient probabilistic 3D object detector for autonomous driving," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, DOI: 10.1109/CVPR.2019.01296.

[73] G. Zamanakos, L. Tsochatzidis, A. Amanatiadis and I. Pratikakis, „A comprehensive survey of LIDAR-based 3D object detection methods with deep learning for autonomous driving," *Computers & Graphics*, vol. 99, pp. 153–181, 2021, DOI: 10.1016/J.CAG.2021.07.003.

[74] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan and O. Beijbom, „nuScenes: A multimodal dataset for autonomous driving," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 11618–11628, 2019, DOI: 10.48550/arxiv.1903.11027. Available: https://openaccess.thecvf.com/content_CVPR_2020/papers/Caesar_nuScenes_A_Multimodal_Dataset_for_Autonomous_Driving_CVPR_2020_paper.pdf.

[75] B. Deng, C. R. Qi, M. Najibi, T. Funkhouser, Y. Zhou and D. Anguelov, „Revisiting 3D Object Detection From an Egocentric Perspective," in *Advances in Neural Information Processing Systems*, 2021.

[76] J. Philion, A. Kar and S. Fidler, „Learning to Evaluate Perception Models Using Planner-Centric Metrics," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020, DOI: 10.1109/CVPR42600.2020.01407.

[77] W. Maddern, G. Pascoe, C. Linegar and P. Newman, „1 year, 1000 km: The Oxford RobotCar dataset," *International Journal of Robotics Research*, vol. 36, no. 1, 2017, DOI: 10.1177/0278364916679498.

[78] J. Xue, J. Fang, T. Li, B. Zhang, P. Zhang, Z. Ye and J. Dou, „BLVD: Building a large-scale 5D semantics benchmark for autonomous driving," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2019, DOI: 10.1109/ICRA.2019.8793523.

[79] P. Sun, 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, „Scalability in Perception for Autonomous Driving: Waymo Open Dataset," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2443–2451, ISBN: 978-1-7281-7168-5. DOI: 10.1109/CVPR42600.2020.00252. Available: https://ieeexplore.ieee.org/document/9156973/.

[80] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng and R. Yang, „The ApolloScape Open Dataset for Autonomous Driving and Its Application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, 2020, DOI: 10.1109/TPAMI.2019.2926463.

[81] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin and R. Yang, „The ApolloScape Dataset for Autonomous Driving," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2018, DOI: 10.1109/CVPRW.2018.00141.

[82] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. Kaesemodel Pontes, D. Ramanan, P. Carr and J. Hays, „Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[83] Unreal Engine. „*The most powerful real-time 3D creation tool - Unreal Engine*," Nov. 2023. Available: https://www.unrealengine.com/en-US.

[84] Unity. „*Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine*,“ 2023. Available: https://unity.com/.

[85] S. R. Richter, V. Vineet, S. Roth and V. Koltun, „Playing for data: Ground truth from computer games,“ in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, DOI: 10.1007/978-3-319-46475-6{_}7.

[86] S. R. Richter, Z. Hayder and V. Koltun, „Playing for Benchmarks,“ in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, DOI: 10.1109/ICCV.2017.243.

[87] Rockstar Games. „*Grand Theft Auto V - Rockstar Games*,“ Available: https://www.rockstargames.com/gta-v.

[88] A. Gaidon, Q. Wang, Y. Cabon and E. Vig, „VirtualWorlds as Proxy for Multi-object Tracking Analysis,“ in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, DOI: 10.1109/CVPR.2016.470.

[89] Y. Cabon, N. Murray and M. Humenberger, „Virtual Kitti 2,“ *Arxiv*, 2020.

[90] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez and V. Koltun, „CARLA: An Open Urban Driving Simulator,“ in *Conference on robot learning*, 2017, pp. 1–16. Available: https://proceedings.mlr.press/v78/dosovitskiy17a/dosovitskiy17a.pdf.

[91] X. Weng, Y. Man, D. Cheng, J. Park, M. O. Toole and K. Kitani, „All-In-One Drive: A Large-Scale Comprehensive Perception Dataset with High-Density Long-Range Point Clouds,“ *arXiv*, 2020.

[92] A. Farahani, S. Voghoei, K. Rasheed and H. R. Arabnia, „A Brief Review of Domain Adaptation,“ 2021, DOI: 10.1007/978-3-030-71704-9{_}65.

[93] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira and J. W. Vaughan, „A theory of learning from different domains,“ *Machine Learning*, vol. 79, no. 1-2, pp. 151–175, 2010, DOI: 10.1007/s10994-009-5152-4. Available: http://link.springer.com/10.1007/s10994-009-5152-4.

[94] L. T. Triess, M. Dreissig, C. B. Rist and J. M. Zöllner, „A Survey on Deep Domain Adaptation for LiDAR Perception,“ in *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, 2021, pp. 350–357, DOI: 10.1109/IVWorkshops54471.2021.9669228. Available: http://dx.doi.org/10.1109/IVWorkshops54471.2021.9669228.

[95] F. Ott, D. Rügamer, L. Heublein, B. Bischl and C. Mutschler, „Domain Adaptation for Time-Series Classification to Mitigate Covariate Shift,“ in *MM 2022 - Proceedings of the 30th ACM International Conference on Multimedia*, 2022, DOI: 10.1145/3503161.3548167.

[96] M. Schwonberg, J. Niemeijer, J. A. Termohlen, J. P. Schafer, N. M. Schmidt, H. Gottschalk and T. Fingscheidt, „Survey on Unsupervised Domain Adaptation for Semantic Segmentation for Visual Perception in Automated Driving,“ *IEEE Access*, vol. 11, pp. 54296–54336, 2023, DOI: 10.1109/ACCESS.2023.3277785.

[97] P. Oza, V. A. Sindagi, V. V. Sharmini and V. M. Patel, „Unsupervised Domain Adaptation of Object Detectors: A Survey,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023, DOI: 10.1109/TPAMI.2022.3217046.

[98] V. M. Patel, R. Gopalan, R. Li and R. Chellappa, „Visual Domain Adaptation: A survey of recent advances,“ *IEEE Signal Processing Magazine*, vol. 32, no. 3, 2015, DOI: 10.1109/MSP.2014.2347059.

[99] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon and S. Birchfield, „Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization,“ in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018, pp. 969–977.

[100] J. Wang, M. M. Cheng and J. Jiang, „Domain Shift Preservation for Zero-Shot Domain Adaptation," *IEEE Transactions on Image Processing*, 2021, DOI: 10.1109/TIP.2021.3084354.

[101] Y. Zhang, P. David, H. Foroosh and B. Gong, „A Curriculum Domain Adaptation Approach to the Semantic Segmentation of Urban Scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, 2020, DOI: 10.1109/TPAMI.2019.2903401.

[102] Q. Wang, J. Gao and X. Li, „Weakly Supervised Adversarial Domain Adaptation for Semantic Segmentation in Urban Scenes," *IEEE Transactions on Image Processing*, vol. 28, no. 9, pp. 4376–4386, 2019, DOI: 10.1109/TIP.2019.2910667.

[103] S. Wang, L. Zhang, W. Zuo and B. Zhang, „Class-Specific Reconstruction Transfer Learning for Visual Recognition across Domains," *IEEE Transactions on Image Processing*, vol. 29, pp. 2424–2438, 2020, DOI: 10.1109/TIP.2019.2948480.

[104] M. Tan, J. Yu, H. Zhang, Y. Rui and D. Tao, „Image Recognition by Predicted User Click Feature with Multidomain Multitask Transfer Deep Network," *IEEE Transactions on Image Processing*, vol. 28, no. 12, pp. 6047–6062, 2019, DOI: 10.1109/TIP.2019.2921861.

[105] W. M. Kouw and M. Loog, „An introduction to domain adaptation and transfer learning," 2018. Available: https://arxiv.org/abs/1812.11806v2.

[106] Y. Wang, X. Chen, Y. You, L. Erran, B. Hariharan, M. Campbell, K. Q. Weinberger and W.-L. Chao, „Train in Germany, Test in The USA: Making 3D Object Detectors Generalize," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11710–11720. Available: https://openaccess.thecvf.com/content_CVPR_2020/papers/Wang_Train_in_Germany_Test_in_the_USA_Making_3D_Object_CVPR_2020_paper.pdf.

[107] F. E. Nowruzi, P. Kapoor, D. Kolhatkar, F. A. Hassanat, R. Laganiere and J. Rebut, „How much real data do we actually need: Analyzing object detection performance using synthetic and real data," in *ICML Workshop on AI for Autonomous Driving,* 2019. Available: http://arxiv.org/abs/1907.07061.

[108] S. Dodge and L. Karam, „Understanding how image quality affects deep neural networks," *2016 8th International Conference on Quality of Multimedia Experience, QoMEX 2016*, 2016, ISBN: 9781509003549. DOI: 10.1109/QOMEX.2016.7498955.

[109] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang and D. Metaxas, „StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, DOI: 10.1109/ICCV.2017.629.

[110] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann and C. Sutton, „VEEGAN: Reducing mode collapse in GANs using implicit variational learning," in *Advances in Neural Information Processing Systems*, 2017.

[111] M. Mathieu, J. Zhao, P. Sprechmann, A. Ramesh and Y. Le Cun, „Disentangling factors of variation in deep representations using adversarial training," in *Advances in Neural Information Processing Systems*, 2016.

[112] Z. Lin, G. Fanti, A. Khetan and S. Oh, „PacGan: The power of two samples in generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2018, DOI: 10.1109/jsait.2020.2983071.

[113] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft and S. Belongie, „Stacked generative adversarial networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, DOI: 10.1109/CVPR.2017.202.

[114] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, „Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, DOI: 10.1007/978-3-658-40442-0{_}9.

[115] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever and P. Abbeel, „InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2016.

[116] L. T. Triess, D. Peter, C. B. Rist, M. Enzweiler and J. M. Zollner, „CNN-based synthesis of realistic high-resolution LiDAR data," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2019, DOI: 10.1109/IVS.2019.8813771.

[117] L. Van Der Maaten and G. Hinton, „Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[118] J. Huang, S. Lu, D. Guan and X. Zhang, „Contextual-Relation Consistent Domain Adaptation for Semantic Segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020, DOI: 10.1007/978-3-030-58555-6{_}42.

[119] Y. LeCun and C. Cortes, „MNIST handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 7, 2010.

[120] L. Schoneveld. „*In Raw Numpy: t-SNE*," Sept. 2017. Available: https://nlml.github.io/in-raw-numpy/in-raw-numpy-t-sne/.

[121] L. T. Triess, C. B. Rist, D. Peter and J. M. Zöllner, „A Realism Metric for Generated LiDAR Point Clouds," *International Journal of Computer Vision*, vol. 130, no. 12, pp. 2962–2979, 2022. Available: http://arxiv.org/abs/2208.14958.

[122] W. Yuan, T. Khot, D. Held, C. Mertz and M. Hebert, „PCN: Point Completion Network," *2018 International Conference on 3D Vision (3DV)*, pp. 728–737, 2018. Available: https://wentaoyuan.github.io/pcn..

[123] Y. Yang, C. Feng, Y. Shen and D. Tian, „FoldingNet: Point Cloud Auto-encoder via Deep Grid Deformation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 206–215, 2017, ISBN: 9781538664209. DOI: 10.48550/arxiv.1712.07262. Available: https://arxiv.org/abs/1712.07262v2.

[124] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao, „3D ShapeNets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, DOI: 10.1109/CVPR.2015.7298801.

[125] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi and F. Yu, „ShapeNet: An Information-Rich 3D Model Repository," 2015. Available: https://arxiv.org/abs/1512.03012v1.

[126] L. Zou, H. Tang, K. Chen and K. Jia, „Geometry-Aware Self-Training for Unsupervised Domain Adaptation on Object Point Clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2021, DOI: 10.1109/ICCV48922.2021.00634.

[127] Y. Ganin and V. Lempitsky, „Unsupervised domain adaptation by backpropagation," in *32nd International Conference on Machine Learning, ICML 2015*, 2015.

[128] J. Gebele, B. Stuhr and J. Haselberger, „CARLANE: A Lane Detection Benchmark for Unsupervised Domain Adaptation from Simulation to multiple Real-World Domains," in *Advances in Neural Information Processing Systems*, 2022, pp. 4046–4058, DOI: 10.34740/kaggle/dsv/3798459. Available: http://arxiv.org/abs/2206.08083%20http://dx.doi.org/10.34740/kaggle/dsv/3798459.

[129] S. Yan, Z. Yang, H. Li, L. Guan, H. Kang, G. Hua and Q. Huang, „IAE: Implicit Autoencoder for Point Cloud Self-supervised Representation Learning," 2022, DOI: 10.48550/arxiv.2201.00785. Available: https://arxiv.org/abs/2201.00785v3.

[130] L. McInnes, J. Healy, N. Saul and L. Großberger, „UMAP: Uniform Manifold Approximation and Projection,“ *Journal of Open Source Software*, vol. 3, no. 29, 2018, DOI: 10.21105/joss.00861.

[131] C. Hubschneider, S. Roesler and J. M. Zöllner, „Unsupervised Evaluation of Lidar Domain Adaptation,“ *2020 IEEE 23rd International Conference on Intelligent Transportation Systems, ITSC 2020*, 2020, ISBN: 9781728141497. DOI: 10.1109/ITSC45102.2020.9294540.

[132] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang and W. Shi, „Photo-realistic single image super-resolution using a generative adversarial network,“ in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, DOI: 10.1109/CVPR.2017.19.

[133] R. Kesten, 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. „*Woven Planet Perception Dataset 2020*,“ 2019. Available: https://woven.toyota/en/perception-dataset.

[134] P. Xiao, Z. Shao, S. Hao, Z. Zhang, X. Chai, J. Jiao, Z. Li, J. Wu, K. Sun, K. Jiang, Y. Wang and D. Yang, „PandaSet: Advanced Sensor Suite Dataset for Autonomous Driving,“ in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2021, DOI: 10.1109/ITSC48978.2021.9565009.

[135] G. Wilson and D. J. Cook, „A Survey of Unsupervised Deep Domain Adaptation,“ *ACM Transactions on Intelligent Systems and Technology*, vol. 11, no. 5, 2020, DOI: 10.1145/3400066.

[136] G. Ros, L. Sellart, J. Materzynska, D. Vazquez and A. M. Lopez, „The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes,“ in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, DOI: 10.1109/CVPR.2016.352.

[137] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth and B. Schiele, „The Cityscapes Dataset for Semantic Urban Scene Understanding,“ in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, DOI: 10.1109/CVPR.2016.350.

[138] A. Dundar, M.-Y. Liu, T.-C. Wang, J. Zedlewski and J. Kautz, „Domain Stylization: A Strong, Simple Baseline for Synthetic to Real Image Domain Adaptation,“ 2018.

[139] F. Yu, V. Koltun and T. Funkhouser, „Dilated residual networks,“ in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, DOI: 10.1109/CVPR.2017.75.

[140] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, „Rethinking Atrous Convolution for Semantic Image Segmentation Liang-Chieh,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, 2018.

[141] G. Adam, V. Chitalia, N. Simha, A. Ismail, S. Kulkarni, V. Narayan and M. Schulze, „Robustness and Deployability of Deep Object Detectors in Autonomous Driving,“ *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, pp. 4128–4133, 2019, ISBN: 9781538670248. DOI: 10.1109/ITSC.2019.8917352.

[142] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu and A. C. Berg, „SSD: Single shot multibox detector,“ in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, DOI: 10.1007/978-3-319-46448-0{_}2.

[143] V. Seib, B. Lange and S. Wirtz, „Mixing Real and Synthetic Data to Enhance Neural Network Training – A Review of Current Approaches,“ 2020. Available: http://arxiv.org/abs/2007.08781.

[144] S. Burdorf, K. Plum and D. Hasenklever, „Reducing the Amount of Real World Data for Object Detector Training with Synthetic Data,“ 2022, DOI: 10.48550/arxiv.2202.00632. Available: https://arxiv.org/abs/2202.00632v1.

[145] M. Ljungqvist, O. Nordander, A. Mildner, T. Liu and P. Nugues, „Object Detector Differences When using Synthetic and Real Training Data,“ in *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2022, pp. 48–59, ISBN: 978-989-758-555-5. DOI: 10.5220/0010778200003124. Available: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0010778200003124.

[146] S. Kornblith, M. Norouzi, H. Lee and G. Hinton, „Similarity of Neural Network Representations Revisited,“ in *International Conference on Machine Learning*, 2019, pp. 3519–3529. Available: http://arxiv.org/abs/1905.00414.

[147] X. Yue, B. Wu, S. A. Seshia, K. Keutzer and A. L. Sangiovanni-Vincentelli, „A LiDAR Point Cloud Generator: from a Virtual World to Autonomous Driving,“ in *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, 2018, pp. 458–464. Available: http://arxiv.org/abs/1804.00103.

[148] S. Spiegel and J. Chen, „Using Simulation Data From Gaming Environments For Training A Deep Learning Algorithm On 3D Point Clouds,“ *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. VIII-4/W2-2021, pp. 67–74, 2021, DOI: 10.5194/isprs-annals-VIII-4-W2-2021-67-2021.

[149] F. Piewak, P. Pinggera and M. Zöllner, „Analyzing the Cross-Sensor Portability of Neural Network Architectures for LiDAR-based Semantic Labeling,“ in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, DOI: 10.1109/ITSC.2019.8917412.

[150] F. Langer, A. Milioto, A. Haag, J. Behley and C. Stachniss, „Domain Transfer for Semantic Segmentation of LiDAR Data using Deep Neural Networks,“ in *IEEE International Conference on Intelligent Robots and Systems*, 2020, DOI: 10.1109/IROS45743.2020.9341508.

[151] C. B. Rist, M. Enzweiler and D. M. Gavrila, „Cross-sensor deep domain adaptation for LiDAR detection and segmentation,“ in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2019, DOI: 10.1109/IVS.2019.8814047.

[152] A. Elhadidy, M. Afifi, M. Hassoubah, Y. Ali and M. Elhelw, „Improved Semantic Segmentation of Low-Resolution 3D Point Clouds Using Supervised Domain Adaptation,“ in *2nd Novel Intelligent and Leading Emerging Sciences Conference, NILES 2020*, 2020, DOI: 10.1109/NILES50944.2020.9257903.

[153] L. Yi, B. Gong and T. Funkhouser, „Complete & Label: A domain adaptation approach to semantic segmentation of LiDAR point clouds,“ in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021, DOI: 10.1109/CVPR46437.2021.01511.

[154] P. Jiang and S. Saripalli, „LiDARNet: A Boundary-Aware Domain Adaptation Model for Point Cloud Semantic Segmentation,“ in *Proceedings - IEEE International Conference on Robotics and Automation*, 2021, DOI: 10.1109/ICRA48506.2021.9561255.

[155] I. Alonso, L. Riazuelo, L. Montesano and A. C. Murillo, „Domain adaptation in LiDAR semantic segmentation,“ *arXiv*, 2020. Available: https://arxiv.org/pdf/2010.12239.pdf.

[156] M. Jaritz, T. H. Vu, R. De Charette, E. Wirbel and P. Perez, „XMUDA: Cross-Modal Unsupervised Domain Adaptation for 3D Semantic Segmentation,“ in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020, DOI: 10.1109/CVPR42600.2020.01262.

[157] S. Zhao, Y. Wang, B. Li, B. Wu, Y. Gao, P. Xu, T. Darrell and K. Keutzer, „ePointDA: An End-to-End Simulation-to-Real Domain Adaptation Framework for LiDAR Point Cloud Segmentation,“ in *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, 2021, DOI: 10.1609/aaai.v35i4.16464.

[158] B. Wu, X. Zhou, S. Zhao, X. Yue and K. Keutzer, „SqueezeSegV2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a LiDAR point cloud,“ in *Proceedings - IEEE International Conference on Robotics and Automation*, 2019, DOI: 10.1109/ICRA.2019.8793495.

[159] D. Dworak, F. Ciepiela, J. Derbisz, I. Izzat, M. Komorkiewicz and M. Wojcik, „Performance of LiDAR object detection deep learning architectures based on artificially generated point cloud data from CARLA simulator,“ *2019 24th International Conference on Methods and Models in Automation and Robotics, MMAR 2019*, pp. 600–605, 2019, ISBN: 9781728109336. DOI: 10.1109/MMAR.2019.8864642.

[160] W. Ali, S. Abdelkarim, M. Zidan, M. Zahran and A. E. Sallab, „YOLO3D: End-to-end real-time 3D oriented object bounding box detection from LiDAR point cloud,“ in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, DOI: 10.1007/978-3-030-11015-4{_}54.

[161] J. Fang, D. Zhou, F. Yan, T. Zhao, F. Zhang, Y. Ma, L. Wang and R. Yang, „Augmented LiDAR Simulator for Autonomous Driving,“ *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1931–1938, 2018, DOI: 10.1109/LRA.2020.2969927. Available: http://dx.doi.org/10.1109/LRA.2020.2969927.

[162] S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W.-C. Ma and R. Urtasun, „LiDARsim: Realistic LiDAR Simulation by Leveraging the Real World,“ in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11164–11173, ISBN: 978-1-7281-7168-5. DOI: 10.1109/CVPR42600.2020.01118. Available: https://ieeexplore.ieee.org/document/9157601/.

[163] D. Tsai, J. S. Berrio, M. Shan, S. Worrall and E. Nebot, „See Eye to Eye: A Lidar-Agnostic 3D Detection Framework for Unsupervised Multi-Target Domain Adaptation,“ *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7904–7911, 2022, DOI: 10.1109/LRA.2022.3185783. Available: https://ieeexplore.ieee.org/document/9804815/.

[164] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang and H. Li, „PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection,“ *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Available: http://arxiv.org/abs/1912.13192.

[165] Y. Yan, Y. Mao and B. Li, „SECOND: Sparsely Embedded Convolutional Detection,“ *Sensors 2018, Vol. 18, Page 3337*, vol. 18, no. 10, p. 3337, 2018, DOI: 10.3390/S18103337. Available: https://www.mdpi.com/1424-8220/18/10/3337/htm.

[166] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand and V. Lempitsky, „Domain-adversarial training of neural networks,“ *Journal of Machine Learning Research*, vol. 17, 2016.

[167] S. J. Pan and Q. Yang, „A Survey on Transfer Learning,“ *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010, DOI: 10.1109/TKDE.2009.191.

[168] S. Chen, X. Jia, J. He, Y. Shi and J. Liu, „Semi-supervised Domain Adaptation based on Dual-level Domain Mixing for Semantic Segmentation,“ in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021, DOI: 10.1109/CVPR46437.2021.01087.

[169] Y. Chen, X. Ouyang, K. Zhu and G. Agam, „Semi-supervised Dual-Domain Adaptation for Semantic Segmentation," in *Proceedings - International Conference on Pattern Recognition*, 2022, DOI: 10.1109/ICPR56361.2022.9956524.

[170] A. Mütze, M. Rottmann and H. Gottschalk, „Semi-Supervised Domain Adaptation with CycleGAN Guided by Downstream Task Awareness," 2023, DOI: 10.5220/0011630900003417.

[171] Z. Wang, Y. Wei, R. Feris, J. Xiong, W. M. Hwu, T. S. Huang and H. Shi, „Alleviating semantic-level shift: A semi-supervised domain adaptation method for semantic segmentation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2020, DOI: 10.1109/CVPRW50498.2020.00476.

[172] N. Hanselmann, N. Schneider, B. Ortelt and A. Geiger, „Learning cascaded detection tasks with weakly-supervised domain adaptation," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2021, DOI: 10.1109/IV48863.2021.9575397.

[173] S. Paul, Y. H. Tsai, S. Schulter, A. K. Roy-Chowdhury and M. Chandraker, „Domain Adaptive Semantic Segmentation Using Weak Labels," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020, DOI: 10.1007/978-3-030-58545-7{_}33.

[174] M. Toldo, A. Maracani, U. Michieli and P. Zanuttigh. „*Unsupervised Domain Adaptation in Semantic Segmentation: A Review*," 2020. DOI: 10.3390/technologies8020035.

[175] G. Csurka, R. Volpi and B. Chidlovskii, „Unsupervised Domain Adaptation for Semantic Image Segmentation: a Comprehensive Survey," 2021. Available: https://arxiv.org/abs/2112.03241v1.

[176] J. Y. Zhu, T. Park, P. Isola and A. A. Efros, „Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, DOI: 10.1109/ICCV.2017.244.

[177] A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira and S. Birchfield, „Structured Domain Randomization: Bridging the Reality Gap by Context-Aware Synthetic Data," in *2019 International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7249–7255. Available: http://arxiv.org/abs/1810.10093.

[178] X. Yue, Y. Zhang, S. Zhao, A. Sangiovanni-Vincentelli, K. Keutzer and B. Gong, „Domain Randomization and Pyramid Consistency: Simulation-to-Real Generalization without Accessing Target Domain Data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2100–2110. Available: http://arxiv.org/abs/1909.00889.

[179] H. Abu Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger and C. Rother, „Augmented Reality Meets Computer Vision : Efficient Data Generation for Urban Driving Scenes," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 961–972, 2017, DOI: 10.48550/arxiv.1708.01566. Available: https://arxiv.org/abs/1708.01566v1.

[180] Z. Yihan, C. Wang, Y. Wang, H. Xu, C. Ye, Z. Yang and C. Ma, „Learning Transferable Features for Point Cloud Detection via 3D Contrastive Co-training," in *Advances in Neural Information Processing Systems*, 2021, pp. 21493–21504. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/b3b25a26a0828ea5d48d8f8aa0d6f9af-Paper.pdf.

[181] Z. Wang, S. DIng, Y. Li, M. Zhao, S. Roychowdhury, A. Wallin, G. Sapiro and Q. Qiu, „Range adaptation for 3d object detection in LiDAR," in *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, 2019, DOI: 10.1109/ICCVW.2019.00285.

[182] R. Debortoli, L. Fuxin, A. Kapoor and G. A. Hollinger, „Adversarial Training on Point Clouds for Sim-to-Real 3D Object Detection," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, 2021, DOI: 10.1109/LRA.2021.3093869.

[183] W. Zhang, W. Li and D. Xu, „SRDAN: Scale-aware and Range-aware Domain Adaptation Network for Cross-dataset 3D Object Detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021, DOI: 10.1109/CVPR46437.2021.00670.

[184] T. Shan, J. Wang, F. Chen, P. Szenher and B. Englot, „Simulation-based lidar super-resolution for ground vehicles," *Robotics and Autonomous Systems*, vol. 134, 2020, DOI: 10.1016/j.robot.2020.103647.

[185] Q. Xu, Y. Zhou, W. Wang, C. R. Qi and D. Anguelov, „SPG: Unsupervised Domain Adaptation for 3D Object Detection via Semantic Point Generation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2021, DOI: 10.1109/ICCV48922.2021.01516.

[186] K. Saleh, A. Abobakr, M. Attia, J. Iskander, D. Nahavandi, M. Hossny and S. Nahvandi, „Domain adaptation for vehicle detection from bird's eye view lidar point cloud data," in *Proceedings - 2019 International Conference on Computer Vision Workshop (ICCVW)*, 2019, DOI: 10.1109/ICCVW.2019.00404.

[187] A. E. Sallab, I. Sobh, M. Zahran and N. Essam, „LiDAR Sensor modeling and Data augmentation with GANs for Autonomous driving," 2019. Available: https://arxiv.org/pdf/1905.07290.pdf.

[188] A. E. Sallab, I. Sobh, M. Zahran and M. Shawky, „Unsupervised Neural Sensor Models for Synthetic LiDAR Data Augmentation," 2019. Available: https://arxiv.org/pdf/1911.10575.pdf.

[189] K. Nakashima and R. Kurazume, „Learning to Drop Points for LiDAR Scan Synthesis," in *IEEE International Conference on Intelligent Robots and Systems*, 2021, DOI: 10.1109/IROS51168.2021.9636747.

[190] C. Saltori, S. Lathuiliere, N. Sebe, E. Ricci and F. Galasso, „SF-UDA3D: Source-Free Unsupervised Domain Adaptation for LiDAR-Based 3D Object Detection," in *Proceedings - 2020 International Conference on 3D Vision, 3DV 2020*, 2020, DOI: 10.1109/3DV50981.2020.00087.

[191] J. Yang, S. Shi, Z. Wang, H. Li and X. Qi, „ST3D: Self-training for Unsupervised Domain Adaptation on 3D Object Detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021, DOI: 10.1109/CVPR46437.2021.01023.

[192] C. Fruhwirth-Reisinger, M. Opitz, H. Possegger and H. Bischof, „FAST3D: Flow-Aware Self-Training for 3D Object Detectors," *32nd British Machine Vision Conference, BMVC 2021*, 2021. Available: https://arxiv.org/abs/2110.09355v1.

[193] Y. You, C. A. Diaz-Ruiz, Y. Wang, W. L. Chao, B. Hariharan, M. Campbell and K. Q. Weinbergert, „Exploiting Playbacks in Unsupervised Domain Adaptation for 3D Object Detection in Self-Driving Cars," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2022, DOI: 10.1109/ICRA46639.2022.9811722.

[194] B. Wu, A. Wan, X. Yue and K. Keutzer, „SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2018, DOI: 10.1109/ICRA.2018.8462926.

[195] P. Morerio, J. Cavazza and V. Murino, „Minimal-entropy correlation alignment for unsupervised deep domain adaptation," in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

[196] S. Ioffe and C. Szegedy, „Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *32nd International Conference on Machine Learning, ICML 2015*, 2015.

[197] Y. Li, N. Wang, J. Shi, J. Liu and X. Hou, „Revisiting batch normalization for practical domain adaptation," in *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings*, 2017.

[198] Y. Wu and K. He, „Group Normalization,“ *International Journal of Computer Vision*, vol. 128, no. 3, 2020, DOI: 10.1007/s11263-019-01198-w.

[199] D. Ulyanov, A. Vedaldi and V. Lempitsky, „Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis,“ in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, DOI: 10.1109/CVPR.2017.437.

[200] H. Nam and H. E. Kim, „Batch-instance normalization for adaptively style-invariant neural networks,“ in *Advances in Neural Information Processing Systems*, 2018.

[201] O. Ronneberger, P. Fischer and T. Brox, „U-net: Convolutional networks for biomedical image segmentation,“ in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, DOI: 10.1007/978-3-319-24574-4{_}28.

[202] A. Milioto, I. Vizzo, J. Behley and C. Stachniss, „RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation,“ in *IEEE International Conference on Intelligent Robots and Systems*, 2019, DOI: 10.1109/IROS40897.2019.8967762.

[203] F. Piewak, P. Pinggera, M. Schäfer, D. Peter, B. Schwarz, N. Schneider, M. Enzweiler, D. Pfeiffer and M. Zöllner, „Boosting LIDAR-based semantic labeling by cross-modal training data generation,“ in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, DOI: 10.1007/978-3-030-11024-6{_}39.

[204] Y. Choi, M. Choi, M. Kim, J. W. Ha, S. Kim and J. Choo, „StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation,“ in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, DOI: 10.1109/CVPR.2018.00916.

[205] A. Royer, K. Bousmalis, S. Gouws, F. Bertsch, I. Mosseri, F. Cole and K. Murphy, „XGAN: Unsupervised image-to-image translation for many-to-many mappings,“ in *Domain Adaptation for Visual Understanding* 2020, DOI: 10.1007/978-3-030-30671-7{_}3.

[206] S. Benaim and L. Wolf, „One-sided unsupervised domain mapping,“ in *Advances in Neural Information Processing Systems*, 2017.

[207] Y. Taigman, A. Polyak and L. Wolf, „Unsupervised cross-domain image generation,“ in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[208] D. Yoon, T. Tang and T. Barfoot, „Mapless online detection of dynamic objects in 3D lidar,“ in *Proceedings - 2019 16th Conference on Computer and Robot Vision, CRV 2019*, 2019, DOI: 10.1109/CRV.2019.00023.

[209] J. Redmon and A. Farhadi, „YOLOv3: An Incremental Improvement,“ 2018. Available: http://arxiv.org/abs/1804.02767.

[210] L. Caccia, H. V. Hoof, A. Courville and J. Pineau, „Deep Generative Modeling of LiDAR Data,“ in *IEEE International Conference on Intelligent Robots and Systems*, 2019, DOI: 10.1109/IROS40897.2019.8968535.

[211] H. Durrant-Whyte and T. Bailey, „Simultaneous localization and mapping: Part I,“ *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, 2006, DOI: 10.1109/MRA.2006.1638022.

[212] Z. Luo, Z. Cai, C. Zhou, G. Zhang, H. Zhao, S. Yi, S. Lu, H. Li, S. Zhang and Z. Liu, „Unsupervised Domain Adaptive 3D Detection with Multi-Level Consistency,“ in *Proceedings of the IEEE International Conference on Computer Vision*, 2021, DOI: 10.1109/ICCV48922.2021.00874.

[213] S. Huch, L. Scalerandi, E. Rivera and M. Lienkamp, „Quantifying the LiDAR Sim-to-Real Domain Shift: A Detailed Investigation Using Object Detectors and Analyzing Point Clouds at Target-Level,“ *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 4, pp. 2970–2982, 2023, DOI: 10.1109/TIV.2023.3251650.

[214] A. Wischnewski, M. Geisslinger, J. Betz, T. Betz, F. Fent, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, F. Nobis, L. Ögretmen, M. Rowold, F. Sauerbeck, T. Stahl, R. Trauth, M. Lienkamp and B. Lohmann. „*Indy Autonomous Challenge - Autonomous Race Cars at the Handling Limits*,“ 2022. DOI: 10.48550/arxiv.2202.03807.

[215] Google Maps. „*Las Vegas Motor Speedway*,“ 2024. Available: https://www.google.com/maps/@36.2726556,-115.0118791,1054m/data=!3m1!1e3?entry=ttu.

[216] M. Zulfaj, „Development of a Pipeline for the Comparison of Real-World and Synthetic Autonomous Driving Data,“ Semesters Thesis, Technische Universität München, München, 2022.

[217] HEXAGON. „*PwrPak7 Produkt Sheet*,“ 2017. Available: https://hexagon.com/company/divisions/autonomy-and-positioning..

[218] M. Schneider, „Enhancing the Unity Autonomous Driving Simulator for the IAC,“ Interdisciplinary Project, Technische Universität München, München, 2022.

[219] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, „PyTorch: An imperative style, high-performance deep learning library,“ in *Advances in Neural Information Processing Systems*, 2019.

[220] K. M. Ting, „Precision and Recall,“ in *Encyclopedia of Machine Learning and Data Mining* 2017, DOI: 10.1007/978-1-4899-7687-1{_}659.

[221] M. Everingham, L. Van Gool, C. K. Williams, J. Winn and A. Zisserman, „The pascal visual object classes (VOC) challenge,“ *International Journal of Computer Vision*, vol. 88, no. 2, 2010, DOI: 10.1007/s11263-009-0275-4.

[222] A. Simonelli, S. R. R. Bulò, L. Porzi, M. López-Antequera and P. Kontschieder, „Disentangling Monocular 3D Object Detection,“ *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. Available: http://arxiv.org/abs/1905.12365.

[223] S. Huch and M. Lienkamp, „Towards Minimizing the LiDAR Sim-to-Real Domain Shift: Object-Level Local Domain Adaptation for 3D Point Clouds of Autonomous Vehicles,“ *Sensors*, vol. 23, no. 24, p. 9913, 2023, DOI: 10.3390/s23249913.

[224] C. Moenning and N. A. Dodgson, „Fast Marching farthest point sampling for implicit surfaces and point clouds,“ *Computer Laboratory Technical Report*, no. 565, 2003.

[225] D. Li, Y. Wei and R. Zhu, „A comparative study on point cloud down-sampling strategies for deep learning-based crop organ segmentation,“ *Plant Methods*, vol. 19, no. 1, pp. 1–26, 2023, DOI: 10.1186/S13007-023-01099-7/FIGURES/13. Available: https://plantmethods.biomedcentral.com/articles/10.1186/s13007-023-01099-7%20http://creativecommons.org/publicdomain/zero/1.0/.

[226] C. Zhou, C. Yunand, Z. Wenqingand, L. Junweiand, T. Tongand and C. Wang, „SeedFormer: Patch Seeds Based Point Cloud Completion with Upsample Transformer,“ in *Computer Vision – ECCV 2022*, 2022, pp. 416–432, ISBN: 978-3-031-20062-5. DOI: 10.1007/978-3-031-20062-5{_}24.

[227] I. Loshchilov and F. Hutter, „Decoupled weight decay regularization,“ in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[228] CARLA. „*Town 5 - CARLA Simulator*,“ Available: https://carla.readthedocs.io/en/latest/map_town05/.

[229] S. Huch, L. Scalerandi, E. Rivera and M. Lienkamp. „*S2R-DAD: Sim-to-Real Distribution-Aligned Dataset*,“ 2023. DOI: 10.14459/2023mp1695833. Available: https://mediatum.ub.tum.de/1631378?query=10.14459%2F2023mp1695833&show_id=1695833&srcnodeid=1631378.

[230] S. Huch, L. Scalerandi, E. Rivera and M. Lienkamp. „*TUMFTM/Sim2RealDistributionAlignedDataset: Sim-to-Real Distribution-Aligned Dataset (S2R-DAD) for Domain Shift and Domain Adaptation Analysis*,“ 2023. Available: https://github.com/TUMFTM/Sim2RealDistributionAlignedDataset.

[231] S. Huch and M. Lienkamp. „*TUMFTM/LOL_DA: LiDAR Object-Level Local Domain Adaptation*,“ 2023. Available: https://github.com/TUMFTM/LOL_DA.

[232] O. Odukha. „*How Will Self Driving Cars Save Money? | Intellias Blog*,“ Aug. 2023. Available: https://intellias.com/self-driving-car-save-money/.

[233] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson and A. Marrs, „Disruptive technologies: Advances that will transform life, business, and the global economy | McKinsey,“ McKinsey Global Institute, 05/2013. Available: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/disruptive-technologies.

[234] R. Zhang, Y. Wu, W. Jin and X. Meng, „Deep-Learning-Based Point Cloud Semantic Segmentation: A Survey,“ *Electronics 2023, Vol. 12, Page 3642*, vol. 12, no. 17, p. 3642, 2023, DOI: 10.3390/ELECTRONICS12173642. Available: https://www.mdpi.com/2079-9292/12/17/3642/htm.

[235] W. Lu, Y. Zhou, G. Wan, S. Hou and S. Song, „L3-net: Towards learning based lidar localization for autonomous driving,“ *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 6382–6391, 2019, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00655.

[236] S. Huch, F. Sauerbeck and J. Betz, „DeepSTEP - Deep Learning-Based Spatio-Temporal End-To-End Perception for Autonomous Vehicles,“ in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2023, DOI: 10.1109/IV55152.2023.10186768.

Prior Publications

During the development of this dissertation, publications and student theses were written in which partial aspects of this work were presented.

Journals; Scopus/Web of Science listed (peer-reviewed)

[9] J. Betz, T. Betz, F. Fent, M. Geisslinger, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, M. Lienkamp, B. Lohmann, F. Nobis, L. Ögretmen, M. Rowold, F. Sauerbeck, T. Stahl, R. Trauth, F. Werner and A. Wischnewski, „TUM autonomous motorsport: An autonomous racing software for the Indy Autonomous Challenge,“ *Journal of Field Robotics*, vol. 40, no. 4, pp. 783–809, 2023, DOI: 10.1002/ROB.22153. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/rob.22153.

[213] S. Huch, L. Scalerandi, E. Rivera and M. Lienkamp, „Quantifying the LiDAR Sim-to-Real Domain Shift: A Detailed Investigation Using Object Detectors and Analyzing Point Clouds at Target-Level,“ *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 4, pp. 2970–2982, 2023, DOI: 10.1109/TIV.2023.3251650.

[223] S. Huch and M. Lienkamp, „Towards Minimizing the LiDAR Sim-to-Real Domain Shift: Object-Level Local Domain Adaptation for 3D Point Clouds of Autonomous Vehicles,“ *Sensors*, vol. 23, no. 24, p. 9913, 2023, DOI: 10.3390/s23249913.

Journals, Conferences, Periodicals, Reports, Conference Proceedings and Poster, etc.; not Scopus/Web of Science listed

[214] A. Wischnewski, M. Geisslinger, J. Betz, T. Betz, F. Fent, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, F. Nobis, L. Ögretmen, M. Rowold, F. Sauerbeck, T. Stahl, R. Trauth, M. Lienkamp and B. Lohmann. „*Indy Autonomous Challenge - Autonomous Race Cars at the Handling Limits*,“ 2022. DOI: 10.48550/arxiv.2202.03807.

Non-thesis-relevant publications; Scopus/Web of Science listed (peer-reviewed)

[27] F. Sauerbeck, S. Huch, F. Fent, P. Karle, D. Kulmer and J. Betz, „Learn to See Fast: Lessons Learned From Autonomous Racing on How to Develop Perception Systems,“ *IEEE Access*, vol. 11, 2023, DOI: 10.1109/ACCESS.2023.3272750.

[236] S. Huch, F. Sauerbeck and J. Betz, „DeepSTEP - Deep Learning-Based Spatio-Temporal End-To-End Perception for Autonomous Vehicles,“ in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2023, DOI: 10.1109/IV55152.2023.10186768.

P. Rosenberger, M. Holder, S. Huch, H. Winner, T. Fleek, M. R. Zofka, J. M. Zollner, T. D'Hondt and B. Wassermann, „Benchmarking and functional decomposition of automotive lidar sensor models," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2019-June, pp. 632–639, 2019, ISBN: 9781728105604. DOI: 10.1109/IVS.2019.8814081.

P. Karle, T. Betz, M. Bosk, F. Fent, N. Gehrke, M. Geisslinger, L. Gressenbuch, P. Hafemann, S. Huber, M. Hübner, S. Huch, G. Kaljavesi, T. Kerbl, D. Kulmer, T. Mascetta, S. Maierhofer, F. Pfab, F. Rezabek, E. Rivera, S. Sagmeister, L. Seidlitz, F. Sauerbeck, I. Tahiraj, R. Trauth, N. Uhlemann, G. Würsching, B. Zarrouki, M. Althoff, J. Betz, K. Bengler, G. Carle, F. Diermeyer, J. Ott and M. Lienkamp. „*EDGAR: An Autonomous Driving Research Platform - From Feature Development to Real-World Application*," 2023. DOI: 10.48550/arXiv.2309.15492.

P. Karle, F. Fent, S. Huch, F. Sauerbeck and M. Lienkamp, „Multi-Modal Sensor Fusion and Object Tracking for Autonomous Racing," *IEEE Transactions on Intelligent Vehicles*, 2023, DOI: 10.1109/TIV.2023.3271624.

S. Huch, A. Ongel, J. Betz and M. Lienkamp, „Multi-Task End-to-End Self-Driving Architecture for CAV Platoons," *Sensors (Switzerland)*, vol. 21, no. 4, 2021, DOI: 10.3390/s21041039.

Thesis-relevant open-source software

[229] S. Huch, L. Scalerandi, E. Rivera and M. Lienkamp. „*S2R-DAD: Sim-to-Real Distribution-Aligned Dataset*," 2023. DOI: 10.14459/2023mp1695833. Available: https://mediatum.ub.tum.de/1631378?query=10.14459%2F2023mp1695833&show_id=1695833&srcnodeid=1631378.

[230] S. Huch, L. Scalerandi, E. Rivera and M. Lienkamp. „*TUMFTM/Sim2RealDistributionAlignedDataset: Sim-to-Real Distribution-Aligned Dataset (S2R-DAD) for Domain Shift and Domain Adaptation Analysis*," 2023. Available: https://github.com/TUMFTM/Sim2RealDistributionAlignedDataset.

[231] S. Huch and M. Lienkamp. „*TUMFTM/LOL_DA: LiDAR Object-Level Local Domain Adaptation*," 2023. Available: https://github.com/TUMFTM/LOL_DA.

Supervised Student Theses

The following student theses were written within the framework of the dissertation under the supervision of the author in terms of content, technical and scientific support as well as under relevant guidance of the author. In the following, the bachelor, semester and master theses relevant and related to this dissertation are listed. Many thanks to the authors of these theses for their extensive support within the framework of this research project.

[216] M. Zulfaj, „Development of a Pipeline for the Comparison of Real-World and Synthetic Autonomous Driving Data," Semesters Thesis, Technische Universität München, München, 2022.

[218] M. Schneider, „Enhancing the Unity Autonomous Driving Simulator for the IAC," Interdisciplinary Project, Technische Universität München, München, 2022.

Y. Tian, „Evaluation of End-to-End Deep Learning Architectures for Autonomous Vehicles," Master's Thesis, Technische Universität München, München, 2020.

B. B. Gnana Sekar, „3D-Object Detection for Autonomous Racing Vehicles," Master's Thesis, RWTH Aachen University, Aachen, 2021.

I. Freitas Sym, „Monocular 3D Object Detection for Autonomous Race Cars Using Deep Neural Networks," Semesters Thesis, Technische Universität München, München, 2021.

R. Essmoum, „Implementation and Optimization of a LiDAR Data Preprocessing Pipeline for Real-Time Object Detection in Autonomous Racing Vehicles," Master's Thesis, Technische Universität München, München, 2021.

K. Hartmann, „Object Detection of Unstructured Objects with Lidar-Sensors for Autonomous Vehicles," Master's Thesis, Technische Universität München, München, 2021.

M. Liu, „Development of a Camera Object Detection Pipeline for Autonomous Race Cars," Semesters Thesis, Technische Universität München, München, 2021.

K. Fleischer, „Investigation of the Influence of Sensor Noise in Synthetic Point Clouds for 3D Object Detection of Autonomous Vehicles," Master's Thesis, Technische Universität München, München, 2021.

K. Fahmy, „Development of a Deep Learning Algorithm for the Sim-to-Real Adaptation of Synthetic Point Clouds," Master's Thesis, Technische Universität München, München, 2022.

S. Lachmann, „Autoencoder based Domain Adaptation of Synthetic 3D Point Clouds for Autonomous Driving," Bachelor's Thesis, Technische Universität München, München, 2022.

M. Schoeffmann, „Domain Adaption of 2D LiDAR Bird's Eye View Images to 3D Point Clouds for Autonomous Driving," Master's Thesis, Technische Universität München, München, 2022.

L. Scalerandi, „Development of a Pipeline for the Comparison of Real-World and Synthetic Autonomous Driving Data," Interdisciplinary Project, Technische Universität München, München, 2023.

F. Kleinsteuber, „Domain Adaptation Of Front View Synthetic Point Clouds Using GANs For Autonomous Driving," Master's Thesis, Technische Universität München, München, 2022.

C. Zern, „Development and Integration of an Autonomous Driving Software Stack in a Simulation Environment,“ Semester Thesis, Technische Universität München, München, 2022.

C. Meng, „Development of a Multi-Sensor LiDAR Data Preprocessing Pipeline for an Autonomous Vehicle,“ Semester Thesis, Technische Universität München, München, 2022.

T. Mascetta, „KPConv-based AI-Architecture for Sim-To-Real Domain Adaptation of LiDAR Point Clouds for Autonomous Driving,“ Master's Thesis, Technische Universität München, München, 2023.

D. Khayrutdinov, „Object Detection for Autonomous Driving based on Clustering of LiDAR Point Clouds,“ Bachelor's Thesis, Technische Universität München, München, 2023.

B. Braun, „Integration and Testing of a Neural Network for 3D Object Detection using LiDAR Point Clouds,“ Bachelor's Thesis, Technische Universität München, München, 2023.

D. Gacic, „LiDAR CenterPoint - Evaluation of LiDAR Object Detection Deep Neural Network,“ Semesters Thesis, Technische Universität München, München, 2023.

Z. Ye, „Development of Perception Algorithms for a Level 5 Research Vehicle,“ Interdisciplinary Project, Technische Universität München, München, 2023.

S. F. Salinas, „Development of a Single-Object Sim-to-Real Neural Network for 3D Object Detection using Point Clouds,“ Master's Thesis, Technische Universität München, München, 2023.

L. A. Stratil, „A LiDAR Point Cloud Pseudo-Labeling Pipeline to Efficiently Generate Data for 3D Object Detection Neural Networks,“ Master's Thesis, Technische Universität München, München, 2023.

R. Hajda, „Learning Dropout of Points in Sim-to-Real Domain Adaptation of LiDAR Point Clouds,“ Interdisciplinary Project, Technische Universität München, München, 2023.

K. Katzkowski, „Learning Sim-to-Real Domain Adaptation for 3D LiDAR Point Clouds,“ Interdisciplinary Project, Technische Universität München, München, 2023.

Appendix

A Appendix

A.1 LiDAR Sim-to-Real Domain Shift Quantification Results

Table A.1: Average Precision and Recall for the two object detection networks PointRCNN and PointPillars trained and tested on four different datasets. The evaluation of both AP and Recall is carried out at different IoU thresholds, specifically $50\,\%$ and $70\,\%$, which are referred to as 3D AP (0.5) and 3D AP (0.7), respectively. The values given are the average results from five identical training runs, with the standard deviation indicated in parentheses. The evaluation is performed over the full range of the point cloud, encompassing points with a radial distance up to $100\,\mathrm{m}$. All values in % (from [213]).

Network	Train dataset	Test dataset	3D AP (0.5)	3D AP (0.7)	Recall (0.5)	Recall (0.7)
PointRCNN	Real	Real	74.33 (1.9)	51.96 (1.21)	92.12 (0.7)	80.92 (1.21)
		Sim	86.56 (1.0)	62.53 (2.62)	93.78 (0.48)	87.26 (0.88)
		Sim Noise	77.27 (2.08)	52.25 (2.15)	87.88 (0.74)	77.48 (0.98)
		Sim Downsampled	83.64 (0.72)	58.17 (3.37)	92.7 (0.5)	84.62 (0.89)
	Sim	Real	56.29 (1.36)	38.23 (0.98)	76.56 (0.78)	65.5 (0.95)
		Sim	97.31 (1.35)	96.82 (1.16)	99.46 (0.1)	98.64 (0.21)
		Sim Noise	88.96 (1.11)	88.48 (1.19)	94.4 (0.82)	92.96 (0.83)
		Sim Downsampled	96.31 (0.18)	96.28 (0.17)	99.04 (0.14)	98.08 (0.26)
	Sim Noise	Real	60.36 (1.35)	41.61 (1.65)	78.3 (0.63)	67.8 (0.78)
		Sim	97.07 (1.07)	96.58 (0.12)	99.42 (0.12)	98.48 (0.16)
		Sim Noise	96.93 (1.29)	96.43 (0.43)	99.08 (0.13)	98.16 (0.21)
		Sim Downsampled	96.21 (0.29)	96.19 (0.29)	99.08 (0.25)	97.92 (0.3)
	Sim Downsampled	Real	55.84 (2.72)	37.57 (2.44)	75.98 (0.55)	65.72 (0.17)
		Sim	97.33 (1.08)	96.82 (0.94)	99.3 (0.11)	98.48 (0.32)
		Sim Noise	86.86 (2.48)	86.38 (3.31)	92.78 (1.91)	90.74 (2.29)
		Sim Downsampled	95.93 (0.5)	95.87 (0.54)	99.02 (0.13)	98.12 (0.19)
PointPillars	Real	Real	69.22 (0.0)	41.1 (0.0)	80.5 (0.0)	55.7 (0.0)
		Sim	68.65 (0.03)	20.68 (0.0)	83.1 (0.0)	41.0 (0.0)
		Sim Noise	67.32 (0.0)	19.98 (0.0)	83.2 (0.0)	40.7 (0.0)
		Sim Downsampled	63.44 (0.0)	18.41 (0.02)	82.3 (0.0)	39.7 (0.0)
	Sim	Real	30.49 (0.02)	13.36 (0.12)	69.7 (0.0)	39.9 (0.0)
		Sim	98.75 (0.0)	98.18 (0.0)	99.7 (0.0)	98.9 (0.0)
		Sim Noise	98.98 (0.0)	98.63 (0.0)	99.8 (0.0)	99.1 (0.0)
		Sim Downsampled	98.63 (0.0)	98.11 (0.0)	99.6 (0.0)	98.7 (0.0)
	Sim Noise	Real	39.81 (0.02)	18.77 (0.08)	68.8 (0.0)	43.2 (0.0)
		Sim	99.41 (0.0)	99.22 (0.0)	99.8 (0.0)	98.7 (0.0)
		Sim Noise	98.95 (0.0)	98.51 (0.0)	99.7 (0.0)	98.3 (0.0)
		Sim Downsampled	99.07 (0.0)	98.71 (0.0)	99.8 (0.0)	98.0 (0.0)
	Sim Downsampled	Real	30.25 (0.16)	14.05 (0.02)	67.0 (0.0)	39.1 (0.0)
		Sim	99.15 (0.0)	94.96 (0.0)	99.6 (0.0)	96.7 (0.0)
		Sim Noise	98.73 (0.0)	95.28 (0.0)	99.5 (0.0)	96.9 (0.0)
		Sim Downsampled	98.31 (0.0)	94.46 (0.0)	99.6 (0.0)	97.5 (0.0)

A.2 LiDAR Sim-to-Real Domain Adaptation Results for Adaptation of 3D Point Clouds at the Object-Level

Table A.2: The IoU thresholds for recall are set at $70\,\%$ and $50\,\%$, referred to as Recall (0.7) and Recall (0.5), respectively. The PointRCNN and PointPillars networks are trained five times with the specified training dataset and then assessed on the test split of the *real* dataset. ↑: Higher values are better, with the highest value in each category highlighted in bold. All values in % (from [223]).

Network	**Train Dataset**	**Recall (0.7) ↑**	**Recall (0.5) ↑**
PointRCNN	**Sim (Source)**	32.22±2.11	44.52±2.91
	Sim-Noise Object	29.92±1.20	43.04±1.44
	Sim-to-Real Object No-GAN	32.18±1.88	**44.20±1.51**
	Sim-to-Real Object $\delta = 3$	31.32±3.56	42.76±3.05
	Sim-to-Real Object $\delta = 5$	32.12±1.64	42.66±0.78
	Sim-to-Real Object Full	**33.00±0.73**	43.22±0.88
	Real (Target)	38.50±2.32	64.32±1.63
PointPillars	**Sim (Source)**	15.50±0.00	26.40±0.00
	Sim-Noise Object	18.00±0.00	32.90±0.00
	Sim-to-Real Object No-GAN	18.70±0.00	34.80±0.00
	Sim-to-Real Object $\delta = 3$	15.90±0.00	33.10±0.00
	Sim-to-Real Object $\delta = 5$	19.40±0.00	36.20±0.00
	Sim-to-Real Object Full	**21.90±0.00**	**36.40±0.00**
	Real (Target)	22.10±0.00	34.40±0.00

A.3 LiDAR Sim-to-Real Domain Adaptation Results for Adaptation of 3D Point Clouds at the Scene-Level

Table A.3: The IoU thresholds for recall are set at $70\,\%$ and $50\,\%$, referred to as Recall (0.7) and Recall (0.5), respectively. The PointRCNN and PointPillars networks are trained five times with the specified training dataset and then assessed on the test split of the *real* dataset. ↑: Higher values are better, with the highest value in each category highlighted in bold. All values in %.

Network	**Train Dataset**	**Recall (0.7) ↑**	**Recall (0.5) ↑**
PointRCNN	**Sim (Source)**	32.22$_{\pm 2.11}$	44.52$_{\pm 2.91}$
	Sim-to-Real Scene $\delta = 7$	35.42$_{\pm 3.30}$	51.46$_{\pm 0.79}$
	Sim-to-Real Scene $\delta = 5$	38.54$_{\pm 2.75}$	55.66$_{\pm 0.89}$
	Sim-to-Real Scene Full	**46.26**$_{\pm \mathbf{1.31}}$	**62.42**$_{\pm \mathbf{0.92}}$
	Real (Target)	38.50$_{\pm 2.32}$	64.32$_{\pm 1.63}$
PointPillars	**Sim (Source)**	15.50$_{\pm 0.00}$	26.40$_{\pm 0.00}$
	Sim-to-Real Scene $\delta = 7$	22.90$_{\pm 0.00}$	39.10$_{\pm 0.00}$
	Sim-to-Real Scene $\delta = 5$	24.30$_{\pm 0.00}$	37.30$_{\pm 0.00}$
	Sim-to-Real Scene Full	**25.80**$_{\pm \mathbf{0.00}}$	**39.70**$_{\pm \mathbf{0.00}}$
	Real (Target)	22.10$_{\pm 0.00}$	34.40$_{\pm 0.00}$

A.4 LiDAR Sim-to-Real Domain Adaptation Results for Adaptation of 3D Point Clouds from Non-Distribution-Aligned Data

Table A.4: The IoU thresholds for recall are set at 70 % and 50 %, referred to as Recall (0.7) and Recall (0.5), respectively. The PointRCNN and PointPillars networks are trained five times with the specified training dataset and then assessed on the test split of the *KITTI* dataset. ↑: Higher values are better, with the highest value in each category highlighted in bold. All values in %.

Network	**Train Dataset**	**Recall (0.7) ↑**	**Recall (0.5) ↑**
PointRCNN	**CARLA (Source)**	38.26$\pm$0.84	55.88$\pm$0.90
	CARLA-to-KITTI Object	38.37$\pm$1.14	56.25$\pm$1.07
	CARLA-to-KITTI Scene	**43.20$\pm$0.66**	**61.59$\pm$0.71**
	KITTI (Target)	76.99$\pm$0.32	84.45$\pm$0.33
PointPillars	**Sim (Source)**	14.16$\pm$0.00	39.61$\pm$0.00
	CARLA-to-KITTI Object	13.93$\pm$0.00	42.13$\pm$0.00
	CARLA-to-KITTI Scene	**16.29$\pm$0.00**	**43.98$\pm$0.00**
	Real (Target)	60.85$\pm$0.00	82.45$\pm$0.00