OLISTER: OBSERVING LIDAR-INDUCED SOURCES FOR TRANSFERABILITY, ESTIMATION AND ROBUSTNESS IN 3D OBJECT DETECTION

By

Onur Can Yücedağ

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science—Master of Science

2025

ABSTRACT

Unsupervised Domain Adaptation (UDA) for 3D object detection in autonomous driving faces challenges due to various sources of domain shift, such as differences in LiDAR resolution, the use of synthetic versus real-world data, scenery variations, and sensor configurations (e.g., sensor placement and number). This thesis systematically investigates these domain shifts through controlled experiments using synthetic datasets generated via the CARLA simulator, enabling precise isolation and quantification of each factor. To facilitate these experiments, two software tools are introduced: `carlaSceneCollector`, designed for efficient synthetic data generation, and `rosbag2nuScenes`, which converts ROSBag data into the widely adopted nuScenes format. The study emphasizes two critical sources of domain shift: LiDAR resolution and the synthetic-to-real data shift. It identifies saturation effects at intermediate LiDAR resolutions (32–64 channels) and analyzes how varying resolution shifts impact detection performance, particularly noting the disproportionate effects on smaller objects. It evaluates various performance metrics, highlighting the robustness of the NuScenes Detection Score (NDS) compared to traditional metrics like mean Average Precision (mAP). Simultaneously, the synthetic-to-real domain shift is analyzed through systematic comparisons across the nuScenes, adaScenes, and carlaScenes datasets. This reveals that synthetic-to-real differences significantly surpass the impact of LiDAR resolution shifts, underscoring profound discrepancies between simulated and real-world LiDAR point clouds. The thesis further addresses limitations in the default voxelization settings of the CenterPoint model by proposing adaptive voxelization techniques and structural enhancements, enhancing model adaptability across resolutions. Finally, it examines real-world datasets like nuScenes, highlighting their complexity and diversity as key factors in achieving robust model performance and improved generalization.

ACKNOWLEDGMENTS

I would like to express my deepest thanks to my supervisor, Prof. Joshua Siegel, for being constantly supportive and constructive towards my work on this thesis. He has been a true inspiration and a great teacher to me during my time in the graduate program.

Special thanks to my committee members, Prof. Philip McKinley and Dr. Yu Kong, for their enthusiasm towards my project and their invaluable expertise and time in evaluating this work.

In addition, I would like to acknowledge Dr. Ali Ufuk Peker, Dr. Kerem Par and ADASTEC Corp. for encouraging me to pursue graduate studies and for their financial support throughout this endeavor.

I sincerely thank my friends and colleagues, who have been by my side whenever I needed them and helped me through times of stress.

Finally, I am incredibly grateful for my loving wife. She is the bright light who guided me throughout this journey, kept my priorities straight and always there whenever I needed her support.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

UDA	Unsupervised Domain Adaptation
LiDAR	Light Detection and Ranging
ROS	Robot Operating System
R_{LR}	LiDAR Resolution
R_{SR}	Synthetic vs. Real data
R_{SC}	Scenery Disparities
R_{LP}	Differences in LiDAR Placement
R_{LC}	Variations in the Number of LiDAR Sensors
mAP	mean Average Precision
NDS	NuScenes Detection Score
AP	Average Precision
IoU	Intersection over Union

CHAPTER 1

INTRODUCTION

Autonomous driving depends on a vehicle's ability to accurately perceive its surroundings. This perception is achieved through a combination of sensors—LiDAR, radar, and cameras—that work together to build a detailed understanding of the environment. 3D object detection, a core component of this perception, involves identifying and localizing objects in three-dimensional space. This is crucial for tasks such as path planning, collision avoidance, and decision-making.

LiDAR sensors are indispensable for 3D object detection. They provide accurate and dense 3D measurements in the form of point clouds. Unlike cameras, which primarily capture 2D color and texture information, LiDAR sensors use laser beams to measure distances, generating a 3D representation of the environment. This geometric information is essential for accurately localizing objects, estimating their size and shape, and determining their distance from the vehicle. LiDAR's superior spatial resolution is critical for precise 3D object detection, especially in complex and dynamic environments. However, each LiDAR sensor has unique characteristics based on its provider, firmware, and measurement mechanism. For instance, mechanical LiDAR products, a common type, exhibit significant variations in their ray patterns—the specific arrangement and angles at which laser beams are emitted by a LiDAR sensor. These ray patterns determine how comprehensively and densely the environment is scanned. Such variations directly impact the density and distribution of point clouds, which are composite data structures. These structures consist of two main types of data. First, there are positional fields—X, Y, and Z coordinates—that indicate each point's location in space, with errors bounded by the sensor's resolution. Second, there are signal noise-related fields that capture additional information about the reflected laser signal. These include intensity (the strength of the reflected signal), elongation (the stretching of the return pulse), ambient (background light levels), and reflectivity (how well a surface reflects the laser). The noise fields are highly provider-dependent, leading to distributional differences in these values. These noise fields are commonly uncalibrated, meaning that the same surface can activate different noise signals depending on environmental conditions such as ambient lighting or

weather situations. This lack of standardization poses challenges for models trained on data from one LiDAR sensor, as they may not generalize well to data from another, even when capturing the same scene. Even when noise fields are calibrated within a provider, other providers do not follow the same mechanisms. Consequently, noise channels exhibit different scales and characteristics across different providers. Finally, variations in how different providers handle outliers further contribute to performance differences, particularly in adverse weather conditions.

A significant challenge in 3D object detection is domain shift, which occurs when a model's performance degrades on data differing from the training set. This is particularly true for LiDAR data. The sensor-specific characteristics described above, along with variations in sensor configuration and environmental conditions, make LiDAR data especially susceptible to domain shift. For example, a model trained on LiDAR data from an urban setting might perform poorly when applied to data from a highway setting. In an urban environment, objects like pedestrians, cyclists, and vehicles are often close together and moving at relatively low speeds. On a highway, however, objects are spaced farther apart, travel at higher speeds, and include different types of road users, such as trucks and motorcycles. These differences alter the density and distribution of the point cloud, making it difficult for the model to accurately detect objects. Another example is a model trained in clear weather that might fail in fog, where the scattering of laser beams changes the point cloud's structure, demonstrating the practical impact of domain shift.

This thesis systematically investigates the impact of domain shift on 3D object detection, with a particular focus on the effects of LiDAR resolution and the challenges of transferring models trained on synthetic data to real-world scenarios. To address this, we employ a controlled experimental framework using synthetic data generated with the CARLA simulator [1]. This framework leverages two software packages developed as part of this research: `carlaSceneCollector`, for efficient synthetic data generation, and `rosbag2nuscenes`, for conversion into the nuScenes dataset format. These tools allow us to isolate and quantify the impact of specific domain shift factors.

Our findings indicate that LiDAR resolution has a notable effect on detection performance, especially for smaller objects. We also observe saturation effects at intermediate resolutions

(32–64 channels) and suggest strategies to mitigate these effects. Furthermore, we quantify the challenges associated with transferring models trained on synthetic data to real-world scenarios, noting the effectiveness of the NuScenes Detection Score (NDS) [2] in capturing this impact.

The remainder of this thesis is structured as follows: Chapter 2 contextualizes our work within the existing literature, providing a review of related work in 3D object detection and domain adaptation. Chapter 3 outlines the experimental setup, detailing the methodology used for data generation and experimentation. Chapter 4 presents and analyzes the key findings, presenting the results of our experiments. Finally, Chapter 5 summarizes the thesis's contributions and suggests avenues for future research, concluding the thesis and discussing future research directions.

CHAPTER 2

BACKGROUND

The advent of autonomous vehicles necessitates robust and reliable perception systems capable of accurately interpreting the surrounding environment. Among the various perception tasks, 3D object detection from point clouds stands out as a crucial component for ensuring safe navigation and preventing collisions. This capability allows autonomous vehicles to classify and precisely locate objects within their three-dimensional surroundings, forming the bedrock for subsequent tasks like motion planning and decision-making. Consequently, the field of 3D object detection has witnessed a surge of research interest and significant advancements in recent years. However, a persistent challenge that hinders the widespread deployment of these systems is the issue of domain shift.

Domain shift occurs when a model trained on a specific dataset or environment experiences a significant drop in performance when applied to a different dataset or environment. This discrepancy often arises due to variations in data characteristics between the training (source) domain and the operational (target) domain. Understanding and effectively quantifying this domain shift is paramount for developing adaptable and generalizable 3D object detection systems. Furthermore, the scarcity and high cost associated with acquiring labeled data in diverse real-world scenarios underscore the importance of Unsupervised Domain Adaptation (UDA). UDA offers a promising avenue to bridge the performance gap by adapting models trained on abundant labeled data from a source domain to an unlabeled target domain.

This background section provides a comprehensive overview of the current research landscape concerning domain shift quantification in 3D object detection for autonomous driving. It delves into the fundamental concepts, the various types of domain shift encountered, existing quantification methods, and the challenges of applying UDA to 3D point clouds. The section also highlights the pivotal role of datasets like nuScenes, recent advancements in UDA techniques, the impact of different sensor modalities, the distinction between the semantic gap and feature distribution shift, and the importance of the structured nature of the nuScenes dataset in facilitating this research.

2.1 Fundamentals of 3D Object Detection from Point Clouds

The task of 3D object detection from point clouds has seen the development of various methodologies, broadly categorized based on how the unstructured point cloud data is processed and represented. General overview of pointcloud processing approaches are shown in 2.1. Here, "unstructured" means that although a point cloud provides 3D Cartesian coordinates, spatial relationships between points, such as neighborhood or similarity, are not explicitly defined and require distance calculations or methods like KD-trees or octrees to be established.

2.1.1 Point-based Methods

Point-based methods directly operates on raw and unprocessed pointcloud data. Pioneering works of point-based methods, PointNet [3] and a following work PointNet++ [4] employs point-wise operations such as Multi-Layer Perceptrons (MLPs) and symmetric functions such as sum and max pooling for generating geometric set of features for each point. The symmetry ensures that the model is invariant to the order of points in the cloud. PointNet++ [4] extends PointNet and further built upon this foundation by introducing a hierarchical network structure which enables the model to capture local spatial patterns at different scales. Operating on different scales is paramount for pointclouds since range drastically effects the density between point patches. This methodology significantly enhancing PointNet++ ability to comprehend complex scenes by aggregating features from neighboring points. Subsequent research has further refined point-based methods to achieve state-of-the-art performance. PointRCNN [5] incorporated a region proposal network(RPN) to generate candidate 3D bounding boxes directly from the point cloud, which are then refined for final detection. 3DSSD [6] focused on improving efficiency by employing sophisticated sampling strategies to select representative points, reducing the computational burden while maintaining accuracy.

A key advantage of point-based methods lies in their ability to handle the inherent unstructured nature of point cloud data without requiring any intermediate representation. However, a notable drawback is their potential computational intensity, as each point in the cloud often needs to be processed individually.

2.1.2 Voxel-based Methods

In contrast to point-based approaches, voxel-based methods adopt a strategy of discretizing the continuous 3D space into a grid of regular voxels. VoxelNet[7] was among the first to demonstrate the effectiveness of this representation by applying 3D Convolutional Neural Networks (CNNs) to the voxelized point clouds. This allowed for leveraging the power of CNNs, which have proven highly successful in 2D image analysis, for the task of 3D object detection. SECOND[8] further advanced the efficiency of voxel-based methods through the introduction of sparse convolution. Sparse convolution techniques are designed to operate only on the occupied voxels, significantly reducing the computational overhead, especially in scenarios with sparse point clouds common in autonomous driving.

While voxel-based methods benefit from the structured representation that is well-suited for CNNs, they may suffer from information loss due to the inherent discretization process. Voxelization strategies are also important for reliable feature extraction from point clouds with varying densities. These density variations can arise from differences in the range of specific regions or the use of different LiDAR sensors. This loss can be particularly pronounced when dealing with sparse point clouds, where fine-grained details might be smoothed out or lost during voxelization.

2.1.3 Hybrid Methods

Hybrid methods seek to capitalize on the complementary strengths of both point-based and voxel-based approaches. PV-RCNN[9] exemplifies this strategy by employing a voxel-based network to efficiently generate high-quality 3D proposals, which are subsequently refined by a point-based network. This allows the model to exploit the computational efficiency of voxelization for initial proposal generation while retaining the fine-grained geometric information from the raw point cloud data during the refinement stage. Similarly, CenterPoint[10] utilizes a Bird's Eye View (BEV) representation, which is obtained by projecting the 3D point cloud onto a 2D plane. This BEV representation offers a compact and efficient way to detect objects, proving particularly effective for tasks like vehicle detection in autonomous driving scenarios.

By strategically combining different representations and processing techniques, hybrid methods

often achieve superior performance, balancing computational efficiency and representational power. Voxel-based methods are more sensitive to changes in point cloud density, as the voxelization process is affected by the number of points within each voxel. Conversely, point-based methods are more robust to variations in overall point density but more susceptible to noise or outliers. Therefore, selecting the appropriate method and domain adaptation strategy requires understanding the expected domain shifts and the inherent vulnerabilities of each detection approach.

Table 2.1 Comparison of Point Cloud Processing Approaches for 3D Detection

Criteria	Point-based Methods	Voxel-based Methods	Hybrid Methods
Data Representation	Raw, unstructured point cloud data	Discretized into a 3D grid of voxels	Combination of voxelized and raw point cloud data
Feature Extraction	Point-wise operations (e.g., MLPs, symmetric functions like max pooling)	3D Convolutional Neural Networks (CNNs) on voxelized data	Voxel-based for initial proposals, point-based for refinement
Computational Efficiency	Can be computationally intensive due to processing each point individually	More efficient with sparse convolution techniques	Balances efficiency and detail by using voxelization for proposals and points for refinement
Sensitivity to Density Variations	More robust to overall density variations but sensitive to noise and outliers	More sensitive to density changes due to voxelization process	Moderately sensitive, depending on the specific hybrid approach
Handling of Unstructured Data	Directly handles unstructured data without intermediate representations	Requires conversion to structured voxel grid	Uses both structured and unstructured representations
Performance on Sparse Data	May struggle with very sparse data due to lack of local context	Can lose fine details in sparse regions due to discretization	Better at retaining details in sparse regions through point-based refinement
Key Advantages	Handles unstructured data directly; Can capture fine details	Leverages powerful CNNs; Efficient with sparse convolution	Combines efficiency of voxelization with detail preservation of point-based methods
Key Disadvantages	Computationally intensive; May overfit to specific point distributions	Information loss due to discretization; Sensitive to density variations	More complex to implement; May still suffer from some limitations of both methods

2.2 Unsupervised Domain Adaptation for 3D Object Detection

Unsupervised Domain Adaptation (UDA) is a critical area of research that aims to adapt machine learning models trained on a source domain, where abundant labeled data is available, to a target domain, where only unlabeled data exists[11]. This is particularly relevant for 3D object detection in autonomous driving because acquiring labeled 3D point cloud data in diverse real-world environments is often a laborious, time-consuming, and expensive endeavor. Therefore, the ability to effectively transfer knowledge learned from a well-annotated source domain (e.g., a synthetic dataset or data collected in a specific geographical location under favorable conditions) to

an unlabeled target domain (e.g., real-world data from a new city or collected under adverse weather) is of paramount importance for the practical deployment of autonomous vehicles[12][13][14][15].

The success of UDA methods often hinges on the assumption that the underlying feature space between the source and target domains exhibits some degree of similarity [16]. If the fundamental features representing objects and scenes differ drastically, simple distribution alignment might not suffice for effective adaptation. While the primary focus of this section is on UDA, it is worth noting that other forms of domain adaptation exist, including Semi-Supervised Domain Adaptation (SSDA) [17], where a small fraction of target domain samples are labeled, Weakly Supervised Domain Adaptation (WSDA) [18], where only weak labels (e.g., image-level tags) are available in the target domain, and Supervised Domain Adaptation, where labeled data is available in both the source and target domains. It's important to note that many of these methods were initially developed for 2D image data.

2.2.1 Traditional Domain Adaptation

Early UDA methods primarily focused on aligning the feature distributions between the source and target domains. One prominent example is Domain Adversarial Neural Networks (DANN) [19], which employed an adversarial training paradigm. In this approach, a feature extractor is trained to produce features that are not only discriminative for the main task (e.g., object classification) but also indistinguishable with respect to the domain they originate from (source or target). A domain classifier is simultaneously trained to distinguish between source and target domain features, and the gradients from this domain classifier are reversed when updating the feature extractor. This forces the feature extractor to learn domain-invariant features that can confuse the domain classifier. However, these traditional methods were primarily designed for 2D image data and often do not effectively capture the unique characteristics and challenges associated with 3D point cloud data, such as its sparsity, irregularity, and lack of inherent order.

2.2.2 Adaptation for 3D Object Detection

Recent advancements in Unsupervised Domain Adaptation (UDA) for 3D object detection have shifted from basic feature alignment to sophisticated techniques that generate robust pseudo-

labels for unlabeled target domains. These approaches leverage temporal, spatial, and synthetic data to bridge domain disparities, such as variations in LiDAR resolution or differences between synthetic and real-world environments. A prominent strategy involves self-training, where a detector pretrained on a labeled source domain produces bounding box predictions for the target domain. These predictions are refined and filtered into pseudo-labels, iteratively retraining the model to enhance its adaptability.

2.2.2.1 Tracking-Based Methods

A notable group of UDA techniques utilizes multi-object tracking (MOT) to exploit motion consistency across frames, improving pseudo-label reliability. MS3D++ [20] exemplifies this approach by combining outputs from an ensemble of pretrained detectors—each trained on distinct source datasets with varying architectures—using a kernel density estimation (KDE) algorithm. The use of ensemble of models that have different network architectures and source domain is for reducing common pitfalls among the detection sets. Examples for common pitfalls could be detecting a false-positive object in the absence of a pointcloud path because of adverse weather or detecting a small sized pedestrian around the specific traffic signs which can differ on regions. These fused detections initialize a 3D MOT tracker, built on SimpleTrack [21], yielding consistent pseudo-labels derived from trajectories, classification scores, and motion cues, with iterative refinement until performance converges. MS3D++ enhances precision through temporal tactics: retroactive object labeling propagates dependable labels from later frames to correct earlier, ambiguous detections impacted by sparse points or occlusions, while static vehicle refinement ensures uniform bounding boxes for stationary objects, improving shape accuracy and detection coherence.

In contrast, CTRL [18] employs a track-centric backtracking technique, atypical for real-time applications. Following an initial forward pass, it revisits earlier frames to recover missed detections, enhancing track continuity and label completeness through bidirectional sequence refinement. Other methods focuses on quantifying the track reliability, SF-UDA 3D [22] employs sophisticated track equations for score labeling to reduce false-positive and enforce temporal consistency throughout the timeline, ST3D++ [23] employs novel voting mechanism powered by hybrid quality-aware

triplet memory (HQTM) to make sure tracklets can explained by their detections consistently.

2.2.2.2 Recent Advancements in UDA Techniques

Beyond tracking-based methods, recent UDA innovations focus on shape preservation, clustering, and extended sequence processing. Auto4D [24] preserves rigid object shapes by collecting point clouds in the object's reference frame, mitigating distortions from shifting centers. A convolutional neural network (CNN) derives shape estimates from these dense clouds, polished via closest-corner alignment. For static objects, aggregating points in world coordinates—enabled by precise ego-vehicle localization—refines size estimates, minimizing noise from erroneous detections. Tracking is supported by AB3DMOT [25].

Once Detected, Never Lost [26] adapts the Fully Sparse Detector (FSD) for offline analysis by incorporating both past and future frames. It uses bidirectional MOT: a forward pass constructs tracklets, followed by a backward pass that retrieves overlooked detections prior to tracklet initiation. A specialized module, integrating UNet for sparse feature extraction and PointNet for bounding box refinement, enhances proposals, with multi-way registration ensuring track consistency as a final step.

An unsupervised method in [27] applies augmentations like ray dropping to bolster generalization, particularly for distant objects. It employs L-shape fitting for box estimation and clustering to detect objects without labels, providing a straightforward response to domain shifts, though it omits temporal refinement.

Offboard 3D Object Detection from Point Cloud Sequences [28] enhances detectors like PointRCNN for multi-frame analysis, compensating for vehicle motion. Using AB3DMOT [25] for tracking, it aggregates point clouds for static objects to form comprehensive shape priors and aligns trajectories for dynamic objects, refining accuracy with lightweight PointNet-based regression networks across sequences.

DetZero [29] integrates an offline tracker with a multi-frame detector to ensure trajectory integrity. An attention-based module sharpens contextual details across extended point cloud sequences, addressing incomplete trajectories and diverse motion states. Decomposed regression

further hones detections, delivering outstanding performance on the Waymo Open Dataset (85.15 mAPH, L2).

2.3 The nuScenes Dataset Format

The nuScenes dataset[2] uses a structured relational database format to organize its sensor data, annotations, and metadata. It is composed of multiple interlinked tables that describe different aspects of the dataset. For example, the `category` table defines a hierarchical taxonomy of object classes (e.g., a top-level class "vehicle" with sub-classes like "vehicle.car" or "vehicle.truck"), and the `attribute` table specifies mutable properties of objects (for instance, whether a vehicle is parked or moving, or whether a bicycle has a rider). The `sensor` table enumerates all sensors employed (such as the LiDAR and each camera), while the `calibrated_sensor` table provides each sensor's calibration parameters (intrinsic settings and extrinsic pose relative to the vehicle), ensuring that data from different sensors can be accurately aligned in a common reference frame. Additionally, the `visibility` table offers a measure of how well an object is observed in the camera views, binned into ranges (e.g., 0–40%, 40–80%, etc.), which gives annotators' assessment of partial occlusions. The `map` table stores environmental context in the form of precomputed semantic maps (such as drivable area masks) associated with each location or log in the dataset. Several tables capture the dynamic, time-indexed elements of nuScenes. The `log` table contains metadata for each recording session (each "log" corresponds to a route driven by the data collection vehicle, with information such as the location, date, and the vehicle used). Each log is subdivided into scenes, and the `scene` table defines these distinct 20-second sequences (each scene is a continuous clip within a log). The `sample` table represents the key frames sampled at 2,Hz in each scene; each sample acts as a synchronized snapshot containing one LiDAR sweep and the set of camera images closest in time, along with all associated annotations. For each sample, the actual recorded sensor readings are listed in the `sample_data` table: for example, a LiDAR point cloud file and several camera image files would be separate entries in `sample_data`, each linked to a specific sensor and accompanied by the relevant calibration and the vehicle pose. The vehicle's pose (position and orientation) at any timestamp is recorded in the `ego_pose` table, which gives the location of the

"ego" vehicle in a global coordinate frame for each sensor reading or sample. The annotations for objects are stored in the `sample_annotation` table, which contains the 3D bounding boxes for all objects present in each sample (key frame), along with pointers linking each box to a particular object instance and the object's category and attributes. nuScenes tracks individual object instances within a scene using the `instance` table, which lists unique instance identifiers for objects (each physical object, such as a specific car, gets an `instance` ID within a scene). It should be noted that instances are not tracked across different scenes; if the same physical car appears in two separate scenes, it will be treated as two distinct instances in the dataset. Together, these tables provide a comprehensive and well-organized structure for the nuScenes data, enabling efficient lookup of sensor information and annotations needed for training and evaluating 3D object detection models.

CHAPTER 3

METHODOLOGY

In this thesis, we address the challenge of Unsupervised Domain Adaptation (UDA) within the realm of 3D Object Detection. Our approach begins by systematically identifying and quantifying potential sources of domain shift, leveraging a carefully curated suite of both tailored and generic datasets. We also present a novel pipeline for generating domain-specific datasets using the CARLA simulator, designed to capture and analyze domain shift characteristics. Subsequently, we train our models on this dataset suite and perform cross-dataset evaluations to uncover key axes of domain shift. These findings underscore the necessity of an auto-labeling pipeline to effectively mitigate UDA challenges. In the following sections, we detail a foundational auto-training pipeline, critique its limitations, and propose targeted enhancements, including an innovative re-detection mechanism driven by tracking priors.

3.1 Sources of Domain Shift

Feature sets extracted from source datasets encapsulate the distinct intrinsic properties inherent to each dataset[12]. Within the domain of 3D object detection, these properties may originate from variations in environmental conditions, scene composition, LiDAR sensor specifications (including type, placement, and channel count), or the fidelity of sensor data[20], particularly when datasets are synthetically generated. Depiction of potential domain shifts sources related to the environment conditions and scene composition between real life datasets shown in 3.1. This study systematically categorizes the recurring patterns that define these intrinsic attributes, establishing a comprehensive framework for analyzing domain shift in 3D object detection.

In the UDA framework, we designate the source dataset, denoted S_i, as the fully annotated dataset employed for initial model training, and the target dataset, S_j, as the unlabeled dataset targeted for adaptation, where domain discrepancies must be minimized. The transition $S_i \rightarrow S_j$ represents the process of training a model on S_i and evaluating its performance on S_j.

To quantify domain shift, we first identify and define potential sources of domain shift. Let R represent the set of domain shift sources, where $R = \{R_{LR}, R_{SR}, R_{SC}, R_{LP}, R_{LC}\}$. Here, R_{LR}

denotes variations in LiDAR resolution (e.g., 16Ch vs. 32Ch), R_{SR} indicates the use of synthetic versus real data in source or target datasets, R_{SC} refers to scenery disparities (e.g., urban vs. highway settings), R_{LP} signifies differences in LiDAR sensor placement, and R_{LC} reflects variations in the number of LiDAR sensors between source and target datasets.

Let D denote any performance metric commonly utilized in 3D object detection for autonomous driving, such as mean Average Precision (mAP), NuScenes Detection Score (NDS), or class-specific Average Precision at a fixed threshold (e.g., $CAR\_AP_{0.5}$). The value $D^{S_i \to S_j}$ represents the performance of a model trained on S_i and tested on S_j. In the context of UDA, the domain shift between datasets S_i and S_j is quantified as the difference between the baseline performance, $D^{S_i \to S_i}$ (when trained and tested on the source), and the adapted performance, $D^{S_i \to S_j}$ (when tested on the target), expressed as:

$$\Delta D^{S_i \to S_j} = D^{S_i \to S_j} - D^{S_i \to S_i}$$

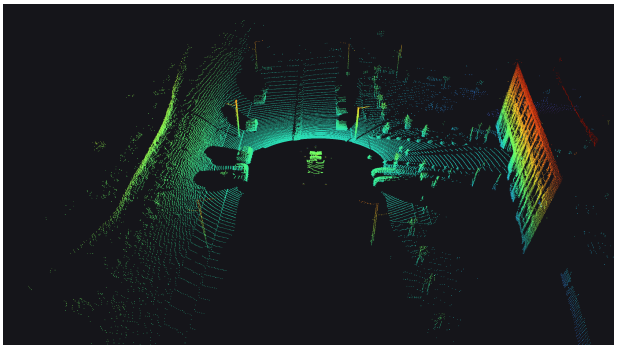
This difference captures the domain shift between source and target datasets as the amount of variation in the chosen metric, whether positive (indicating improvement) or negative (indicating performance degradation).

This $\Delta D^{S_i \to S_j}$ quantifies the aggregate domain shift projected onto metric D, encompassing contributions from all potential sources in R. Recognizing $\Delta D^{S_i \to S_j}$ as a composite variable influenced by multiple factors, we decompose it into contributions from individual sources:

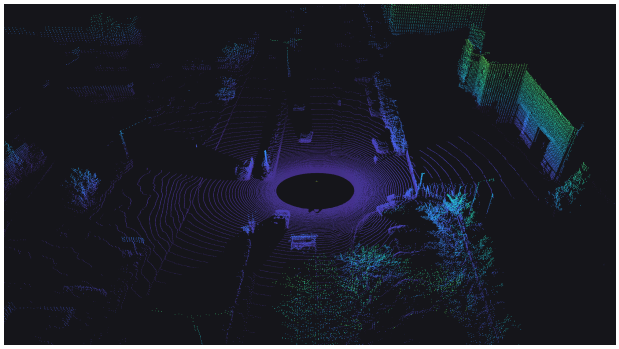
$$\Delta D^{S_i \to S_j} = \sum_{k \in R} \Delta D_{R_k}^{S_i \to S_j}$$

where $\Delta D_{R_k}^{S_i \to S_j}$ represents the domain shift attributed to source R_k between datasets S_i and S_j.

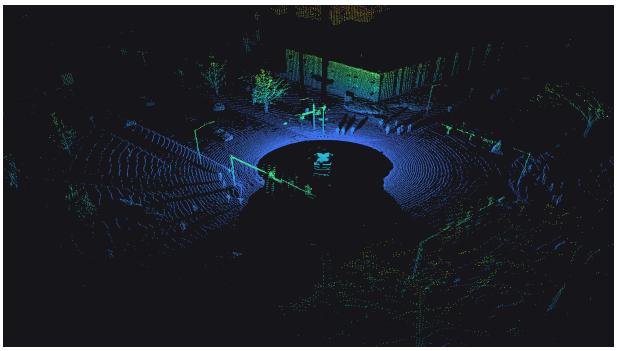
To isolate and quantify the impact of each R_k, this work employs a strategy of meticulously tailoring datasets such that only a single domain shift source varies between S_i and S_j, while other sources remain controlled. For instance, to assess R_{LR}, we generate datasets differing solely in LiDAR resolution, holding factors like scenery and sensor count constant. This controlled approach enables precise measurement of $\Delta D_{R_k}^{S_i \to S_j}$ for each source, facilitating a detailed understanding of their individual contributions to the total domain shift.



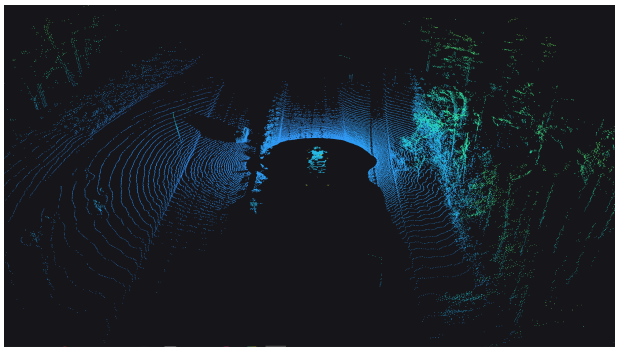
(a) Industrial Site



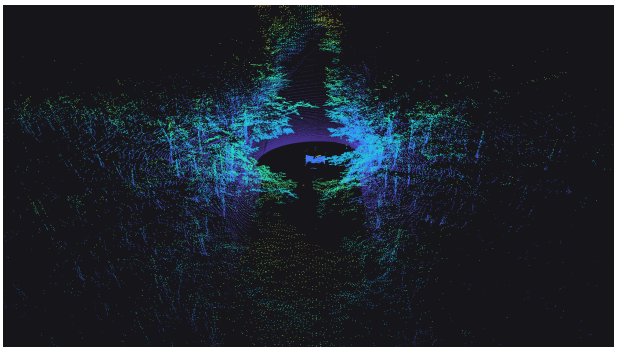
(b) Passenger Car



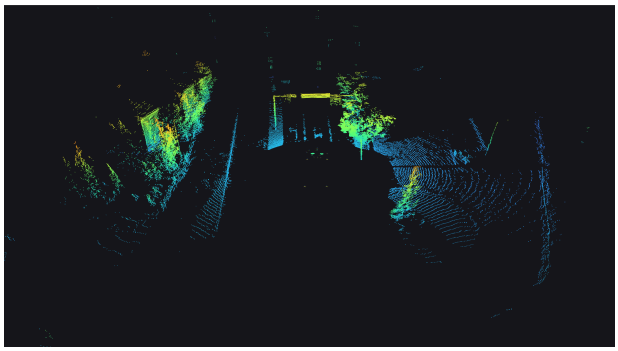
(c) University



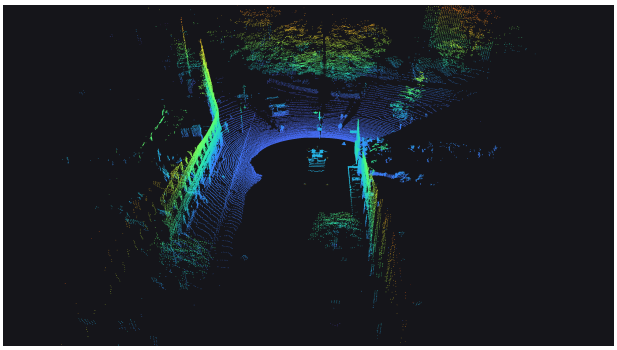
(d) Highway



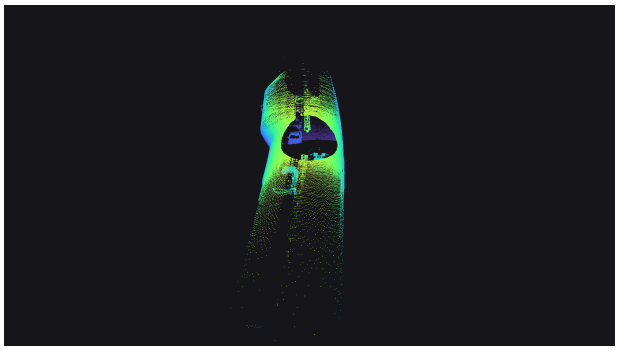
(e) Forest



(f) Rainy Weather



(g) Urban



(h) Tunnel

Figure 3.1 Various driving scenarios that contributes to domain shift between real-life datasets

3.2 Data Collection

To isolate the impact of a single domain shift source, R_k, it is imperative to create tailored datasets where extraneous domain shift sources do not contribute to the overall domain shift effect. The design and generation of such datasets hinge on several key considerations and requirements:

1. The datasets must be tailored to the autonomous driving domain, incorporating sensor configurations relevant to this context.

2. They should be straightforward to generate and distribute efficiently.

3. They must be compatible with prevalent 3D object detection frameworks to facilitate seamless training and evaluation.

4. Ground truth annotations for 3D object detection must be provided to ensure reliable assessment.

After careful evaluation, the nuScenes dataset format was selected as it satisfies all these criteria. The nuScenes format is widely adopted in the autonomous driving research community, offering a standardized structure that supports diverse sensor data and comprehensive ground truth annotations, thereby aligning with the needs of this study. However, a challenge remains: generating the requisite raw data to populate this format. To address this, we opted for the CARLA simulator as the primary data generation source. CARLA was chosen due to its extensive community support, rich ecosystem of libraries, and flexibility in simulating a wide range of autonomous driving scenarios, making it an ideal tool for producing controlled, high-fidelity sensor data.

While CARLA effectively generates raw data in this work, real-world applications might draw data from diverse sources, such as physical sensor deployments or other simulators. To address this heterogeneity and ensure versatility across projects, we advocate for a generalized approach to raw data storage. We adopt the ROSBag format, a widely recognized standard in robotics and autonomous systems. ROSBag supports the storage of raw sensor data, including LiDAR point clouds, camera images, and vehicle pose estimates, alongside metadata like 3D rigid body

transformations and camera intrinsic calibration data (e.g., focal length, distortion coefficients). Compatible with both real-world and simulated data from CARLA, ROSBag offers a flexible, interoperable solution that maintains spatial and temporal relationships critical for 3D object detection and localization. To convert this data into the nuScenes format, we introduce two new packages, `carlaSceneCollector` and `rosbag2nuScenes` package, a set of modules that processes and transforms ROSBag data, including sensor streams, transformations, and localization, to create tailored datasets for 3D object detection experiments.

3.3 Details on the carlaSceneCollector Package

The CARLA simulator, designed specifically for the autonomous driving domain, benefits from active maintenance and a robust community of contributors and users, ensuring its reliability for research purposes. Built on Unreal Engine [30], a high-fidelity game engine widely utilized across industries, CARLA provides powerful APIs that enable users to interact with its physics-based environment seamlessly. CARLA supports an extensive array of road agents, configurable as either the ego vehicle or other road users, and includes an autonomous traffic management system [1]. This system simplifies the automation of both the ego vehicle and the surrounding traffic, enhancing scenario realism. Additionally, CARLA offers a bridge module for integration with the ROS middleware[31], facilitating data exchange. Spawning agents is straightforward, as CARLA provides predefined safe spawning points to ensure reliable agent placement. The `carlaSceneCollector` package, developed as part of this work, leverages these capabilities by accepting a configuration file that defines the data collection schema. This file specifies parameters such as the target ego vehicle, map selection, sensor setup, asset choice for the ego vehicle, and the number of scenes to collect, where each scene comprises 20 seconds of data formatted according to the nuScenes standard.

The `carlaSceneCollector` package integrates a suite of modules to orchestrate data generation and collection from a running CARLA instance. The `setAutopilot` module enables or disables autopilot mode for the ego vehicle, ensuring a safe reset of any residual velocity or acceler-

[1] CARLA Traffic Manager https://carla.readthedocs.io/en/latest/tuto_G_traffic_manager

ation inputs. The `generateTraffic` module populates the scene with a user-specified number of pedestrians, bicycles, motorcycles, buses, cars, and trucks. The `removeAllActors` module clears all non-ego actors from the scene, allowing a fresh start for each scenario without carryover from prior configurations. The `setEgoVehicleRandomPose` module queries the map for safe spawning locations and randomly repositions the ego vehicle to one of these points. The `collector` module functions as a ROSBag recorder, capturing all sensor data—including LiDAR point clouds, camera images, and localization ground truth—along with frame transformations such as sensor calibration and pose information. A runner script within `carlaSceneCollector` coordinates these modules to execute the pipeline, achieving the desired number of scenes efficiently. A depiction of the `carlaSceneCollector` package's pipeline is shown in Figure 3.2

3.4 Details on rosbag2nuScenes package

After collecting set of ROSBags, `rosbag2nuscenes` package is responsible for converting the raw data to nuScenes format. Package consists of many individual components and also components that are interactively work with each other in order to distribute single understanding of the entire dataset. In this chapter, we plan first to explain configuration step of the pipeline.Subsequently, we elaborate on the generation of metadata tables that establish the dataset's temporal structure, namely `log`, `scene`, and `sample`. We then explore the data-related tables, encompassing `sensor`, `sample_data`, `calibrated_sensor`, and additional tables such as `category`, `ego_pose`, `instance`, `map` and `sample_annotation`, which collectively define the nuScenes dataset format.

3.4.1 Configuration

The `rosbag2nuscenes` package maintains a global parameter set to configure a single dataset conversion session, designed to process any ROSBag data, not solely those collected from CARLA. The parameter `rosbag_paths` specifies a set of ROSBag file paths to be considered for dataset generation. The `annotation_type` parameter defines the ROS message type for the annotation topic, which provides ground truth data including bounding boxes, velocities, and object classifications. To enhance compatibility, we developed conversion functions that transform various 3D object de-

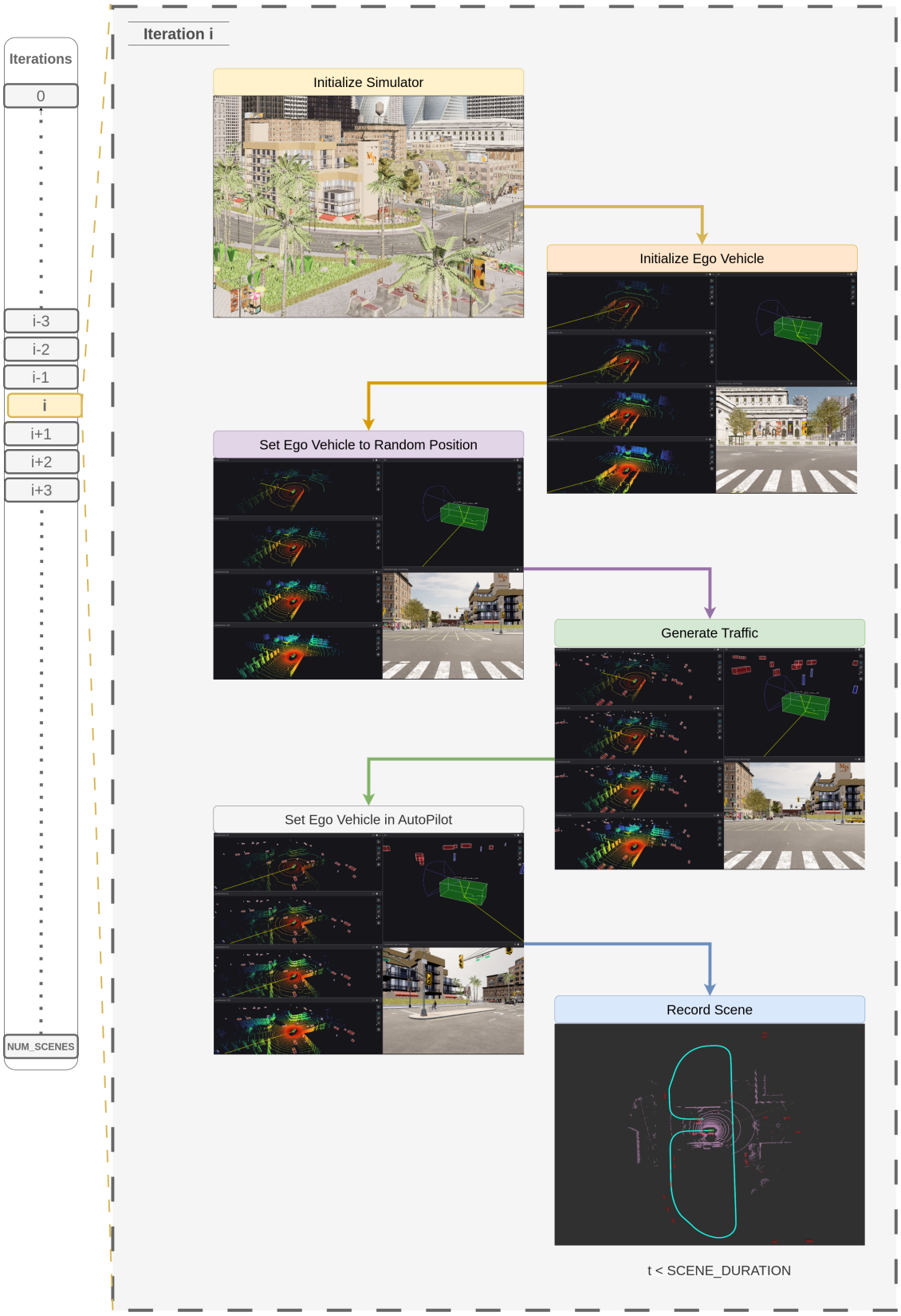


Figure 3.2 Depiction of carlaSceneCollector packages pipeline

tection message types commonly used in the community into a unified `derived_object_array` format. This format is widely adopted and straightforward for ROS developers to utilize.

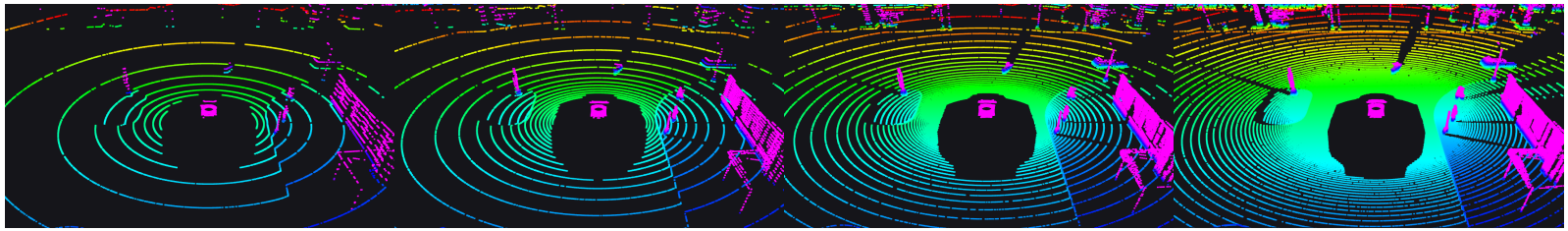
The parameters `global_frame_id` and `ego_frame_id` designate the frame names for the global reference frame (to which localization messages refer) and the ego vehicle frame, respectively. The ego frame is defined as the ground projection of the midpoint between the two rear wheels of the ego vehicle. To prevent unintended domain shifts, we ensured that the ego vehicle frame remains consistently positioned relative to the vehicle body across all datasets in this study. This consistency, for instance, maintains ground plane points at a uniform z-coordinate regardless of sensor configuration. The `sensors_of_interest` parameter identifies the set of sensors that `rosbag2nuscenes` processes.

Additional sensor-specific parameters include `modality`, `topic_name`, `is_anchor`, and an optional `sensor_info_topic`. The `is_anchor` boolean indicates whether a sensor's timestamps serve as the reference for defining a sample. When a sensor is designated as an anchor, `rosbag2nuscenes` synchronizes all other sensor data to its timestamps, discarding any sample where a match cannot be found. Furthermore, we define `sample_duration` as the maximum allowable time difference between a message and its nearest anchor timestamp for inclusion in a sample, and `scene_duration` as the total duration of a scene, set to 20 seconds in accordance with the nuScenes standard.

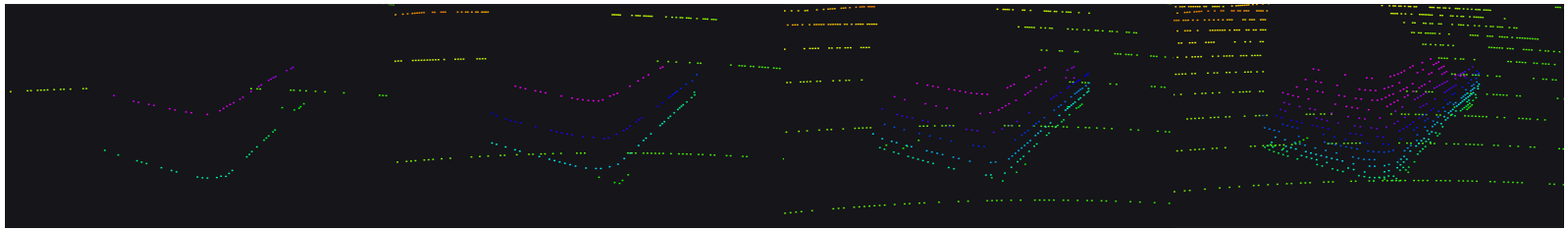
The `rosbag2nuscenes` package incorporates several post-processing modules to refine samples, annotations, and point clouds. The `annotation_filters` module includes a collection of filters tailored for annotation data, while `sample_filters` targets the generation of sample data, and `pointcloud_filters` focuses on processing point cloud data. These filters are applied sequentially according to user-defined specifications within the `rosbag2nuscenes` pipeline.

3.4.2 Pipeline

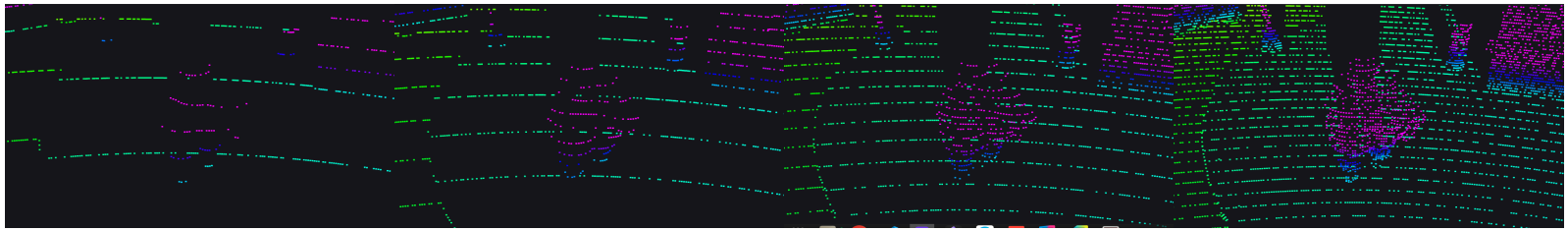
The pipeline developed in this work is organized into three distinct class categories to facilitate nuScenes dataset generation. The first category encompasses classes that manage data storage and mapping for the nuScenes format, the second includes classes that facilitate coordination among



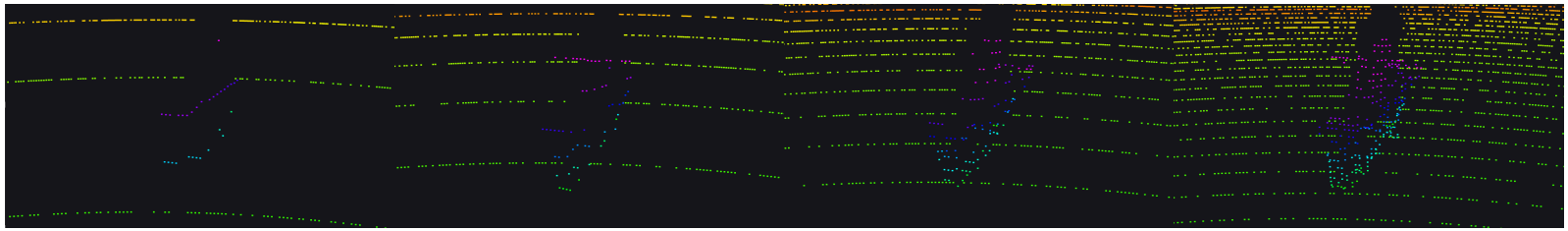
(a) General — 16 ch (b) General — 32 ch (c) General — 64 ch (d) General — 128 ch



(e) Cars — 16 ch (f) Car — 32 ch (g) Car — 64 ch (h) Car — 128 ch



(i) Pedestrian — 16 ch (j) Pedestrian — 32 ch (k) Pedestrian — 64 ch (l) Pedestrian — 128 ch



(m) Bicycle — 16 ch (n) Bicycle — 32 ch (o) Bicycle — 64 ch (p) Bicycle — 128 ch

Figure 3.3 Comparison of different object classes across LiDAR channel configurations in CARLA simulator

other classes, and the third comprises utility classes that assist in various pipeline operations. For example, the `ContextManager` class, part of the second category, oversees the execution flow and relays critical data between classes to support subsequent tasks. At initialization, `ContextManager` parses the `sensors_of_interest` parameter to create `Sensor` objects for each designated sensor. The `Sensor` class, an instance of the first category, holds sensor-specific attributes (token, modality, and channel) and directly aligns them with the nuScenes format without complex processing. Subsequently, `ContextManager` instructs the storage of this data to disk, generating the `sensor.json` file. This sensor information, retained in memory by `ContextManager`, is shared with later stages, such as the creation of the `calibrated_sensor` and `sample_data` tables, ensuring cohesive dataset assembly. The initial sensor set definition is crucial, as it remains constant across ROSBags and establishes the dataset's sensor framework. In scenarios requiring a heterogeneous sensor configuration across scenes, `rosbag2nuscenes` must receive the superset of sensors during the configuration phase. While autonomous driving datasets typically feature homogeneous sensor setups, `rosbag2nuscenes` is fully equipped to handle heterogeneous configurations when necessary.

After creating the sensor set, `ContextManager` creates `Log` object for each of the ROSBag files. `Log` class is one of the most complex classes of the `rosbag2nuscenes` package since it holds the most generic and interconnected information for the nuScenes dataset. A depiction of the `rosbag2nuscenes` package's pipeline is shown in Figure 3.4

3.4.2.1 Creation of Logs

The `Log` class handles a ROSBag file by utilizing its file path and a `scene_start_index` parameter, an offset that differentiates scenes across various logs to ensure unique `name` fields in the nuScenes format, calculated and supplied by the `ContextManager` to each `Log` instance. It segments the ROSBag into uniform portions according to the `scene_duration` parameter, keeping any remaining data if the total length isn't perfectly divisible, thereby retaining all data rather than omitting leftovers, despite scenes typically lasting 20 seconds for standardization. Next, it generates `EgoPose` objects for each piece of localization data in the ROSBag, which

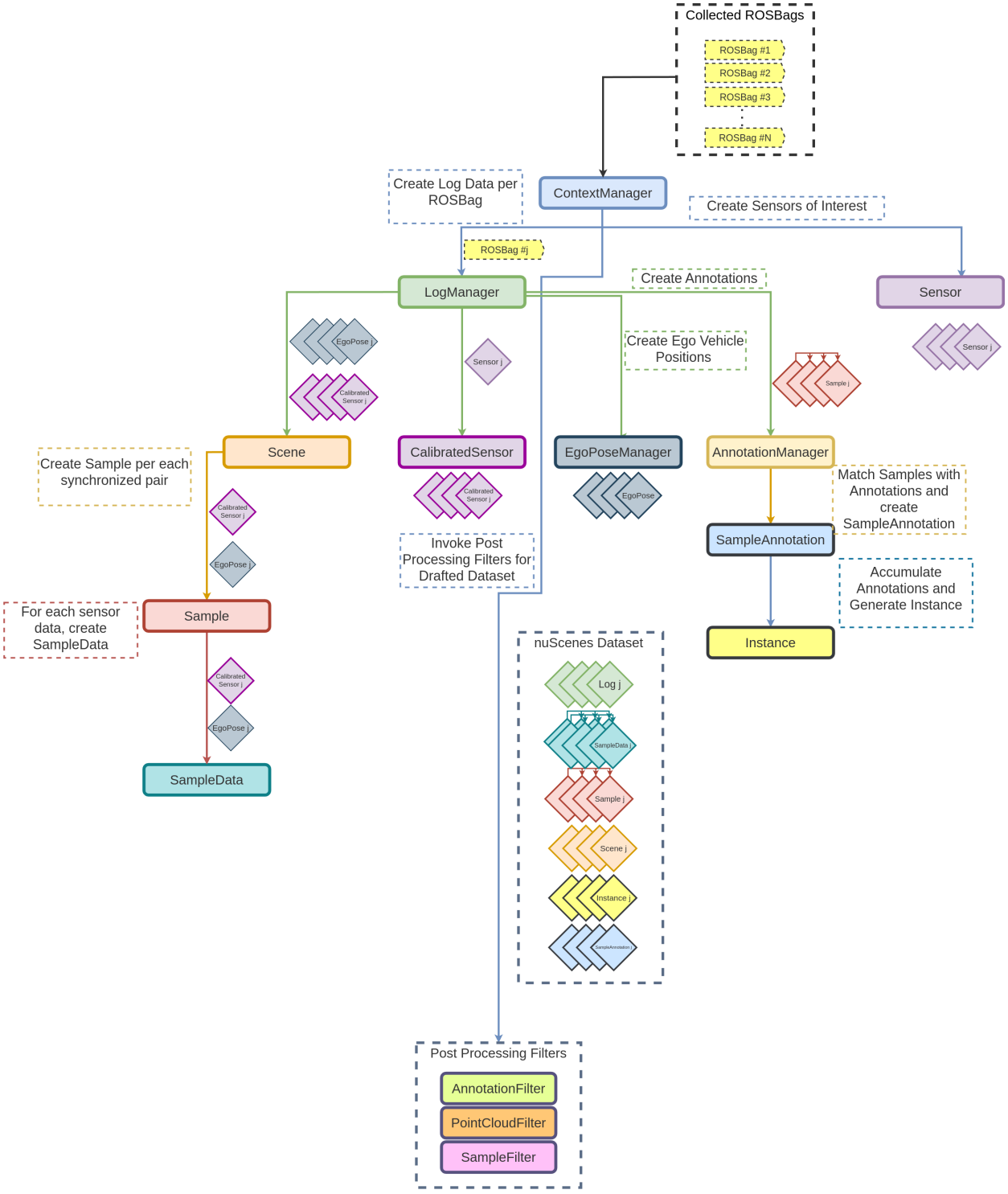


Figure 3.4 Depiction of rosbag2nuscenes packages pipeline

supports Scene objects in producing SampleData objects representing raw sensor outputs. The EgoPose class contains the necessary details to populate the ego_pose table in the nuScenes format, capturing the ego vehicle's position relative to a global frame. Following this, the Log class creates CalibratedSensor objects that define a sensor's specific state, including intrinsic and extrinsic details, corresponding to the calibrated_sensor table in nuScenes.

In the nuScenes format, log data represents a continuous data collection session within a global timeline, encompassing a single interval of recorded activity. While the log spans the entire session, scenes represent smaller portions within it, meaning a log consists of multiple scenes, so the Log class manages all ego positions and calibrated sensor information, distributing these details to individual Scene objects created for each segment, along with the ROSBag object and its specific start and end times.

The Log class also initiates the AnnotationManager object, which oversees the generation of the instance and sample_annotation tables that store ground truth object information. Using the pre-existing list of Scene objects, the Log class activates the AnnotationManager with all sample data from the current log to connect frame-specific objects in the sample_annotation table to the comprehensive timeline of road agents in the instance table.

3.4.2.2 Scene, Sample and SampleData

In the nuScenes format, the scene table describes a 20-second portion of a data collection session, tied to a specific log entry. Each scene identifies a start and end sample, where a sample captures a single frame in the scene's timeline and connects to synchronized sensor data stored in sample_data. For easy navigation, sample includes links to the previous and next samples within the scene. The sample_data table ties this sensor data to ego_pose and calibrated_sensor entries for accurate positioning and calibration. It also contains timestamp (when the data was captured), filename, and fileformat (indicating the sensor type and data location), with timestamps that may vary across a sample's sensors. Additionally, sample_data has an is_key_frame flag to show if a frame is labeled. While nuScenes collects data at 20 Hz, only 2 Hz frames are labeled, leaving most with a false is_key_frame value. These unlabeled frames remain

useful for multi-frame detection models like CenterPoint, which we explore in later experiments.

To create these tables, the `Scene` class gathers all sensor messages from the ROSBag between the given start and end times. It then picks out `anchor` topic messages based on the `is_anchor` setting in the sensor setup, typically the highest-resolution LiDAR in multi-LiDAR cases, as it's key for localization when using one LiDAR. Samples are formed by setting a time window around each `anchor` message using `sample_duration`, matching messages from different sensors listed in `sensors_of_interest` if their timestamps fit within this window. For 20 Hz data, `sample_duration` is about 0.05 seconds; for 10 Hz, it's around 0.1 seconds. We opted for 20 Hz data collection in CARLA to match the nuScenes standard, maintaining timestamp intervals of 0.05 seconds, since we observed that differing frequencies introduce unintended domain shifts in multi-frame detectors like CenterPoint, which rely on multi-frame data paired with relative timestamp differences calculated from the earliest point cloud's timestamp. Adjusting these time gaps isn't helpful since CenterPoint also predicts per-object velocity, tied to point feature shifts over specific time intervals, affecting performance consistency across datasets.

During the pairing of sensor data with the `anchor` topic, the `Scene` class creates `Sample` and `SampleData` objects, linking their `next` and `prev` fields in a two-way queue structure. The `Scene` class also attaches each `SampleData` object to its corresponding `ego_pose` entry from the `EgoPose` object during this process, while its dual-queue and reference-based design ensures clear connections between `Sample` and `SampleData` objects, making it a key module that manages both these relationships and its own environmental data for the nuScenes format.

After generating `Sample` objects, each `Sample` creates individual `SampleData` objects for every piece of synchronized sensor data, linking each one to the corresponding entry in the `calibrated_sensor` table by utilizing the `CalibratedSensor` object to ensure proper calibration details are attached. Unlike other components in the `rosbag2nuscenes` package, the `SampleData` class is unique because it directly writes the sensor data to disk instead of holding it in memory, a choice driven by the `num_features` configuration parameter that determines the target dimensions of the point cloud data to be saved, such as deciding whether to include all five fields—x, y, z, intensity,

and timestamp—or just a subset like the first three if we exclude intensity. We require this parameter to be specified because 3D detection frameworks, like mmdetection3d, depend on consistent point cloud parsing rules for both training and evaluation, and mismatched dimensions can disrupt these processes. For example, if we don't need the `intensity` field, we can set `num_features` to use only x, y, and z, tailoring the data to our needs, but this flexibility demands that all input point clouds share the same field structure across the dataset. To achieve this uniformity, the `SampleData` class first reads the incoming point cloud data, then adjusts it by either adding padding or trimming columns as necessary to match the specified `num_features`, ensuring every saved point cloud has the same format. In our work, we set `num_features` to 5, covering the full set of {x, y, z, intensity, timestamp}, since this is a widely used configuration for multi-frame detection models like CenterPoint, which we explore later, and saving directly to disk after processing helps manage memory efficiently by avoiding the need to retain the already-processed data in memory.

3.4.2.3 AnnotationManager

The `AnnotationManager` class is tasked with managing the ground truth data, ensuring it is properly structured and stored within the `sample_annotation` and `instance` tables of the nuScenes dataset format. It begins this process by collecting all messages published to the topic specified by the `annotation_type` parameter, which defines the type of ROS message carrying annotation information, and then converts these messages into a standardized `derived_object_array` format for consistency across the pipeline. Following this conversion, the class carefully processes each object message by extracting and storing its `id`—a unique identifier assigned to each frame-specific object that corresponds to a specific CARLA agent—into a comprehensive list; this list serves as the foundation for generating `Instance` objects, where each `Instance` object represents a distinct agent in the dataset, while the `sample_annotation` messages provide snapshots of that agent's state at particular points in time. In the nuScenes framework, every `sample_annotation` must be associated with a specific `sample` entry, which represents a single frame in the timeline, so the `AnnotationManager` systematically works through the full list of instances, gathering all annotations tied to each instance and then pairing these annotations with

the appropriate `sample` entries; it does this by applying a time window defined by the `sample_duration` parameter, matching annotations to samples if their timestamps fall within this window, a method akin to how samples are initially created from sensor data. This pairing approach, although thorough, demands significant computational effort because it requires iterating over the entire set of annotations for each instance and then aligning them with every `sample` entry in the dataset, leading to a time complexity of $O(n^2)$, where n denotes the total number of annotations or samples, making it one of the more resource-intensive operations in the pipeline.

3.4.2.4 Post-processing Filters

The `rosbag2nuscenes` package incorporates three distinct post-processing modules to enhance the dataset's quality after it has been saved to disk —`AnnotationFilter`, `SampleFilter`, and `PointCloudFilter`. These modules are configured using a global parameter list, consistent with the setup of other components within this package suite, and they operate on the dataset by leveraging the tools and context provided by the nuScenes-devkit, allowing for additional refinement of the data entities stored on disk.

The `AnnotationFilter` module is composed of four specialized submodules designed for filtering: `BoxElevationShiftFilter`, `RangeFilter`, `AnnotationRelationCorrector` and `PointsFilter`. The `BoxElevationShiftFilter` adjusts the height of ground truth bounding boxes for specific object classes when necessary, addressing a quirk in CARLA where a bounding box, defined as $bbox = \{x, y, z, l, w, h, o\}$—with x, y, z as the center coordinates, l, w, h as the length, width, and height, and o as the yaw orientation—positions z at ground level rather than the box's center; to align with nuScenes' center-based standard, it adds $h/2$ to the z-coordinate, ensuring the $bbox_{xyz}$ accurately reflects the box's midpoint. The `RangeFilter` is a straightforward tool that takes `min_range` and `max_range` values along with a `channel` input (defaulting to LiDAR, though it must match a `sensor` entry name) to exclude annotations falling outside these specified distance boundaries, helping to focus on relevant objects within a sensor's effective range. The `PointsFilter` removes annotations that lack sufficient points within their bounding box, determined by a `min_points` threshold, and it supports multiple LiDAR inputs via `channel_`-

list, counting the total points inside the box across all listed sensors to ensure meaningful data density. The AnnotationRelationCorrector is a more intricate submodule that addresses the ripple effects of prior filters deleting sample_annotation entries; in nuScenes, each instance entry points to a starting and ending sample_annotation, so if one is removed, the entire instance can be lost, and consecutive sample_annotation entries rely on next and prev pointers for easy traversal, which can break when entries disappear; this filter meticulously scans the full sample_annotation table to mend these gaps, either by finding the next valid annotation or adjusting pointers if no further entries exist, marking the current one as the last if needed.

This correction process is both critical and time-intensive because CARLA's object messages, which form the initial basis for our ground truth data, list all agents in a scene—whether they're visible or not—without checking if sensors can detect them due to occlusions or being out of range, requiring us to refine the dataset post-collection. Additionally, the AnnotationRelationCorrector evaluates the velocities attached to sample_annotation entries, which the nuScenes-devkit calculates by interpolating the three nearest states of an instance to estimate movement; when an instance has too few annotations, this can lead to unrealistic velocity values that don't make sense geometrically, and in such cases, the filter removes the entire instance from the dataset to preserve accuracy and reliability, ensuring the ground truth reflects observable and feasible object behavior.

SampleFilter module includes a single submodule, UnsyncedSamplesFilter, which examines all sample entries to detect those lacking annotations while their previous and next samples both contain related annotation data. This filter is essential because, in rare instances, all sample_annotation entries tied to a sample might be removed—often due to CARLA occasionally repeating an object message for the same timestamp, which the AnnotationManager then discards as duplicates—leaving an empty sample that doesn't reflect meaningful information and could indicate an outlier rather than valuable data; the SampleFilter identifies such cases, reconnects the previous and next samples by updating their pointers, and removes the empty sample from the dataset to maintain its integrity.

PointCloudFilter module features a single submodule called SelfCropBoxFilter, which

uses `min` and `max` vectors to define a bounding box ($bbox$) that outlines the ego vehicle's boundaries, along with a `channel_list` parameter to specify which LiDAR sensor or sensors' data should be processed, and it removes any points falling within this defined $bbox$. This filtering step is necessary because we found that when a model trained on a dataset without visible ego vehicle parts in the sensor data is tested on a dataset where the ego vehicle is detectable by LiDAR sensors, it often generates persistent false-positive detections around the ego vehicle's location; this unwanted behavior introduces a bias in performance metrics, skewing results in a way we aim to prevent by ensuring the point cloud data reflects only the external environment and not the vehicle itself.

3.5 Quantification of Domain Shift

To explore domain shift, we carefully selected a subset of potential sources—specifically R_{LR} (LiDAR resolution) and R_{SR} (synthetic versus real data)—and designed our datasets to isolate their effects. When building datasets to examine R_{LR} (LiDAR resolution), we equipped the ego vehicle with a sensor setup that includes one RGB camera and four LiDAR sensors, all fixed at the same position relative to the vehicle's frame to maintain consistency. Even though these LiDAR sensors share the same location, they differ in resolution, operating at 16, 32, 64, and 128 channels, allowing us to test how resolution impacts detection performance. Pointclouds collected from the CARLA simulator for estimation of R_{LR} shown in Figure 3.3 for comparison. We crafted this synthetic sensor arrangement within a uniform scenario to remove influences from other domain shift factors, such as R_{SR} (synthetic vs. real data), R_{SC} (variations in scenery), R_{LP} (differences in LiDAR placement), and R_{LC} (number of LiDAR units), ensuring that only R_{LR} drives any observed domain shift. We created four distinct datasets, each tailored to a specific LiDAR resolution (16, 32, 64, and 128 channels), resulting in the carlaScenes datasets named accordingly—carlaScenes 16, carlaScenes 32, carlaScenes 64, and carlaScenes 128—to assess the individual effect of each LiDAR's resolution. This deliberate and controlled approach lets us accurately measure how R_{LR} affects key performance metrics in 3D object detection, providing clear insights into its role. To maintain consistency with the nuScenes dataset, we positioned the LiDAR sensors at the same location and orientation relative to the ego vehicle frame, which is defined as the ground projection

of the midpoint between the two rear wheels, mirrored accordingly. Overall, we gathered 1000 scenes and sampled them to achieve approximately 28,000 samples, aligning with the sample count of the nuScenes dataset; we further emphasize maintaining a similar or identical number of training samples across all datasets, and although the sample size alone doesn't guarantee model success without considering other hyperparameters, we intentionally standardized this aspect to ensure more reliable and comparable training sessions. Each scene features a random assortment of agents—including the ego vehicle—placed and acting unpredictably across the map, which boosts the dataset's variety and strength for robust analysis. To investigate R_{SR} (synthetic vs. real data), we utilized the nuScenes dataset and created a custom dataset by labeling real-world data from ADASTEC CORP using Segments.AI. The datasets employed in this work are detailed in Table 3.1.

Table 3.1 Datasets Used in This Study

Dataset Name	**Num LiDARs**	**LiDAR Resolution**	**Synthetic**	**Number of Samples**
nuScenes	1	32	No	28130
adaScenes	5	128+32	No	19727
carlaScenes 16	1	16	Yes	27902
carlaScenes 32	1	32	Yes	27902
carlaScenes 64	1	64	Yes	27902
carlaScenes 128	1	128	Yes	27902

Although the adaScenes dataset has fewer samples compared to the others, we addressed this difference by randomly selecting an equal number of samples from the nuScenes and carlaScenes datasets to match adaScenes' size, ensuring a fair comparison without the influence of dataset length. Also, we only used the single top LiDAR data from the adaScenes to not further add a potential domain shift sources such as R_{LP} and R_{LC}. Once the datasets were prepared, we made thoughtful decisions about the 3D detection framework and neural network model for our experiments, choosing the CenterPoint model within the mmdetection3d library [32], a 3d detection framework built on PyTorch [33] that simplifies working with pre-trained models across various architectures and datasets, especially since all our datasets follow the nuScenes format; notably, CenterPoint already has a pre-trained version for nuScenes, though it relies on the dataset's inclusion of point cloud intensity data.

The intensity value in a point cloud is a measure of how much a surface reflects the LiDAR signal, influenced by distance because signal strength weakens over range, but this measurement varies between LiDAR manufacturers due to differences in their hardware and calibration methods, making it inconsistent across devices. A key challenge arises with CARLA's LiDAR simulator, which assigns intensity using a basic formula, $I = e^{-\alpha d}$, where α is a fixed attenuation rate and d is the point's distance; this oversimplified approach produces intensity values that don't match real-world conditions [34], lacking the complexity of actual sensor behavior [35]. Because of this limitation and the variability in real LiDAR intensity, we chose to retrain the CenterPoint model without using the intensity channel, ensuring our results depend on more reliable features like position and avoid potential inaccuracies introduced by this noisy and simulator-specific data.

For training sessions, we have used in total of 20 epochs with learning rate auto scaling and only kept the $[car, motorcycle, pedestrian]$ heads during training. This class truncation is done because CARLA does not provide any distinguishing classes between four wheeled objects such as $truck, bus$ or two wheeled objects such as $bicycle, scooter$. For concrete model configuration, we have used `centerpoint_pillar02_second_secfpn_8xb4-cyclic` which corresponds to the CenterPoint model with Pillar encoding with a 0.2 voxel resolution, SECOND backbone, SECONDFPN neck, batch normalization applied throughout the model and a cyclic learning rate schedule over 20 epochs. Remaining identifier, `8xb4` refers to having 8 samples per GPU across 4 GPUs, which we do share in our training sessions. We have trained and tested all of our models on machine equipped with 4 x V100 GPU. Execution times are shown in Table 3.2.

Table 3.2 Datasets Used in This Study

Dataset Name	**Number of Samples**	**Number of Epochs**	**LiDAR Resolution**	**Time(hours)**
nuScenes	28130	20	32	36
adaScenes	19727	20	128	42
carlaScenes 16	27902	20	16	11
carlaScenes 32	27902	20	32	14
carlaScenes 64	27902	20	64	22
carlaScenes 128	27902	20	128	33

CHAPTER 4

EVALUATION AND RESULTS

We evaluate model performance across datasets by presenting key metrics, including mean Average Precision (`mAP`), NuScenes Detection Score (`NDS`), and class-specific Average Precision at a 0.5 IoU threshold for cars (`Car AP 0.5`), pedestrians (`Pedestrian AP 0.5`), and motorcycles (`Motorcycle AP 0.5`), as detailed in Tables 4.1, 4.2, 4.3, 4.4, and 4.5, respectively. To further assess domain shift impacts, we report the performance differences, $\Delta D^{S_i \to S_j}$, in Tables 4.7, 4.6, 4.8, and 4.9, highlighting how these variations influence detection accuracy across datasets.

Table 4.1 Trained/Tested mAP Across Datasets

Train/Test	**carlaScenes 16**	**carlaScenes 32**	**carlaScenes 64**	**carlaScenes 128**	**nuScenes**	**adaScenes**
carlaScenes 16	0.8398	0.8265	0.7761	0.6684	0.0973	0.0998
carlaScenes 32	0.7714	0.9405	0.9474	0.9337	0.1148	0.1922
carlaScenes 64	0.6618	0.9089	0.9636	0.9727	0.1254	0.2227
carlaScenes 128	0.5541	0.8547	0.9493	0.9721	0.1014	0.22
nuScenes	0.5574	0.695	0.7134	0.6771	0.5011	0.4957
adaScenes	0.3898	0.4849	0.5201	0.5262	0.1619	0.542

Table 4.2 Trained/Tested NDS Across Datasets

Train/Test	**carlaScenes 16**	**carlaScenes 32**	**carlaScenes 64**	**carlaScenes 128**	**nuScenes**	**adaScenes**
carlaScenes 16	0.7518	0.7319	0.6987	0.6282	0.2694	0.2147
carlaScenes 32	0.7178	0.8135	0.8147	0.7961	0.2976	0.2767
carlaScenes 64	0.6641	0.8017	0.8368	0.8386	0.3187	0.2996
carlaScenes 128	0.6009	0.7639	0.8249	0.8404	0.3077	0.3031
nuScenes	0.4939	0.5691	0.573	0.5554	0.5743	0.4884
adaScenes	0.3798	0.4314	0.4566	0.4649	0.3243	0.5339

Table 4.3 Trained/Tested CAR AP 0.5 Across Datasets

Train/Test	**carlaScenes 16**	**carlaScenes 32**	**carlaScenes 64**	**carlaScenes 128**	**nuScenes**	**adaScenes**
carlaScenes 16	0.863	0.8203	0.8364	0.809	0.134	0.0444
carlaScenes 32	0.7635	0.9075	0.9336	0.9071	0.1259	0.0647
carlaScenes 64	0.6127	0.8548	0.9511	0.9624	0.1178	0.065
carlaScenes 128	0.4639	0.7669	0.9353	0.963	0.0854	0.046
nuScenes	0.5169	0.6036	0.6219	0.5475	0.5220	0.4795
adaScenes	0.2934	0.3828	0.4388	0.4394	0.1942	0.543

Table 4.4 Trained/Tested PEDESTRIAN AP 0.5 Across Datasets

Train/Test	carlaScenes 16	carlaScenes 32	carlaScenes 64	carlaScenes 128	nuScenes	adaScenes
carlaScenes 16	0.6678	0.6936	0.6275	0.517	0.0155	0.1586
carlaScenes 32	0.6181	0.899	0.9218	0.925	0.1002	0.4053
carlaScenes 64	0.6017	0.8947	0.9332	0.9543	0.1464	0.4951
carlaScenes 128	0.5184	0.8471	0.9184	0.9519	0.1193	0.5055
nuScenes	0.4167	0.6549	0.704	0.7481	0.5800	0.6942
adaScenes	0.3331	0.5816	0.6288	0.7015	0.1547	0.7688

Table 4.5 Trained/Tested MOTORCYCLE Across Datasets

Train/Test	carlaScenes 16	carlaScenes 32	carlaScenes 64	carlaScenes 128	nuScenes	adaScenes
carlaScenes 16	0.8898	0.8442	0.7598	0.576	0.0041	0.0012
carlaScenes 32	0.8289	0.9493	0.9394	0.9081	0.0033	0.0077
carlaScenes 64	0.6607	0.8959	0.9745	0.9782	0.0046	0.0171
carlaScenes 128	0.5589	0.8355	0.9499	0.9787	0.0012	0.0276
nuScenes	0.4752	0.6075	0.5954	0.5459	0.1579	0.0706
adaScenes	0.3014	0.2588	0.2897	0.2716	0.0052	0.1249

Table 4.6 mAP Differences (±) Across Datasets

Train/Test	carlaScenes 16	carlaScenes 32	carlaScenes 64	carlaScenes 128	nuScenes	adaScenes
carlaScenes 16	0	-0.0133	-0.0637	-0.1714	-0.7425	-0.74
carlaScenes 32	-0.1691	0	0.0069	-0.0068	-0.8257	-0.7483
carlaScenes 64	-0.3018	-0.0547	0	0.0091	-0.8382	-0.7409
carlaScenes 128	-0.418	-0.1174	-0.0228	0	-0.8707	-0.7521
nuScenes	0.0563	0.1939	0.2123	0.176	0	-0.0054
adaScenes	-0.1522	-0.0571	-0.0219	-0.0158	-0.3801	0

Table 4.7 NDS Differences (±) Across Datasets

Train/Test	carlaScenes 16	carlaScenes 32	carlaScenes 64	carlaScenes 128	nuScenes	adaScenes
carlaScenes 16	0	-0.0199	-0.0531	-0.1236	-0.4824	-0.5371
carlaScenes 32	-0.0957	0	0.0012	-0.0174	-0.5159	-0.5368
carlaScenes 64	-0.1727	-0.0351	0	0.0018	-0.5181	-0.5372
carlaScenes 128	-0.2395	-0.0765	-0.0155	0	-0.5327	-0.5373
nuScenes	-0.0804	-0.0052	-0.0013	-0.0189	0	-0.0859
adaScenes	-0.1541	-0.1025	-0.0773	-0.069	-0.2096	0

Table 4.8 CAR AP 0.5 Differences (±) Across Datasets

Train/Test	carlaScenes 16	carlaScenes 32	carlaScenes 64	carlaScenes 128	nuScenes	adaScenes
carlaScenes 16	0	-0.0427	-0.0266	-0.054	-0.729	-0.8186
carlaScenes 32	-0.144	0	0.0261	-0.0004	-0.7816	-0.8428
carlaScenes 64	-0.3384	-0.0963	0	0.0113	-0.8333	-0.8861
carlaScenes 128	-0.4991	-0.1961	-0.0277	0	-0.8776	-0.917
nuScenes	-0.0051	0.0816	0.0999	0.0255	0.0000	-0.0425
adaScenes	-0.2496	-0.1602	-0.1042	-0.1036	-0.3488	0

Table 4.9 PED AP 0.5 Differences (±) Across Datasets

Train/Test	carlaScenes 16	carlaScenes 32	carlaScenes 64	carlaScenes 128	nuScenes	adaScenes
carlaScenes 16	0	0.0258	-0.0403	-0.1508	-0.6523	-0.5092
carlaScenes 32	-0.2809	0	0.0228	0.026	-0.7988	-0.4937
carlaScenes 64	-0.3315	-0.0385	0	0.0211	-0.7868	-0.4381
carlaScenes 128	-0.4335	-0.1048	-0.0335	0	-0.8326	-0.4464
nuScenes	-0.1633	0.0749	0.1240	0.1681	0.0000	0.1142
adaScenes	-0.4357	-0.1872	-0.14	-0.0673	-0.6141	0

The following figures explore how LiDAR resolution influences different performance metrics within the carlaScenes datasets. In these visualizations, the x-axis represents the LiDAR resolution, which includes 16, 32, 64, and 128 channels, while the y-axis displays the corresponding metric values. To emphasize specific results, a marker and an arrow are used to highlight test outcomes when the training and testing datasets share the same LiDAR resolution. This highlighting is applied to same-dataset evaluations, such as carlaScenes 16 tested on carlaScenes 16, as well as cross-dataset evaluations, like nuScenes (with 32 channels) tested on carlaScenes 32, and adaScenes (with 128 channels) tested on carlaScenes 128, even though these datasets are not identical.

More specifically, Figure 4.3 provides a detailed view for models both trained and tested on carlaScenes datasets. On the left side, it features a heatmap showing the performance differences across various carlaScenes train-test pairs, with colors indicating the magnitude of these differences. On the right side, a line chart illustrates the class-specific Average Precision at 0.5 IoU for cars (Car AP 0.5), pedestrians (Pedestrian AP 0.5), and motorcycles (Motorcycle AP 0.5). Similarly, Figures 4.1 and 4.2 present the mean Average Precision (mAP) and NuScenes Detection Score (NDS), respectively, for models trained and tested on carlaScenes datasets. In both figures, the left side shows a heatmap representing the performance differences as listed in Tables 4.6 and

4.7, while the right side displays a line plot that tracks how these metrics change across different LiDAR resolutions. Together, these figures demonstrate the effect of LiDAR resolution on model performance within the carlaScenes dataset family.

On the other hand, Figure 4.4 examines the same set of metrics—mAP, NDS, Car AP 0.5, Pedestrian AP 0.5, and Motorcycle AP 0.5—but for models trained on nuScenes and adaScenes and then tested on carlaScenes datasets. This figure also highlights points where the LiDAR resolutions match between the training and testing datasets, making it easier to spot these specific comparisons. By doing so, it reveals how performance varies due to domain shift when models are applied across different datasets, offering insights into the challenges of cross-dataset generalization.

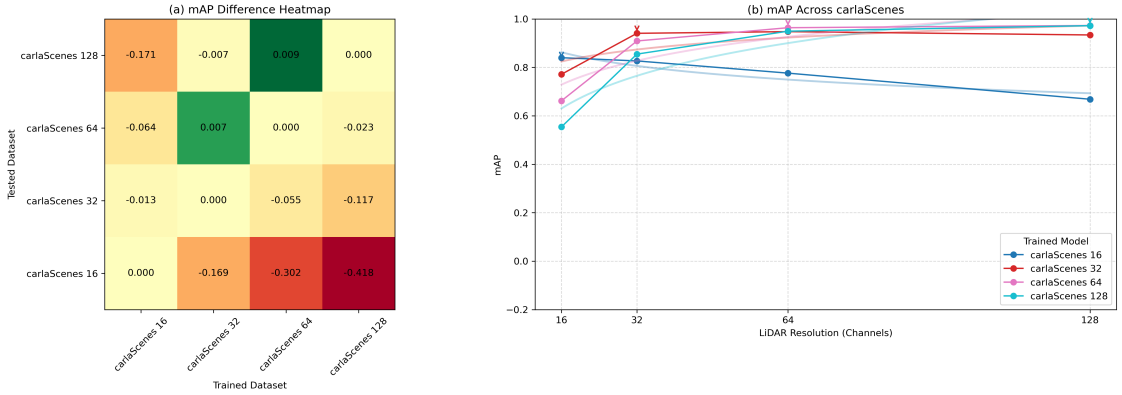


Figure 4.1 Heatmap (from Table 4.6) and line plot illustrating the effect of LiDAR resolution (R_{LR}) on mean Average Precision (mAP) across carlaScenes datasets, with highlighted points for same-resolution train-test pairs

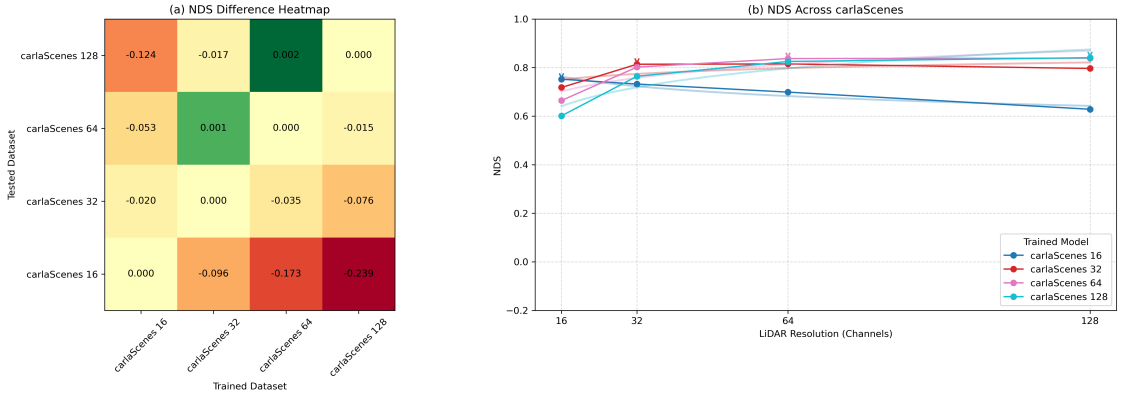


Figure 4.2 Heatmap (from Table 4.7) and line plot depicting the influence of LiDAR resolution (R_{LR}) on NuScenes Detection Score (NDS) across carlaScenes datasets, with highlighted points for same-resolution train-test pairs

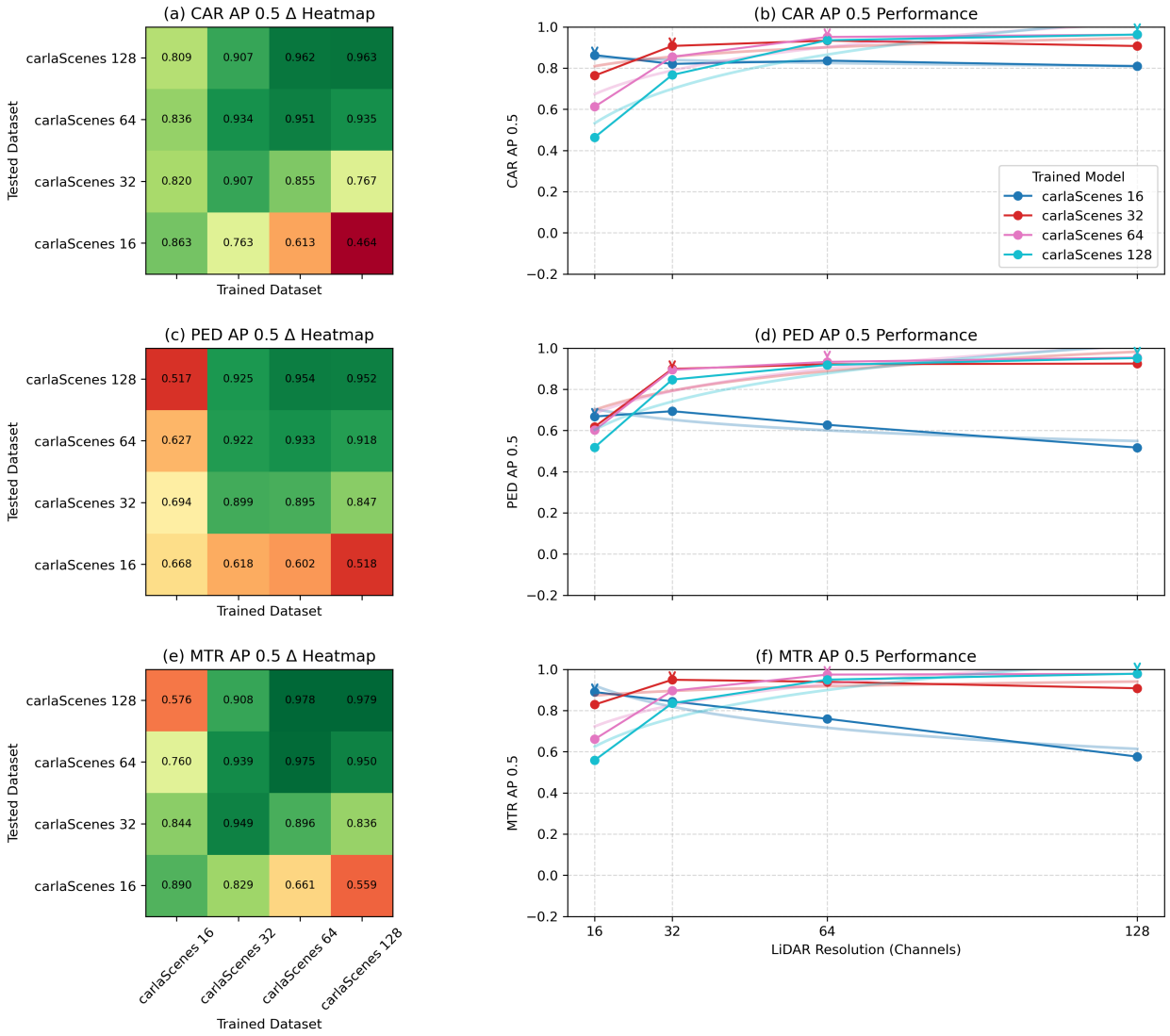


Figure 4.3 Heatmaps and line plots showing the impact of LiDAR resolution (R_{LR}) on class-specific metrics (Car AP 0.5, Pedestrian AP 0.5, Motorcycle AP 0.5) across carlaScenes datasets, with highlighted points indicating same-resolution train-test pairs

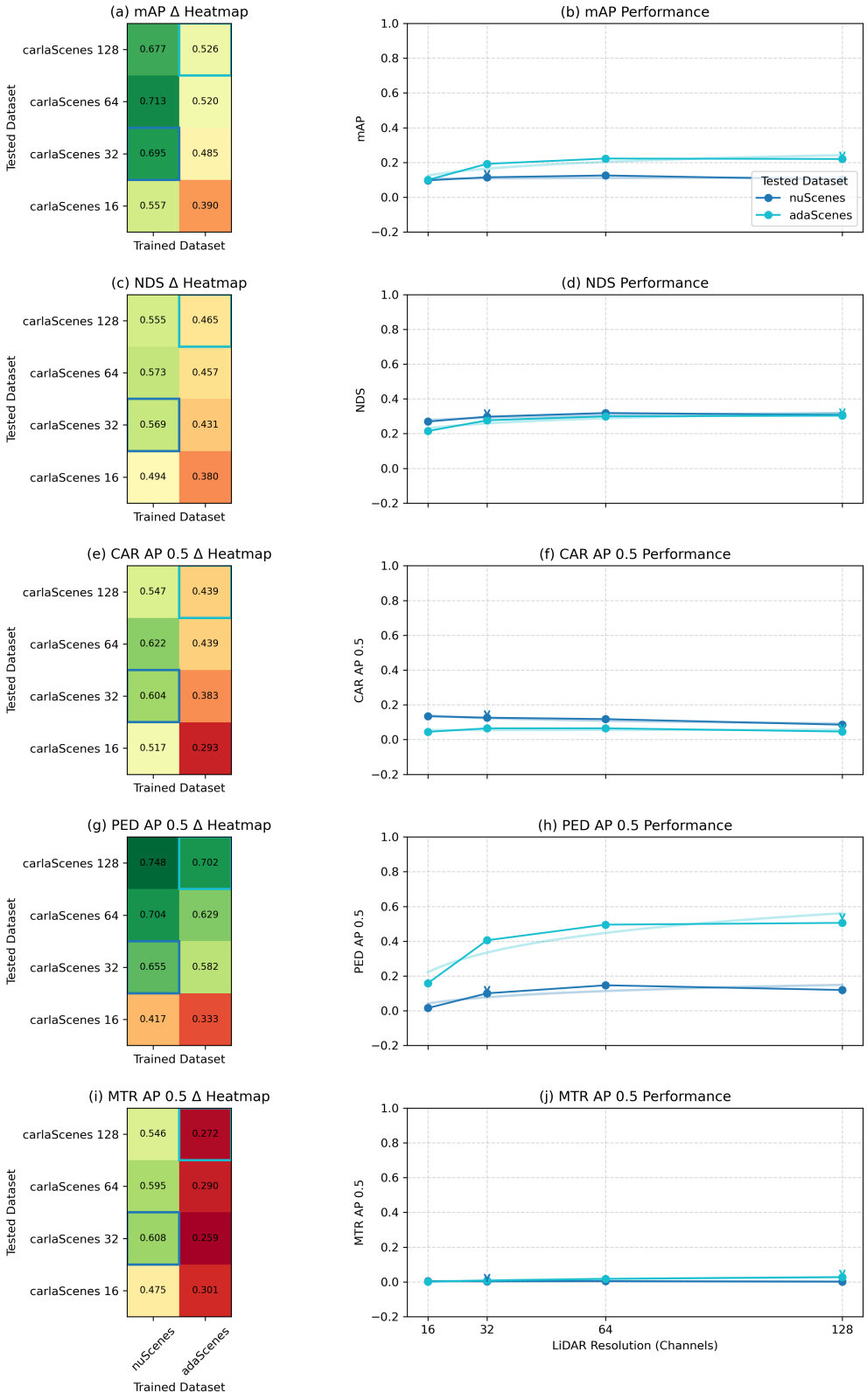


Figure 4.4 Heatmaps and line plots showing the performance of models trained on nuScenes and adaScenes and tested on carlaScenes datasets for mAP, NDS, Car AP 0.5, Pedestrian AP 0.5, and Motorcycle AP 0.5, with highlighted points for matching LiDAR resolutions

4.1 Effect of LiDAR Resolution, R_{LR}

To elucidate the influence of LiDAR resolution (R_{LR}) on detection performance, a detailed examination of Figures 4.3, 4.1, and 4.2 are warranted. These figures illustrate the performance of the CenterPoint model across carlaScenes datasets, which share identical environmental settings but differ in LiDAR resolutions (16, 32, 64, and 128 channels). For aggregate performance metrics such as mean Average Precision (mAP) and NuScenes Detection Score (NDS)—which combine detailed measures of detection accuracy across object classes and attributes like translation, scale, and orientation—a clear trend of logarithmic-like saturation is observed when comparing models trained and evaluated on their source datasets for carlaScenes 32, 64, and 128. This saturation indicates that beyond a certain resolution threshold, additional increases in LiDAR channels yield diminishing improvements in performance. For instance, a model trained on carlaScenes 64 achieves an mAP of 0.9636 on carlaScenes 64 (Table 4.1), with only marginal gains to 0.9727 on carlaScenes 128, despite the doubled resolution. When assessing the resilience of these models to resolution shifts in cross-dataset evaluations, single-step changes—such as from 32 to 64 channels or 64 to 128 channels—demonstrate minimal impact on performance metrics. This is evident in the results for models trained on carlaScenes 32, which achieve an mAP of 0.9474 on carlaScenes 64 (a single-step increase), compared to 0.9405 on their source dataset (Table 4.1). Similarly, a carlaScenes 64-trained model maintains an NDS of 0.8017 on carlaScenes 32 (a single-step decrease), close to its source NDS of 0.8368 (Table 4.2). In contrast, larger resolution changes, such as two-step shifts (e.g., from 64 to 16 channels or 128 to 32 channels), result in substantial performance degradation. For example, a carlaScenes 64-trained model, which achieves an mAP of 0.9636 on its source dataset, drops to an mAP of 0.6618 on carlaScenes 16 (Table 4.1), a two-step decrease, highlighting a significant loss in detection capability of approximately 0.3018 in mAP. Similarly, a carlaScenes 128-trained model, which achieves an mAP of 0.9721 on its source dataset, drops to an mAP of 0.8547 on carlaScenes 32 (Table 4.1), a two-step decrease, reflecting a notable decline of 0.1174 in mAP, though less extreme than the drop observed with carlaScenes 16. This pattern implies that single-step transitions between typical LiDAR resolutions

(e.g., 16 to 32, 32 to 64, or 64 to 128 channels) do not severely compromise model efficacy, whereas larger shifts do. A practical illustration is a model trained on 64-channel LiDAR, which performs robustly on both 32-channel (mAP of 0.9089) and 128-channel (mAP of 0.9727) point clouds, yet falters significantly on 16-channel data (mAP of 0.6618), as shown in Table 4.1. Consequently, when designing a training dataset for the CenterPoint model to ensure robust performance across a spectrum of LiDAR resolutions, these findings suggest that intermediate resolutions, such as 32 or 64 channels, may offer a balanced trade-off between performance and adaptability, with further analysis to follow.

Nevertheless, an exception to this trend is observed with carlaScenes 16, where the model's behavior diverges markedly from the patterns seen in higher-resolution datasets. We hypothesize that this anomaly stems from the CenterPoint model's configuration, which struggles to extract generalizable features from the sparse 16-channel LiDAR data across diverse object classes, leading to overfitting to the specific point cloud distributions of carlaScenes 16. This overfitting is particularly pronounced in class-specific metrics for smaller objects, such as pedestrians and motorcycles, as depicted in Figures 4.3(d) and 4.3(f). For instance, a model trained on carlaScenes 16 achieves a Pedestrian AP 0.5 of 0.6678 on its source dataset, but this plummets to 0.518 when tested on carlaScenes 128—a three-step resolution increase (Table 4.4). Similarly, Motorcycle AP 0.5 drops from 0.8898 on carlaScenes 16 to 0.558 on carlaScenes 128 (Table 4.5). Conversely, the Car AP 0.5 exhibits greater stability, saturating around 0.82 across resolutions; for example, it reaches 0.8364 on carlaScenes 64 and 0.809 on carlaScenes 128 for a carlaScenes 16-trained model (Table 4.3). This resilience likely arises from the larger physical size of car objects, which ensures their shapes remain discernible even in sparser point clouds, unlike smaller objects that demand denser data for accurate detection. The saturation of Car AP 0.5, rather than a steep decline, also sheds light on the limitations of the model's complexity and its default configuration. The CenterPoint model employed here mirrors one of the default training setups for the nuScenes dataset (32 channels) from mmdetection3d, utilizing pillar-based voxelization with fixed parameters: `voxel_size` = $[0.2, 0.2, 10]$ and `max_voxels` = $[30000, 40000]$. For carlaScenes 16, the

sparsity of the 16-channel point clouds results in fewer points per voxel, potentially causing the model to overfit by memorizing resolution-specific patterns rather than learning broadly applicable features. In higher-resolution datasets like carlaScenes 64 and 128, the denser point clouds overwhelm these fixed parameters. The `voxel_size`, optimized for 32-channel data, becomes too coarse for 64- and 128-channel inputs, failing to capture the finer details available in these denser clouds. Additionally, the `max_voxels` limit triggers random truncation of points in approximately 30% of object related voxels in these higher-resolution datasets, discarding valuable information. This truncation skews the model toward learning localized relationships within truncated point cloud patches, rather than fostering a holistic, resolution-agnostic understanding of object shapes within the environment. As a result, the model's generalization across resolutions is impaired, particularly for smaller objects like pedestrians and motorcycles. These findings indicate that modifying the voxelization parameters—such as adopting a resolution-dependent `voxel_size` or implementing a dynamic `max_voxels` threshold—could improve the model's capacity to learn robust and transferable features across diverse LiDAR resolutions, thereby alleviating the observed domain shift effects. However, for a more straightforward solution, we suggest increasing `max_voxels` to $[100000, 100000]$ from the default $[30000, 40000]$ to reduce truncation in dense point clouds like carlaScenes 64 and 128, allowing the model to retain more information from high-resolution data. Additionally, we propose adjusting `voxel_size` to $[0.1, 0.1, 10]$ from $[0.2, 0.2, 10]$, keeping the z-dimension unchanged. This finer horizontal resolution in x and y enables better capture of small objects like pedestrians and motorcycles, which benefit from increased cell occupation per object rather than vertical detail. Since CenterPoint employs pillar-based encoding, where the z-axis is collapsed into a single pillar, refining the z-resolution offers no advantage and aligns with the pillar feature encoder's design, unlike an alternative version of model with voxel feature encoder where z-resolution might matter. Furthermore, to complement the increased complexity of the feature extraction process, we propose enhancing the depth of the class-specific `SeparateHead` components within the `CenterHead` of the CenterPoint model. Specifically, we recommend increasing the number of convolutional layers—each consisting of Conv + BatchNorm + ReLU—from the

original 2 to at least 4. This adjustment provides the model with greater capacity to process and distill the more intricate feature sets generated by the proposed adaptive feature extraction, thereby improving its ability to generalize across varying LiDAR resolutions.

In order to find the R_{LR} distribution, we have used the carlaScenes 32 and carlaScenes 128 as test datasets, renamed as `carlasc32` and `carlasc128` in this section to avoid clutter, because the real-life datasets in this work also use the same resolutions. We aim to approximate a Gaussian distribution for R_{LR} for each performance metric; to achieve this, we calculate the difference between the minimum and maximum performance metrics for each trained model across the test datasets `carlasc32` and `carlasc128`. Here, R_{LR} is an abstract definition representing the domain shift due to LiDAR resolution, and we assume that any performance metric (e.g., NDS, mAP, Car AP 0.5) provides an equally acceptable approximation of R_{LR}. Specifically, for each trained model in the set {carlasc32, carlasc64, carlasc128}, we compute the difference $\min D^{S_i \to S_j} - \max D^{S_i \to S_j}$, where D is treated as a variable representing the performance metric, and $S_j \in \{\text{carlasc32}, \text{carlasc128}\}$, excluding `carlasc16` due to the model not learning properly, as discussed in previous paragraphs. These differences form a set of samples used to approximate the R_{LR} distribution as a Gaussian for each metric, and ultimately, R_{LR} relates to these distributions we approximate, providing insight into the domain shift caused by LiDAR resolution:

$$R_{LR} \sim \text{Gaussian}\left(\left\{\min_{S_j} D^{S_i \to S_j} - \max_{S_j} D^{S_i \to S_j} \;\middle|\; \begin{smallmatrix} S_i \in \{\text{carlasc32,carlasc64,carlasc128}\} \\ S_j \in \{\text{carlasc32,carlasc128}\} \end{smallmatrix}\right\}\right) \tag{4.1}$$

The Gaussian distributions in Figure 4.6 illustrate the combined effects of LiDAR-resolution shift (R_{LR}) and synthetic vs. real data shift (R_{SR}) on various performance metrics across datasets. In the context of R_{LR}, we observe that NDS exhibits the smallest standard deviation ($\sigma = 0.025$) among the metrics, indicating that it is the most consistent in capturing the effect of LiDAR resolution on performance. This suggests that NDS, despite being a composite metric derived from multiple true-positive metrics, effectively reflects the performance loss due to domain shift and is a more reliable choice for comparing domain shifts between dataset pairs when a single metric is needed. Following NDS, Pedestrian AP 0.5 ($\sigma = 0.032$) and Motorcycle AP 0.5 ($\sigma = 0.042$) also

show relatively low deviations, meaning they are more confident in distinguishing the domain shift between the two LiDAR resolutions. Although their mean differences ($\mu = -0.063$ for Pedestrian AP 0.5 and $\mu = -0.089$ for Motorcycle AP 0.5) are closer to zero compared to Car AP 0.5, their smaller standard deviations indicate that these metrics, which focus on smaller objects, are less variable and thus provide a clearer signal of the domain shift caused by LiDAR resolution. In contrast, Car AP 0.5 has the largest standard deviation ($\sigma = 0.080$) and the furthest mean from zero ($\mu = -0.101$), suggesting greater variability in its estimation of domain shift. We attribute this higher variability to the two-step resolution jump between carlaScenes 32 and carlaScenes 128 (from 32 to 128 channels). As discussed in previous sections, while car detection performance tends to saturate between carlaScenes 32 and carlaScenes 64, this larger resolution jump significantly impacts models trained on high-resolution datasets (e.g., carlaScenes 128) when tested on lower-resolution datasets (e.g., carlaScenes 32), leading to more pronounced and variable performance drops for larger objects like cars.

4.2 Thoughts on Performance Metrics

Among the evaluation metrics we consider, the NuScenes Detection Score (NDS) proves substantially more robust to outliers than mean Average Precision (mAP). To demonstrate this, we performed a sensitivity analysis by fitting Gaussian curves to each test-column in the performance difference tables ($\Delta D^{S_i \to S_j}$) for mAP (Table 4.6) and NDS (Table 4.7), and plotting the resulting distributions (Figure 4.5). The noticeably narrower distributions for NDS confirm its lower variability and greater resilience to large performance deviations. Consequently, when comparing overall 3D detection performance without focusing on a particular object class, NDS is the preferred metric because it aggregates multiple true-positive sub-metrics into a single, stable score.

The NuScenes Detection Score (NDS) is defined as a weighted sum of mean Average Precision (mAP) and five true-positive error metrics — mean Average Translation Error (mATE), mean Average Scale Error (mASE), mean Average Orientation Error (mAOE), mean Average Velocity Error (mAVE), and mean Average Attribute Error (mAAE) — collectively denoted by TP:

$$TP = \{\text{mATE, mASE, mAOE, mAVE, mAAE}\}.$$

$$\text{NDS} = \frac{1}{10}\left[5\,\text{mAP} + \sum_{mTP \in TP}\left(1 - \min(1, mTP)\right)\right] \tag{4.2}$$

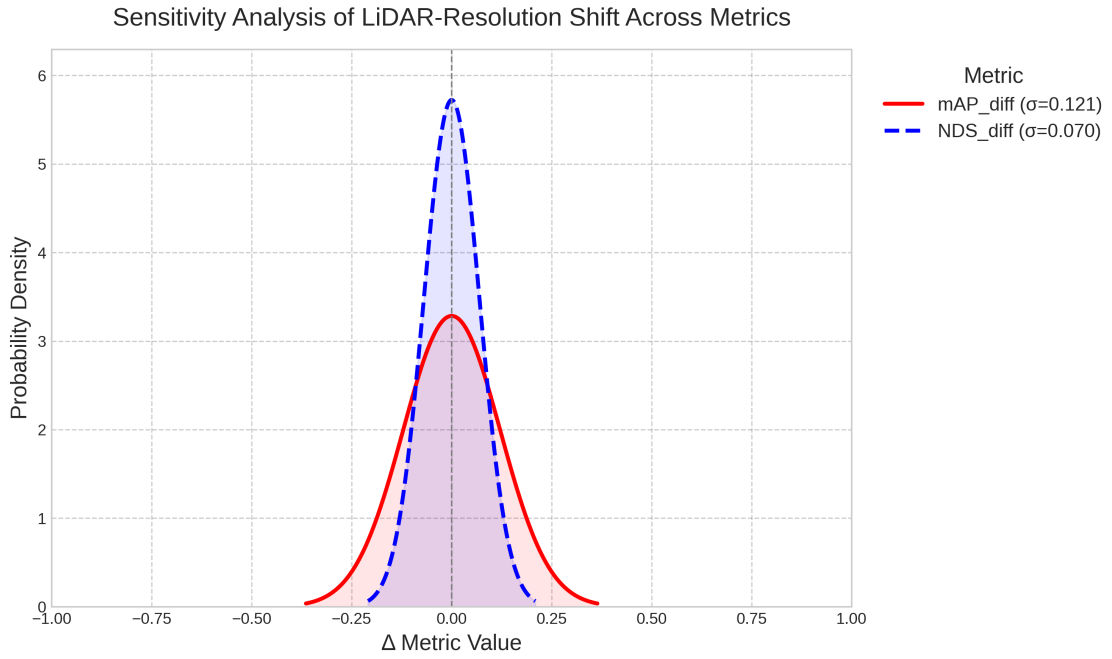


Figure 4.5 Gaussian distributions comparing the variability of performance differences ($\Delta D^{S_i \to S_j}$) for mAP and NDS across carlaScenes datasets, illustrating the robustness of NDS to LiDAR-resolution shift (R_{LR}) with standard deviations $\sigma = 0.121$ for mAP and $\sigma = 0.070$ for NDS (see Tables 4.6 and 4.7)

4.3 Effect of Synthetic vs Real Life, R_{SR}

In addition to the carlaScenes datasets, we incorporated the adaScenes and nuScenes datasets, which were collected using sensors mounted on ego vehicles operating in real-world environments. The adaScenes dataset is a custom dataset generated from real-life sensors mounted on a bus measuring 8 meters in length and 2.3 meters in width, equipped with a top-mounted 128-channel LiDAR sensor, which we exclusively used to avoid introducing R_{LC} (number of LiDAR units) as an additional domain shift factor. However, the difference in ego vehicle dimensions between adaScenes and other datasets, such as nuScenes or carlaScenes, inevitably introduces R_{LP} (differences in

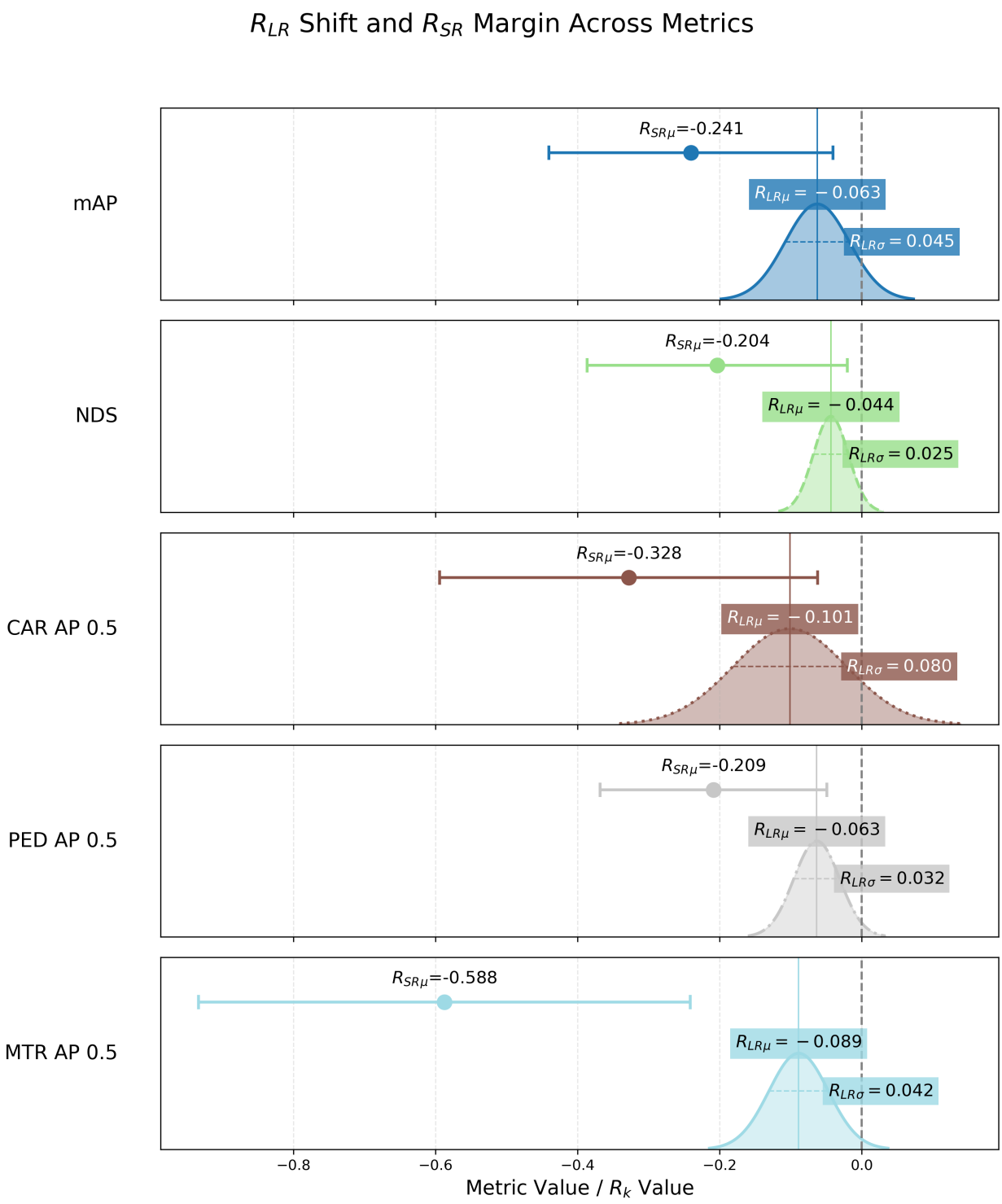


Figure 4.6 Gaussian sensitivity analysis of performance differences ($\Delta D^{S_i \to S_j}$) for multiple metrics across carlaScenes test resolutions, highlighting the variability in LiDAR-resolution shift (see Tables 4.6 and 4.7)

LiDAR placement) as a domain shift source. Comparing these real-life datasets (adaScenes and nuScenes) with the synthetic carlaScenes datasets introduces multiple potential sources of domain shift, complicating the analysis of performance differences, as detailed in Table 4.10, which lists the domain shift factors between each pair of datasets. For instance, while the nuScenes dataset employs a single 32-channel LiDAR, similar to the carlaScenes 32 dataset, a direct comparison reveals at least two distinct domain shift factors. First, R_{SR} (synthetic vs. real data) arises because the carlaScenes 32 dataset is synthetically generated, whereas the nuScenes dataset is derived from real-world data. Second, R_{SC} (variations in scenery) emerges due to differences in the environments: the nuScenes dataset was collected in urban settings across Singapore and Boston, while carlaScenes 32 was generated using `Town10`, a pre-existing map provided by the CARLA simulator, which mimics a different urban landscape. Similarly, when comparing adaScenes and carlaScenes 128, both equipped with a 128-channel LiDAR, at least two domain shift factors are present: R_{SR} (synthetic vs. real data) due to carlaScenes 128 being synthetically generated while adaScenes is real-world data, and R_{SC} (variations in scenery) because adaScenes was collected in real-world environments, while carlaScenes 128 uses the `Town10` map in the CARLA simulator, representing a different urban setting. Beyond these two domain shift sources, additional factors could further influence the results. For example, R_{LP} may play a role; although we positioned the LiDAR sensor in carlaScenes 32 to match the placement in the nuScenes dataset, the ego vehicles in the simulator and real-world settings differ. These differences affect the relative positioning of the LiDAR sensor with respect to the ego vehicle's surface, potentially altering the point cloud data for nearby objects (e.g., the closest objects and their corresponding point clouds relative to the ego vehicle frame may vary if the ego vehicle's boundaries differ). Specifically, the carlaScenes datasets were collected using an Audi vehicle model within the CARLA simulator, whereas the real-world datasets involve distinct vehicle models: nuScenes uses a smaller Renault different vehicle, and adaScenes employs a bus, adding another layer of complexity to the domain shift analysis.

Table 4.10 Domain Shift Factors Between Dataset Pairs

Train/Test	**nuScenes**	**adaScenes**	**carlaScenes 32**	**carlaScenes 128**
nuScenes	-	$R_{LR} + R_{SC} + R_{LP}$	$R_{SR} + R_{SC}$	$R_{LR} + R_{SR} + R_{SC}$
adaScenes	$R_{LR} + R_{SC} + R_{LP}$	-	$R_{SR} + R_{SC} + R_{LR} + R_{LP}$	$R_{SR} + R_{SC} + R_{LP}$
carlaScenes 32	$R_{SR} + R_{SC}$	$R_{SR} + R_{SC} + R_{LR} + R_{LP}$	-	R_{LR}
carlaScenes 128	$R_{LR} + R_{SR} + R_{SC}$	$R_{SR} + R_{SC} + R_{LP}$	R_{LR}	-

To estimate the impact of R_{SR}, we utilize known domain shift sources across datasets to isolate its effect. For instance, to eliminate the influence of R_{SC} from $\Delta D^{\text{nusc}\rightarrow\text{carlasc32}}$, we subtract $\Delta D^{\text{nusc}\rightarrow\text{adasc}}$, which introduces additional R_{LR} and R_{LP} terms into the equation. To address the R_{LR} component, we incorporate $\Delta D^{\text{carlasc32}\rightarrow\text{carlasc128}}$, leveraging the carlaScenes datasets with 32 and 128 channels. Furthermore, to mitigate the R_{LP} term, we include $\Delta D^{\text{adasc}\rightarrow\text{carlasc32}}$. This method serves as an approximation to quantify the relative contributions of different domain shift sources, aiming to understand the extent of challenges each source poses during training and testing phases. For a more precise calculation of R_{LP} in the context of R_{SR}, we propose the creation of a carlaScenes 128 Bus dataset, which would allow a focused analysis of R_{LP}. However, integrating a new bus asset into the CARLA simulator presents challenges, as it requires modeling expertise and familiarity with Unreal Engine. While CARLA provides a pre-existing bus asset, the Fuso Rosa from Mitsubishi Motors, it differs significantly from the bus in adaScenes. The Fuso Rosa measures 6.9 meters in length and 2.7 meters in height, whereas the adaScenes bus is 8.3 meters long, with its LiDAR mounted at 3.1 meters above the ground. These discrepancies suggest that using the Fuso Rosa as a substitute for the adaScenes bus would introduce further approximations, potentially increasing ambiguity in estimating R_{LP}. Therefore, a more accurate representation of the adaScenes bus is necessary to minimize such uncertainties in the analysis.

$$
\begin{aligned}
R_{SR} \sim\ & \Delta D^{\text{nusc}\rightarrow\text{carlasc32}} && \longleftarrow && +R_{SR} && +R_{SC} && && \\
& - \Delta D^{\text{nusc}\rightarrow\text{adasc}} && \longleftarrow && && - R_{SC} && - R_{LR} && - R_{LP} \\
& - \Delta D^{\text{nusc}\rightarrow\text{carlasc128}} && \longleftarrow && -R_{SR} && - R_{SC} && - R_{LR} && \\
& + \Delta D^{\text{adasc}\rightarrow\text{carlasc32}} && \longleftarrow && +R_{SR} && +R_{SC} && + R_{LR} && + R_{LP} \\
& + \Delta D^{\text{carlasc32}\rightarrow\text{carlasc128}} && \longleftarrow && && && + R_{LR} &&
\end{aligned}
$$

The same estimation of R_{SR} would also work in reverse order of the datasets:

$$
\begin{aligned}
R_{SR} \sim\ & \Delta D^{\text{carlasc32}\rightarrow\text{nusc}} && \longleftarrow && -R_{SR} && - R_{SC} && && \\
& - \Delta D^{\text{adasc}\rightarrow\text{nusc}} && \longleftarrow && && + R_{SC} && + R_{LR} && + R_{LP} \\
& - \Delta D^{\text{carlasc128}\rightarrow\text{nusc}} && \longleftarrow && +R_{SR} && + R_{SC} && + R_{LR} && \\
& + \Delta D^{\text{carlasc32}\rightarrow\text{adasc}} && \longleftarrow && -R_{SR} && - R_{SC} && - R_{LR} && - R_{LP} \\
& + \Delta D^{\text{carlasc128}\rightarrow\text{carlasc32}} && \longleftarrow && && && - R_{LR} &&
\end{aligned}
$$

In Figure 4.6, we can observe that R_{SR} effect on the amount of metric loss is far greater then the effect of R_{LR} which suggests LiDAR pointcloud from CARLA simulator differs from the real-life profoundly. This difference alone shows that for Unsupervised Domain Adaptation problems, it is better and more reliable to use real-life data as the training source in order to distill information to the target datasets.

4.4 Generalization of a Dataset

In this study, we explore how well real-world datasets, such as nuScenes, generalize compared to synthetic datasets in the context of domain shift for autonomous driving applications. The nuScenes dataset is widely valued within the autonomous driving community for several key reasons. Firstly, it captures data from diverse locations, featuring a variety of traffic patterns and complex decision-making scenarios that challenge autonomous vehicles. Secondly, with approximately 28,000 samples, nuScenes occupies a middle ground in terms of dataset size. For comparison, the Waymo

dataset[36] contains a much larger 390,000 samples, while the KITTI dataset[37] is smaller with 15,000 samples, and the Lyft dataset[38] is closer to nuScenes with 55,000 samples. This moderate size makes nuScenes a practical choice for research purposes. Additionally, while the nuScenes dataset is collected at a high frequency of 20Hz, its annotations are provided at a lower rate of 2Hz. This difference creates an interesting opportunity for multi-sweep models, which use temporal information from multiple point cloud sweeps to improve densification. Even though the extra sweeps between the labeled key frames do not come with their own annotations, this setup can actually be a strength. It encourages the models to depend on the key frame's annotations to figure out object states in the unlabeled intermediate sweeps, helping them learn more robust temporal patterns. As a result, this feature of nuScenes could make models better at generalizing to real-world autonomous driving situations, where not every frame has full labels—a common scenario that tests a model's adaptability.

Evidence from Tables 4.1 and 4.2 demonstrates that models trained on nuScenes perform robustly not only on their own dataset but also when evaluated on synthetic datasets, such as carlaScenes. This strong cross-dataset performance indicates that nuScenes enables models to learn features that are not overly specific to its own characteristics, suggesting a high capacity for generalization across different domains. Another compelling indicator of nuScenes' generalization is its performance on adaScenes, a distinct real-world dataset. As shown in Tables 4.6 and 4.7, the performance of nuScenes-trained models on adaScenes remains close to that of models both trained and tested on adaScenes. For instance, the NDS score for a nuScenes-trained model tested on adaScenes is 0.4884, which is notably close to the 0.5339 achieved by an adaScenes-trained model on its own dataset. This relatively small performance gap highlights the ability of nuScenes-trained models to adapt effectively to other real-world environments.

On the other hand, models trained on synthetic datasets like carlaScenes struggle significantly when evaluated on real-world datasets such as nuScenes or adaScenes. For instance, a model trained on carlaScenes 128, which uses the same 128-channel LiDAR as adaScenes, only achieves an NDS of 0.3031 when tested on adaScenes. This is much lower than the 0.5339 scored by a model trained

and tested on adaScenes itself. This notable performance drop emphasizes the challenges synthetic data faces in matching real-world conditions, even when the LiDAR resolution is identical.

However, it's important to note that while models trained on nuScenes perform well when tested on synthetic datasets, their scores don't match the results of models trained directly on those synthetic datasets. For example, a nuScenes-trained model tested on carlaScenes 32 earns an NDS of 0.5691, which is solid but still below the 0.8135 achieved by a model trained and tested on carlaScenes 32. This difference matters: the real power of nuScenes isn't in beating synthetic models on their own turf, but in equipping models to tackle a broad variety of scenarios—both synthetic and real-world—much better than synthetic-only training can.

Likewise, adaScenes, another real-world dataset, shows some ability to generalize, though not as strongly as nuScenes. Unlike models trained on synthetic data, which suffer huge performance drops when tested on real-world sets, adaScenes-trained models hold up better. For example, when tested on carlaScenes 128—which matches its 128-channel LiDAR—an adaScenes-trained model scores an NDS of 0.4649. This is decent but well below the 0.8404 of a carlaScenes 128-trained model, showing that adaScenes offers moderate generalization to synthetic data, though less effectively than nuScenes. On the flip side, when adaScenes-trained models are tested on nuScenes, they struggle, achieving an NDS of just 0.3243 compared to 0.5743 for a nuScenes-trained model. This big gap suggests that nuScenes might have greater complexity—think diverse road users, varied environments, or unique ego-vehicle movements—that adaScenes lacks. It seems a dataset's ability to generalize could hinge on how complex it is: richer datasets like nuScenes train models that adapt well across domains, while less complex ones like adaScenes leave models less prepared for tougher, more varied test conditions.

Both nuScenes and adaScenes were collected from multiple cities, exposing their models to a broad spectrum of environmental conditions and urban layouts. In contrast, carlaScenes is derived from a single simulated city, potentially limiting the variety of scenarios it represents. This broader real-world exposure in nuScenes and adaScenes likely aids in training models that learn more robust and transferable features, better equipping them to handle domain shifts across diverse test datasets.

CHAPTER 5

CONCLUSION

This thesis addressed the significant challenge of Unsupervised Domain Adaptation (UDA) within the domain of 3D object detection, specifically focusing on the systematic quantification and analysis of domain shifts between datasets. We began by identifying key sources of domain shift relevant to 3D object detection in autonomous driving, such as LiDAR resolution variations, differences between synthetic and real-world data, sensor placement, and scenery discrepancies.

To systematically investigate these domain shift sources, we developed a comprehensive methodological framework. Central to this framework was the generation of carefully curated datasets using the CARLA simulator, enabling precise control over domain shift factors. We introduced two novel packages, `carlaSceneCollector` and `rosbag2nuScenes`, specifically designed for this research. The `carlaSceneCollector` package streamlines the process of data generation in CARLA by automating sensor data collection, scenario configuration, and ROSBag recording, thus facilitating the creation of controlled, synthetic raw data. The `rosbag2nuScenes` package provides a unique, generic solution for converting ROSBag data into the widely-used nuScenes format, accommodating various sensor setups and ensuring compatibility with prevalent 3D detection frameworks. This represents a significant contribution, as no other publicly available tool currently offers such comprehensive and adaptable functionality, making these packages invaluable for synthetic data-driven research and development in autonomous systems.

Through rigorous experimentation, we revealed crucial insights into the relationship between LiDAR sensor resolution (R_{LR}) and detection performance. Specifically, we demonstrated a clear performance saturation effect beyond certain LiDAR resolutions, highlighting that intermediate resolutions (such as 32 or 64 channels) provide an optimal trade-off between accuracy and generalizability. However, it is essential to emphasize that this saturation effect is closely related to the underlying model and its hyperparameters. As such, these findings should not be generalized universally across all 3D detection models. We also laid out potential sources for this observed saturation effect and proposed methods to mitigate it, including adaptive voxelization parameters

and increased model complexity.

Our investigation also highlighted pronounced differences between synthetic and real-world datasets (R_{SR}). Models trained on synthetic CARLA-generated datasets showed substantial performance drops when evaluated on real-world datasets (nuScenes and adaScenes). Conversely, models trained on real-world datasets exhibited considerably better generalization capabilities, reinforcing the importance of real-world training data for robust adaptation.

Additionally, we identified the NuScenes Detection Score (NDS) as a particularly robust and reliable metric for capturing the aggregate impact of domain shift. Compared to other metrics such as mean Average Precision (mAP), NDS proved less susceptible to variability and outliers, making it well-suited for comparative studies across datasets.

Lastly, the broader generalization capabilities of real-world datasets, particularly nuScenes, were underlined. This dataset's diverse real-world scenarios and moderate complexity provided the foundation for training models with robust adaptability across varied domains, both synthetic and real. In contrast, simpler datasets like adaScenes demonstrated limited adaptability, emphasizing that dataset complexity and scenario diversity are critical for fostering model generalization. Notably, synthetic datasets generated as part of this research (carlaScenes datasets) demonstrated even lower adaptability compared to adaScenes, suggesting a significant gap in realism and scenario complexity. This limited adaptability of synthetic datasets can be attributed to the random sampling approaches used for agent motion, agent count, and ego vehicle movements, as well as the inherent limitations of the LiDAR simulator, which employs a simplified ray-casting method. Addressing these limitations and enhancing the realism and complexity of synthetic datasets remain important areas for future investigation.

In conclusion, this thesis contributes to a deeper understanding of domain shift phenomena in 3D object detection and provides a clear methodology for its quantification. Our findings underscore the critical importance of careful dataset selection, thoughtful sensor configuration, and robust metric choice when addressing the challenges posed by Unsupervised Domain Adaptation. The novel software packages developed in this research, `carlaSceneCollector` and `rosbag2nuScenes`,

significantly enhance the process of dataset generation and conversion, laying a strong foundation for future synthetic data-driven research and development in autonomous systems. Future research may explore further refinement of adaptive methodologies, leveraging real-world data more effectively, improving synthetic dataset realism, and extending these insights to broader contexts within autonomous systems and robotics.

Future Work

Given the observed performance degradation when using synthetic datasets, future work should prioritize the development and refinement of self-labeling techniques for real-world, unlabeled datasets. These approaches could leverage the superior generalization capabilities of real-world data to generate high-quality pseudo-labels, thereby reducing reliance on synthetic data and improving model adaptability across domains.

Furthermore, the superior generalization observed with the nuScenes dataset compared to both synthetic datasets and other real-world datasets like adaScenes raises important questions about the factors that contribute to a dataset's generalizability. Future research should aim to identify and quantify these factors—such as scenario diversity, data complexity, and sensor fidelity—and develop a framework for evaluating and comparing the generalization potential of different datasets. Such a framework would be invaluable for selecting optimal training datasets and establishing criteria for the collection of new datasets tailored to specific autonomous driving applications.

Additionally, the logarithmic saturation effect observed in synthetic training performance underscores the need for more sophisticated approaches to dataset creation and model training. Future work should explore adaptive dataset generation techniques that dynamically adjust to the model's learning progress, as well as augmentation strategies that introduce targeted variability to counteract saturation and enhance model robustness across a wider range of conditions.

To further enhance the applicability of these findings, future studies should incorporate a diverse array of LiDAR sensors, including solid-state LiDAR, which are becoming increasingly prevalent in autonomous systems. Moreover, efforts should be directed toward improving the fidelity of simulated LiDAR data by developing methods to accurately match the ray patterns and

noise characteristics of real-world LiDAR sensors, thereby closing the gap between synthetic and real-world data.

By pursuing these avenues, future research can build upon the insights gained in this thesis, advancing the field of 3D object detection and contributing to the development of more robust and adaptable autonomous systems.

BIBLIOGRAPHY

[1] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[2] H. Caesar, V. Bankiti, A. Lang, S. Vora, V. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving. arxiv," 2019.

[3] C. Qi, H. Su, K. Mo, and L. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation, corr abs/1612.00593," *arXiv preprint arXiv:1612.00593*, 2016.

[4] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[5] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud. corr, vol. abs/1812.04244 (2018)," *arXiv preprint arxiv:1812.04244*, 2018.

[6] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 040–11 048.

[7] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[8] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[9] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. L. Pv-rcnn, "Pointvoxel feature set abstraction for 3d object detection. in 2020 ieee," in *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 526–10 535.

[10] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11 784–11 793.

[11] J. Yang, H. Qian, Y. Xu, K. Wang, and L. Xie, "Can we evaluate domain adaptation models without target-domain labels?" *arXiv preprint arXiv:2305.18712*, 2023.

[12] Y. Wang, X. Chen, Y. You, L. E. Li, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao, "Train in germany, test in the usa: Making 3d object detectors generalize," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 713–11 723.

[13] Y. You, K. Luo, C. P. Phoo, W.-L. Chao, W. Sun, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Learning to detect mobile objects from lidar scans without labels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1130–1140.

[14] Z. Ding, Y. Hu, R. Ge, L. Huang, S. Chen, Y. Wang, and J. Liao, "1st place solution for waymo open dataset challenge–3d detection and domain adaptation," *arXiv preprint arXiv:2006.15505*, 2020.

[15] Q. Xie, E. Hovy, M. Luong *et al.*, "Self-training with noisy student improves imagenet classification. arxiv," *Learning*, 2019.

[16] J. Li, R. Xu, X. Liu, J. Ma, B. Li, Q. Zou, J. Ma, and H. Yu, "Domain adaptation based object detection for autonomous driving in foggy and rainy weather," *arXiv preprint arXiv:2307.09676*, 2023.

[17] S. Ahmed, A. Al Arafat, M. N. Rizve, R. Hossain, Z. Guo, and A. S. Rakin, "Ssda: Secure source-free domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 180–19 190.

[18] N. Hanselmann, N. Schneider, B. Ortelt, and A. Geiger, "Learning cascaded detection tasks with weakly-supervised domain adaptation," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 532–539.

[19] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks (2016)," *URL https://arxiv.org/abs/1505.07818*, 2015.

[20] D. Tsai, J. S. Berrio, M. Shan, E. Nebot, and S. Worrall, "Ms3d++: Ensemble of experts for multi-source unsupervised domain adaptation in 3d object detection," *IEEE Transactions on Intelligent Vehicles*, 2024.

[21] Z. Pang, Z. Li, and N. Wang, "Simpletrack: Understanding and rethinking 3d multi-object tracking. arxiv," *arXiv preprint arXiv:2111.09621*, 2021.

[22] C. Saltori, S. Lathuiliére, N. Sebe, E. Ricci, and F. Galasso, "Sf-uda 3d: Source-free unsupervised domain adaptation for lidar-based 3d object detection," in *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020, pp. 771–780.

[23] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi, "St3d++: Denoised self-training for unsupervised domain adaptation on 3d object detection," *arXiv preprint arXiv:2108.06682*, 2021.

[24] B. Yang, M. Bai, M. Liang, W. Zeng, and R. Urtasun, "Auto4d: Learning to label 4d objects from sequential point clouds," *arXiv preprint arXiv:2101.06586*, 2021.

[25] X. Weng, J. Wang, D. Held, and K. Kitani, "3d multi-object tracking: A baseline and new evaluation metrics," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 359–10 366.

[26] L. Fan, Y. Yang, Y. Mao, F. Wang, Y. Chen, N. Wang, and Z. Zhang, "Once detected, never lost: Surpassing human performance in offline lidar based 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 820–19 829.

[27] L. Zhang, A. J. Yang, Y. Xiong, S. Casas, B. Yang, M. Ren, and R. Urtasun, "Towards unsupervised object detection from lidar point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9317–9328.

[28] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov, "Offboard 3d object detection from point cloud sequences," in *Proceedings of the IEEE/CVF Conference*.

[29] T. Ma, X. Yang, H. Zhou, X. Li, B. Shi, J. Liu, Y. Yang, Z. Liu, L. He, Y. Qiao *et al.*, "Detzero: Rethinking offboard 3d object detection with long-term sequential point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 6736–6747.

[30] Epic Games, "Unreal engine." [Online]. Available: https://www.unrealengine.com

[31] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.

[32] M. Contributors, "MMDetection3D: OpenMMLab next-generation platform for general 3D object detection," https://github.com/open-mmlab/mmdetection3d, 2020.

[33] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[34] D. Yang, X. Cai, Z. Liu, W. Jiang, B. Zhang, G. Yan, X. Gao, S. Liu, and B. Shi, "Realistic rainy weather simulation for lidars in carla simulator," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 951–957.

[35] F. Goudreault, D. Scheuble, M. Bijelic, N. Robidoux, and F. Heide, "Lidar-in-the-loop hyperparameter optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 404–13 414.

[36] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[37] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[38] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," in *Conference on Robot Learning*. PMLR, 2021, pp. 409–418.