Frankfurt University of Applied Sciences
Faculty of Computer Science and Engineering

Enabling scenario-based testing for point cloud based Vulnerable Road Users Detection in the context of highly automated driving

Master Thesis

presented by

Bishal Gautam

Matrikelnummer: 1387308

Date:	08.04.2024
First Professor:	Prof. Dr. Andreas Pech
Second Professor:	Prof. Dr. Peter Nauth
External Supervisor:	M.Eg. Hubert Padusinski

Contact Details

Prof. Dr. Andreas Pech
Frankfurt University of Applied Sciences
Computer Science and Engineering
pech@fb2.fra-uas.de

Prof. Dr. Peter Nauth
Frankfurt University of Applied Sciences
Computer Science and Engineering
pnauth@fb2.fra-uas.de

M.Eg. Hubert Padusinski
FZI Forschungszentrum Informatik, Karlsruhe
Embedded Systems and Sensors Engineering
padusinski@fzi.de

Acknowledgements

I would like to express my heartfelt gratitude to Prof. Dr. Andreas Pech for his invaluable support, guidance, and encouragement throughout the process of completing this master thesis. His expertise, insightful feedback, and unwavering commitment have been instrumental in shaping this work. I am deeply grateful to Prof. Dr. Peter Nauth for his supervision, and teaching, and for believing in my abilities, which have been a constant source of my motivation.

I am also indebted to Hubert Padusinski for the countless hours spent discussing ideas, refining methodologies, and for providing constructive criticism. His clarification and guidance has helped me explore more in the subject matter.

Finally, I would like to express my gratitude to my family and friends for their unwavering encouragement and understanding throughout this endeavor.

This thesis would not have been feasible without the support of these individuals, and for that, I am truly grateful.

Declaration of Authorship

I hereby declare that I wrote my Master's Thesis on my own and that I have followed the regulations relating to good scientific practice of the Frankfurt University of Applied Sciences in its latest form. I did not use any unacknowledged sources or means and I marked all references I used literally or by content.

Frankfurt am Main, 08.04.2024

_____

Bishal Gautam

Abstract

Recent years have seen a significant advancement in autonomous driving technology, which has great promise for improving both the efficiency and safety of transportation. However, maintaining autonomous vehicles' dependability is still a major problem. Autonomous driving vehicles distinguish the objects in the surroundings with the help of pre-trained neural network models and sensor data. Achieving high accuracy in the training of a Deep Neural Network (DNN) necessitates exposure to a substantial volume of meticulously annotated data devoid of bias. Generating LiDAR-based datasets for training deep neural networks entails considerably greater time investment and elevated personal and manual workload costs compared to other machine vision functions such as image datasets. SOTA datasets like SemanticKITTI are publicly available to forward the development of Highly Automated Driving (HAD) system research in tasks like semantic segmentation, and object detection. However, because of the vast amount of interacting possibilities between the environment and objects, publicly accessible datasets such as SemanticKITTI do not suffice in providing the comprehensive range of scenarios necessary to thoroughly assess the functionality of Highly Automated Driving (HAD) systems within the realm of machine vision. Researchers and practitioners also decided on synthetic data from simulations. As a consequence of inadequacies in scenery assembly, sensor realism, and sensor fidelity, coupled with environmental variability and the modeling of dynamic objects, a persistent domain gap endures between simulated and real-world environments. Despite improvements in the performance of models, such as those used for object detection, these advancements are still deemed insufficient. Creating sufficiently robust training and testing datasets for DNN poses a formidable challenge, demanding state-of-the-art methodologies to ensure validity across diverse operational scenarios within its domain.

This thesis investigates methods to enhance the variety of circumstances between foreground objects and background scenery of LiDAR test data within the scope of scenery assembly, particularly focusing on the interplay between Vulnerable Road Users (VRUs), such as pedestrians, and their surrounding background. The adopted approach entails extracting foreground objects as representative prototypes or clusters by leveraging the 3D geometric information of points obtained from a source point cloud. Subsequently, this information is utilized to augment another

point cloud, referred to as the target, by accurately situating the object within the target environment. This thesis introduces a method for recombining point clouds through a statistical approach aimed at diversifying the variety of foreground objects, encompassing their appearance and positional relationship concerning the LiDAR sensor and the scene. These methodologies hold applicability in testing scenarios involving both real-world driving situations and virtual test environments, particularly in cases where LiDAR data is utilized. The utilization of 3D geometric information from points facilitates the extraction of prototypes (VRUs) from a source scene cloud. Thereafter, the extracted prototype is relocated to a specified target location within the target scene cloud. Following the surface reconstruction of the "transformed" prototype, the point cloud associated with the prototype is recalculated, considering its new position within the target scene cloud. Furthermore, shadow-casting computations are performed to assess the prototype's influence on the target location. Consequently, a novel scenario point cloud is generated through the recombination of the prototype extracted from a source scene cloud and positioned within the target scene cloud.

Contents

List of Figures

List of Tables

List of Acronyms

ADAS Advanced Driver Assistance Systems

BR Back to Reality

CAD Computer-Aided Design

CARLA CAR Learning to Act

CG Computer Graphics

DA Data Augmentation

DADA Dual Adaptive Data Augmentation

DNN Deep Neural Network

ECUs Electronic Control Units

FN False Negative

FP False Positive

GPS Global Positioning System

GT Ground Truth

GUI Graphical User Interface

HAD Highly Automated Driving

HIL Hardware-in-the-Loop

HPR Hidden Point Removal

ICP Iterative Closest Point

IoU Intersection over Union

MIL Model-in-the-Loop

ML Machine Learning

MSDA Mixed Sample Data Augmentation

NMS Non-maximum Suppression

NMVCCS National Motor Vehicle Crash Causation Survey

ODD Operational Design Domain

PA-AUG Part-Aware Data Augmentation

PCD Point Cloud

PPBA Progressive Population Based Augmentation

ROI Region of Interest

ROS Robot Operating System

SDV Self-Driving Vehicle

SE-SSD Self-Ensembling Single-Stage object Detector

SIL Software-in-the-Loop

SOTA State of the Art

TN True Negative

TP True Positive

VIL Vehicle-in-the-Loop

VRUs Vulnerable Road Users

XIL X-in-the-Loop

1 Introduction

1.1 Motivation

A survey report (NMVCCS) presented by the U.S. Department of Transportation, National Highway Traffic Safety Administration in 2008 (Transport, 2008) shows that about 94% of road accidents are caused by human errors. Failure to use a seatbelt, excessive speeding, distractions from various sources, and tailgating are among the causes of road incidents. False assumptions about other drivers' habits and neglecting to check traffic before pulling out also contribute to these incidents. With the goal of preventing collisions, cutting emissions, assisting the mobility-impaired in getting around, and easing the stress associated with driving, Highly Automated Driving (HAD) Systems are being researched. The development of deep learning and the accessibility of sensors such as LiDAR, radar, camera, etc have led to significant advancements in computer vision and accelerated the study and application of HAD in industry. Autonomous robotaxis are operating in major cities like San Francisco, Phoenix, Beijing, Shanghai, etc (Yang, 2024). Among the sensors such as camera, ultrasonic, radar, and LiDAR, the depth information is captured by all the sensors except camera. Among the depth information capturing sensors, LiDAR has the highest accuracy for a range less than 200 meters (Yurtsever et al., 2020). LiDAR has depth information in comparison to single cameras. It allows to estimate the reflectivity of object's surface based on absorption intensity. Newer LiDAR sensor types also include velocity measurement. However, companies have different carlines and the setup for each carline is different. For detection in short to middle range (Up to 150 meters (Molom-Ochir, 2019)), cameras are dominantly being used (e.g. Tesla Carlines, Mercedes Class S) in the HAD industry. LiDAR and Radar are used in carlines mostly for middle to long-range detection (Up to or over 250 meters (Molom-Ochir, 2019)). A LiDAR-equipped Lexus and a Tesla Model 3 were tested in the dark by Luminar to demonstrate the variations in safety measures (Neves, 2022). They encountered a dummy pedestrian. The Tesla mowed down the dummy while the Lexus stopped in time. While the system's usefulness cannot be disputed, this seems to indicate that the camera-equipped Tesla is less effective at night. The 2007 DARPA Grand Challenge showcased the capabilities of LiDAR perception systems and was a significant event in the field of autonomous driving. Each of the top three teams'

systems was harnessed with LiDAR. Since 2015 (Guo et al., 2020), deep learning-based point cloud analysis, including object categorization and segmentation, has advanced significantly thanks to the availability of publically available datasets and progress in deep learning methodology for 3D. Deep learning on 3D PCDs is still beset by a number of serious difficulties, including the chaotic structure of 3D point clouds, the high dimensionality, and the small size of datasets (Charles R. Qi et al., 2017). In the context of LiDAR sensors like Velodyne HDL-64E, the sensor reads more than 1.3 million points per second from space giving an output of 100MBPS UDP Packets (Hyper-Tech, 2014). Annotating and verifying each and every point of the LiDAR point cloud is time-consuming. It took annotators about 1700 hours to label and verify the labeling process in the SemanticKITTI dataset (Behley et al., 2019b).

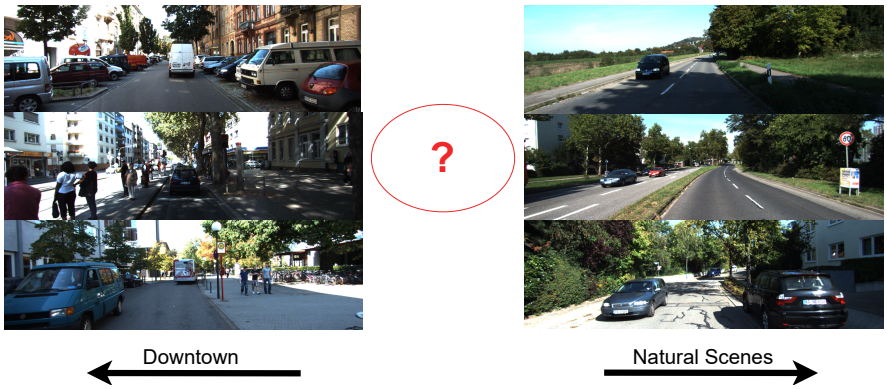


Figure 1.1: Images from KITTI Vision Benchmark Suite(Geiger et al., 2012).

The distribution of points in the LiDAR point cloud exhibits a notable imbalance between foreground and background elements. Analysis of the KITTI training dataset reveals an average of 19,047 points per scene within the camera's field of view, with approximately 1,382 points (about 7.25%) falling within 3D bounding box annotations, indicating foreground objects. Similarly, examination of the KITTI validation set shows an average of 18,888 points per scene in the camera field of view, with around 1,641 points (approximately 8.69%) classified as foreground within 3D bounding box annotations (Hahner, Dai, et al., 2020). This observation highlights a class imbalance between foreground objects and the background scene. According to (Lin et al., 2017), class imbalance poses several challenges. Training efficiency is compromised as a majority of locations represent easy negatives, providing minimal learning signals. The prevalence of these easy negatives can dominate the training process, resulting in the development of imbalanced models. Figure 1.1, shows a brief look into the diversity of data in SemanticKITTI. It is inferred from the figure that the data in SemanticKITTI is mostly concentrated around two main types of scenery, i.e. either downtown scenes or natural scenes. So there lies a question of how to close the gap concerning class imbalance in

State of the Art (SOTA) datasets like SemanticKITTI that is used as a base for semantic understanding of HAD system benchmarking (PapersWithCode, 2023). Because of the interactions between vehicles driven by humans, the chances of accidents are low. As a result, the dataset collected in an open-world environment may not represent complete possible scenarios. Such conditions lead to faulty tests of HAD systems. For example in 2022, Cruise Robotaxis had 546,492 driverless miles in California and even though the number of miles driven by the autonomous vehicles of Cruise in California for 2023 was a total of 583,624 miles with 510 Vehicles (Department of Motor Vehicles, 2023), accident occurred on February 10, 2023. A woman was struck by a human-driven Nissan at a red light, which caused her to fall under the passenger side of the autonomous Cruise Robotaxi that was moving parallel to the Nissan. When the cab attempted to pull over to the edge of the road, it dragged the woman a few meters instead of stopping instantly (Herger, 2024). Even though the Cruise Robotaxis had driven millions of miles in the real world as HAD system, such a horrific situation occurred due to the result of a scenario that was never seen by the Cruise Robotaxis. Enhancing the correlation between foreground and background circumstances is imperative for achieving robust machine vision capabilities. This necessity is underscored in (K. Y. Xiao et al., 2020), which highlights the significant impact of backgrounds on the performance of neural network object detectors. The validation of simulation-based training for DNN intended for real-world driving applications remains incomplete, revealing notable reality gaps (Prabhu et al., 2023) that impede its efficacy. Neural networks play a crucial role in addressing the class imbalance issue by extracting foreground objects and recombining them with background scenes. However, a central challenge arises from the necessity of a flawlessly trained neural network to accurately discern and extract all foreground objects. Moreover, there is also a need to inject objects into the dataset that are unrecognizable by any existing neural network. Deficits of the variety of scenarios (positions, behavior or pose, the appearance of pedestrians, and also the scenery like the background or surroundings of pedestrians) necessitate a new method of scenario generation that represents the authentic world better than the synthetic simulations and can be employed to evaluate the HAD systems effectiveness and harmlessness.

1.2 Applications

Annotated data is necessary to apply deep learning technology. Deep neural network models are trained, validated, and tested using the known labels. A predominant strategy for mitigating the workload associated with data annotation involves the generation of synthetic data through environmental simulations. This

approach exhibits potential by streamlining the acquisition of all ground-truth labels. Even though the performance of the object detection models is improved (Johnson-Roberson et al., 2017), there still exists a domain disparity among the simulated and real-world data (Prabhu et al., 2023). There exists a question about the fidelity of sensors in a simulation involving LiDAR because of reasons such as unreturned pulses, multiple echos from the environment, retroreflectors, spurious returns, etc (Manivasagam, Bârsan, et al., 2023). As a result of these factors, achieving high-fidelity modeling of simulators involving LiDAR sensors remains unattained. These factors have prompted the utilization of simulations, such as Model-in-the-Loop (MIL), Software-in-the-Loop (SIL), Hardware-in-the-Loop (HIL), Vehicle-in-the-Loop (VIL), etc during the early phases of HAD system development and research (Riedmaier et al., 2018). Another strategy to deal with limited annotated data is data augmentation (Fang, Zuo, et al., 2021). We ask the research question :

- How to address the class imbalance issue i.e. between foreground objects and background scene?

- How to mitigate the deficiency in scenarios in the context of HAD?

1.3 Contributions

This thesis outlines a methodology for generating concrete scenarios utilizing available LiDAR data. Unlike approaches that operate at a higher level, such as simulation, for scenario generation and testing, this research emphasizes a lower-level approach, specifically augmenting point clouds through LiDAR augmentation techniques.

Figure 1.2 illustrates current methods employed in augmenting LiDAR Point Cloud (PCD), where blue-colored blocks represent existing techniques. These methods predominantly rely on annotated datasets, Semantic ML models, or pre-existing CAD models for extracting Ground Truth (GT) objects (foreground objects), as indicated by the red-colored blocks within the "GT Object Source" block. The green-colored blocks in the figure signify the approaches adopted in this thesis for scenario generation. Regardless of certain augmentation methods (Ren et al., 2022; J. Lee et al., 2023), this thesis does not require prior knowledge of the characteristics of the LiDAR in the target dataset to be augmented. Instead, it leverages the distribution of the target point cloud in the augmentation process to accommodate these characteristics. A distinctive contribution of this work involves leveraging 3D geometric features of points within a local neighborhood

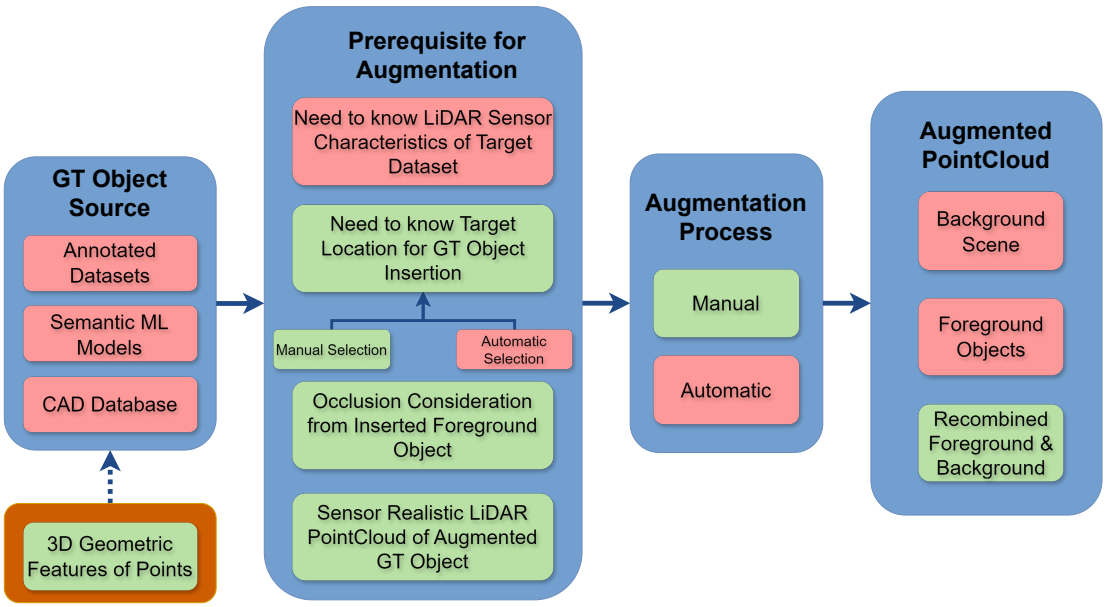


Figure 1.2: Augmentation process of Scenes by injecting extracted Point Cloud Prototypes.

for the extraction of ground truth objects from a scene, depicted by the dark orange-colored block on the far left side in figure 1.2. Implementing this method has the potential to enhance dataset diversity by injecting foreground point cloud objects into scenarios lacking foreground elements, thus addressing the class imbalance issue. Importantly, this approach eliminates the necessity for collecting and annotating new LiDAR data for scenario generation.

2 Basics

2.1 LiDAR

LiDAR refers to Light detection and ranging (Laser imaging, detection, and ranging). It can measure an object's range, angle, material reflectance, and, in certain situations, radial velocity in relation to the sensor. Numerous fields, including oceanography, meteorology, security, augmented and virtual reality, mapping, and autonomy, employ it.

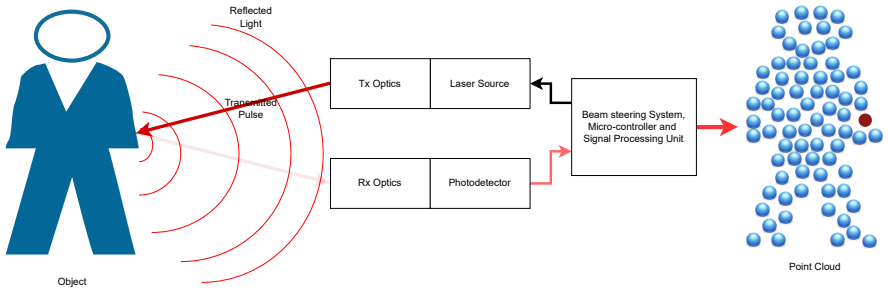


Figure 2.1: Operating principle of LiDAR.

Laser is an electromagnetic radiation similar to that of radio frequency waves. Lasers are far shorter in wavelength than radio frequency waves, making it simple to use lenses to collimate them into narrow beams. This greatly improves LiDAR's resolution, with less than 2cm accuracy in the case of Velodyne HDL-64E LiDAR (Hyper-Tech, 2014), over MIMO imaging radar and reduces the difficulty of estimating the direction of arrival. As shown in figure 2.1, laser light is transmitted from the laser source, which is directed by the transmitter optics. The laser ray hits an object, reflecting the laser depending on the properties of the surface. The reflected light is captured by the photodetector system. The time of flight is calculated based on the time between the sending and receiving of the pulse. Based on the time of flight and speed of the light, the range is determined. For a round trip time t, with the velocity of laser light c, the distance d between the

object and the laser source is determined as shown in equation 2.1.

$$d = \frac{c \cdot t}{2} \tag{2.1}$$

For Speed of light(Approx.) $= 3 * 10^8 m/s^2$ and TOF $= 10ns = 10 * 10^{-9} second$. We can use equation 2.1 to find the distance between the laser source and the target point. Using the formula we get, the distance between the laser source and target = 0.15 meters. So it means that the distance between the laser source and the target hit point is 0.15 meters. By using the orientation of the LiDAR sensor, the exact position of the hit point is calculated. For example, if the LiDAR sensor was oriented along the x-axis, then the coordinate of the hit point would be $(x, y, z) = (0.15, 0, 0)$. Finding each point from the environment gives a point cloud representation of the environment. Data from LiDAR sensors are stored in the form of binaries. The point cloud from the LiDAR can also be represented in formats such as LAS/ LAZ, PLY, ASCII, BIN, text, etc.

2.2 Testing Highly Automated Driving Systems

Since autonomous driving systems are getting more complicated, they need to be extensively examined before being released for public use. Safeguarding the passengers' safety within an autonomous driving system presents a formidable challenge. Throughout the whole phases of development, including functional engineering and testing, integration of systems and validation, test drive and validation, etc., a comprehensive profile for testing of autonomous vehicle procedures is critically needed. Figure 2.2 shows pipelines in autonomous driving development and testing.

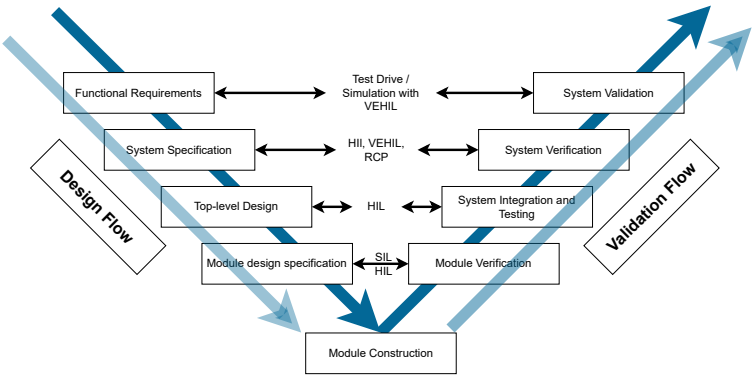


Figure 2.2: Design and Testing pipeline for Autonomous vehicles, adapted from (Huang et al., 2016).

Various steps involve from describing functional requirements to development, validation, and verification of the requirement. Model-in-the-Loop (MIL), Software-in-the-Loop (SIL), Hardware-in-the-Loop (HIL), Vehicle-in-the-Loop (VIL), or mixed simulation techniques are used to guarantee HAD systems functionality, dependability, and safety.

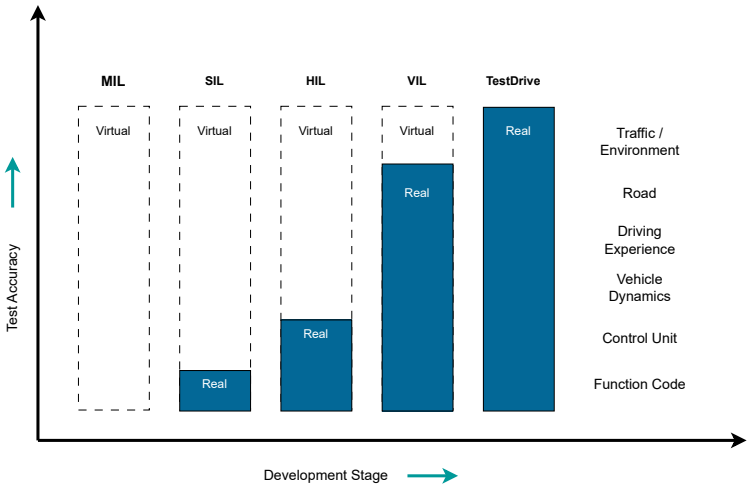


Figure 2.3: X-in-the-Loop (XIL), adapted from (C. Park et al., 2020).

A comparison of various tests concerning their reliability and development lifecycle can be viewed in figure 2.3. Model-in-the-Loop (MIL) is a highly helpful tool for rapidly creating and evaluating algorithms during the prototyping or initial test stage. Pseudo code that is ready to be implemented on the HAD system needs to be tested. Instead of testing directly on the HAD system, which might be hazardous, the Software-in-the-Loop (SIL) approach is carried out. In SIL, only the pseudo-code that is to be implemented in the actual automotive controller in the HAD system is not synthetic, other domains such as Electronic Control Units (ECUs), Vehicle Dynamics, Driving experience, road, traffic, and environment are all implemented in simulations. Numerous situations can be checked quickly because trials can be simulated more quickly than in real life. Hence, MIL and SIL are executed at the initial development phase and also have lower test accuracy in comparison to real-drive testing. Based on environmental data gathered by LiDAR, radar, stereo cameras, and other sensors, the Electronic Control Units (ECUs) controller manages the vehicle's supervisory control. With the gathered data, ECU needs to be able to understand the surrounding environment where it acts as a perception unit. It might need to be able to make decisions about safety rules, traffic laws, etc where it supports decision-making. It might also support on sending control signals to the actuators of the HAD system. An HAD system comprises multiple ECUs each focused on specific tasks, and this number only rises when pursuing towards higher level of autonomy in the HAD system.

In a lab setting, Hardware-in-the-Loop (HIL) verifies the real ECU design and operational characteristics at the hardware level. Testing multiple ECUs in a synthetic environment and vehicle makes the development and verification phase of ECUs faster. Vehicle-in-the-Loop (VIL) is the testing of a real vehicle in the real world but with a virtual environment. Traffic around a vehicle in the real world is simulated in VIL. It can be utilized for the evaluation of the HAD system behaviors under various critical situations without the threat of collision or hazards. Test drive comes at the final stage of the development phase. The actual vehicle is tested for the verification of the complete system. The reliability of the HAD system can be emphasized by real drive testing. During real-drive testing, an Advanced Driver Assistance Systems (ADAS) system can get feedback from the takeover or by the engagement of the driver. This makes the process ever-learning, which improves the reliability factor of the HAD system under various scenarios.

Real-world validation of autonomous vehicles through real-drive testing is a time-consuming endeavor, as evidenced by the findings presented in (Kalra and Paddock, 2016), which suggest that it would take approximately 400 years of continuous driving by a fleet of 100 autonomous driving (AD) vehicles, each traveling at an average speed of 25 miles per hour, to achieve a statistically significant comparison with human driver fatality rates, with a confidence level of 95%. However, the advent of sophisticated simulation environments developed by industry leaders has provided a promising alternative. According to a Waymo blog post (TheWaymoTeam, 2020), their vehicles accrue over 100 years of simulated driving experience every single day. Despite these advancements and accumulation of significant experiences, incidents such as those reported in (Peña, 2024) involving Waymo vehicles raise concerns about the existence of the lack of tested scenery. Domain gaps (Manivasagam, Bârsan, et al., 2023) between real-world scenarios and those replicated in virtual environments is also a major concern. Limitations in accurately mimicking real-world landscapes and scenarios within virtual simulations may contribute to such discrepancies.

2.3 SemanticKITTI Dataset

It is an extensive dataset based on laser technology to advance semantic segmentation research in the field of autonomous driving. Comprehensive point-wise annotations are available for every sequence in the KITTI Vision Odometry Benchmark dataset (Geiger et al., 2012). The public dataset was created by gathering real-world data to meet the rigorous requirements for functional testing and improvement of autonomous driving vehicles. The dataset contains rural roads, residential regions, freeway scenes, and inner-city traffic in Karlsruhe, Germany.

For 28 classes; such as person, car, road, and vegetation; point-wise annotation is offered. This is the current state-of-the-art dataset for 3D semantic segmentation. The current SOTA 3D semantic models for autonomous driving can also be viewed at (PapersWithCode, 2023), which are also compared using the SemanticKITTI dataset. About more than 1700 hours (Behley et al., 2019b) was required by a group of annotators to label and verify the labeling of the benchmark dataset. Though collected in a variety of different scenes, it does not contain all the variation of data.

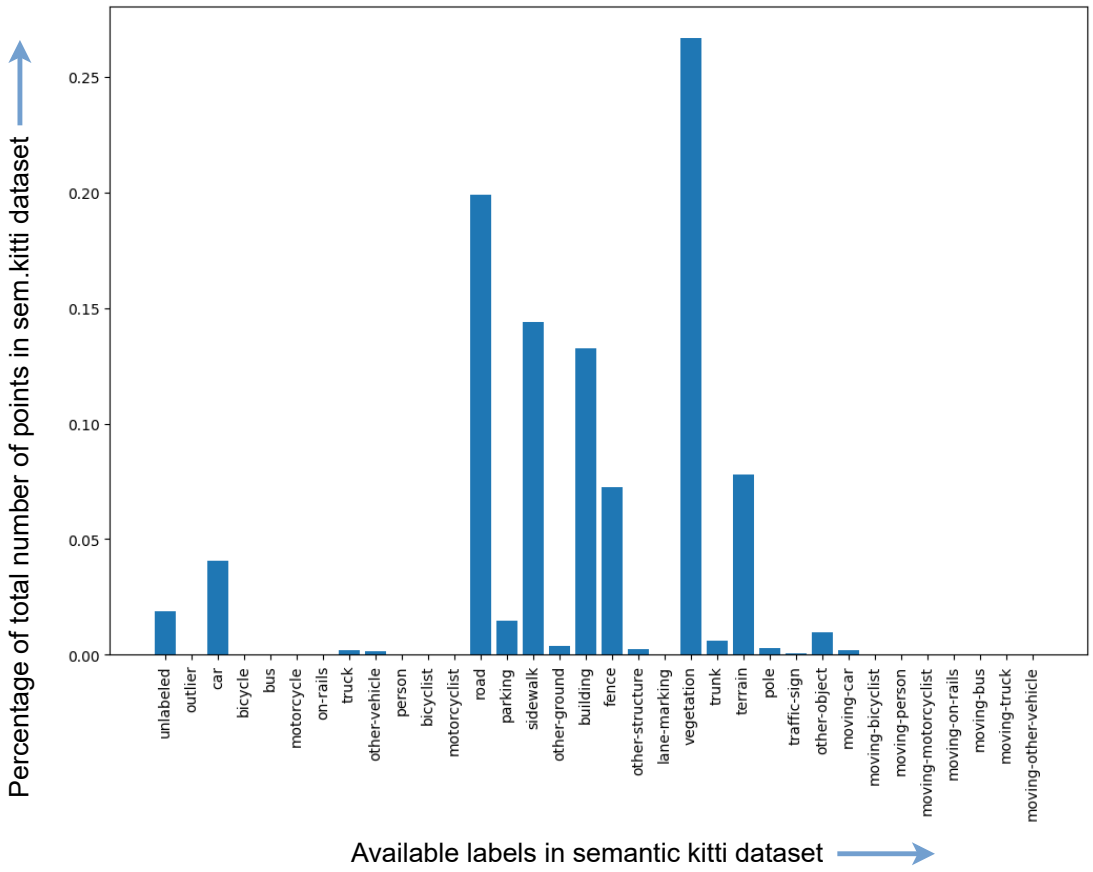


Figure 2.4: Class distribution in SemanticKITTI Dataset in comparison with complete dataset points.

Figure 2.4 shows variants of classes in comparison with total labeled classes available in the SemanticKITTI dataset (Behley et al., 2019b; Behley et al., 2019a). From the figure, it can be visualized that vegetation is the most labeled class, after that are roads, sidewalks, buildings, fences, etc. The number of labeled points for classes like person, car, trucks, etc is relatively low in comparison with classes like road, building, fence, vegetation, terrain, etc. The dataset contains scenes with either a higher density of natural scenes or scenes of downtowns, as shown in figure 1.1. For a HAD system, any situation can occur when navigating

in an environment such as across a river, along a mountain trail, through a desert, alongside an under-construction region, or through a huge mass of people due to demonstration, etc. Such scenarios are not made available in the SemanticKITTI dataset as the data was collected in a controlled setting in Karlsruhe, Germany. This presents a gap in the dataset to represent scenarios to a maximum extent and hence only represent the real world in fragments. Also since the performance of LiDAR is also affected by environmental factors (J. Park et al., 2023), the dataset is not enough to represent the real world to a maximum extent in the HAD context.

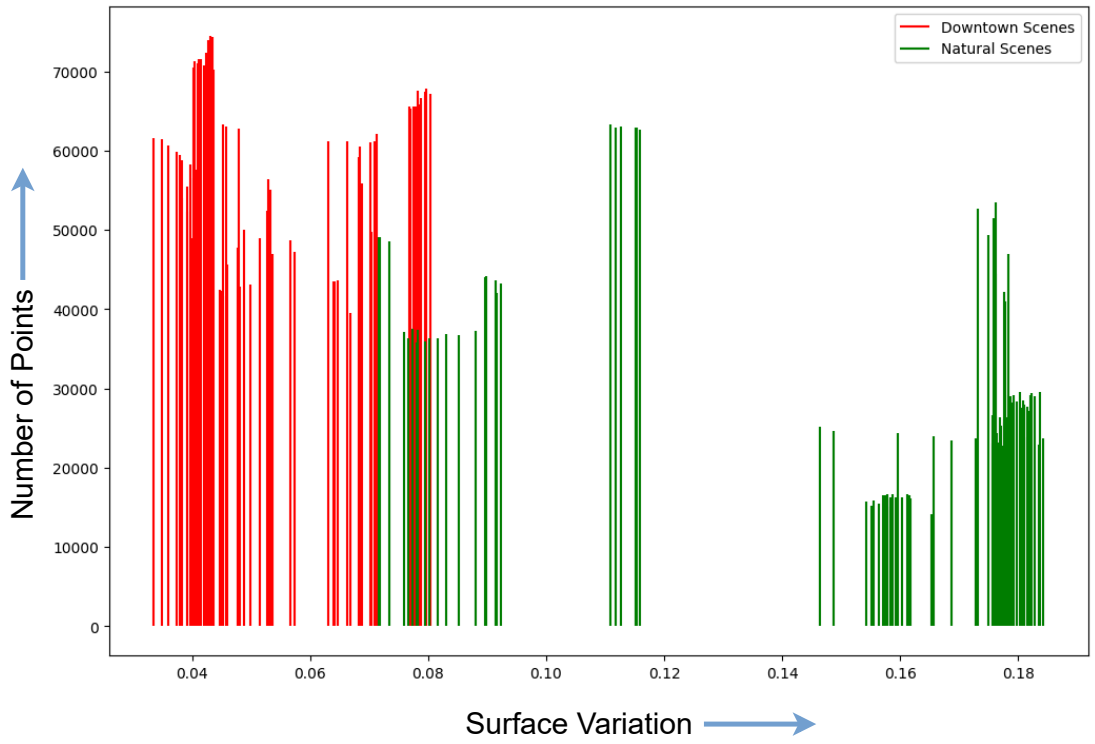


Figure 2.5: Distribution of SemanticKITTI Dataset.

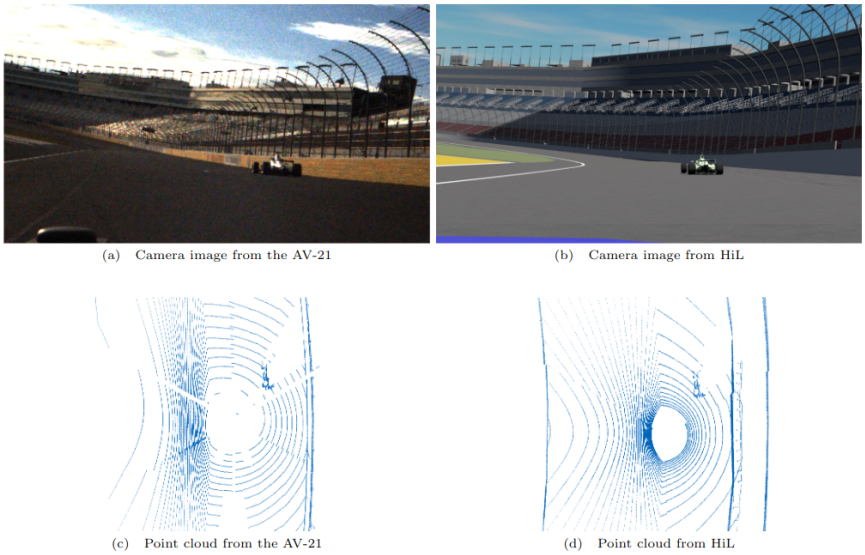
From a randomly selected 100 LiDAR sequences from the SemanticKITTI dataset, point clouds for downtown scenes were calculated by accumulating points for buildings, fences, vehicles, parking, etc, unlabeled points, outliers, roads, vegetation, trunks, terrain, other-objects points were not taken to be the points for the downtown scene for the calculation in figure 2.5. For the point clouds of natural scenes; vegetation, trunk, and terrain point clouds were combined. Point cloud aggregation is performed independently for downtown scenes and natural scenes categories within each sequence. Using hybrid search in open3d (Q. Y. Zhou et al., 2018) with a sphere of radius 30 cm and maximum nearest neighbor of 6 for nearest neighbor search, surface variation was calculated using the equation 2.20. The average was taken after dividing the sum of the surface variation of each

point by the total number of points for a scene for either downtown or natural scenes separately. Figure 2.5 shows the distribution of geometric features (surface variation) of the dataset for each scene. The red lines represent downtown scenes and the green lines represent natural scenes. It can be analyzed that the dataset is not evenly distributed when comparing the geometric features. The available SOTA dataset does not represent diverse scenes and it is risky, expensive, and not usually reproducible to physically test HAD vehicles on public roads. Thus, rather than doing real-world testing, simulations and other methods are opted for the initial testing phase.

2.4 Simulations

Collection of large annotated datasets like SemanticKITTI requires extensive labor work(Behley et al., 2019b). The situation becomes considerably more problematic when sparse LiDAR data is included. In order to prevent overfitting and enhance neural networks' capacity for generalization by intentionally increasing the number and variety of training data, simulations are used. Simulation are very useful in generating a variety of scenarios which is risky, costly, and rare to get from the real world. However, the variation and unpredictability of real-world situations, such as interactions between automobile drivers, pedestrians, and environmental elements, may not always be adequately captured by simulations. A variety of simulators are currently available for testing autonomous vehicles, including CARLA, DeepDrive, Nvidia Drive Sim, Vista, VI-WorldSim, and others. (Yueyuan Li et al., 2023) provides a detailed analysis of various simulators and identifies CARLA as the sole open-source, fully functional simulator that is still being maintained. As described in (Yueyuan Li et al., 2023), the examination of synthetic and real-world images within CARLA highlights the existence of disparities between simulated settings and reality. Additionally, it also discusses the shortcomings of current weather simulations, which frequently only concentrate on visual effects while not including comparable noise disturbances to sensors like LiDAR, radar, etc. There are also issues with how accurately car dynamics are simulated; numerous physics models only behave reliably when the parameters of steering angle, acceleration, and velocity stay low (Kong et al., 2015).

Data captured on a series of Indy Autonomous Challenge 2021 (Sauerbeck et al., 2023) shows the difference between the real environment and synthetic simulation. The assumptions that underpin virtual sensor models do not always stand valid in practical situations. Content and Appearance gap between the synthetic and real-world is also discussed in (Prabhu et al., 2023). For high accuracy, we need simulation with a low domain gap concerning the real world. It is difficult to model



(a) Camera image from the AV-21 (b) Camera image from HiL

(c) Point cloud from the AV-21 (d) Point cloud from HiL

Figure 2.6: Real world vs Simulation difference (Sauerbeck et al., 2023).

HAD simulators involving LiDAR because of the following reasons (Manivasagam, Bârsan, et al., 2023).

- Unreturned pulses because of pulse's power decay.
- Multiple echoes from the environment resulting in multiple peaks in the returning signal.
- Spurious returns due to volume scattering, blooming, beam divergence, multi-echo, etc.
- Noisy points because of randomness in the real world, retro-reflectors.
- Rolling shutter and motion blur in moving vehicle.

2.5 Scenario-Based Testing

According to (Ulbrich et al., 2015), a scene is a description of a particular moment in time of the environment that includes the scenery, dynamic aspects, self-representations of all the performers and spectators, and the interactions between those entities. A scenario explains how various scenes in a series of sequences

of scenes evolve over time. Three distinct scenarios are defined according to (Menzel et al., 2018). They are functional scenarios, logical scenarios, and concrete scenarios.

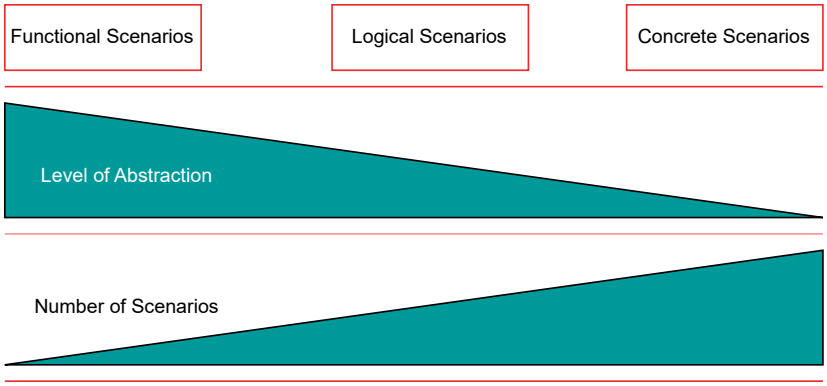


Figure 2.7: Three different scenarios, adapted from (Menzel et al., 2018).

As shown in figure 2.7, the level of abstraction decreases with an increase in several scenarios from functional to concrete scenarios. Functional scenarios come in the concept phase where the scenarios can be described by human experts in the natural language. A logical scenario appears in the system development phase where the scenario is explained with various parameter ranges. Concrete scenarios are represented for the test phase where the concrete state values are defined to guarantee scene replicability and facilitate testing techniques. For describing the parameters of the logical and concrete scenario, a five-layer model is explained in (Bagschik et al., 2018), as shown in figure 2.8.

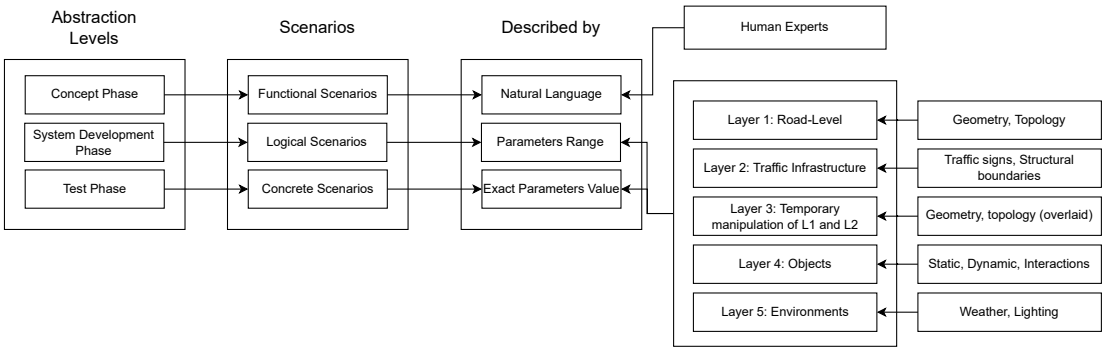


Figure 2.8: Abstraction levels, scenarios, and explanation terms. Adapted from (Bagschik et al., 2018) and (Menzel et al., 2018).

To arrange all the data in a database of knowledge to represent scenes, a model with multiple layers is explained as illustrated in the figure 2.8. The topology and markings of the road are described in the first layer. The model's hierarchical

ideas describe how to construct legitimate road fragments for traffic scenes by assembling geometries that represent lanes. Road-level traffic infrastructure is added in the second layer. Not every stationary object, such as vegetation, is a part of the infrastructure for traffic. Classes that reflect the marking, routing, and security requirements for building sites are included in the third layer. In the ensuing traffic scenarios, the modifications made to the original layout are identified as manipulation. The fourth layer models every object that may not be included in the transportation infrastructure. It is possible to arrange both stationary and mobile objects without significantly altering the connections within the traffic infrastructure. Participants in traffic are divided into object classes, such as vehicles, trucks, bicycles, etc. Environmental conditions like weather and lightning are considered on the fifth layer. Additionally, the impact on the interactions amongst traffic participants is taken into account, leading to varying parameter ranges for relative parameters.

With the help of the scenarios database and knowledge, new scenarios are generated. The information content and level of abstraction in each generated or extracted scenario categorize the scenario types within the Operational Design Domain (ODD). The scenarios database is created with expert knowledge or by varying the parameter range and values. As described on (Bagschik et al., 2018), concrete scenarios are selected based on a testing-based approach or falsification-based approach. In the testing-based approach, range sampling and distribution sampling are done from a logical scenario database, and concrete scenarios are extracted. Based on macroscopic and microscopic assessment, a scenario is selected. A substantial quantity of data needs to be made available to make a macroscopic (statistical) assessment regarding the overall effect of autonomous vehicles on traffic. Specific traffic scenarios are put to the test and assessed. Microscopic evaluation is the assessment of a single scenario. The purpose of the falsification techniques is to locate counterexamples that break safety regulations while doing microscopic evaluations. Based on criticality, complexity, etc concrete scenarios are selected from the scenario database which undergoes microscopic assessment. Scenarios could be samples from the accident database or by changing the criticality of available concrete scenarios. Increasing the scenario's complexity can likewise raise the likelihood of discovering a counterexample. The chosen specific scenarios can be executed in a variety of testing environments. These can be carried out using X-in-the-Loop (XIL) simulation (Riedmaier et al., 2018), which offers a higher level of virtualization, or in the actual world through field or proving-ground experiments. Since XIL offers numerous benefits in terms of, for example, expenses, risks, and safety, nearly all references employ simulation as part of their proof of concept. Scenario-based testing involves organizing circumstances relevant to the testing system in a systematic manner. Implementing a derived logical scenario necessitates keeping irrelevant circumstances constant for a specific test argumentation while adjusting relevant ones through the execution of concrete

scenarios. Achieving this demands positioning prototypes such as persons within the scene in a controlled manner, which is impractical in reality and feasible in simulation, though without valid sensor data.

2.6 Enhancing Methods for Testing LiDAR-based Functions

Enhancing LiDAR-based functions involves improving the capabilities and performance of LiDAR systems for various applications. The different methods used to improve the LiDAR-based functions are described in below points :

- **LiDAR Data Augmentation :** Capturing data representing diversities of scenes in the real world is expensive and laborious. Simulations are also utilized in addition but because of the domain gap between the real and synthetic world, the captured data do not accurately represent the real world environment. LiDAR data augmentation could be approached where data within the LiDAR is changed like scaled, transformed, rotated, etc. Combining synthetic data of objects with real-world point clouds is also an option.

- **Sensor Data Fusion :** The system's reliability is increased by integrating LiDAR data with information from other sensors, including cameras, radar, and GPS, which also improves perception accuracy and redundancy. Data from the camera sensor can be mapped to the LiDAR point cloud to make use of a robust image segmentation model for the localization of objects in the point cloud. (Vora et al., 2019) make use of segmentation information obtained from the image detection algorithm to "paint" the points of a 3D point cloud. In the context of integrating data from various sensors, challenges such as ensuring the synchronization of data streams must be addressed.

- **Security testing and Attack Simulations :** Testing LiDAR systems for possible security risks like jamming or spoofing guarantees their robustness and resilience when used in real-world scenarios. Adversarial scenario simulation is used to evaluate LiDAR performance under several attack vectors, which aids in finding weaknesses and strengthening system defenses.

- **Resolution Enhancement and Noise Reduction :** Post-processing techniques like interpolation or the use of multi-beam LiDAR may be approached to enhance the resolution of LiDAR data. Accurate object detection and mapping can be improved by using point clouds with higher resolutions,

which capture finer aspects of the scene. Different techniques can be employed to remove noise from LiDAR point clouds to make the data more accurate and clear. Strategies like Gaussian filtering and statistical outlier elimination can assist in lowering noise brought on by flaws in the sensor or environmental influences.

- **Use of Contextual Information :** Using additional environmental cues to enhance the comprehension of the 3D point clouds is an effective way to employ contextual information in LiDAR data analysis. LiDAR-based algorithms can achieve better scene understanding and object recognition by adding contextual information such as spatial interconnections, object interactions, and scene interpretation. Contextual indications, such as building structures, traffic signs, and road markings, can be particularly helpful in determining the presence and placement of things like cars, pedestrians, and other obstacles. Contextual data also makes LiDAR systems more resilient and reliable in real-world applications like autonomous driving, urban planning, and environmental monitoring by allowing them to adjust dynamically to changing environmental conditions. Local neighborhood features of points play a vital role in understanding the context.

2.7 Statistical Computation of LiDAR Point Cloud Descriptors

2.7.1 Mathematical Operations in 3D Space

2.7.1.1 Scalars and Vectors

A scalar is a quantity that has only magnitude or size without any associated direction. Scalars are often employed to quantify quantities like mass, temperature, energy, speed, or time since they are represented by a single real integer. A quantity having both magnitude and direction is called a vector. An ordered collection of real numbers, or components, and a direction in space are used to represent vectors. Vectors are commonly depicted in 3D space as arrows that point from the origin to a designated location in the space.

In the figure 2.9, the volume of the cuboid represents the scalar quantity, and the black arrow pointing to a red point in the cuboid represents a vector. Scalars can be added, subtracted, multiplied, or divided just like ordinary numbers. Various

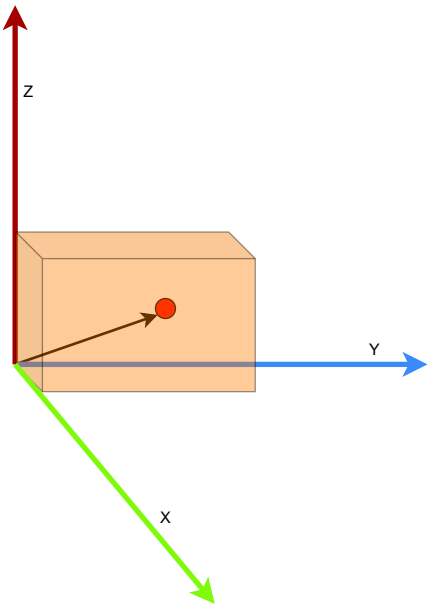


Figure 2.9: A 3D Cube in XYZ Coordinate System.

mathematical operations can be performed on vectors like scalar multiplication, vector addition, dot product, cross product, etc. Equation 2.2, 2.3, 2.4, 2.5, 2.6 gives a general idea of calculating the angle between two vectors using the vector operations.

- **Dot Product**:

$$\mathbf{p} \cdot \mathbf{q} = p_x \cdot q_x + p_y \cdot q_y + p_z \cdot q_z \tag{2.2}$$

- **Cosine of Angle between two vectors**:

$$\cos\theta = \left(\frac{\mathbf{p} \cdot \mathbf{q}}{|\mathbf{p}| \cdot |\mathbf{q}|} \right) \tag{2.3}$$

- **Cross Product**:

$$\mathbf{p} \times \mathbf{q} = (p_y q_z - p_z q_y, p_z q_x - p_x q_z, p_x q_y - p_y q_x) \tag{2.4}$$

- **Sine of Angle between two vectors**:

$$\sin\theta = \left(\frac{|\mathbf{p} \times \mathbf{q}|}{|\mathbf{p}| \cdot |\mathbf{q}|} \right) \tag{2.5}$$

- **Angle between two vectors**:

$$\theta = \arctan\left(\frac{\sin(\theta)}{\cos(\theta)}\right) \tag{2.6}$$

2.7.1.2 Translation, Rotation, and Transformation

Moving an object from one place to another in three dimensions without modifying its shape or orientation is referred to as translation. It entails moving every point on the item in a certain direction by a predetermined amount. In three-dimensional space, rotation is the process of shifting an object's orientation around a designated axis or point. It involves rotating the object by a certain angle about the selected axis or point. A combination of translation, rotation, scaling, and other operations performed on an object is referred to as transformation in 3D space. It enables one-step changes to the object's size, location, and orientation. The operations are represented by the equation 2.7, 2.8, 2.9. Where (x,y,z) represents a point in cylindrical coordinates and T represents any transformation matrix.

- **Translation**:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x+a \\ y+b \\ z+c \end{pmatrix} \tag{2.7}$$

- **Rotation**:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.8}$$

- **Transformation**:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = T \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.9}$$

2.7.2 Nearest Neighbour Search

A basic issue in mathematics and computer science, especially in the areas of computational geometry and data structures, is nearest neighbor search. A

collection of points P serves as the problem's input. The objective is to locate the point in the set P that is closest to a given query point, q. This necessitates O(n) comparisons naively since we must compare q to every point in P. Using the brute-force technique is not computationally possible when the number of points is very huge. As a result, our objective is to pre-process the input and produce a data structure that can respond to queries rapidly. Distance metric, such as the Minkowski, Manhattan, or Euclidean distances, is usually utilized to determine the separation between two positions, which can be used to create new data structures for effective search. The Euclidean distance, which is the square root of the total squared differences in each coordinate, is frequently used in the context of three-dimensional space. Nearest Neighbor has countless applications and is used in numerous computer science fields, including computational linguistics, machine learning, and image recognition. It's crucial to remember that, despite all recent developments in the field, exhaustive search remains the only technique that can reliably retrieve the precise nearest neighbor. Because of the "curse of dimensionality", finding the exact nearest neighbor is impractical when the dimensionality of the data is large and so an Approximate Nearest Neighbor was introduced. Techniques like Vector Transformation, Vector Encoding, None Exhaustive Search Components were introduced to preprocess the data into an efficient index. Numerous space-partitioning techniques have been created to address the nearest neighbor search issue. The k-d tree (a binary tree), which repeatedly splits the search space into two areas with half of the points of the parent region, is arguably the most straightforward. By analyzing the query point at each split, queries are executed by traversing the tree from the root to a leaf. It might also be necessary to assess nearby branches that could contain hits, depending on the query's distance specification. Parameters, such as radius and maximum number of nearest neighbors, could be adjusted for a search of the nearest neighbor of a query point.

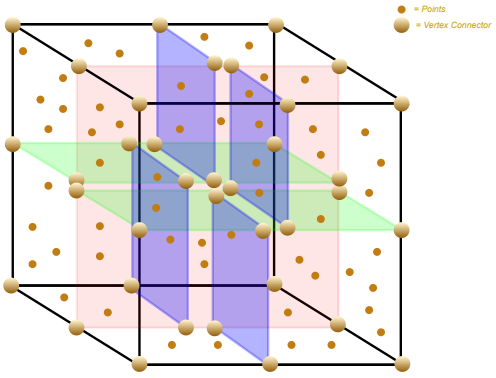


Figure 2.10: 3D Space containing Points with dividing Hyperplanes.

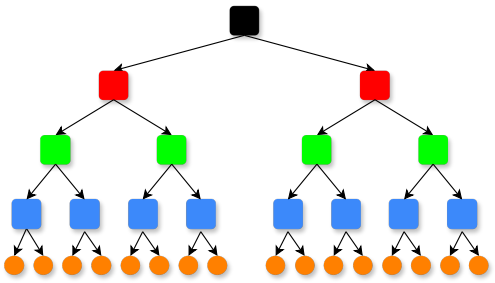


Figure 2.11: Tree structure showing leaf nodes, root nodes.

In figure 2.10, the initial partition (the red vertical plane) divides the root cell into two subcells. The subsequent division (the green horizontal planes) further divides each of those subcells into two subcells, and so on. The splitting plane in the three-dimensional space represents the non-leaf node. All non-leaf nodes can be viewed as inherently creating a dividing hyperplane, or half-space, that divides the available region into two halves. Points to the left of the hyperplane are represented by the subtree on the left side of that node, while points to the right of the hyperplane are represented by the subtree on the right side. Here's how to figure out the orientation of the hyperplane: each node in the tree corresponds to one of the k dimensions, and the hyperplane is oriented perpendicularly to the axis of that dimension. This splitting of 3D dimension into cells can be discussed with the figure 2.11. The square boxes in the figure represent non-leaf nodes. The circular shapes in orange color represent the points in the 3D space. A 3D space is split into two halves by a red plane in figure 2.10. This is represented by the black root node separated into two red-colored child nodes - left and right node. The red plane is further divided into two green planes in the 3D space, which is represented by two green-colored child nodes of red parent nodes. The space separated by green planes is further divided by blue planes, which contain the points. A search of the nearest neighbor is done by traversing the tree from the root to the leaf nodes, where each comparison is done for each dimension of k-dimensions alternatively when traversing the parent nodes.

2.7.3 Variance, Covariance, and Covariance Matrix

The dispersion or spread of a single random variable is measured by variance. The variance of X in 3D space expresses the degree to which the values of X differ from their mean value. The amount that two random variables fluctuate together is measured by covariance.

- **Variance of variable X:**

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2 \tag{2.10}$$

- **Covariance between X and Y:**

$$\mathrm{Cov}(X, Y) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y}) \tag{2.11}$$

- **Covariance Matrix in 3D Spaces:** The covariance matrix C is given by:

$$C = \begin{bmatrix} \text{Var}(X) & \text{Cov}(X,Y) & \text{Cov}(X,Z) \\ \text{Cov}(Y,X) & \text{Var}(Y) & \text{Cov}(Y,Z) \\ \text{Cov}(Z,X) & \text{Cov}(Z,Y) & \text{Var}(Z) \end{bmatrix} \tag{2.12}$$

The equation 2.12 is very useful in giving us insights about the dataset of variables by helping us to understand the relation between variables, in our case in 3D space. It can be summarized in below points :

- Negative covariances show that variables move in the opposite way.
- Positive covariances show that variables tend to rise or decrease together.

It can be utilized to construct a transformation matrix known as a whitening transformation, which, from another angle, enables one to evaluate the best foundation for compactly describing the data or, alternatively, to fully decorrelate the data. Therefore, the covariance matrix can be viewed as a linear transformation that maps vectors from one d-dimensional space to another d-dimensional space. Dimensionality Reduction, financial portfolio optimizations, clustering, statistical inference, and calculation of surface orientations are some use cases from the use of covariance matrix.

2.7.4 Eigen Values and Eigen Vectors

The vectors that a linear transformation works on are rotated, stretched, or sheared. The vectors that are merely stretched; they do not rotate or shear; are known as its eigenvectors. The associated eigenvalue for an eigenvector is the value that influences an eigenvector to change in length or direction. If the eigenvalue is negative, the eigenvector's direction is transformed. Non-zero vectors that, upon application of a linear transformation (expressed by a matrix), are scaled by the associated eigenvalues are Eigenvectors. Eigenvectors depict the directions that a transformation acts in, just extending or compressing; they do not alter the direction. For a transformation matrix "T", the corresponding eigen values and eigen vectors are represented by equation 2.13. If λ_1, λ_2, and λ_3 represents the eigen values of a 3D covariance matrix then a relation can be showed by equation 2.14.

- Eigenvalue (λ) and Eigenvector ($\mathbf{v}^{\rightarrow}$):

$$\text{T}\mathbf{v}^{\rightarrow} = \lambda\mathbf{v}^{\rightarrow} \tag{2.13}$$

- Relation between eigen values for a 3D Covaraiance matrix :

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0 \tag{2.14}$$

By undergoing the process of eigen decomposition using the equation 2.13, eigen vectors and eigen values can be calculated from the covariance matrix. Principal components analysis makes use of eigenvectors and eigenvalues, where eigenvectors are associated with the principal components, and eigenvalues explain the discrepancies of the principal components. Some Key features can be described as :

- High value of eigenvector captures directions of maximum variance of data.
- Low value of eigenvector captures directions of minimum variance of data.
- Sum of eigenvalues gives the total variation of points from the center of gravity.
- For n-dimensional data, at most n eigenvalues and corresponding eigenvectors is possible.

Selecting a subset of the eigenvectors that represent the maximum variance of data allows for dimensionality reduction while preserving the most important features. It is used in applications like normal calculation, feature extraction, vibration analysis, etc.

2.7.5 Normal Estimation

In computer vision, computer graphics, and geometric modeling, normal estimation is an essential procedure. To accurately recreate the underlying surface geometry, surface reconstruction techniques frequently begin with normal estimation, which involves determining the normals of a point cloud or a group of surface points. When working with scarce or noisy data, normal information helps create surfaces that are accurate and smooth. Normals aid in the identification of sharp edges, the determination of surface features, and the direction of different mesh processing activities. Rendering scenes with accurate normal estimates makes them look more realistic. For robots to navigate challenging environments and interact with items effectively, accurate normal information is necessary.

In figure 2.12, the black curve represents a surface represented by a group of red points. The orientation of the surface in the region bounded by the yellow color

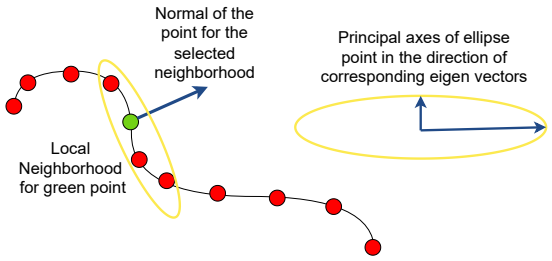


Figure 2.12: Surface represented by points and selection of local neighborhood, adapted from (Platt, 2016).

curve represents the normal of the surface for the yellow-bounded region, this is represented by the direction of the blue arrow. We can find the normal of the surface at a point by the following steps (Platt, 2016) referencing the figure 2.12 :

1. Identify the neighbors for the point (represented by a green-colored point in the figure) through which the normal is to be calculated. The neighborhood is represented by the yellow ellipse.

2. Calculation of Covariance Matrix from the local neighborhood.

3. Calculation of Eigen Values and Eigen Matrix by Eigen decomposition of Covariance Matrix.

4. The Direction of the eigenvector corresponding to the least eigenvalue is the surface normal. (i.e. direction of least variance of the features)

2.7.6 Statistical Point Cloud Measurement

The spatial placement of individual 3D points implicitly represents important geometric information found in 3D point clouds. These attributes should be calculated with particular spatial support because context is not taken into account when considering individual points (Weinmann et al., 2013). Usually, the local neighborhood surrounding each 3D point is retrieved in order to explain this implicit information sufficiently. For example, defining a fixed size sphere around a point in a point cloud could give a local neighborhood where all the points lying in the sphere are the neighbors (I. Lee and Schenk, 2002). The local 3D structure can be described directly by the eigenvalues of a 3D covariance matrix which follows the equation 2.13. Alternatively, additional measures that capture unique geometric features can be deduced based on these eigenvalues. Utilizing

the relation of eigenvalues from equation 2.14, the definition of various features is referenced from (Weinmann et al., 2013).

- Linearity (L_λ):

$$L_\lambda = \frac{\lambda_1 - \lambda_2}{\lambda_1} \tag{2.15}$$

- Planarity (P_λ):

$$P_\lambda = \frac{\lambda_2 - \lambda_3}{\lambda_1} \tag{2.16}$$

- Scatter or Sphericity (S_λ):

$$S_\lambda = \frac{\lambda_3}{\lambda_1} \tag{2.17}$$

- Omnivariance (O_λ):

$$O_\lambda = (\lambda_1 \cdot \lambda_2 \cdot \lambda_3)^{\frac{1}{3}} \tag{2.18}$$

- Anisotropy (A_λ):

$$A_\lambda = \frac{\lambda_1 - \lambda_3}{\lambda_1} \tag{2.19}$$

- Surface Variation (Rusu, 2009) (C_λ):

$$C_\lambda = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \tag{2.20}$$

Various measures like F-score, Gini Index, Information Gain, Pearson correlation coefficient, and ReliefF were considered from the training data for the ranking of features. From the features ranking as shown in figure 2.13, $R_{\lambda,2D}$, V and C_λ are the most prominent features. $R_{\lambda,2D}$ is a robust feature for detecting the exterior of the building, the verticality V is a promising feature in differentiating facades from the ground, and C_λ is a powerful feature in understanding the difference between planar and non-planar structures (Weinmann et al., 2013). Among the prominent features, an in-depth review of surface variation (C_λ) is explained.

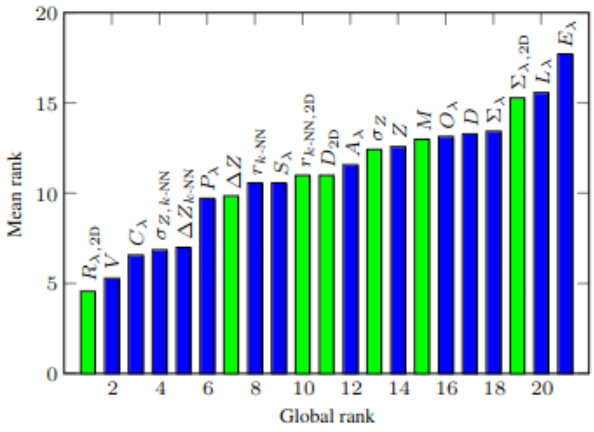
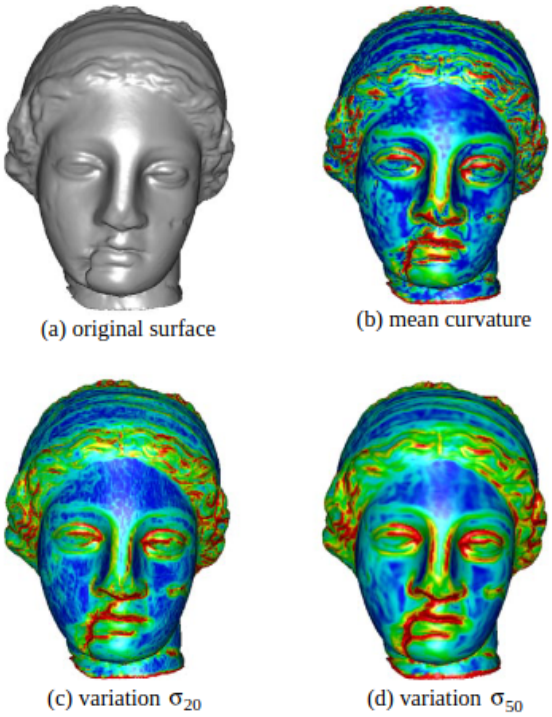


Figure 2.13: Mean ranking of the derived features - 3D by blue and 2D by green (Weinmann et al., 2013).



(a) original surface (b) mean curvature

(c) variation σ_{20} (d) variation σ_{50}

Figure 3: *Comparison of surface curvature and surface variation on the igea model (a). In (b), curvature is computed analytically from a cubic polynomial patch fitted to the point set using moving least squares (see Section 2.2). (c) and (d) show surface variation for different neighborhood sizes.*

Figure 2.14: Visualization of Surface Variation, referenced from (Pauly et al., 2002).

2.7.6.1 Surface Variation

As shown in equation 2.20, surface variation, also known as surface roughness or surface irregularity, is the ratio between the minimum eigenvalue and the sum of eigenvalues. The surface variation of a point centered in a local neighborhood gives the change in curvature. This feature is invariant to scaling. A smaller value of the surface variation for that point indicates that all points in the local neighborhood lie on a tangential plane to the surface. The value of the surface variation of a point is dependent on the selected neighborhood. The selected neighboring point influences the covariance matrix, which influences the eigenvalues and hence the surface variation is varied. If $C_\lambda = 0$, then all the points of the neighborhood are located in the plane. $C_\lambda = \frac{1}{3}$ is considered to be the value of isotropically distributed points (Pauly et al., 2002). Figure 2.14 shows the variation of features for an Igea model. Various applications are possible by the analysis of surface variation such as quality control in manufacturing, surface reconstruction, designing components, etc.

2.8 Surface Reconstruction

The process of turning a collection of discrete points or a point cloud into a digital representation of an uninterrupted surface is known as surface reconstruction. It is essential in computer vision, computer graphics, and geometric modeling because it makes it possible to create 3D models from the real-world data that is acquired by sensors like depth cameras, LiDAR scanners, or other photogrammetry methods. Among various available methods for surface reconstruction, three are discussed.

2.8.1 Alpha Shapes

Alpha shape as explained in (Edelsbrunner et al., 1983), is the generalization of a convex hull. As clarified on (Fischer, 2011), $\alpha - shape$ can be described as follows. Envision a massive ice cream mass forming a region, with the "hard" chocolate chunks represented as points on the region. All the areas of the ice cream mass are carved out that can be reached without running into pieces of chocolate by using ice cream spoons shaped in the form of a sphere. As a result, we even create holes inside the block (i.e., portions that are not accessible by just sliding the spoon from the outside). Eventually, what we have will be an object surrounded by caps, arcs, and points, but not necessarily convex. The $\alpha - shape$ of the points(chocolate

toppings) could be understood by adjusting all the "round" faces into triangles and line segments. α in $\alpha - shape$ represents the radius of the carving spoon, which determines the level of detail of the constructed surface. A higher value of alpha results in larger and simpler shapes with lower details. A lower value of alpha results in shapes with finer details. In figure 2.15, points represent chocolate toppings and circles represent spherical ice-cream spoons in 2D space.

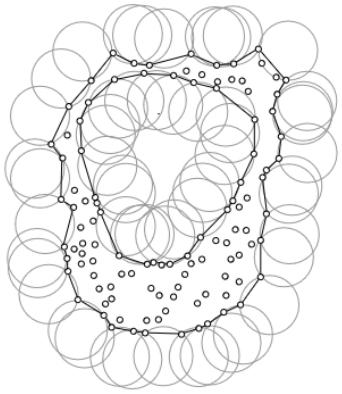


Figure 2.15: Alpha Shapes Surface Reconstruction Method (Fischer, 2011).

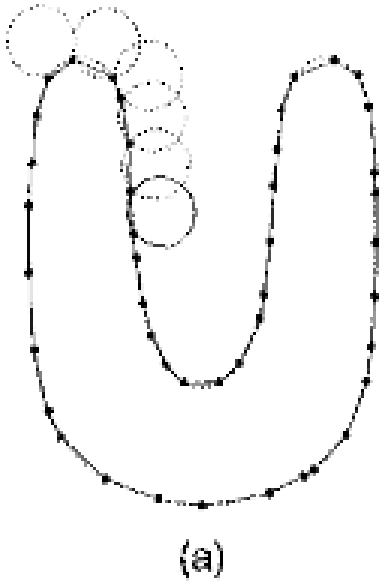


Figure 2.16: Ball Pivoting Algorithm (Bernardini et al., 1999).

2.8.2 Ball Pivoting

Triangle mesh is constructed using the Ball Pivoting algorithm from a set of points (Bernardini et al., 1999). The idea behind the ball algorithm is straightforward. Starting from a single edge of the points in the unstructured points, a ball of a specified radius is moved or pivoted without leaving contact with the edge. A single triangle is formed if the ball touches three points, without considering any other points and without falling through. This process is carried on until all reachable edges have been tried. This process can also be viewed in figure 2.16. Ball size can be varied to handle the varying point densities. For example, a larger-sized ball can be used to handle points that are very sparse.

2.8.3 Poisson Surface Reconstruction

By solving a regularized optimization problem, the Poisson surface regeneration approach produces a smooth surface (Michael Kazhdan and Hoppe, 2006). Because of this, this method for surface regeneration can be superior to the previously discussed techniques, which results in non-smooth outcomes because the points of the raw point cloud are also the location of the vertex of the final triangle mesh without any change.

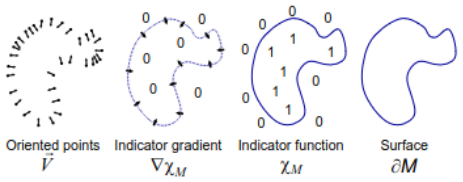


Figure 2.17: Poisson reconstruction illustrated in 2D (Michael Kazhdan and Hoppe, 2006).

Figure 2.18: Poisson Reconstruction illustration (Michael Kazhdan and Hoppe, 2006).

A model's indicator function and the oriented points, as shown in figure 2.17, that are collected from its surface are inextricably linked. In particular, the slope of the indicator function is a vector field that is equivalent to the inner surface normal nearly throughout (because the indicator function remains unchanged in almost every place), except in places that are close to the surface. Consequently, the directed point specimens can be considered instances regarding the model's indicator function gradient. Additionally, the indication function can be seen along the plane shown in the figure 2.18 rightmost image and can be viewed by the middle image of the same figure in black color. Triangle meshes are constructed even in low-point density locations and also infer the surface in such locations. The density value of each point indicates if the neighborhood of the point is crowded by other points. Maximum tree depth for the octree defines the resolution of the final reconstructed triangle meshes.

The reconstructed surfaces represent triangle mesh. For each triangle in the triangle mesh, the centroid can be calculated. Centroid refers to the center of the geometry. It can be thought of as the location where a geometry with an evenly dispersed mass could rest ideally on a needle spike. For $\triangle PQR$ as shown in figure 2.19, point "C" represents the centroid where the line bisecting the surfaces of the triangle intersects. The vertex of the triangle $\triangle PQR$ is represented by $P(x_0, y_0, z_0)$, $Q(x_1, y_1, z_1)$ and $R(x_2, y_2, z_2)$. Centroid $C(x_c, y_c, z_c)$ can be calculated using the equation 2.21

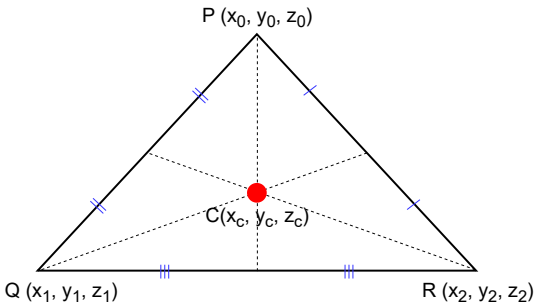


Figure 2.19: Centroid of Triangle.

$$C(x_c, y_c, z_c) = \left(\frac{x_0 + x_1 + x_2}{3}, \frac{y_0 + y_1 + y_2}{3}, \frac{z_0 + z_1 + z_2}{3}\right) \tag{2.21}$$

2.9 Hidden Point Removal Algorithm

The "Hidden" Point Removal operator (Katz et al., 2007)is used to identify the points in a point cloud that are observable when perceived from a specific angle. Even in the absence of recreating a surface or measuring normals, visibility is ascertained. This operator can be used on point clouds of different sizes, on sparse and thick point clouds, and on views both inside and outside the cloud. This algorithm is helpful for shadow casting, view-dependent reconstruction, and point cloud visualization. No point is genuinely hidden since points cannot obscure each other (unless they happen to fall along the exact same beam from the viewer). It is possible to identify which points are visible once a surface has been rebuilt from the points. This suggests that the cloud of points has information inherent to it that can be used to extract the point's visibility. Using the intrinsic information of points in the point cloud, the visibility of the points is determined without the need for surface reconstruction. For a point cloud with n number of points, the time complexity of the algorithm is $O(n \log(n))$ (Katz et al., 2007).

For a point cloud with a "P" number of points, to find the points that are visible from a given viewpoint "C", the HPR operator can be used. This HPR operator consists of two main steps as shown in figure 2.21. First is Spherical Flipping and second is the creation of a convex hull.

2.9.1 Spherical Flipping

This step is also called inversion of points. The point "P" is in the geometry where the viewpoint is the Origin of the geometry. A function that maps a point p_i along the direction of the ray from the viewpoint and is declining in $||\mathbf{p}_i||$ monotonically, results in spherical flipping. It can be given by equation 2.22. Where $\hat{\mathbf{p}}_i$ is the point after spherical flipping of point p_i. R is the radius of the sphere and $||\mathbf{p}_i||$ represents euclidean norm or magnitude of vector p_i.

$$\hat{\mathbf{p}}_i = f(\mathbf{p}_i) = \mathbf{p}_i + 2(R - ||\mathbf{p}_i||)\frac{\mathbf{p}_i}{||\mathbf{p}_i||} \tag{2.22}$$

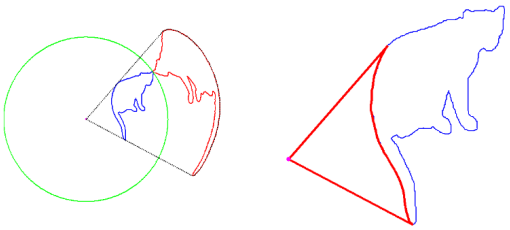


Figure 2.20: Spherical Flipping and Back Projection(Katz et al., 2007).

In the figure 2.20, the green circle represents the Sphere whose center is the viewpoint C or Origin. For a noisy point cloud, the possible alternatives for the radius of the sphere can be found in (Mehra et al., 2010). Blue curves, internal to the sphere, are projected along the ray from the viewpoint of its image external to the sphere. This is represented by the red curve on the left side image of figure 2.20. The right image of the figure represents the back project of the convex hull by the red-colored curve.

2.9.2 Calculation of Convex Hull

In the next step of HPR, the convex hull of the transformed point $\hat{\mathbf{p}}_i$ is calculated. The transformed point cloud and the center of viewpoint are used to calculate the convex hull. Visibility of the point $\hat{\mathbf{p}}_i$ is determined using the location of points respective to the convex hull. According to the algorithm, all the points are visible if the transformed points are located on the convex hull. Selection of the viewpoint C is crucial in the creation of a convex hull because, when C is external to P, points on the object's back may otherwise sit on the convex hull. The two steps can also be viewed in figure 2.21.

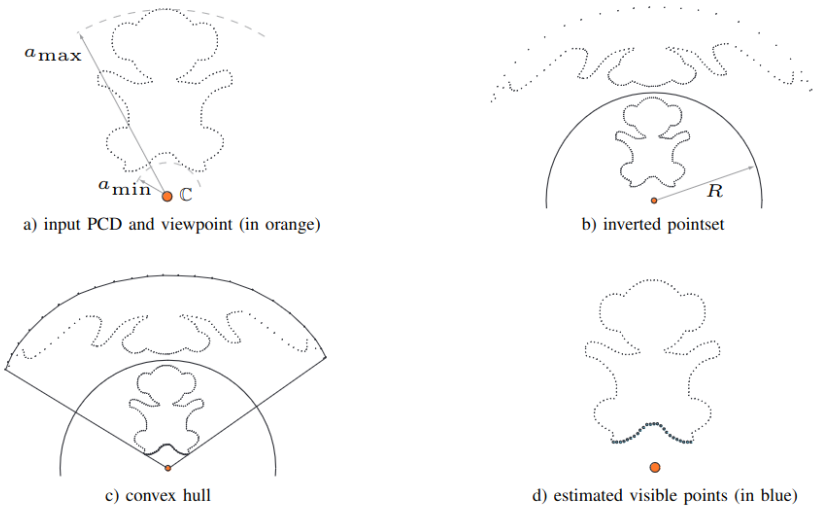


Figure 2.21: Steps in HPR Algorithm (Mehra et al., 2010).

2.10 Performance Metrices

2.10.1 Confusion Matrix

The confusion matrix is the foundation of quality assessment matrices. It is a table that combines the test datasets' actual and anticipated values. Confusion matrices are typically employed for classification models and are used to determine the accuracy and correctness of the model. The projected classifications are in rows, and the actual ones are in columns. It is possible to switch the order of the anticipated and actual columns. In the figure, the positive class refers to the "YES" category, and the negative class refers to the "NO" category.

- **True Positive (TP) :** If the predicted positive value is the correct classification, these are TP.
- **True Negative (TN) :** If the predicted negative value is the correct classification, these are TN.
- **False Positive (FP) / Type I Error :** For any classification task, if the positive class was predicted which is the incorrect classification. These are FP.

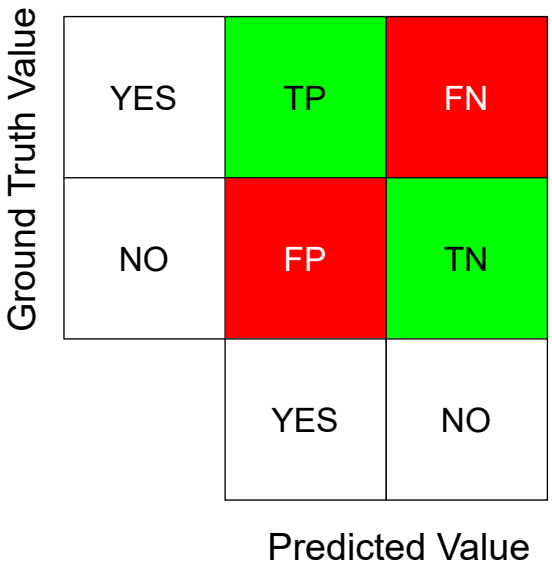


Figure 2.22: Confusion Matrix.

- **False Negative (FN) / Type II Error :** For any classification task, if the negative class was predicted which is incorrect classification. These are FN.

- **Accuracy :** Tells about the percentage of correct predictions made by the model. Accuracy can be utilized as a benchmark to assess a classification model's performance in a balanced dataset; however, if the dataset is imbalanced (i.e., there is a significant actual difference in the number of positive and negative examples) then choosing a classification model based solely on accuracy is misleading.

$$Accuracy = \frac{\sum TP + \sum TN}{\sum TP + \sum TN + \sum FP + \sum FN} \tag{2.23}$$

- **Precision**: The precision of a classifying model indicates the fraction of positively anticipated data that it could correctly identify as positive. Positive Predicted Rate is another name for it.

$$\text{Precision} = \frac{\sum TP}{\sum TP + \sum FP} \tag{2.24}$$

- **Recall**: It provides us with the percentage of real positive cases that our algorithm was able to accurately anticipate. True Positive Rate or sensitivity are other terms for recall. A higher recall score indicates that our algorithm can accurately anticipate the majority of positive classes. A lower recall score

indicates an inability of our model to accurately forecast the real positive examples.

$$\text{Recall} = \frac{\sum TP}{\sum TP + \sum FN} \tag{2.25}$$

- **F1 Score**: A model's accuracy can be determined by calculating the F1-score, which takes into account both the precision and recall of the model's predictions. When working with unbalanced datasets or when the costs associated with false positives and false negatives differ, it is especially helpful. It is the harmonic mean calculated between precision and recall. The best case value of f1-score is 1 and the worst case value is 0. Optimizing recall and precision at the same time increases the value of the f1-score.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.26}$$

Let's discuss the concept of performance metrics with an example. Consider a HAD system that needs to detect between a person and not a person. Out of 100 observations, 97 observations do not contain a person, and the remaining 3 observations contain a person. If predicting a person is a positive class and predicting a non-person is a negative class, then the metrics can be evaluated. If the HAD system predicted all the 100 observations as non-person then we have $TP = 0$, $TN = 97$, $FP = 0$, and $FN = 3$. We get an $Accuracy = 97\%$. Because the dataset is unbalanced, as the number of positive and negative classes is not comparable, considering Accuracy as the determining factor is misleading. So options like F1-Score, Precision, and Recall should be further analysed.

2.10.2 Intersection over Union

Intersection over Union (IoU), also known as the Jaccard Index, is used to compare how similar two irregular shapes are to one another. It is a performance metric that is used to assess how well object detection algorithms perform, especially when it comes to computer vision and machine learning applications. By assessing the regions of overlap between the anticipated and ground truth bounding boxes, an object detection algorithm's accuracy can be determined. It can be calculated using the equation 2.27. IoU is also utilized in Non-maximum Suppression (NMS). It is a post-processing method for object detection that helps to choose the most pertinent detected items by removing duplicate detections. A predicted bounding box with a higher IoU has maximum confidence in detecting the object and based on thresholding the IoU value, suitable predictions are finalized.

$$IoU = \frac{\text{Region of Intersection}}{\text{Region of Union}} \tag{2.27}$$

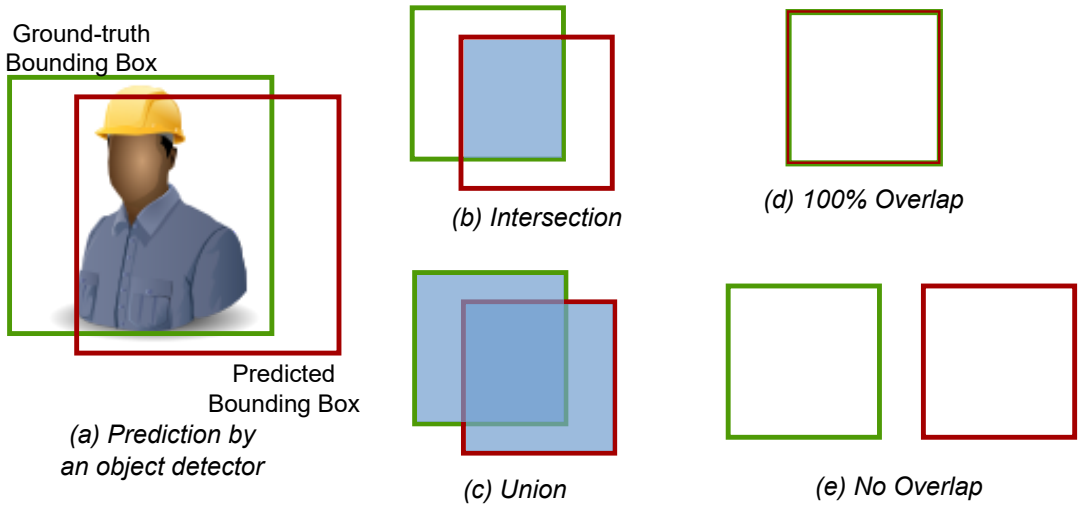


Figure 2.23: Intersection over Union.

Example image (a) in figure 2.23 shows the result of an object detection algorithm. The green colored box is the ground truth and the red colored box is the prediction by the algorithm. By calculating the area of intersection and area of union between the two boxes, IoU can be calculated. Image (d) represents completely overlapped bounding boxes. The IoU in such a case is 1. In case of no overlap between the two regions (image (e)), the IoU is 0. It is necessary to select an accuracy threshold when utilizing IoU as an evaluation metric. Based on the threshold, the result from an algorithm can be either accepted or rejected.

3 Related Work

The Electronic Control Units (ECUs) of the HAD system are in charge of comprehending the environment and making the right decisions. To accomplish high-level tasks like course planning and collision avoidance, the HAD system has to estimate a more detailed 3D bounding box within the real world. The fundamental part of the ECUs that manages perception, judgment, route planning, etc. is the Deep Neural Network (DNN). The quantity and quality of annotated training data are the primary determinants of DNN performance. The main issue that arises while training a DNN based on point clouds is the scarcity of ground truths, which severely restricts the network's ability to converge quickly and perform at its best. More data is required for DNN to reduce the problem of class imbalance, enhance generalization, and prevent overfitting of the model.

One interesting approach that makes it simple to collect all ground truth labels is synthetic data simulation. Utilizing computer graphics techniques, efforts have been made to produce artificially labeled data (Su et al., 2015; B. Wu, Wan, et al., 2018; B. Wu, X. Zhou, et al., 2018). Using the video game engine's APIs from Grand Theft Auto V (GTA-V), (Yue et al., 2018) gathers the calibrated pictures and point cloud and uses this information for vehicle recognition in subsequent projects (B. Wu, Wan, et al., 2018). Simulators like AutonoVi-Sim, CARLA (Best et al., 2018; Dosovitskiy et al., 2017) are also employed for purely synthetic data generation. The Computer Graphics world has a restricted scale and complexity and is primarily created by hand. Although it is demonstrated that using these simulated data might enhance DNN performance, there are still certain issues like domain gaps(Manivasagam, Bârsan, et al., 2023; Prabhu et al., 2023) that need to be resolved.

3.1 Data Augmentation

Data Augmentation (DA) has been extensively investigated for 2D images (Simard et al., 2000; Chapelle et al., 2000) and has shown enormous possibilities in training effective DNN models with restricted training images. DA tries to modify and create new training samples in order to extend the variety of the training data

(A. Xiao et al., 2022). DA has been a standard pre-processing step when training a neural network, as its efficacy has been demonstrated across a wide range of tasks (Fang, Zuo, et al., 2021; Yan et al., 2018; Chen et al., 2020). In (Alhaija et al., 2017), the real-world image dataset is enhanced by the addition of rendered automobiles, which benefits from the foreground's variance and the background's realism. For the 2D image domain, several data augmentation methods are covered. In Cutout (Devries and Taylor, 2017), sections from images are taken off to enhance an image patch. MixUp (H. Zhang et al., 2017) combines two images and the corresponding labels. CutMix (Yun et al., 2019) is a technique that replaces a part of an image with a patch from another image, combining Cutout and MixUp. Though many efforts have been made on image augmentation, very few concentrate on point cloud data augmentation.

(Hahner, Dai, et al., 2020) delves into a variety of augmentation strategies, such as global augmentation, local augmentation, GT-filtering, and GT-sampling, to enhance data diversity in 3D point clouds. While local augmentation methods are specific to the points associated with certain objects in the scene, global augmentation strategies work on the point cloud representing the whole scene. Global augmentations are applied concurrently to every point in the point clouds and every annotation. Several methods, including global translation, rotation, scaling, random flipping, and ground removal, are included in global augmentation. In contrast to global augmentation, local augmentation techniques also involve translation, rotation, and scaling. However, the key distinction lies in their application: rather than applying these transformations to every point indiscriminately, they are only used with specific labels and the points that are included in those labels. By using distinct random values for every label class, this method guarantees that each label class is enhanced individualistically from the others. The GT-Filtering method serves as a means to filter ground truth objects. This filtering process can be contingent upon various factors, such as difficulty level (difficulty level as described in KITTI (Geiger et al., 2012)) or the extent of points contained within the objects. The practice of incorporating more reference instances, such as individuals, from other scenes into the present scene is known as GT-Sampling. In order to do this, a database linking class labels to points must be created and all labels must be reprised over once. This strategy is used to rectify the disparity between the "background" and "foreground" in datasets like SemanticKITTI (Behley et al., 2019b), thereby aiming for more balanced datasets. The subsequent section delves into various LiDAR augmentation strategies, exploring methods to enhance LiDAR data for improved performance in tasks such as object detection, segmentation, and classification.

3.2 Augmentation Strategies involving LiDAR point cloud

This section provides a detailed examination of different augmentation strategies concerning LiDAR point clouds. The emphasis is placed on elucidating the key features and characteristics of each strategy, delineating their significance in enhancing the quality and diversity of LiDAR data for various applications.

- **SECOND: Sparsely Embedded Convolution Detection** : In this study (Yan et al., 2018), a database was constructed to store the labels of all reference (GT) objects and their corresponding point cloud data derived from the training dataset's 3D bounding boxes. Throughout the learning process, a subset of ground truth objects was arbitrarily chosen from this repository using GT-Sampling methodology and integrated into the present training point cloud by using the join operation. This augmentation strategy aimed to augment the number of GT reference objects per point cloud, thus facilitating the simulation of objects prevalent in varied surrounding conditions. Consequently, the detection performance was enhanced; nevertheless, it was observed that the methodology failed to consider the obstruction interaction that exists between various enhanced objects as well as among surrounding regions and enhanced objects, thereby disregarding the importance of rational placement and occlusion considerations. Overall, the augmentation technique primarily relied on the simple reuse of ground truth objects.

- **Mix3D** : In this methodology (Nekrasov et al., 2021), several augmentation techniques is employed to enhance the diversity and robustness of the LiDAR point cloud data. Firstly, the translation of multiple scenes to the origin was carried out by subtracting the centroid from all point positions, ensuring uniformity in spatial orientation. Subsequently, random transformations such as flipping in both horizontal directions and random rotations along different axes is applied to introduce variability. Additionally, random subsampling, elastic distortion, and random scaling is implemented, along with possible contrast augmentation, random brightness adjustments, and color jittering. Post-augmentation, the augmented point clouds from both scenes is merged by simply combining their points, effectively forming the union of the two sets. Furthermore, to augment the data out-of-context, points from two scenes is concatenated, serving as a valuable augmentation strategy, particularly useful for tasks like semantic segmentation.

- **LiDAR-Aug** : This research endeavor (Fang, Zuo, et al., 2021), introduces a rendering-based framework specifically designed to augment LiDAR point cloud data. The automatic implementation of obstruction impediment is ensured by the seamless integration of supplemented elements into genuine

background scenes through the utilization of rendering methodologies. Furthermore, a novel method is proposed for incorporating Computer-Aided Design (CAD) models into scenes and generating augmented LiDAR point clouds using a rendering module. The framework also employs the lightweight "ValidMap" strategy to impose stances for the supplemented objects, ensuring collision avoidance and achieving realistic obstruction insertion. Moreover, improvements are made over GT sampling by implementing measures to avoid object collisions during the object insertion process.

- **Augmented LiDAR Simulator for Autonomous Driving** : This study (Fang, D. Zhou, et al., 2020) presents a LiDAR point cloud simulation framework designed to enrich real point clouds by incorporating synthetic obstacles. The synthetic obstacles, represented by movable CAD models, encompass various types, with 500 models dedicated to pedestrians, 45 models allocated for cars, SUVs, etc. Real-world background models acquired using the RIEGL scan device, renowned for its high-speed, high-performance dual scanner mobile mapping system capabilities, are seamlessly integrated with semantic information. Background annotation using semantic segmentation information obtained through PointNet++ (Charles Ruizhongtai Qi et al., 2017) is employed, with annotations meticulously refined through filtration by groups of annotators. Additionally, the framework incorporates realistic obstacle placement learned from real traffic scenes to ensure the authenticity and accuracy of the augmented point cloud environment.

- **LiDARsim** : In this simulation-based approach (Manivasagam, S. Wang, et al., 2020), the scan background is substituted with a method involving registration techniques for LiDAR and diverse frames aggregation, while dynamic objects are regenerated out of PCD data rather than relying on artificial CAD models. To facilitate this, a substantial catalog comprising 3D static maps and dynamic objects is first compiled through extensive drives across different cities with the self-driving fleet. Scenarios are subsequently generated by selecting scenes from this catalog and virtually placing a Self-Driving Vehicle (SDV) and dynamic objects within the chosen scenes, ensuring plausible and realistic arrangements.

- **PointMixup** : PointMixup (Chen et al., 2020) introduces an interpolation method influenced by MixUp (H. Zhang et al., 2017) in image enhancement strategy. This approach discovers novel instances by efficiently assigning the path function between the labeled training PCDs.

- **PointAugment** : This approach (R. Li et al., 2020)is utilized for automatically optimizing and augmenting point cloud samples during the training of classification networks through the use of an augmentor to enhance data di-

versity. This method is employed to learn augmentation functions tailored to individual samples or classes, ensuring specificity in augmentation strategies. This approach integrates sample-aware auto-augmentation techniques and incorporates objectwise augmentor learning alongside classification learning using labeled training samples. However, the inclusion of an additional augmentor network and the intricate adversarial training process may introduce practical challenges to implementation.

- **PointPainting** : The proposed approach (Vora et al., 2019) involves initially executing the camera image through an image categorization algorithm. Subsequently, the anticipated categorization class grades are projected onto the LiDAR point cloud, effectively "painting" the point cloud with the segmentation results obtained from the camera image. This method offers an enhancement in 3D detection capabilities compared to the augmentation techniques.

- **PointAugmenting** : This research (C. Wang et al., 2021) introduces an object detection algorithm focusing on integrating 2D camera images with 3D LiDAR point clouds. In this methodology, deep features extracted by a 2D Object Detector from camera images are projected onto the corresponding points within the LiDAR point cloud. This process facilitates the fusion of information from the 2D image domain into the 3D LiDAR space, enhancing the feature representation and potentially improving object detection performance.

- **Part-Aware Data Augmentation (PA-AUG)** : The proposed method (Choi et al., 2020) leverages the structural information of 3D ground-truth boxes to enable the network to learn intra-object relationships. It prolongs the GT-Sampling technique through creating sub-division within GT objects and introducing random augmentation to every division. Objects such as cars, cyclists, and pedestrians from the KITTI dataset are utilized for this purpose. By dividing objects into multiple partitions and probabilistically applying LiDAR data augmentation operations like dropout, swapping, mixing, sparsifying, and introducing noise to each partition region, the method not only enhances the accuracy of the dataset but also demonstrates robust performance on corrupted data.

- **Augmentation of LiDAR for Fog Simulation** : A novel fog simulation method suitable for any LiDAR dataset is discussed in (Hahner, Sakaridis, Dai, et al., 2021), focusing on achieving physically valid simulations. This method involves augmenting clear weather point cloud data with artificial fog, thereby enhancing the realism of the dataset. The strategy has enhanced the efficacy of 3D object detection under challenging atmospheric conditions

characterized by the presence of fog.

- **Augmentation of LiDAR for Snowfall Simulation** :(Hahner, Sakaridis, Bijelic, et al., 2022) discusses augmenting the LiDAR point cloud to simulate real-world snowfall. Using this method, artificial snow is added to clear weather point clouds. With this method, each LiDAR line's snow particle sample is taken in two dimensions, and the resulting geometry is used to modify the measurement for each LiDAR beam.

- **Patch-based Progressive 3D Point Set Upsampling** : (Y. Wang et al., 2018) discusses the method of upsampling from lower resolution points to enhance resolution, aiming to reveal detailed geometric structures from sparse and noisy inputs.

- **Self-Ensembling Single-Stage object Detector (SE-SSD)** : In the proposed methodology (Zheng et al., 2021), the ground truth bounding box is partitioned into six pyramid shapes by connecting the centroid with the faces of the box. Subsequently, random dropout, swap, and sparsify operations are applied to each subset of points derived from these divisions, effectively simulating partial object occlusion, increasing object diversity, and mimicking variations in point sparsity. This process treats the divided point subsets as disassembled parts, allowing for targeted manipulation. Prior to shape-aware data augmentation, a series of global transformations are applied to the input point cloud, including random translation, flipping, and scaling. Overall, this approach integrates both shape-aware and global transformations to augment the data effectively.

- **Progressive Population Based Augmentation (PPBA)** : PPBA (Cheng et al., 2020) uses automated data enhancement methods to find the ideal settings for augmentation. During this procedure, the augmentation schedule is optimized by reducing the size of the search space and using the most effective parameters from previous rounds. It is discussed that sampling from individual frames of data from sensors results in a tenfold gain in data efficiency.

- **Dual Adaptive Data Augmentation (DADA)** : (J. Lee et al., 2023) In this approach, transformation of orthogonal coordinates of the point cloud into a spherical coordinate system is carried out, followed by sampling based on the LiDAR characteristics specific to the dataset, such as Velodyne HDL-64E for KITTI (Geiger et al., 2012) or a 40-beam LiDAR for the ONCE dataset (Mao et al., 2021). The object is evenly sliced and sampled in accordance with the LiDAR resolution corresponding to the dataset's LiDAR characteristics. In the case of existing ground truth (GT) objects, points are simulated

while adhering to the LiDAR characteristics to maintain the object's shape. Subsequently, downsampling or upsampling is performed based on the point density requirements.

- **PolarMix** : This method (A. Xiao et al., 2022) presents an innovative way to mix in cylindrical coordinates at the object and scene levels. Swapping at the scene level involves slicing along the azimuth axis to exchange point cloud regions of two circular LiDAR scans. Mixing at the instance level involves copying the rotated point objects into additional scans after cropping them from one LiDAR scan and rotating them by various azimuth degrees to make several copies. It should be noted, nevertheless, that this approach does not take shadowing effects into consideration and can lead to the illogical positioning of instance objects.

- **PatchAugment** : The focus of this methodology (Sheshappanavar et al., 2021) is on enhancing data inside the immediate area or neighborhood of the sampled object, which principle is similar to the PA-AUG data augmentation technique.

- **PointCutMix** : Inspired by Mixed Sample Data Augmentation (MSDA) in the image domain, this method (J. Zhang et al., 2021) generates new training data by mixing the original training samples. Specifically, subsets of point objects are replaced with those of other objects to enrich the training data. To utilize this approach, annotated point cloud objects from two point clouds with corresponding labels are required.

- **RS-Aug** : This approach (An et al., 2023) involves constructing an augmented reality scene to enhance the diversity of the training dataset. Initially, a pre-trained semantic segmentation model, RangeNet++ (Andres Milioto et al., 2019), is utilized to estimate semantic labels. This step serves as a preprocessing stage for the creation of a database in the auto-annotation process. Subsequently, the database is sampled for rendering augmentation in the subsequent step.

- **Back to Reality (BR)** : In this approach (X. Xu et al., 2022), position-level annotation is employed to build a virtual scene using an existing object database, facilitating weakly supervised 3D object detection. This process entails labeling the center of each object in the 3D space, followed by generating a virtual scene from a repository of 3D shapes.

- **Object Insertion Based Data Augmentation for Semantic Segmentation** : In this research (Ren et al., 2022), annotated LiDAR point clouds are used to extract foreground objects, which are then upsampled to create an object

library. Objects are dynamically introduced into the LiDAR point cloud during training. Parameters similar to actual LiDAR settings are used to mimic realistic scanning lines and shadows by using range images, which are generated depending on the resolution of the LiDAR used for data gathering.

- **PointWOLF** : To enhance the variation of known 3D objects, this approach (Kim et al., 2021) uses weighted alterations within local neighborhoods. It might be used, for example, on objects like wolves to produce different poses. The needed diversity is obtained by carefully combining several local transformations that are done with regard to anchor points in smoothly varied ways.

- **Real3D-Aug** : In this method (Šebek et al., 2022), objects are strategically placed at the same distance with identical observation angles to ensure consistent object point density and LiDAR intensity across the dataset. Additionally, collision avoidance measures are implemented by analyzing overlapping bounding boxes within semantic datasets.

- **Contextual Ground Truth Sampling** : In order to rectify data imbalances and insert ground truth GT objects in realistic placements, this method (D. Lee et al., 2022) uses contextual GT sampling (sampling ground truth objects from a pre-saved GT database).

3.3 Need of the Thesis work

The majority of augmentation methods primarily aim to enhance model performance for detecting specific classes using pre-labeled class instances. Many existing strategies assume prior knowledge of ground truth objects, with some leveraging state-of-the-art semantic models to obtain this information (An et al., 2023). However, occlusions are often overlooked in some augmentation strategies (Yan et al., 2018; A. Xiao et al., 2022), and accurate 3D bounding boxes are typically required for appropriate object insertion into the scene. While some approaches utilize a database of rendered graphics models of objects (Yan et al., 2018; An et al., 2023), simulators often rely on CAD models, which may exhibit unrealistic appearances, necessitate costly manual construction, and lack scalability to represent the diversity and complexity of real-world scenarios. Additionally, certain augmentation methods require knowledge of the LiDAR configurations used for the target dataset to facilitate augmentation effectively (Ren et al., 2022; J. Lee et al., 2023). Semantic segmentation of LiDAR frequently exhibits a decrease in accuracy when applied to LiDAR setups not previously encountered, as elucidated

in (Ryu et al., 2023). This discrepancy results in an accuracy decline in LiDAR data-based semantic classification tasks when the LiDAR operated for gathering the evaluation set contradicts the LiDAR utilized for the learning set. The variance in sampling patterns across various LiDAR configurations, characterized by differences in vertical and horizontal resolution, contributes to the emergence of sensor bias issues in 3D perception algorithms (Triess et al., 2021; Yi et al., 2020). Certain augmentation techniques, like PA-AUG and PointWolf, concentrate only on enhancing the ground truth (GT) objects, they do not handle how these objects are combined with the surrounding scene. Current augmentation strategies rely on either known labeled data or semantic models to extract ground truth objects. However, there could be situations where the label of the ground truth objects to be extracted is unknown. SOTA semantic models may also prove ineffective, as the model may not have encountered the specific object during the training. Even though the 3D semantic models have been trained on relevant objects before, using 3D semantic models is still ineffective as it requires an ideal model for the extraction of the specific object and the present SOTA 3D Semantic Models performance (Mean IoU) on real-world datasets like SemanticKITTI is below 80% (PapersWithCode, 2023). This highlights the need for alternative approaches to handle scenarios where ground truth information is unavailable or difficult to obtain.

4 Concept

We aim to manipulate the position of a VRUs (i.e. prototype or person) within a point cloud without physically relocating objects(prototypes) in the real world. This process involves extracting the point cloud corresponding to a prototype by leveraging geometric characteristics of points in three-dimensional space, followed by transforming the prototype's point cloud to a specified location. This approach enables evaluation of the response of a 3D object detection model to different scenarios wherein the prototype is situated in varied locations or within distinct background point cloud scenarios.

In this project, we refer to the source scene point cloud as the point cloud (PCD) from where an object or prototype needs to be extracted. A prototype is an object or a part of the source scene PCD that represents the point cloud of an actual object or object of our interest (person or prototype). The prototype is later also referred to as the original prototype to indicate it has not undergone transformation. The target scene cloud is the cloud where we insert the extracted prototype PCD, creating a new scenario PCD. Generally, at a high level, the system consists of two major steps which consist of sub-steps. The first step is represented by prototype PCD extraction, which is represented by green color in the figure 4.1. In this step, the prototype is extracted from the source scene cloud. The second major step is called the recombining of points cloud that is represented by red color in figure 4.1. In this step, the prototype PCD extracted from the source scene PCD is merged or recombined with the target scene PCD. Further recalculation of the point cloud of the prototype and calculation of shadow casting on the target scene PCD is done in this step. The result of the system is a target scene consisting of a prototype cloud that resembles as if there was an actual prototype object on the target scene PCD in real world. In the following section, we discuss each step in detail.

4.1 Prototype Extraction

The thesis aim is to create new point clouds by extracting prototypes from one point cloud (source scene PCD) and placing the extracted prototype in a desired

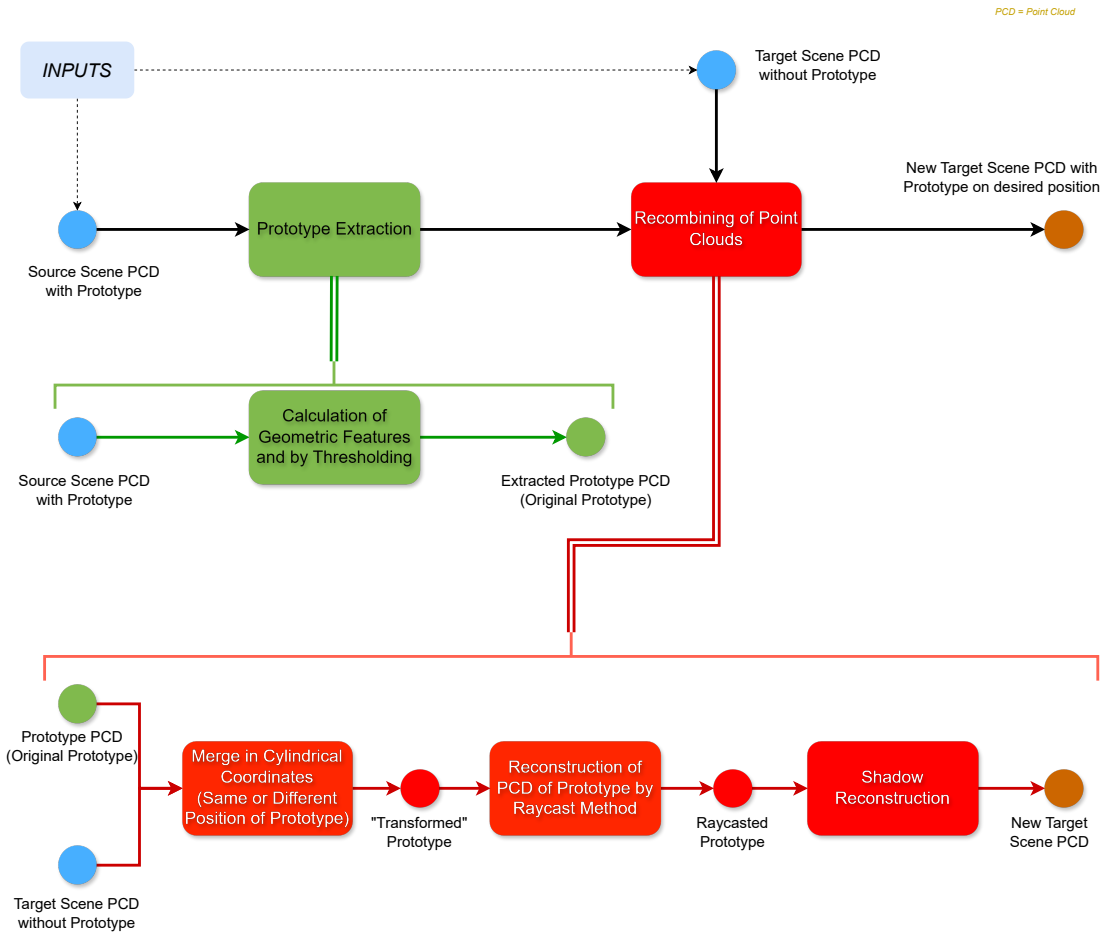


Figure 4.1: Concept graph representing the workflow of the project.

location on another point cloud (the target scene PCD). This facilitates the augmentation of an existing point cloud with additional objects or artifacts within the scene. Enabling such a process would increase the information content of the point cloud, which could be utilized to train various point cloud models or could be used to test the feasibility of a HAD system that would act when the prototypes are situated on the positions. We have multiple options for prototype extractions from the source scene PCD. To extract the prototype PCD from a source scene PCD, we first select a region of interest from where we need to extract the prototype PCD. For our project, we had mainly three options of prototype extractions from the region of interest of the source scene cloud.

1. Extraction of Prototype using known Labels
2. Extraction of Prototype using semantic ML models
3. Extraction of Prototype using geometric features

4.1.1 Extraction of Prototype using known Labels

Given a source scene cloud, a prototype could be extracted from a source scene PCD using the known labels of the point cloud. There is one precondition that needs to be met before using this approach. The precondition is that the point cloud from where the prototype is to be extracted should be labeled. Also, the label of the prototype to be extracted should be known. If we know the label of the prototype on a labeled point cloud then it would be straightforward and a perfect way to extract the prototype from the source scene PCD. However, this may not typically be the situation. Considering the time and cost effort required to get completely correct labeled data, most of the point cloud datasets available publicly online are just raw point clouds without labels. So choosing this approach is not feasible as a precondition has to be met before following this approach, which is not always the case. So we explored the next approach.

4.1.2 Extraction of Prototype using semantic ML models

We also have an option of using state-of-the-art 3D semantic models for the extraction of prototypes. A semantic model from (X. Wu et al., 2023) could be used to extract prototypes PCD from the source scene PCD. The source scene PCD does not need to be labeled for the extraction of the prototype. However, one precondition has to be met to use Machine Learning (ML) models for the

prototype PCD extraction. The ML model needs to be trained with the prototype-like objects and this requires a labeled dataset. Also, it requires an ideal semantic ML model to accurately extract the prototypes PCD, which in the present context is not feasible. Considering the laborious task of point cloud data annotations and the requirement of an ideal semantic ML model, we explored next on a simple yet effective approach to prototype extraction using geometric features.

4.1.3 Extraction of Prototype using geometric features

Geometric features such as surface variation, planarity, and linearity are used for this step to extract the prototype point cloud from a scene. We mainly focused on surface variation. Considering a part of the point cloud selected from a source scene PCD, surface variation tells us the change in surface features for a selected neighborhood. For the goal of extracting a prototype PCD from a region of interest, we mostly see an object standing on a flat surface in our experiment. The roughness (surface variation) for the flat surface is low in comparison with the feature of the prototype to be extracted. This is because of the orientation of the points in the 3D space. Instead of calculating geometric features for a whole scene, we selected a region of interest where the prototype is located. Geometric features are calculated for the selected region of interest with appropriate parameters for the nearest neighbor search. Thus calculated geometric feature is filtered with appropriate values by trial and error. The result is an almost perfect extracted prototype. Thus we do not need to rely on known labels or on using SOTA semantic 3D models for prototype extractions as for a selected Region of Interest (ROI), just using the geometric features is also good enough. As a result, a prototype is extracted from the source scene PCD by thresholding the calculated geometric features. Thus extracted prototype PCD will be combined with the target scene PCD in the next step to generate new scenarios in the target scene PCD.

4.2 Recombining of Point Clouds

This step is represented by blocks with red color in the figure 4.1. This stage involves the relocation of the prototype object to a specified target location within the target scene point cloud. Subsequently, the point cloud associated with the prototype is recalculated based on its new position, followed by the computation of shadow casting projected by the transformed prototype. Each of these sub-steps can be described in detail below.

4.2.1 Merge of Prototype and Target Scene Point Cloud

The goal of this step is to place the extracted prototype on a selected region of the target scene point cloud (PCD). Initially, experiments were done with registration algorithms like the Iterative Closest Point (ICP) algorithm to find the optimal transformation of the prototype point cloud to the target location. This method proved to be ineffective, particularly in scenarios where we intend to relocate the prototype to a region within the target cloud characterized by sparse point density, such as areas distant from the origin of the point cloud. For the ICP algorithm to work, we need to have a set of correspondence between the points in the prototype PCD and the target scene PCD.

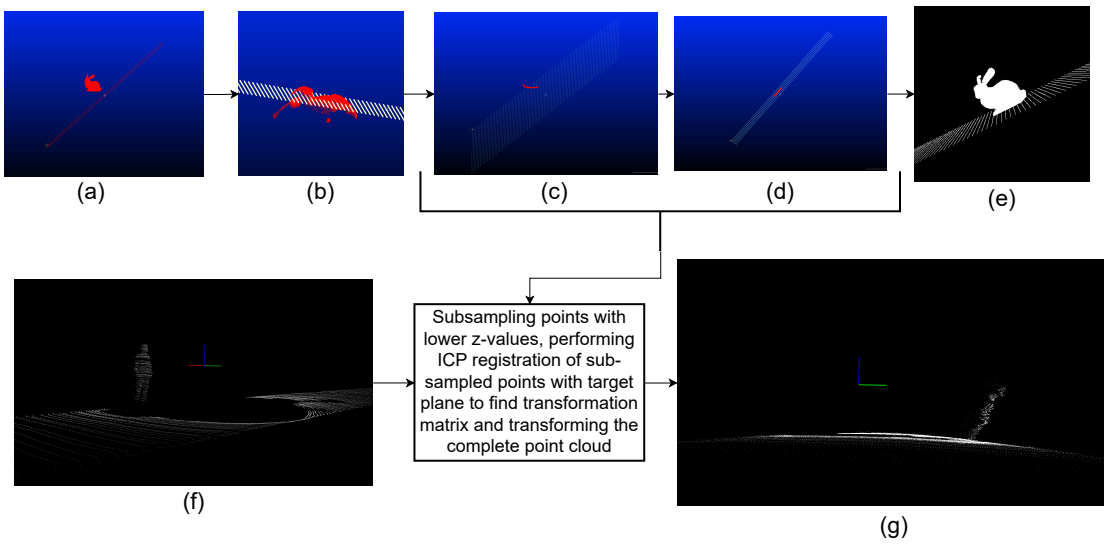


Figure 4.2: Transformation of Prototype PCD to Target Scene Plane using ICP Registration Algorithm (a)Bunny cloud needs to be placed on target inclined plane (b) Simple ICP registration with complete bunny cloud (c) Subsampling lower z-values points of bunny cloud (d) ICP registration of sampled points with the target plane (e) Bunny point cloud transformed using the transformation matrix calculated from subsampled ICP registration step (f) Point cloud of a person on a source scene (g) Point cloud of the person (inclined) after ICP registration on target scene cloud.

Taking the lower points of the prototype (i.e. points with lower z-values) as correspondence points on the target region works with the ICP algorithm as visualized in the upper row images in figure 4.2. But this step has a major drawback. This step assumes that the lower point(i.e. points with low z-values) of the prototype lies on the target region. This method fails to work if the point with lower z-values does not lie on the target location and is located slightly above the ground. This can be shown by the registration of the person point cloud

in the bottom row images in figure 4.2. As shown in figure 4.2, even though the transformation of the bunny cloud to the inclined plane worked with some changes in the registration process, the transformation of the person's point cloud on the target scene did not work as expected. The vertically oriented person was transformed to a required location but the end result was that the person was inclined. This is because the points sub-sampled from the lower z-values of the person's point cloud did not lie on a horizontal plane. ICP registration worked on finding the minimum distance between the sub-sampled points and the target plane. Since the sub-sampled points do not lie on the same plane, the resulting transformation matrix resulted in the inclination of the person point cloud, even though the sub-sampled point cloud aligned perfectly on the target scene. So we approach a different methodology of transformation of the prototype to a desired location on a target scene. It can shown with a flow diagram using figure 4.3

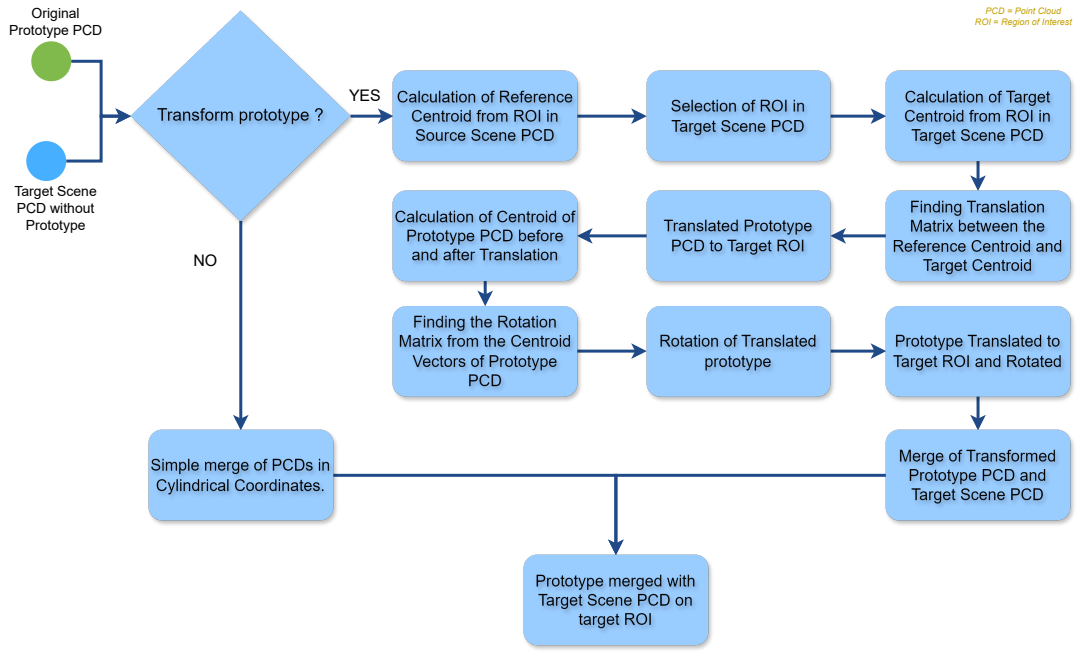


Figure 4.3: Flowchart for the Transformation of Prototype PCD to Target ROI and Merged with Target Scene PCD.

A simple geometrical approach is used for the transformation of the prototype to a region on the target scene PCD. If the prototype does not need to be transformed then it could be directly merged with the target scene and proceed with the next step. If the prototype needs to be transformed to a different location then first the centroid of reference is calculated from the region selected on the source scene PCD. This is calculated by adding all the points lying on the selected region on the source scene cloud that does not contain points from the prototype cloud. The resulting sum is divided via the overall count of considered points to get the

average value of the x, y, and z axis or the centroid of the ground plane. This calculation helps to get the orientation of the ground plane in the selected region of interest where the prototype is situated. Since the reference centroid is calculated from the points in the region of interest but excluding the prototype points, when performing this step, one needs to be careful so that no prototype points or high z-values points are considered for centroid calculation. Under such circumstances, the resulting centroid may not accurately represent the ground plane. Such a situation could lead to errors in the transformation process, potentially causing the prototype to be positioned in invalid positions such as the prototype positioned above the ground plane in the designated region of the target scene PCD.

After the calculation of the reference centroid, a desired region of interest is selected on the target location, and the target centroid is calculated for the selected region. Using the two centroids, a translation matrix is created with which the prototype is translated to the target region in the target scene PCD. Using the vectors representing the reference centroid and target centroid, from equation 2.6 the rotation matrix is calculated.

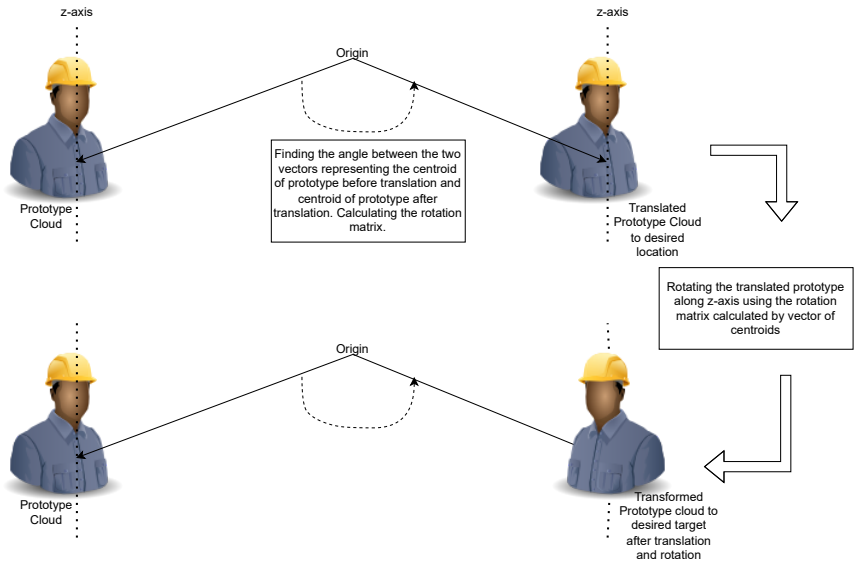


Figure 4.4: Translation of Prototype, Calculation of Rotation Matrix and Rotation of Translated Prototype Point Cloud.

As shown in figure 4.4, the calculated rotation matrix is then used to rotate the translated prototype PCD along the z-axis so that the transformed prototype object faces the origin. This is among the severely crucial actions. If the prototype is only translated to a desired location then the prototype might be facing in an incorrect direction (i.e. not viewing towards the origin). This false orientation of the prototype point cloud would result in a false calculation of points in the further steps as no points exist for surfaces that do not face the origin.

The transformed prototype cloud is then merged with the target scene PCD. The result is a prototype point cloud lying on top of the target region in the target scene PCD.

4.2.2 Reconstruction of Prototype Point Cloud on Target Scene PCD based on distance from the Origin by Raycasting

After the prototype point cloud has been placed in a target position on the target scene cloud, we need to find new point clouds of the prototype that accurately represent the prototype. We need to consider the following points for the reconstruction of points for the "transformed" prototype point cloud.

- Number of point cloud points for the prototype should vary based on distance from the origin.
- Need to consider the sparsity of prototype PCD points based on distance from the origin.
- New points representing the prototype surface should accurately represent the surface.
- Recalculated points should not be random. It should be a collection of points that follow the laser rays that originate from the origin in the target scene cloud.

In raycasting, a series of rays, originating from a viewpoint is passed to a certain distance. The point of intersection of rays to the obstacle is noticed. Based on the distance from the obstacle point, a 3D perspective is reconstructed in a 2D map for the viewpoint. Similar to such a principle, we cast rays to the target scene that contains the "transformed" prototype point cloud. But before the emission of rays, we reconstructed the surface of the "transformed" prototype cloud. We have options such as alpha shapes, ball pivoting, and poisson surface reconstructions that are used for the reconstruction of the surface from the point cloud. The poisson surface reconstruction method, as outlined by (Michael Kazhdan and Hoppe, 2006), tackles the task of generating a smooth surface by addressing a regularized optimization problem. This approach contrasts with other methods that may yield less refined results, as they directly use the original points of the point cloud as the vertices of the resulting triangle mesh without any adjustments. Such reason leads us to use the poisson surface reconstruction for better construction of the surfaces. The Poisson surface reconstruction method extends its surface

generation process to regions with sparse point density and even extrapolates into certain areas to create triangle meshes. The created triangle meshes also represent surfaces where there are minimum to no densities of points. To tackle such an issue, the reconstructed surface mesh is filtered. Filtering based on the density of the point also works but it requires manual setting of the threshold value. Instead of filtering by the density threshold value, triangles whose vertex do not contain the original point cloud points are removed. Following the later approach, no manual adjustment of the density threshold value is required. After the surface mesh is filtered out and the resulting surface represents the mesh more accurately, rays are projected from the origin to a target region on the target point cloud scene through the prototype surface mesh. To make the calculations faster, we select a region on the target scene point cloud.

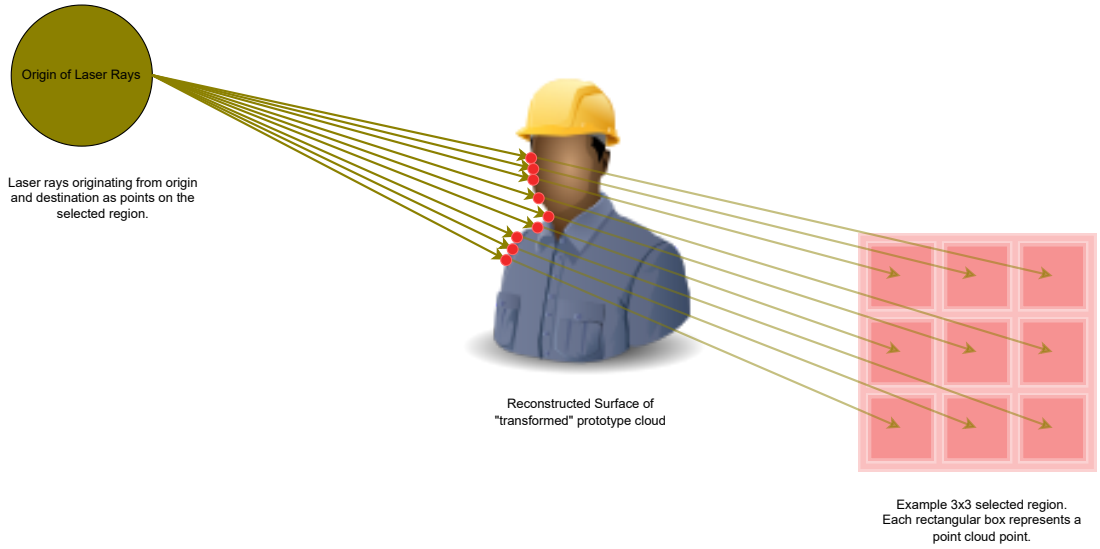


Figure 4.5: Rays traversing from the origin and terminating towards the selected points. Rays intersect with the reconstructed surface of the prototype, which is represented by red points.

As shown in figure 4.5, rays start from the origin and terminate at the selected region of interest. 3x3 rectangular box represents an example region that is selected on the target scene point cloud. The region is selected in the target scene such that rays traversing from the origin to the designated points intersect the surface of the prototype(reconstructed surface of the Prototype). Each 9 red box inside the selected region represents a point cloud. Thus in the example in figure 4.5, we are representing 9 points of a part of the selected target scene point cloud. The rays travel toward the destination and some of the rays intersect the triangular mesh/surface of the prototype. Intersected point is calculated between the reconstructed surface mesh of the prototype and the casted rays. This intersected point gives us the new point cloud of the prototype. We call this new

point cloud as Raycasted point cloud of prototype. This is represented by red points on the figure 4.5.

Two methods are explored for the calculation of the raycasted point cloud of the prototype. Since the reconstructed surface is represented by triangle mesh, when a single "laser beam" hits the surface of the prototype, it intersects with triangle in the triangle mesh.

4.2.2.1 Calculation of Point based on Centroid of Triangle

Figure 4.6 demonstrates a single ray originating from the origin and terminating at a point in the selected region of interest in the target scene point cloud. The ray intersects the triangle mesh at (x, y, z). The vertex of the triangle is represented by $(x0, y0, z0)$, $(x1, y1, z1)$ and $(x2, y2, z2)$. To calculate the point (x_c, y_c, z_c), we compute the centroid of the triangle mesh using equation 2.21. These calculated points give us the raycasted point coordinates. This is represented by the red colored point in figure 4.6. It can be viewed that the centroid of the point is not always the point of intersection. Here in the figure, the green point (x, y, z) is the point of intersection to the triangle but this approach gives us the red color point (x_c, y_c, z_c). To reduce the errors, other methods are experimented with.

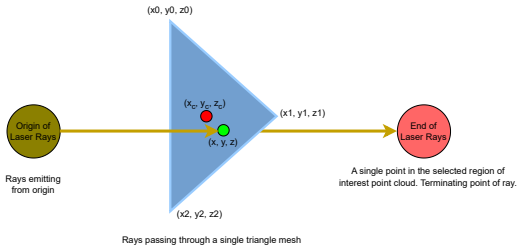


Figure 4.6: Calculation based on Centroid of the Triangle.

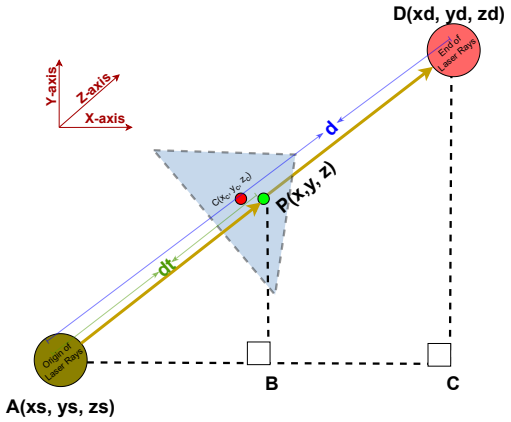


Figure 4.7: Calculation based on hit distance to the Triangle.

4.2.2.2 Calculation of Point based on hit distance to the Triangle Mesh

Using the centroid method to find the raycasted point is not always correct. The accuracy of the centroid method would depend on the proportion of triangles

in the triangle mesh. The larger size of the triangle in the triangle mesh would result in a higher error rate. To eliminate such dependencies and error rates, the distance-based method is approached. In the figure 4.7, right-angle triangles are constructed. $\triangle ABP$ represents the smaller triangle. $\triangle ACD$ represents the larger triangle. dt represents the hit distance from the origin to the surface of the triangle. d represents the distance between the origin and the terminating point of the laser ray. Since two triangles are similar, using the law of similarity in triangles, we can write as follows :

$$\frac{AB}{AC} = \frac{AP}{AD} \quad \text{(Ratio of sides of triangle)}$$

$$\frac{x}{x_d} = \frac{d_t}{d} \quad \text{(Inserting length of the sides)}$$

$$x = x_d \times \frac{d_t}{d} \quad \text{(Result after Cross multiplication)}$$

$X - axis$, $Y - axis$, and $Z - axis$ coordinates of the green color point can be obtained using the same approach.

$$P(x, y, z) = \left(x_d \times \frac{d_t}{d},\ y_d \times \frac{d_t}{d},\ z_d \times \frac{d_t}{d} \right) \tag{4.1}$$

$$\text{distance(d)} = \sqrt{(x_d - x_s)^2 + (y_d - y_s)^2 + (z_d - z_s)^2} \tag{4.2}$$

Equation 4.2 represents the Euclidean distance between the two points in space (i.e. distance between vertex A and vertex D in figure 4.7). The d_t distance from the origin to the triangle can be obtained by raycasting. Hence, by using the equation 4.1, the coordinates of point $P(x, y, z)$ can be calculated. This is represented by the green color point in figure 4.7. As shown in the figure, this point accurately represents the intersected point of the "laser ray" with a triangle mesh. For the calculation of the raycasted point cloud of the prototype, this approach is followed. As a final result, the raycasted point can be obtained which is represented by the red colored points in figure 4.5.

4.2.3 Shadowcasting on the Target Scene PCD by Raycasted Prototype Point Cloud

After the points of the prototype PCD are recalculated based on the distance and orientation from the origin, we call the points raycasted PCD of the prototype. In the shadowcasting step, we reconstruct the shadow projected by the raycasted prototype on the target scene. Two methods for the calculation of shadow casting are experimented with. Two methods are the Hidden Point Removal (HPR) algorithm and calculation based on the Raycasted Rays.

4.2.3.1 Approach using HPR algorithm

Based on a certain viewpoint in a point cloud, the HPR algorithm tells us if a point in the point cloud is hidden or visible. Consider an example as shown in figure 4.8. In the figure, the left image of Armadilos is shown without applying the HPR algorithm. The right image is shown after applying the HPR algorithm. In the figure 4.8, we could not distinguish if the left image of Armadilos is facing backward or frontwards (toward the direction of the reader). Since all the points are practically visible, the left image of the Armadillos in the figure seems to be facing backward and frontwards at the same time, creating confusion. After applying the HPR algorithm, the output is represented by the right side of the image in figure 4.8. We can clearly say that the Armadilo is facing backward with its back towards the reader. On applying the HPR, for the figure 4.8, the viewpoint is selected to be a point in the direction of the reader or outwards from the screen. This resulted in points being removed that are hidden from the viewpoint.

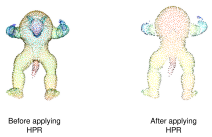


Figure 4.8: Point cloud of Armadilos (Q. Y. Zhou et al., 2018).

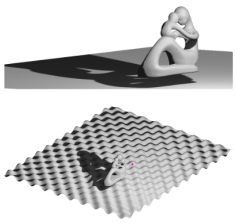


Figure 4.9: Shadow casting by HPR (Katz et al., 2007).

In figure 4.9, the shadow is projected based on the visibility of points from the viewpoint. The viewpoint is defined by a pink color point on the bottom image of figure 4.9. Based on the visibility of the points from the viewpoint, the points are

assigned low-intensity values or left unchanged. This resulted in the projection of shadow on the surface. A similar principle is experimented for the casting of shadow by the raycasted point cloud on the target scene point cloud. The viewpoint for HPR is set to the origin of the target scene point cloud. Instead of changing the intensity values of the hidden points, the points are removed from the point cloud and as a result, shadow-projected by the raycasted prototype on the target scene is acquired as output.

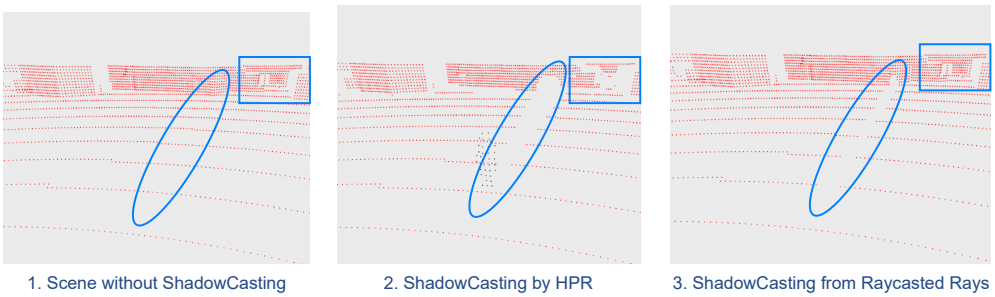


Figure 4.10: Shadow Casting Difference between two methods.

4.2.3.2 Approach using Raycasted Rays

This approach is simple yet very effective. Instead of using the HPR algorithm for the calculation of shadow projection on the scene, the ray casted rays are considered for further calculations. Rays originate from the origin of the target scene point cloud and terminate at the points on the ROI in the target scene point cloud. Any rays intersecting the surface of the prototype give the raycasted point. Rays intersecting with the prototype surface means that the terminating point of the ray will be hidden by the surface of the prototype. Some rays do not intersect with the surface of the prototype. The terminating points of such rays are not hidden by the surface of the prototype. As a result, an accurate shadow of the prototype surface can be calculated. As shown in figure 4.10, this approach gives a perfect shadow of the reconstructed prototype surface. The blue rectangular box in the figure shows that applying the HPR algorithm affected the points that were not in the shadow region of the raycasted prototype. However, points not lying in the shadow region of the prototype were not affected using the raycasted method for shadow calculation. This approach is implemented for shadow calculation.

Figure 4.11 shows an example of shadow casting on the target ROI in target scene PCD. The green-colored rays, originating from the origin to the point in the selected region, were not intersected with the reconstructed prototype surface(triangle mesh). So the terminating points of the green colored rays are not hidden from the origin. This is represented by the green-colored square on the

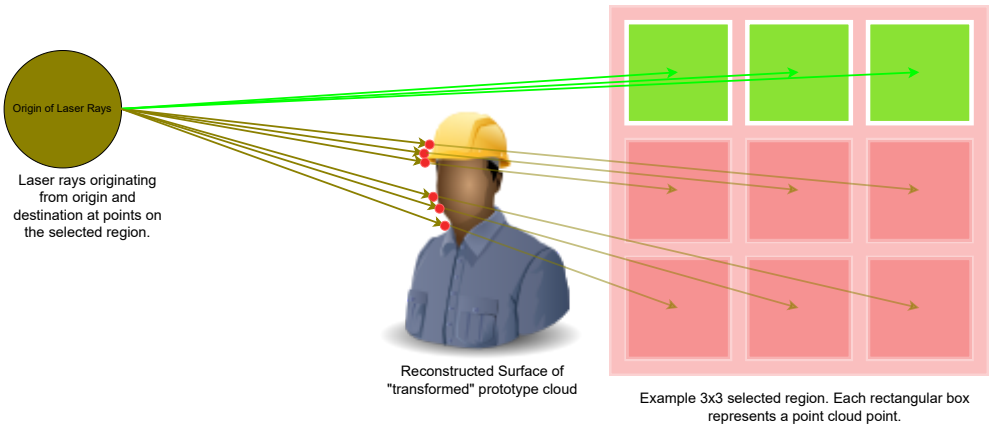


Figure 4.11: Shadow Casting Approach using Raycasted Rays.

figure. The red-colored square represents that the point is hidden by the surface. The rays that originate from the origin designated to the red-colored square are blocked by the surface of the prototype. The intersected point of the rays with the surface is represented by red-colored points and is called the raycasted points. The terminating point of the rays blocked by the surface or prototype is the shadowed region.

4.3 Functional Description of the Implementation

Figure 4.12 shows an example of high-level implementation details of the project. The images present in the top row of the figure 4.12 represent scenes. The top left image represents a source scene. An Asset (Person) is present on the scene, which is represented in the red rectangular box. The top middle image of the figure represents a target scene. The goal of the project is to extract a person from the source scene and insert the extracted person into the target scene (without manipulating the objects in simulations or real-world). This results in a new scene. In the figure, the target scene is modified in such a way that a person is now present on the target scene. So the recombination between the two upper left scenes would result in a new scene similar to the top right of the figure 4.12. These different scenes were captured in the CAR Learning to Act (CARLA) simulator (Dosovitskiy et al., 2017). An ego vehicle was summoned on CARLA using carla-ros-bridge. The Asset or Person was summoned to a specified location near the ego vehicle.

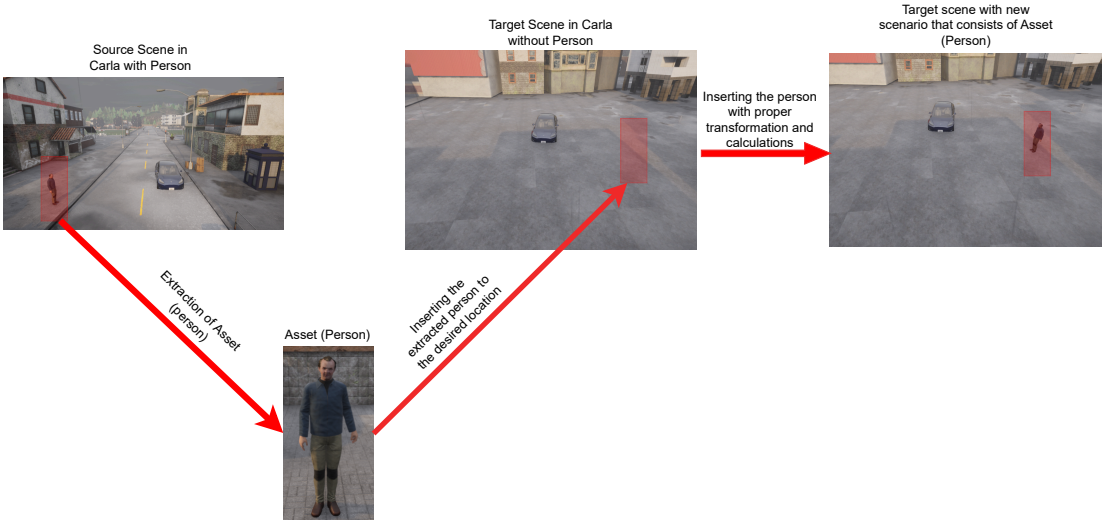


Figure 4.12: Functional description of the implementation.

4.4 Technical Description of the Implementation

The aim of the project is the creation of a novel concrete scenario. This project works on the point cloud level. Functional description of the projects shows a higher level overview of the project using images from CARLA. This section describes the technical aspects of the project. The point cloud images shown on the top row in figure 4.13 correspond to the point cloud of scenes represented by the top row images in figure 4.12. The ego vehicle is attached with a lidar sensor, which gives a lidar point cloud of the current scene continuously. The point cloud data is made available to Robot Operating System (ROS) topic by carla-ros-bridge, which is then captured by written scripts.

The image on the top row leftmost side in figure 4.13 represents a source scene point cloud. The task of the project is to extract the prototype point cloud from the source scene PCD. The prototype PCD is represented by red-colored points on the source scene PCD. The prototype PCD is to be extracted using the geometric features of the point cloud. The region of interest from where we would like to extract and insert a prototype is represented by a red rectangular box in all the top row images in figure 4.13. In the source scene PCD, geometric features on the selected region of interest are calculated. The separation of the prototype point cloud from the target scene is done by thresholding the geometric features. This gives us an extracted prototype PCD represented by the image on the bottom row of figure 4.13. This prototype needs to be placed on the target scene region of interest in the top row middle image of the same figure. With the proper

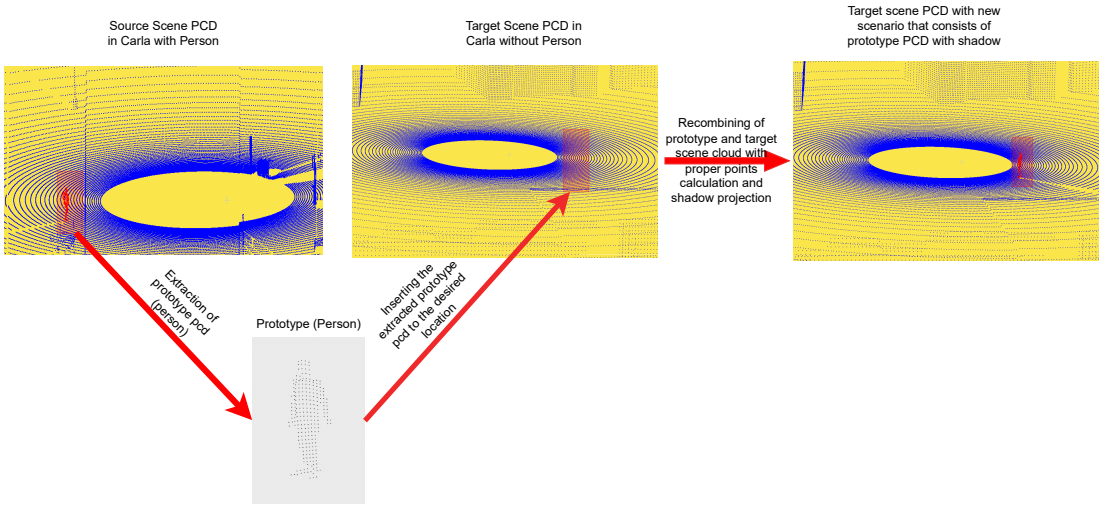


Figure 4.13: Technical description of the implementation.

transformation of the prototype PCD according to the geometry of the target scene PCD, the prototype is placed, and the projected shadow is calculated. A final output point cloud is created by combining the extracted prototype from the source scene point cloud into the target scene point cloud. This novel point cloud is represented by the top row rightmost image in the figure 4.13.

5 Prototypical Implementation of the Concept

From the source scene point cloud, a prototype point cloud (person PCD) is extracted. A region of interest (ROI) is defined in the source scene point cloud which contains the prototype point cloud. Geometric features are calculated for the selected region of interest. Non-prototype points are filtered out based on the geometric feature values of the points. Thus a prototype PCD is obtained from the source scene PCD. The acquired prototype point cloud is transformed into a target location on the target scene PCD. The point cloud for the prototype is recalculated on the target scene PCD based on the location of the prototype on the target scene PCD. Surface reconstruction and Raycasting are used for the recalculation of points of the prototype point cloud. Thus obtained point cloud of the prototype after raycasting is termed as Raycasted Prototype PCD. Shadow projected by the raycasted prototype point cloud on the target scene cloud is computed. Hence a new scenario of the target scene point cloud is created by the recombining of source scene PCD and target scene PCD. The following section explains each step in detail. For the purpose of the experiment, semantic-lidar data captured from CARLA was used for both the source scene point cloud and target scene point cloud.

5.1 Loading a Source Scene Point Cloud

In the figure 5.1, we can see points in the point cloud that are either black or green in color. To color the scene, we have used the ObjIdx value from the semantic-lidar data obtained from CARLA. The semantic LIDAR data is sourced from CARLA to only facilitate the visualization of the prototype point cloud. The value of the ObjIdx is zero for stationery objects and the value of ObjIdx is non-zero for non-stationery objects. Based on this value, if the ObjIdx is zero for the point, the point is colored green and if the ObjIdx is non-zero then the point is colored black. In figure 5.1, we can see a prototype (person), located on the left part of the figure, represented by black in color. The point cloud in figure 5.1 represents the CARLA source scene as shown in functional implementation figure 4.12.

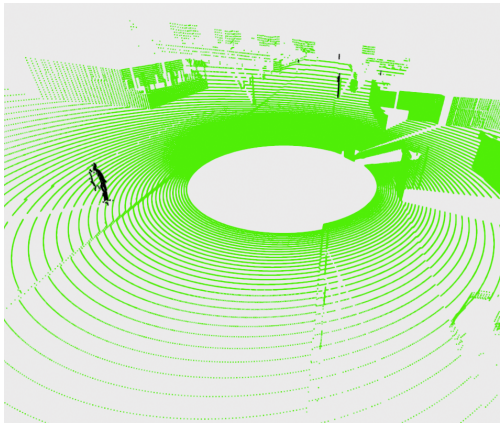


Figure 5.1: Prototype in Source Scene PCD.

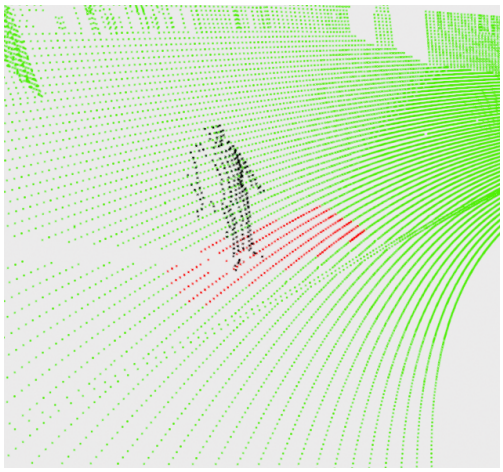


Figure 5.2: Selection of ROI on Source Scene PCD.

5.2 Selection of Region of Interest on Source Scene Point Cloud

Since we plan on extracting the prototype from the source scene cloud, we need to specify the region on the source scene where we would like to extract the prototype PCD from.

In figure 5.2, the region of interest is represented by red-colored points plus any higher z-values points lying within the region. For ease of clarity, we have not changed the color of the prototype. First, a region is selected around the prototype manually. This is done manually by using the mouse cursor. At first, the position of the cursor is noted on the screen, if something is clicked on the screen then the relative position of the cursor on the world coordinates (the world where the point cloud is being visualized, the GUI for visualization) is calculated. Squared distances between the points on the point cloud and the world coordinate point are calculated. Based on minimum distance, a point on the point cloud is selected and marked with red color. Since manually selecting a list of points in a source scene cloud would take a lot of time, minimum x, minimum y, maximum x, and maximum y are calculated from the manually selected boundary region of interest. All the points that lie within the boundary value are finally selected as a region of interest on the source scene PCD, from where a prototype PCD is to be extracted.

5.3 Extraction of Prototype from Source Scene Point Cloud

Geometric features are calculated for the selected region of interest (ROI) from the source scene PCD. Based on the number of nearest neighbors as 5 using "KDTreeSearchParamKNN" in open3d (Q. Y. Zhou et al., 2018), the nearest neighbor is searched for each point in the selected ROI. Using the kd-search tree, the covariance matrix and then the eigenvalue are computed after finding the local neighbors for each point in ROI. Utilizing the formulas for the calculation of geometric features as in equation 2.20, geometric features(surface variation) are calculated for each point in the ROI point cloud. The geometric feature values are filtered out by the appropriate threshold and finally, the remaining point having the valid geometric features in the selected ROI on the source scene PCD is extracted, this extracted point cloud represents the prototype point cloud or original prototype point cloud.

Figure 5.3: Extracted Prototype Point Cloud from Source Scene PCD for different threshold of Surface Variation of points in selected ROI (a) $SurfaceVariation \geq 0$ (b) $SurfaceVariation \geq 0.05$ (c) $SurfaceVariation \geq 10^{-8}$.

Trial and error were employed to explore different threshold values for surface variation, aiming to filter the prototype's point cloud optimally. As depicted in Figure 5.3, a range of filtered point clouds for the prototype are displayed alongside their respective threshold values. Notably, the analysis reveals the utilization of a remarkably low threshold value 10^{-8} for extracting the prototype's point cloud. This decision is attributed to the synthetic origin of the point cloud sourced from CARLA, which lacks the randomness typically observed in the real-world LiDAR point clouds. The resulting point cloud obtained from this process is referred to as the "extracted prototype point cloud" or simply the "(original) prototype point cloud."

5.4 Transformation of Prototype to a position on Target Scene Point Cloud

The extracted prototype cloud could either be merged with the target scene PCD on the original location (if the extracted prototype PCD aligns properly with the target scene PCD) or the prototype could also be transformed to a different location on the target scene PCD.

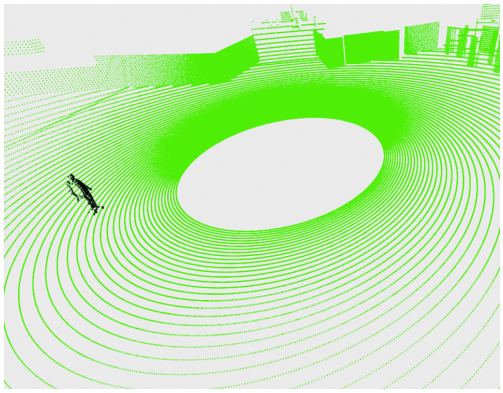


Figure 5.4: Prototype PCD on Target Scene PCD without transformation.

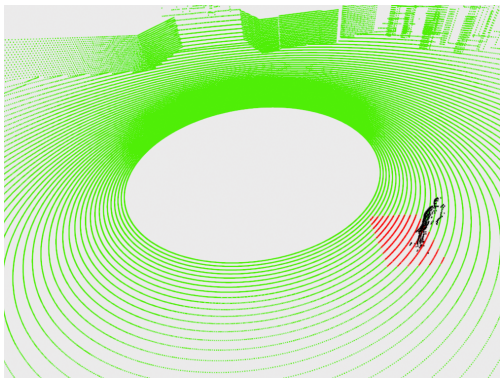
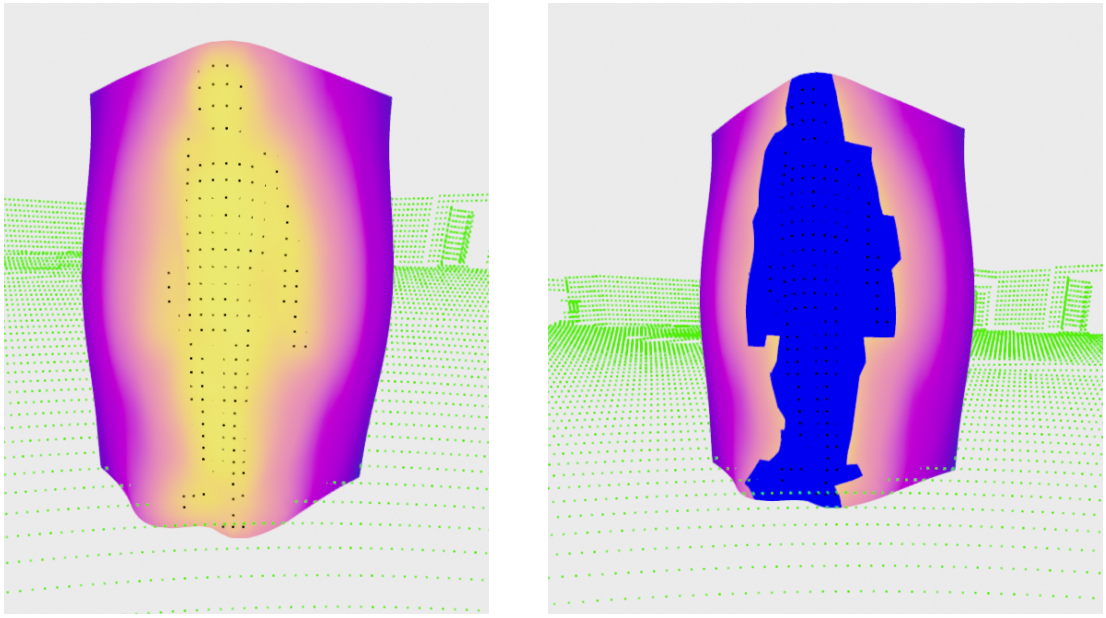


Figure 5.5: Prototype PCD on Target Scene PCD to a different position after transformation.

Figure 5.4 shows a prototype PCD (black points) without any transformation located on a target scene PCD. Since the prototype is standing on top of the target scene and no irregularities were seen when concatenating the prototype point cloud (PCD)to the target scene point cloud (without transformation), we could also continue working with this position. Examples of irregularities encompass instances where a prototype appears submerged within the ground plane or traverses through a wall. If the prototype needs to be transformed to a different position, Region of Interest (ROI) is selected on the target scene point cloud. The selected ROI is shown by red colored points in figure 5.5. The translation and rotation matrix is calculated based on the reference centroid and target centroid as explained in figure 4.4. One thing to note here is that when the position of the prototype is changed in figure 5.5, the prototype is also rotated according to the position change from the origin. Using the calculated rotational matrix, the prototype is rotated around the z-axis through the centroid of the prototype. This is an important step. Since we are not recalculating the side points or backside surfaces of a prototype, it is crucial to maintain the direction of the prototype relative to the origin. In our case, the prototype is still facing the origin.

5.5 Surface Reconstruction and Filtering

After the position of the prototype PCD has been finalized on the target scene PCD, the next step is the reconstruction of the surface. Using the Poisson reconstruction method, the surface of the prototype is reconstructed.



(a) Reconstructed Surface

(b) Reconstructed Surface with filtered surface mesh

Figure 5.6: Reconstructed Surface of the Prototype on the Target Scene PCD (a) Reconstructed Surface with varying density of points (b) Reconstructed surface with filtered surface mesh (blue).

Figure 5.6 (*a*) represents the total reconstructed surface visualized by varying density of points on the surface. The yellow color surface represents the region with a higher point density. The purple-colored surface in the figure represents the region with lower point density. The reconstructed surface of the prototype represents a triangle mesh. Since the prototype is relatively smaller than the reconstructed surface as shown in figure 5.6 (*a*), we need to filter the undesired surface of the reconstructed prototype. Using density value for filtering also works but requires lots of trial and error to get an optimal value, so we filtered the undesired reconstructed surface by removing all the triangles in the triangle mesh that do not contain the point cloud of the prototype. This filtered surface of the prototype is represented by the blue color region in the figure 5.6 (*b*). It can be viewed that the filtered surface (represented by the blue color) tries to resemble the shape of the prototype with a higher concentration of point cloud. The depth

information of the prototype points is represented by the varying depth of the triangles in the filtered reconstructed surface (triangle mesh).

5.6 Raycasting

Rays originating from the origin of the target scene PCD and directed toward the point clouds in the region of interest are "created". The region of interest on the target scene PCD is selected in such a way that the rays cover the surface of the filtered surface mesh as much as possible. The selected region on the target scene is represented by red color points.

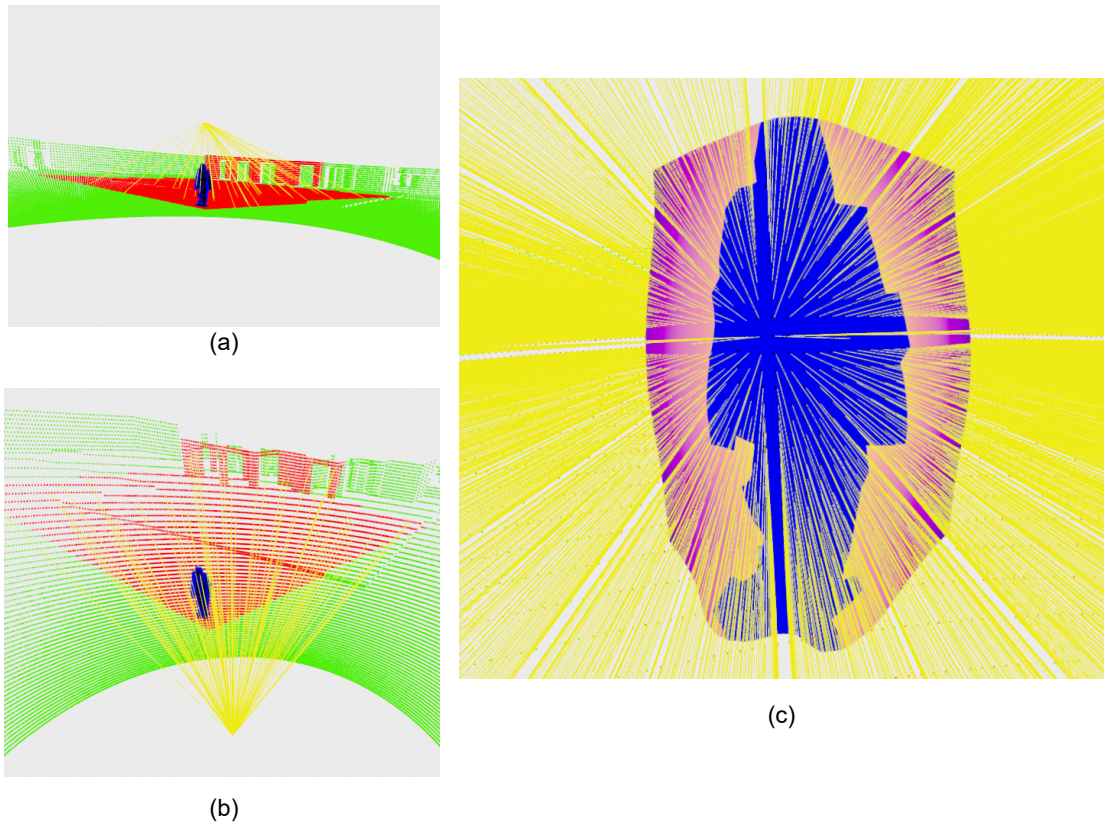


Figure 5.7: Raycasting from Origin to the target ROI on the Target Scene Point Cloud (PCD). (a) and (b) shows raycasting with a lower ray count. (c) shows rays intersect with the surface of the Prototype.

In the figure 5.7, the yellow color lines mimic the laser rays from the LiDAR sensor. The blue color represents the surface mesh of the prototype after filtering. When the ray traverses from the origin to the target region of interest (ROI), it

intersects the surface mesh. An example illustrating the triangle mesh and rays traversal is shown in figure 5.8.

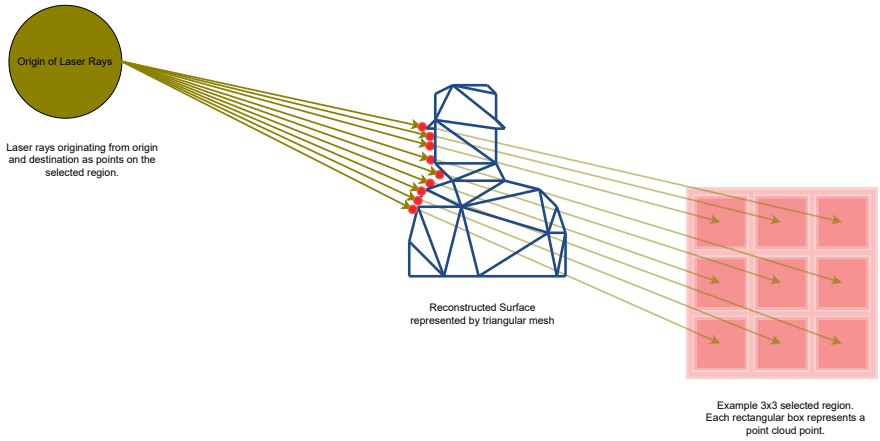


Figure 5.8: Rays casting towards Prototype surface represented by a group of triangles in the triangle mesh.

Figure 5.8 gives an example of a prototype surface represented by a group of triangles of a triangle mesh. Triangle intersected by the rays are calculated by casting rays as shown in figure 5.7 and figure 5.8. From the intersected triangles, new points of the prototype are calculated.

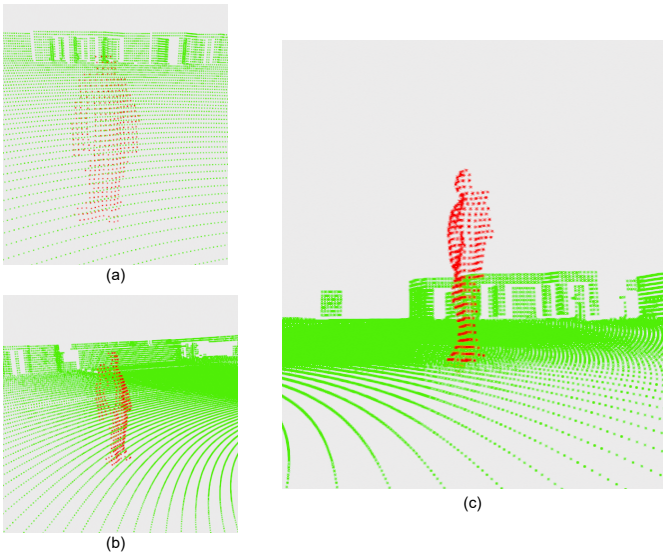


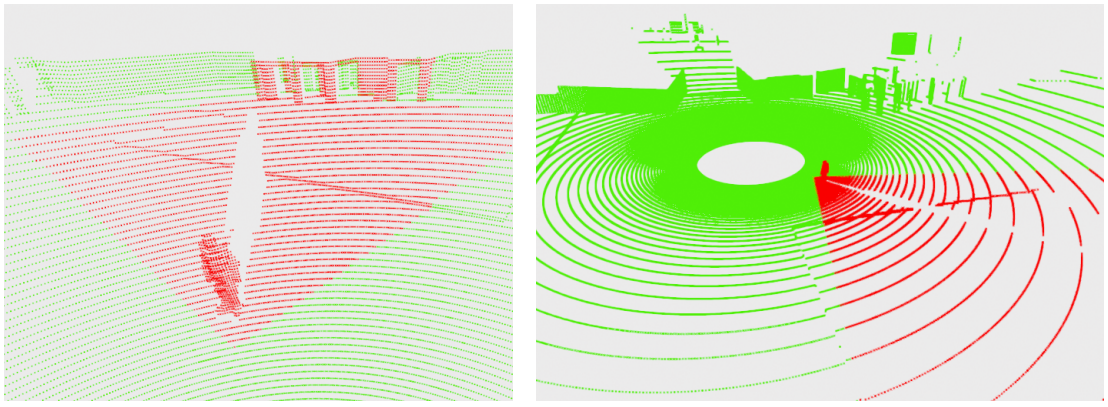
Figure 5.9: Raycasted Prototype visualized by red colored point cloud on the target scene cloud from different viewpoints.

The raycasted point cloud of the prototype is calculated by using the equation 4.1 as shown in figure 4.7. As a final result, a point cloud of the prototype is created.

We call the point cloud received after raycasting a raycasted prototype point cloud. Figure 5.9 shows the raycasted point cloud of the prototype calculated after the raycasting method. The red points point cloud represents the raycasted prototype in figure 5.9. It can be observed that the calculated point cloud of the surface mimics the surface of the original prototype (person). The raycasted prototype lies on the target scene PCD. Shadow projection by the raycasted prototype on the target scene PCD needs to be calculated. This is done in the next section.

5.7 Shadow Casting on Target Scene by Raycasted Prototype Point Cloud

The input to this step is a raycasted prototype point cloud on a target location (ROI) of the target scene PCD. An experiment was done with the hidden point removal algorithm. Important parameters for the HPR algorithm are the viewpoint and the radius of the sphere for spherical flipping. Visibility of the points is determined by looking from the viewpoint of the target scene cloud. The viewpoint is set to the origin so that the process mimics the shadow projection process when a LiDAR sensor is placed at the origin. Appropriate parameters for the HPR algorithm need to be chosen to increase the shadow projection accuracy. As shown in figure 4.10, instead of using the HPR algorithm, the raycasted method is a more accurate approach for shadowcasting in our experiment.



(a) (b)

Figure 5.10: Shadow Casting by the Prototype on the Target Scene Point Cloud (PCD). (a) and (b) shows projected shadow from different viewpoints.

Using the raycasted method as explained in figure 4.11, the shadow projected by the prototype surface in the target region of the target scene cloud is calculated. The result of shadow casting is shown in figure 5.10. Points lying on the shadow

region of the prototype are removed. Shadow was calculated for ROI to make the computation faster (shown by red colored region in figure 5.10).

6 Evaluation

In this thesis, the evaluation of the project output is conducted with a focus on validating the accuracy of the obtained results, rather than demonstrating improvements over existing SOTA 3D object detection models. To evaluate the results obtained from the study, a two-step analysis is conducted. Firstly, the raycasted point cloud is analyzed, followed by an examination of the casted shadow.

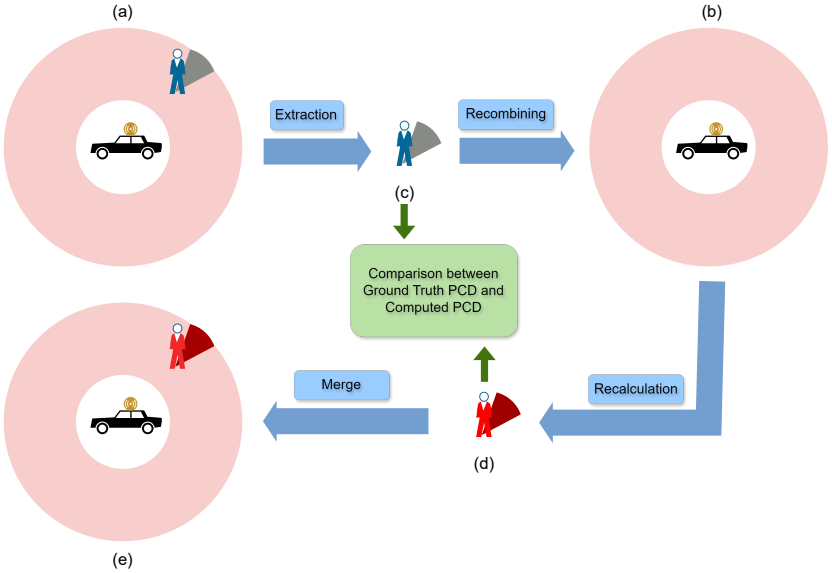


Figure 6.1: Evaluation Concept Diagram. (a) and (b) present two distinct point clouds featuring equivalent background contexts, with the sole disparity being the presence or absence of a Prototype (person) and its cast shadow. (c) showcases the isolated point cloud representing the Prototype (Original Prototype) and the ground truth shadow. (d) displays the computed point cloud of the person(raycasted point cloud) along with its newly projected shadow (anticipated shadow) onto the Target Scene PCD ROI(b). (e) Augmented Target Scene Point Cloud.

The assessment of the thesis output involves a comparative analysis between the point clouds represented in figure 6.1 (c) and 6.1 (d). Figure 6.1 (c) represents the ground truth point cloud of the prototype (person or original prototype) and the shadow projected by the prototype. It is obtained by extracting the desired

ground truth point clouds from the source scene point cloud (*a*). (*d*) represents the prototype point cloud generated through raycasting (raycasted point cloud of prototype), accompanied by the shadow projected by this new prototype point cloud on the target scene (*b*). This output encapsulates the culmination of the thesis work. (*e*) represents the final augmented target scene point cloud. The source scene point cloud (*a*) and the target scene cloud (*b*) are acquired from CARLA.

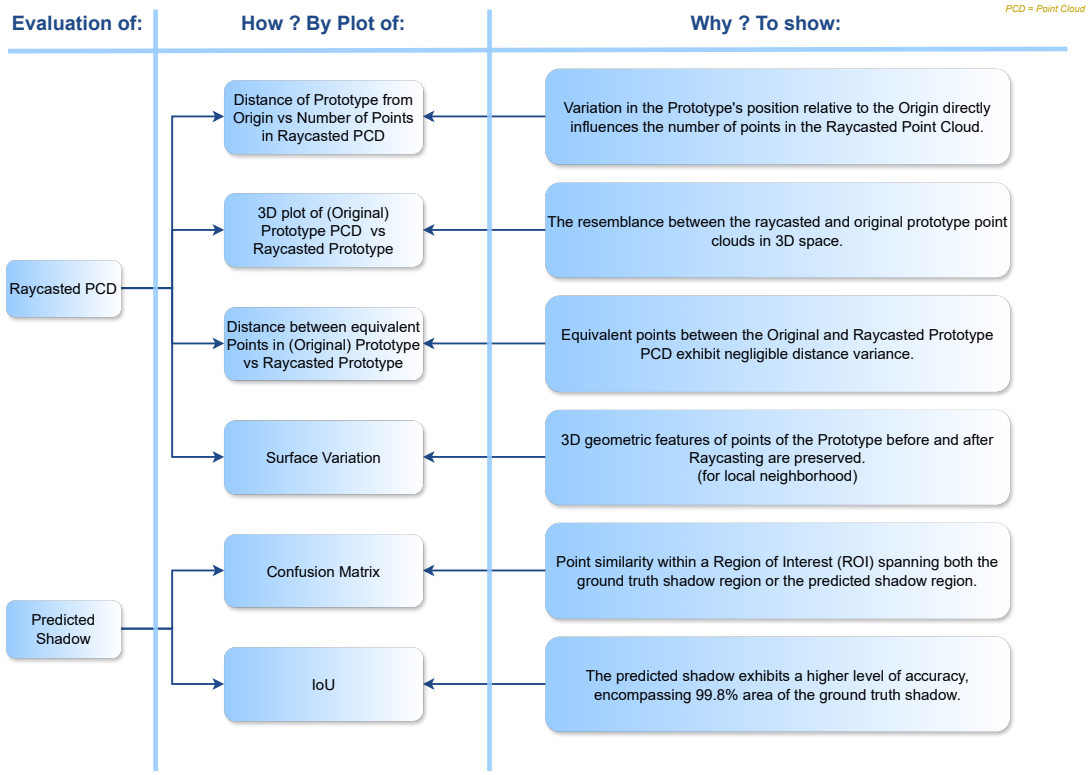


Figure 6.2: Evaluation Overview.

As outlined in figure 6.2, raycasted point cloud is analyzed through four graphical representations. A scatter plot is generated wherein the distance of the prototype (person / original prototype PCD) is varied within the target scene cloud from the origin, and subsequently, the raycasted point cloud is calculated. This analysis demonstrates a correlation between the number of raycasted points on the prototype surface and their proximity to the origin, indicating a decrease in the number of points as the distance from the origin increases. Additionally, a 3D plot comparing the original prototype and the raycasted prototype point cloud is constructed without transforming the prototype location to a different position, facilitating visual differentiation between the two sets of points in the cartesian coordinate system. This is represented visually as a comparison between the point cloud of the prototype (person) represented in figure 6.1 (*c*) and 6.1 (*d*).

Consistency between the original and raycasted prototype point clouds is affirmed by overlapping points observed in the 3D plot. Furthermore, a graph plotting the distance between corresponding points in the original and raycasted prototype PCD illustrates minimal distance discrepancies between equivalent points. Geometric feature distribution analysis, conducted using equivalent parameters, further confirms the similarity in local geometric features of points between the original prototype point cloud and the raycasted prototype point cloud.

To assess the precision of the predicted shadow, two distinct point clouds are captured: one containing a person and their casted shadow (referred to as the source scene cloud, represented by figure 6.1 (a)), and another identical point cloud lacking the person or their shadow but maintaining similar background information (termed the target scene cloud, represented by figure 6.1 (b)). A prototype (person) point cloud is extracted from the source scene cloud and seamlessly inserted into the target scene cloud without any transformation. Subsequently, the shadow cast by the prototype on the target scene cloud is computed. This is visually depicted as a comparison between the ground truth shadow (the shadow projected by the prototype as shown in figure 6.1 (c)) and the predicted shadow (the computed shadow of the raycasted prototype as shown in figure 6.1 (d)).

The correctness of the predicted shadow is observed by plotting the confusion matrix and by computing the IoU score as outlined in figure 6.2. A confusion matrix is constructed, wherein equivalent points between the two point clouds are identified within a Region of Interest (ROI) using nearest neighbor search. The confusion matrix reveals a high degree of similarity between corresponding points in the selected ROI of both point clouds. Additionally, a plot demonstrating the distance between points in the ROI of the predicted shadow cloud and their equivalents in the original shadow cloud further validates the accuracy of the confusion matrix. Intersection over Union (IoU) between the predicted shadow and the ground truth shadow region is calculated, confirming the precise prediction of the shadow cast by the prototype. The discrepancy in the area of the predicted shadow compared to the ground truth shadow is attributed to the larger surface area of the reconstructed surface (triangle mesh of person surface) relative to the original prototype(person).

6.1 Evaluation of Raycasted Point Cloud

A raycasted point cloud is generated by projecting rays from the origin to the surface of the prototype. The prototype surface is obtained through surface reconstruction into a triangular mesh. Methods like calculation of average distance,

comparing the number of points, and comparing the vanilla 3D points were used for the evaluation of raycasted point cloud. These methods are discussed below.

6.1.1 Average distance from the Origin vs Number of Points

With this step, we check the variation of the number of points in the raycasted point cloud when changing the position of the prototype before raycasting concerning origin. The distance between the prototype PCD and the origin of the target scene PCD was changed by transforming the extracted prototype to a different location on the target scene PCD. The distance from a prototype to the origin is calculated by calculating the Euclidean distance from the origin to each point in the prototype PCD. The average was obtained after dividing the sum of the distances of each point in the prototype PCD by the number of points. Rays were cast to the surface of the prototype after surface reconstruction. Change in points number based on different locations of the transformed prototype on the target scene PCD is observed.

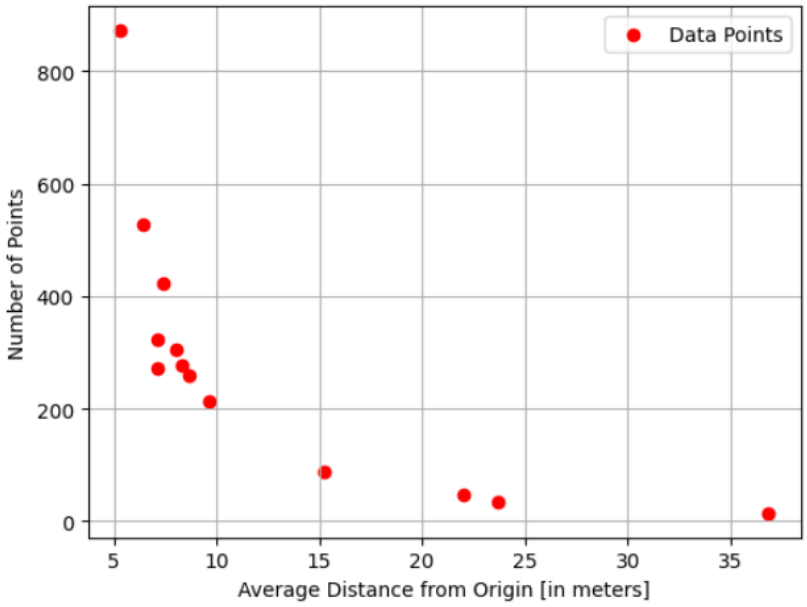


Figure 6.3: Averge distance of Prototype (Person) PCD from the Origin [in meters] vs Number of Points in Raycasted PCD.

In this figure 6.3, the original prototype is positioned at about 7 meters from the origin. The prototype was moved closer toward the origin and farther away from the origin of the target scene PCD, and the raycasted PCD of the prototype was calculated. From the graph, as shown in figure 6.3, it can be seen that the number

of points on the raycasted prototype decreases after an increase in the average distance between the prototype and the origin (more than 7 meters distance). The number of points of the raycasted prototype PCD increases when the prototype is transformed closer to the origin (less than 7 meters distance).

6.1.2 3D plot of the "Original" Prototype and Raycasted Prototype PCD

To visualize the similarities between the corresponding points on the original and raycasted prototype point cloud, a simple 3D plot in the XYZ axis can be analyzed. For this evaluation, the difference between the original prototype and the raycasted prototype is shown where the raycasting was done without transforming the position of the prototype PCD on the target scene cloud.

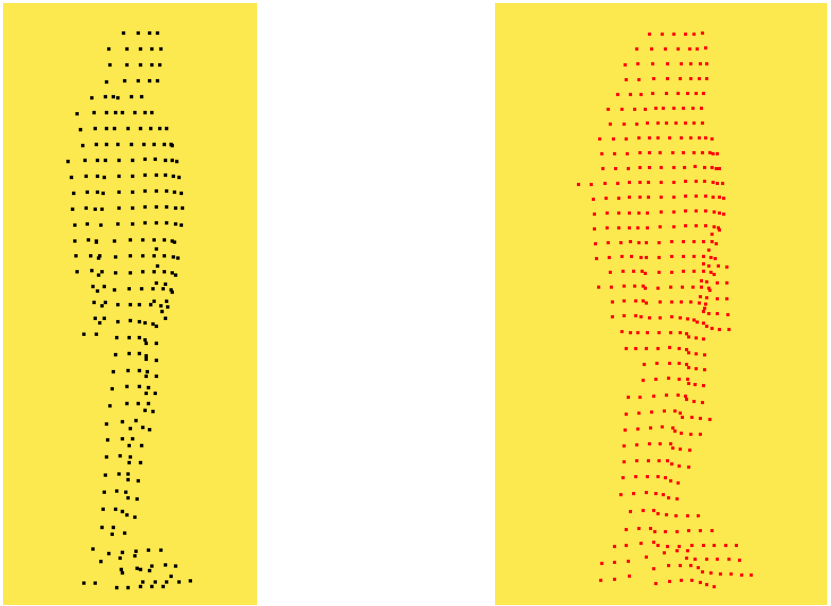


Figure 6.4: Visualization of Point Clouds (a) Original Prototype PCD (b) Raycasted PCD of Prototype.

Figure 6.4 (*a*) shows the point cloud of the prototype (person) before raycasting (by the black color points). (*b*) shows the point cloud of the prototype evaluated after raycasting (by the red color points). The raycasted point cloud in figure 6.4 (*b*) is computed without transforming the orientation of the prototype to compare the difference in corresponding points between the two PCD. Symbolically, it is depicted through the comparison between figure 6.1 (*c*) and 6.1 (*d*).

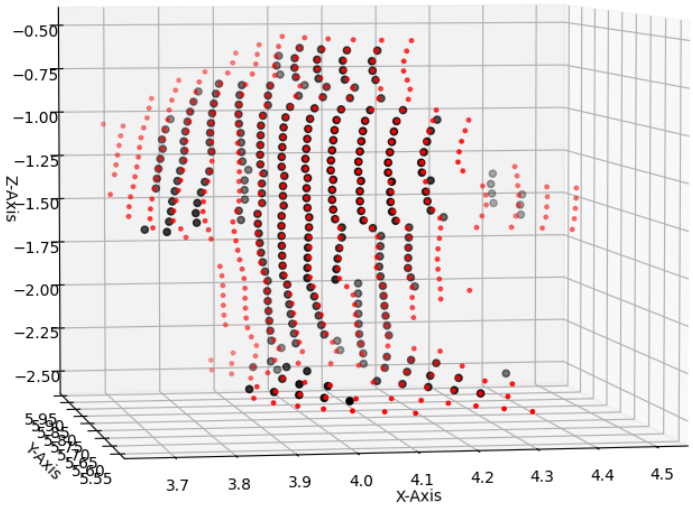


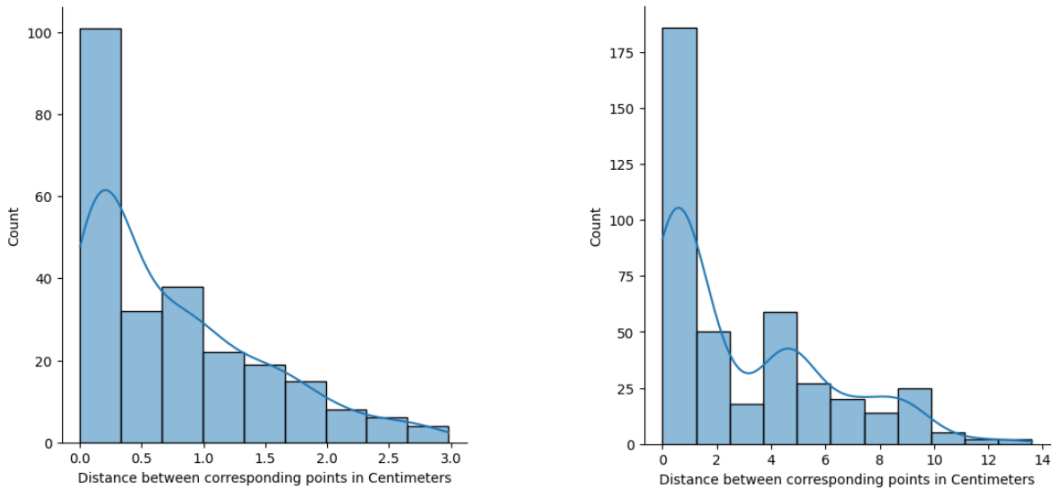
Figure 6.5: 3D XYZ Plot of Original Prototype PCD (black) vs Raycasted Prototype PCD (red).

The graph can be viewed in figure 6.5. The black points in the graph correspond to the point in the prototype before raycasting (representing the PCD shown in figure 6.4 (*a*)). The red points in the graph correspond to the point of the prototype after raycasting (representing the PCD shown in figure 6.4 (*b*)). It can be observed that most of the red points overlap the black points in space, which proves that there is a low difference between the original prototype PCD and the prototype PCD after raycasting. Most of the red points that do not overlap with the black points are due to the bigger area of the reconstructed surface of the prototype. Proper filtering of the reconstructed surface triangle mesh could result in the removal of the red outliers.

6.1.3 Distance between the "Equivalent" Points in the "Original" Prototype and Raycasted Prototype PCD

Distance between the corresponding points was calculated between the original prototype PCD and the prototype PCD after raycasting. The raycasting was performed on the original prototype surface without transformation. We calculated the average distance between the black points and the corresponding red points of figure 6.5. To find the equivalent points between two clouds, the nearest neighbor points between the two PCD is computed. Spheres with different values of radius

were experimented with for the calculation of the nearest neighbor. Neighbors for a point in the original prototype PCD were calculated from the raycasted PCD that falls within the radius of the sphere, centered around the ego point. The neighbor that has the minimum distance from the ego point is chosen. This point in the raycasted prototype PCD is considered the equivalent point of the ego point in the original prototype PCD. Sphere size can be lower or higher for neighbor search. This is because whatever the size of the sphere, we are extracting a point from the neighborhood inside the sphere that has the least distance from the ego point or center of the sphere. However, too low value for radius might lead to not finding any neighbors.



(a) Search Radius for Nearest Neighbor is 3 cm.

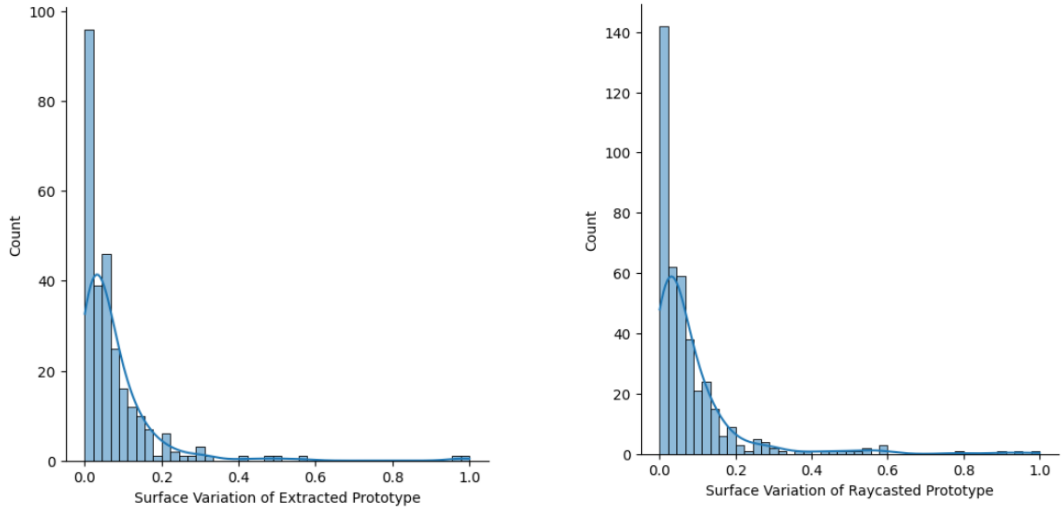
(b) Search Radius for Nearest Neighbor is 20 cm.

Figure 6.6: Distance (in centimeters) between the "equivalent" points in the Prototype(Original) and Raycasted Prototype PCD.

Figure 6.6 shows plots of the distance between the black points and their equivalent red points from figure 6.5. Figure (a) is plotted after finding the nearest neighbor with a search within the sphere of radius 3 cm centered around an ego point. When searching the neighbor with a sphere of radius 3cm, out of 408 points in the raycasted PCD, 163 number of equivalent points were not found in the original PCD. Using a higher radius value for neighbor search (20 cm), all the points in the red points were found to have their equivalent black points. This is shown by part (b) of the figure. The figure shows that a high concentration of equivalent points is situated within 0-3 cm distance apart from eachother.

6.1.4 Surface Variation Plot

Surface variation for the point cloud represented by red points and black points in figure 6.5 was calculated and plotted.



(a) Surface Variation Plot of Extracted Prototype

(b) Surface Variation Plot of Raycasted Prototype

Figure 6.7: Surface Variation Plot between the Extracted (Original) Prototype and Raycasted Prototype PCD.

For this experiment, a radius of 30cm with a maximum value of the nearest neighbor was taken to be 6 as a parameter to calculate the nearest neighbor using KDTree in open3d. Using the formula for surface variation as described in equation 2.20, surface variation for each point in the PCD is calculated. The obtained result is normalized on a scale of 0 to 1 and the corresponding plot is displayed as shown in figure 6.7. The figure shows the similarities in local geometric features (surface variation) of points between the two point clouds.

6.2 Evaluation of Casted Shadow

For the evaluation purpose, point cloud representing a "flat" surface was captured from the CARLA. This point cloud represents a target scene where we would like to do a shadow casting. A prototype "person" was spawned in some location in the target region without changing the orientation of the lidar sensor and the point cloud was saved. This point cloud corresponds to the source scene PCD.

The prototype PCD is extracted from the source scene cloud and placed in the target scene PCD. Shadow casting by the prototype is performed on the target scene PCD and the final augmented point cloud is saved. The prototype PCD of the person is not transformed in the target scene. The difference between the target scene cloud and the source scene cloud is the availability of the person and its casted shadow, the rest of the points are similar in both point clouds, i.e. background information is similar on both point clouds. Figuratively, it is illustrated by assessing the person's cast shadow in figure 6.1 (*c*) and 6.1 (*d*). A region is selected from the source scene cloud which constitutes of shadow projected by the prototype. This region is the ground truth shadow region as shown in figure 6.8 (*a*). The corresponding region is selected on the target scene cloud where the raycasted point cloud of the prototype is computed and the shadow projected by the new prototype point cloud is calculated. This region is the predicted shadow region as shown in figure 6.8 (*b*).

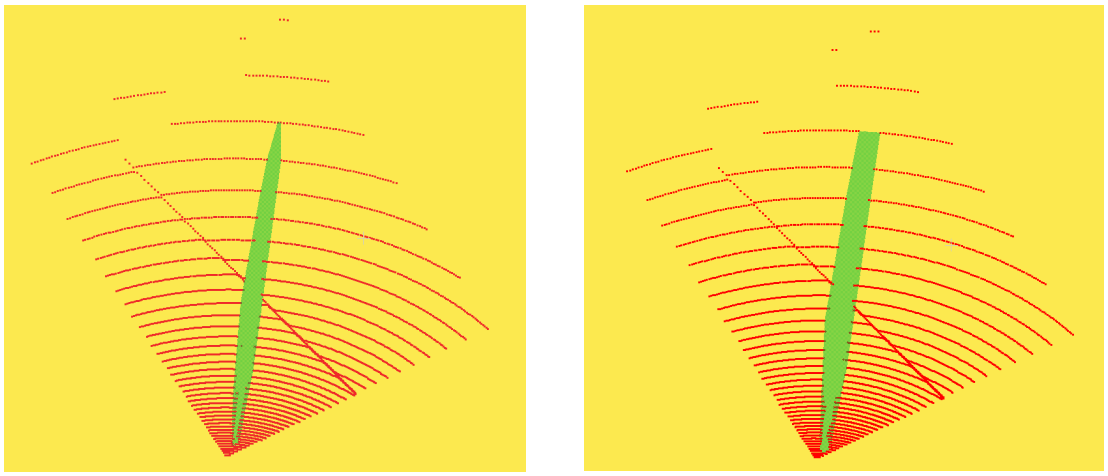


Figure 6.8: Shadow Casted by the Prototype (person) PCD on a ROI, represented by green color region (a) Ground Truth Shadow (b) Predicted Shadow.

Figure 6.8 shows the shadow projected by the prototype. The green-colored region on both (*a*) and (*b*) represents the approximate shadow projected by the prototype. (*a*) represents ground truth shadow region and (*b*) represents predicted shadow region. Shadow is calculated and computed for only a selected region on the target scene cloud. The selected Region of Interest (ROI) is shown by the red color point in figure 6.8.

6.2.1 Confusion Matrix

Comparison between the shadow is done by finding the corresponding points in ROI where the ground truth shadow and the anticipated shadow is projected, i.e. between points in figure 6.8 (*a*) and (*b*). The nearest neighbor search was done to find the equivalent points in the selected ROI of the ground truth shadow and anticipated shadow cloud. The radius of the sphere for the neighbor search was taken to be 3 centimeters. If no neighbors were found on the corresponding point cloud within this region, the point was considered to be not found. From the found neighbors, based on some threshold value, the neighbors were filtered. If there exists a neighbor after filtering with some threshold distance, the point was considered to be found on both point clouds. If not, the point was labeled as not found. A confusion matrix is plotted as shown in figure 6.9.

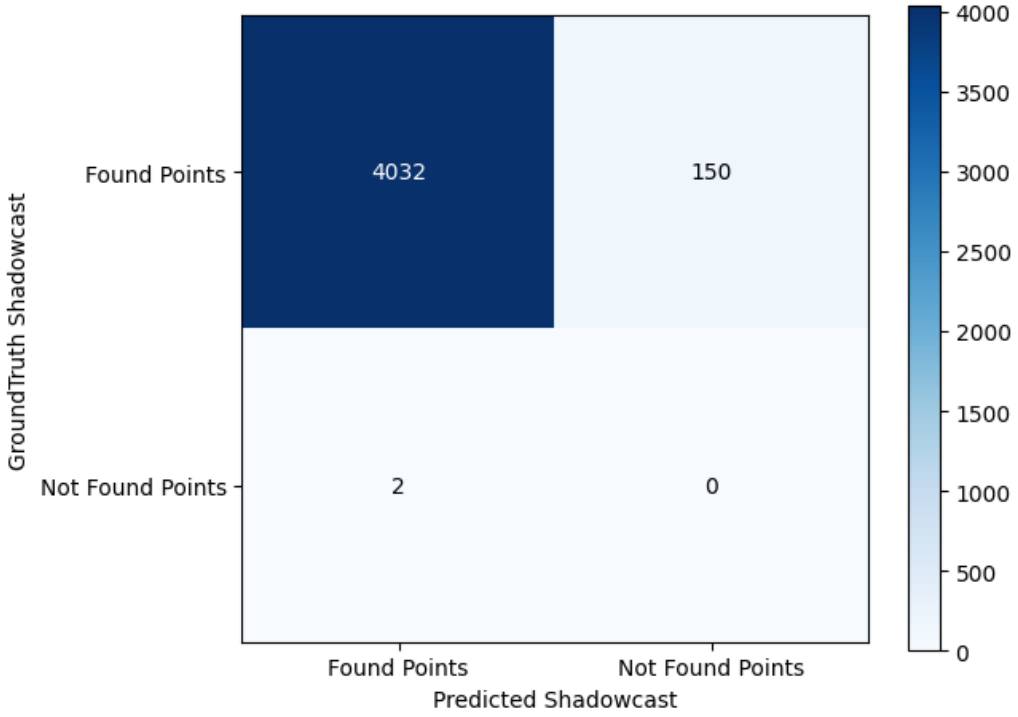


Figure 6.9: Confusion matrix.

Some observations can be made from the confusion matrix in figure 6.9. Equivalent points for 150 points from the ROI containing the ground truth shadow were not found in the ROI containing the predicted shadow region, i.e between 6.8 (*a*) and 6.8 (*b*). 2 points available on the ROI containing predicted shadow PCD were not found in the ROI containing the ground truth shadow region. 4032 corresponding points were found in both regions. Unable to say about the points not found on

both, the true negative section in the confusion matrix is labeled as 0. From the confusion matrix and using equations 2.23, 2.24, 2.25, and 2.26, the following metrics are observed as shown in table 6.1.

Metrics	Value
True Positive	4032
True Negative	0
False Positive	2
False Negative	150
Accuracy	0.963
Precision	0.999
Recall	0.964
F1-Score	0.981

Table 6.1: Performance Metrics Calculated for ROI containing GT Shadow Region and Predicted Shadow Region.

The calculation proves that the shadow projection has a high accuracy and a very low error rate.

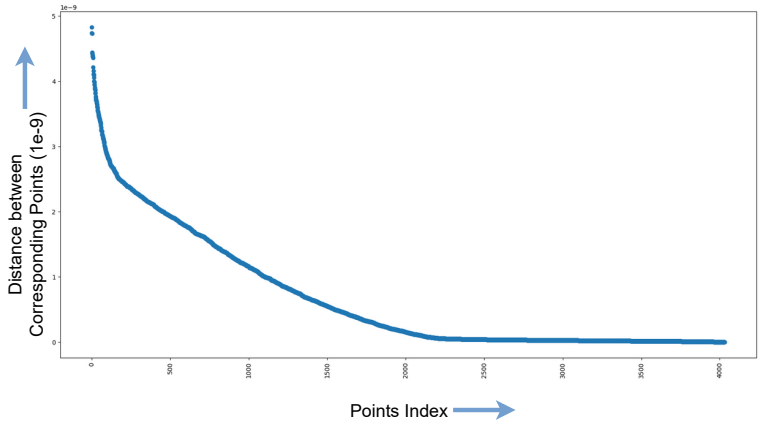


Figure 6.10: Distance between corresponding points for selected ROI containing GT Shadow and Predicted Shadow of Prototype PCD.

Figure 6.10 shows the distance between the corresponding points in the ROI containing the predicted shadow region and the ground truth shadow region. The figure verifies that the points that were considered to be similar in both regions

are very similar as the distance between the equivalent points is in the order of 10^{-9} meters.

6.2.2 Intersection over Union

For the calculation of Intersection over Union (IoU), the contour points of the shadow region were first extracted for both the point clouds(ground truth shadow and predicted shadow region). The contour points are represented by the boundary points of the green-colored region of figure 6.8. The extracted 3d-points were projected to the XY plane; i.e. by making the z-value of each point zero. The area of the region occupied by the surface bounding the contour points (i.e. the shadow region) was calculated. This is shown approximately by green-colored region in figure 6.8. The absolute region representing the ground truth shadow and predicted shadow is shown in figure 6.11.

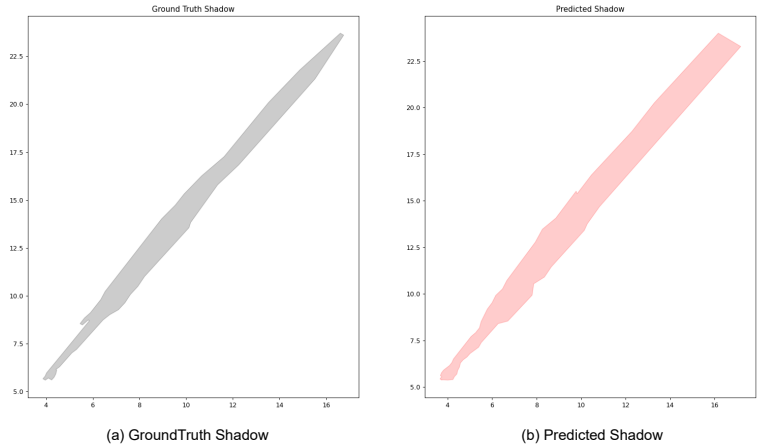
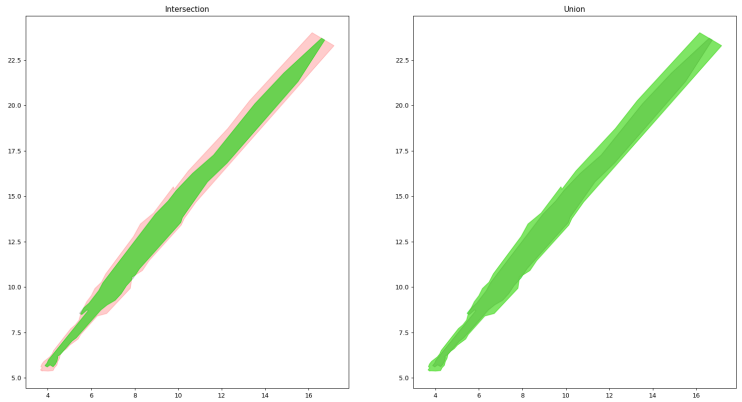


Figure 6.11: Plot of Ground Truth Shadow and Predicted Shadow.

Figure 6.12 (a) shows the intersection region between the ground truth shadow and predicted shadow by green color. (b) shows the union area between the two regions.

$$PP_{\mathrm{GT}} = \left(\frac{A_{\mathrm{GT}} - A_{\mathrm{GT\backslash Pred}}}{A_{\mathrm{GT}}} \right) \times 100\% \tag{6.1}$$

Equation 6.1 gives the percentage of GT shadow accurately predicted from the project. PP_{GT} represents the Prediction Percentage of GT Shadow that was



(a) Area of Intersection (b) Area of Union

Figure 6.12: Intersection and Union between the GT Shadow Region and Predicted Shadow Region.

accurately predicted. A_{GT} represents the Area of the Ground Truth (GT) shadow region. $A_{\mathrm{GT}\backslash\mathrm{Pred}}$ is the Area in GT shadow region that is not predicted as the shadow region of Prototype.

Observations can be made as shown in table 6.2.

Metrics	**Value**
Area of Ground Truth (GT) Shadow Region	17.318 m^2
Area of Predicted Shadow Region	25.886 m^2
Area of Intersection	17.293 m^2
Area of Union	25.910 m^2
Intersection over Union (IoU)	0.667
Area in Ground Truth Shadow not in Predicted Shadow	0.0248 m^2
Area in Predicted Shadow not in Ground Truth Shadow	8.592 m^2
Percentage of GT Shadow Region accurately Predicted	99.856 %

Table 6.2: Metrics assessing the Intersection and Union between GT Shadow Region and Predicted Shadow Region.

Using equation 6.1, it is calculated that about 99.856% of the ground truth shadow region was predicted. The IoU score was 0.667. It can be improved by filtering the reconstructed surface mesh to represent the prototype accurately.

7 Discussion and Conclusion

Novel point cloud scenarios within the context of Highly Automated Driving (HAD) systems were generated by injecting a Point Cloud (PCD) object from a source scene PCD into a target scene PCD. By using the orientation of the prototype in the source scene PCD, the prototype was transformed into a target location on the target point cloud. After the transformation, a new point cloud for the transformed prototype was calculated and the shadow casted by the newly computed prototype PCD on the target scene PCD was observed.

7.1 Challenges

The initial phase of experimental data collection from CARLA posed hurdles due to the necessity of meeting system requirements for running CARLA and ensuring compatibility with all requisite versions of dependent software such as ROS2, carla-ros-brige, and the operating system. Placing the extracted prototype point cloud data (PCD) into a designated target location also presented difficulties. Initial attempts using the Iterative Closest Point (ICP) registration algorithm proved ineffective, particularly in areas with sparse points in the point cloud. Instead, successful placement of the extracted prototype PCD was achieved by transforming it based on the centroid of the filtered ground plane. To address the challenge of unknown surfaces of a prototype that are not visible to the LiDAR sensor, the extracted prototype point cloud is rotated during transformation within the target scene PCD. This rotation is determined by utilizing the centroid of the prototype both before and after translation. Additionally, obtaining a realistic point cloud of the prototype that accurately replicated the real LiDAR beam scan patterns at various positions within the target scene Region of Interest (ROI) posed a challenge. This was addressed by leveraging the points within the target point cloud to simulate the laser beams for raycasting, thereby generating a realistic LiDAR point cloud of the prototype at a desired ROI without requiring knowledge of the LiDAR configuration. Furthermore, accurately calculating the realistic shadow projected by the prototype onto the target ROI of the scene cloud initially proved to be error-prone when using the Hidden Point Removal (HPR) algorithm. However, the process was refined by tracking casted rays to the target

ROI, facilitating the computation of a realistic shadow projected by the prototype onto the ROI of target scene PCD.

7.2 Future Works

The project could be improved in various ways.

- Extraction of the prototype PCD from the source scene PCD is done by using surface variation only. Several other geometric features could be used to remove the false positives. This is required in the case when working with real-world LiDAR data.

- Experiment is done for the extraction of the pedestrian point cloud. Methods need to be tested for various objects such as cars, bikes, etc.

- By scanning an object from a different angle view and by the point cloud registrations technique, a point cloud representing a complete hollow 3D object could be constructed. Such objects could be rotated in any way in the target location. This results in an increase in the variability of scenarios.

- The centroid of points from the remaining points in the ROI on the source scene PCD is calculated after cropping the prototype PCD, which is later used in the transformation of the prototype on the target scene PCD. If the remaining points after filtering the prototype PCD consist of points from the prototype or higher z-values than the ground plane, then the transformation would be wrong. Eg. the transformed prototype point cloud lies 1 meter above the target ground plane i.e. levitating in the air.

- Experiment is done on flat surfaces point cloud. Improvements could be made in the future by adding support for non-planar surfaces.

- The required placement of objects needed for scenario-based testing should be properly analyzed.

- The Prototype PCD could only be extracted if it is standing on the ground plane point cloud. There could be a scenario where a prototype could be very near to the LiDAR sensor where there is no ground plane PCD. Because of the field of view of the LiDAR sensor, only the upper region of prototype surface is scanned by the LiDAR scanner. The ground plane where the prototype is situated or the lower region of prototype points is

not visible. Current method needs to be updated for extracting such objects as there is no ground plane point cloud to calculate a reference centroid for transformation.

- The Raycasting process depends on the original direction of the laser on the target scene PCD. If the points represented on the target scene PCD are erroneous, then the resulting raycasted points will be erroneous.

- Accurately representing the surface of the prototype PCD after surface reconstruction would result in a reduced error during raycasting and shadowcasting.

- To replicate the behavior of LiDAR sensors in real-world settings, random dropout, and noise can be introduced.

- At present, the intensity of points can be determined by assigning an intensity function to points according to how far they are from the LiDAR source. However, there is a need to develop methodologies that take into account and preserve the reflective characteristics of the prototype surface.

- Currently, manual selection of a region is done for the insertion of the extracted prototype in a desired region of the target scene PCD. Finding an appropriate region automatically for the insertion of the prototype on the target region of the target scene PCD could remove the step for manual intervention. If the manual selection of the ROI step is eliminated then by using the pre-saved extracted prototype PCD, real-time injection on the target scene PCD could be possible.

- In this thesis, the challenge of class imbalance is addressed by leveraging the reuse of existing scenes. However, it is imperative to further investigate methods that accommodate scenarios without the reuse of pre-existing scenes.

7.3 Conclusion

Thus, novel scenario-driven test data, in the context of Highly Automated Driving (HAD) Systems, were synthesized through LiDAR data augmentation. This involved extracting a prototype point cloud utilizing the geometric features of 3D points in the local neighborhood of the point cloud. The prototype surface was then reconstructed at a designated location within the target scene PCD, and new point

clouds for the prototype were calculated based on the original direction of laser rays in the target scene PCD. Subsequently, shadows projected by the newly computed prototype PCD onto the target scene PCD were determined. Thus, resulting in the creation of a new point cloud representing the altered scene. Evaluation of the approach was conducted by analyzing two point clouds representing identical backgrounds: one with a ground truth prototype (foreground object) and another without. This methodology allowed for the examination of discrepancies between the ground truth and the calculated new point cloud without the necessity of considering differing background point clouds. (Padusinski et al., 2024) explains the necessity of interpreting objects at different levels of granularity grades to identify deficiencies in known general machine vision strategies. Disregarding the intensity data associated with the prototype surface would lead to the loss of crucial granularity information intrinsic to the prototype's structure. Using the reconstructed surface to calculate the point cloud of the object gives a raycasted point cloud with some discrepancies due to varying surface area. Most importantly, the intensities of the original PCD are not taken into account in the raycasted PCD as the surface reconstruction step removes that information by creating a CAD model from extracted prototype PCD. In case of the recombining of the extracted prototype PCD to the target scene PCD in its original position without transformation, the extracted prototype PCD can be merged with the target scene PCD instead of the raycasted PCD to save the information inherent to the original prototype object (information such as point cloud intensity based on the reflectivity of the prototype surface). In the pursuit of generating a highly accurate augmented point cloud, reconstructed mesh can be used in calculation of casted shadow, while retaining the original prototype points in their initial orientation. This approach preserves the integrity of the method, as disregarding intensity values and utilizing mesh in a different position would compromise the validity of the process. By incorporating the mesh for shadow casting only and maintaining the original position of the prototype points results in a high fidelity augmented point cloud.

The study effectively tackled the challenge of class imbalance between foreground objects and background scenes within Highly Automated Driving (HAD) systems through an systematic strategy generating as valid as possible augmented LiDAR data leveraging LiDAR data augmentation. Significantly, this was achieved without the reliance on annotated LiDAR data during prototype extraction and without prior knowledge of the LiDAR characteristics of the target dataset to be augmented. This novel process circumvents the necessity for 3D semantic models or pre-existing CAD models. Furthermore, the augmentation process demonstrated its potential to mitigate deficiencies within scenarios by seamlessly integrating foreground objects into background scenes, thereby generating diverse and novel concrete scenarios. The methodology investigated in this study could benefit from improvements in the surface reconstruction phase. By incorporating the automated selection of

regions within the target scene, the process could facilitate real-time injection of foreground objects and the generation of extensive datasets featuring diverse scenarios, all without requiring manual intervention. Such enhancements hold promise for integration into testing phases aimed at assessing the performance of Highly Automated Driving (HAD) systems, thereby advancing the evaluation of system efficacy.

Bibliography

[1] Hassan Abu Alhaija et al. “Augmented Reality Meets Computer Vision : Efficient Data Generation for Urban Driving Scenes”. In: *CoRR* abs/1708.01566 (2017). arXiv: 1708.01566. url: http://arxiv.org/abs/1708.01566.

[2] Pei An et al. “RS-Aug: Improve 3D Object Detection on LiDAR With Realistic Simulator Based Data Augmentation”. In: *IEEE Transactions on Intelligent Transportation Systems* 24.9 (2023), pp. 10165–10176. doi: 10.1109/TITS.2023.3266727.

[3] Gerrit Bagschik, Till Menzel, and Markus Maurer. “Ontology based Scene Creation for the Development of Automated Vehicles”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 1813–1820. doi: 10.1109/IVS.2018.8500632.

[4] J. Behley et al. *API for SemanticKITTI*. [accessed 01 January 2024]. 2019. url: https://github.com/PRBonn/semantic-kitti-api/blob/master/config/semantic-kitti-all.yaml.

[5] J. Behley et al. “SemanticKITTI A Dataset for Semantic Scene Understanding of LiDAR Sequences”. In: *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*. 2019.

[6] F. Bernardini et al. “The ball-pivoting algorithm for surface reconstruction”. In: *IEEE Transactions on Visualization and Computer Graphics* 5.4 (1999), pp. 349–359. doi: 10.1109/2945.817351.

[7] Andrew Best et al. “AutonoVi-Sim: Autonomous Vehicle Simulation Platform with Weather, Sensing, and Traffic Control”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 1161–11618. doi: 10.1109/CVPRW.2018.00152.

[8] Olivier Chapelle et al. “Vicinal Risk Minimization”. In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2000. url: https://proceedings.neurips.cc/paper_files/paper/2000/file/ba9a56ce0a9bfa26e8ed9e10b2cc8f46-Paper.pdf.

[9] Yunlu Chen et al. “PointMixup: Augmentation for Point Clouds”. In: *CoRR* abs/2008.06374 (2020). arXiv: 2008.06374. url: https://arxiv.org/abs/2008.06374.

[10] Shuyang Cheng et al. “Improving 3D Object Detection through Progressive Population Based Augmentation”. In: *CoRR* abs/2004.00831 (2020). arXiv: 2004.00831. url: https://arxiv.org/abs/2004.00831.

[11] Jaeseok Choi, Yeji Song, and Nojun Kwak. "Part-Aware Data Augmentation for 3D Object Detection in Point Cloud". In: *CoRR* abs/2007.13373 (2020). arXiv: 2007.13373. url: https://arxiv.org/abs/2007.13373.

[12] State of California Department of Motor Vehicles. *Disengagement Reports*. [accessed 11 February 2024]. 2023. url: https://www.dmv.ca.gov/portal/vehicle-industry-services/autonomous-vehicles/disengagement-reports/.

[13] Terrance Devries and Graham W. Taylor. "Improved Regularization of Convolutional Neural Networks with Cutout". In: *CoRR* abs/1708.04552 (2017). arXiv: 1708.04552. url: http://arxiv.org/abs/1708.04552.

[14] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[15] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. "On the shape of a set of points in the plane". In: *IEEE Transactions on Information Theory* 29.4 (1983), pp. 551–559. doi: 10.1109/TIT.1983.1056714.

[16] Jin Fang, Dingfu Zhou, et al. "Augmented LiDAR Simulator for Autonomous Driving". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1931–1938. doi: 10.1109/LRA.2020.2969927.

[17] Jin Fang, Xinxin Zuo, et al. "LiDAR-Aug: A General Rendering-based Augmentation Framework for 3D Object Detection". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 4708–4718. doi: 10.1109/CVPR46437.2021.00468.

[18] Kaspar Fischer. *Introduction to Alpha Shapes*. [accessed 03 February 2024]. Stanford University. 2011. url: https://graphics.stanford.edu/courses/cs268-11-spring/handouts/AlphaShapes/as_fisher.pdf.

[19] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3354–3361.

[20] Yulan Guo et al. *Deep Learning for 3D Point Clouds: A Survey*. 2020. arXiv: 1912.12033 [cs.CV].

[21] Martin Hahner, Dengxin Dai, et al. "Quantifying Data Augmentation for LiDAR based 3D Object Detection". In: *CoRR* abs/2004.01643 (2020). arXiv: 2004.01643. url: https://arxiv.org/abs/2004.01643.

[22] Martin Hahner, Christos Sakaridis, Mario Bijelic, et al. *LiDAR Snowfall Simulation for Robust 3D Object Detection*. 2022. arXiv: 2203.15118 [cs.CV].

[23] Martin Hahner, Christos Sakaridis, Dengxin Dai, et al. "Fog Simulation on Real LiDAR Point Clouds for 3D Object Detection in Adverse Weather". In: *CoRR* abs/2108.05249 (2021). arXiv: 2108.05249. url: https://arxiv.org/abs/2108.05249.

[24] Mario Herger. *Cruise Pusblishes Crash Report*. [accessed 01 February 2024]. 2024. url: https://thelastdriverlicenseholder.com/2024/01/25/cruise-publishes-crash-report/.

[25] WuLing Huang et al. "Autonomous vehicles testing methods review". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 163–168. doi: 10.1109/ITSC.2016.7795548.

[26] Hyper-Tech Advanced Systems Ltd Hyper-Tech. *Velodyne HDL-64E Datasheet*. [accessed 03 February 2024]. 2014. url: https://hypertech.co.il/wp-content/uploads/2015/12/HDL-64E-Data-Sheet.pdf.

[27] Matthew Johnson-Roberson et al. *Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks?* 2017. arXiv: 1610.01983 [cs.CV].

[28] Nidhi Kalra and Susan M. Paddock. "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" In: *Transportation Research Part A: Policy and Practice* 94 (2016), pp. 182–193. issn: 0965-8564. doi: https://doi.org/10.1016/j.tra.2016.09.010. url: https://www.sciencedirect.com/science/article/pii/S0965856416302129.

[29] Sagi Katz, Ayellet Tal, and Ronen Basri. "Direct visibility of point sets". In: vol. 26. July 2007. doi: 10.1145/1275808.1276407.

[30] Sihyeon Kim et al. "Point Cloud Augmentation with Weighted Local Transformations". In: *CoRR* abs/2110.05379 (2021). arXiv: 2110.05379. url: https://arxiv.org/abs/2110.05379.

[31] Jason Kong et al. "Kinematic and dynamic vehicle models for autonomous driving control design". In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. 2015, pp. 1094–1099. doi: 10.1109/IVS.2015.7225830.

[32] Daeun Lee, Jongwon Park, and Jinkyu Kim. *Resolving Class Imbalance for LiDAR-based Object Detector by Dynamic Weight Average and Contextual Ground Truth Sampling*. 2022. arXiv: 2210.03331 [cs.CV].

[33] Impyeong Lee and Anton Schenk. "Perceptual organization of 3D surface points". In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 34 (Jan. 2002).

[34] Joohyun Lee et al. "Dual Adaptive Data Augmentation for 3D Object Detection". In: *2023 14th International Conference on Information and Communication Technology Convergence (ICTC)*. 2023, pp. 1732–1737. doi: 10.1109/ICTC58733.2023.10393861.

[35] Ruihui Li et al. "PointAugment: an Auto-Augmentation Framework for Point Cloud Classification". In: *CoRR* abs/2002.10876 (2020). arXiv: 2002.10876. url: https://arxiv.org/abs/2002.10876.

[36] Yueyuan Li et al. *Choose Your Simulator Wisely: A Review on Open-source Simulators for Autonomous Driving*. 2023. arXiv: 2311.11056 [cs.RO].

[37] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *CoRR* abs/1708.02002 (2017). arXiv: 1708.02002. url: http://arxiv.org/abs/1708.02002.

[38] Sivabalan Manivasagam, Ioan Andrei Bârsan, et al. "Towards Zero Domain Gap: A Comprehensive Study of Realistic LiDAR Simulation for Autonomy Testing". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 8272–8282.

[39] Sivabalan Manivasagam, Shenlong Wang, et al. "LiDARsim: Realistic LiDAR Simulation by Leveraging the Real World". In: *CoRR* abs/2006.09348 (2020). arXiv: 2006.09348. url: https://arxiv.org/abs/2006.09348.

[40] Jiageng Mao et al. "One Million Scenes for Autonomous Driving: ONCE Dataset". In: 2021.

[41] Ravish Mehra et al. "Visibility of noisy point cloud data". In: *Computers and Graphics* 34.3 (2010). Shape Modelling International (SMI) Conference 2010, pp. 219–230. issn: 0097-8493. doi: https://doi.org/10.1016/j.cag.2010.03.002. url: https://www.sciencedirect.com/science/article/pii/S0097849310000397.

[42] Till Menzel, Gerrit Bagschik, and Markus Maurer. "Scenarios for Development, Test and Validation of Automated Vehicles". In: *2018 IEEE Intelligent Vehicles Symposium (IV)* (2018), pp. 1821–1827. url: https://api.semanticscholar.org/CorpusID:13905150.

[43] Matthew Bolitho Michael Kazhdan and Hugues Hoppe. "Poisson Surface Reconstruction". In: *Eurographics Symposium on Geometry Processing* (2006).

[44] Andres Milioto et al. "RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation". In: Nov. 2019, pp. 4213–4220. doi: 10.1109/IROS40897.2019.8967762.

[45] Tergel Molom-Ochir. *Autonomous Driving and Its Sensor Technology*. [accessed 18 March 2024]. 2019. url: https://orise.orau.gov/resources/stem/documents/scholarships/autonomous-driving-and-its-sensor-technology.pdf.

[46] Alexey Nekrasov et al. "Mix3D: Out-of-Context Data Augmentation for 3D Scenes". In: *CoRR* abs/2110.02210 (2021). arXiv: 2110.02210. url: https://arxiv.org/abs/2110.02210.

[47] Jarryd Neves. *Official: Pedestrian Detection Is Useless In The Dark*. [accessed 18 February 2024]. 2022. url: https://carbuzz.com/news/official-pedestrian-detection-is-useless-in-the-dark.

[48] Hubert Padusinski et al. *The Machine Vision Iceberg Explained: Advancing Dynamic Testing by Considering Holistic Environmental Circumstances*. 2024. arXiv: 2401.14831 [cs.RO].

[49] PapersWithCode. *3D Semantic Segmentation*. [accessed 05 February 2024]. 2023. url: https://paperswithcode.com/task/3d-semantic-segmentation.

[50] Changwoo Park, Seunghwan Chung, and Hyeongcheol Lee. "Vehicle-in-the-Loop in Global Coordinates for Advanced Driver Assistance System". In: *Applied Sciences* 10 (Apr. 2020), p. 2645. doi: 10.3390/app10082645.

[51] Jewoo Park et al. "An Automotive LiDAR Performance Test Method in Dynamic Driving Conditions". In: *Sensors* 23.8 (2023). issn: 1424-8220. doi: 10.3390/s23083892. url: https://www.mdpi.com/1424-8220/23/8/3892.

[52] Mark Pauly, Markus H. Gross, and Leif Kobbelt. "Efficient simplification of point-sampled surfaces". In: *IEEE Visualization, 2002. VIS 2002.* (2002), pp. 163–170. url: https://api.semanticscholar.org/CorpusID:14952977.

[53] Mauricio Peña. *Voluntary recall of our previous software.* [accessed 20 March 2024]. 2024. url: https://waymo.com/blog/2024/02/voluntary-recall-of-our-previous-software/.

[54] Robert Platt. *Point Cloud Processing.* [accessed 01 February 2024]. Northeastern University. 2016. url: https://www.khoury.northeastern.edu/home/rplatt/cs5335_fall2016/slides/point_clouds.pdf.

[55] Viraj Prabhu et al. *Bridging the Sim2Real gap with CARE: Supervised Detection Adaptation with Conditional Alignment and Reweighting.* 2023. arXiv: 2302.04832 [cs.CV].

[56] Charles R. Qi et al. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.* 2017. arXiv: 1612.00593 [cs.CV].

[57] Charles Ruizhongtai Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *CoRR* abs/1706.02413 (2017). arXiv: 1706.02413. url: http://arxiv.org/abs/1706.02413.

[58] Yuan Ren, Siyan Zhao, and Liu Bingbing. "Object Insertion Based Data Augmentation for Semantic Segmentation". In: *2022 International Conference on Robotics and Automation (ICRA).* 2022, pp. 359–365. doi: 10.1109/ICRA46639.2022.9811816.

[59] Stefan Riedmaier et al. "Validation of X-in-the-Loop Approaches for Virtual Homologation of Automated Driving Functions". In: May 2018.

[60] Radu Bogdan Rusu. "Semantic 3D object maps for everyday manipulation in human living environments". PhD thesis. Computer Science department, Technische Universität München, Germany, 2009.

[61] Kwonyoung Ryu, Soonmin Hwang, and Jaesik Park. "Instant Domain Augmentation for LiDAR Semantic Segmentation". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2023, pp. 9350–9360. doi: 10.1109/CVPR52729.2023.00902.

[62] Florian Sauerbeck et al. "Learn to See Fast: Lessons Learned From Autonomous Racing on How to Develop Perception Systems". In: *IEEE Access* 11 (2023), pp. 44034–44050. doi: 10.1109/ACCESS.2023.3272750.

[63] Petr Šebek et al. *Real3D-Aug: Point Cloud Augmentation by Placing Real Objects with Occlusion Handling for 3D Detection and Segmentation.* 2022. arXiv: 2206.07634 [cs.CV].

[64] Shivanand Venkanna Sheshappanavar, Vinit Veerendraveer Singh, and Chandra Kambhamettu. "PatchAugment: Local Neighborhood Augmentation in Point Cloud Classification". In: *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2021, pp. 2118–2127. doi: 10.1109/ICCVW54120.2021.00240.

[65] Patrice Simard, Yann Lecun, and John Denker. "Transformation Invariance in Pattern Recognition - Tangent Distance and Tangent Propagation". In: (May 2000).

[66] Hao Su et al. "Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views". In: *CoRR* abs/1505.05641 (2015). arXiv: 1505.05641. url: http://arxiv.org/abs/1505.05641.

[67] TheWaymoTeam. *Off road, but not offline: How simulation helps advance our Waymo Driver*. [accessed 20 March 2024]. 2020. url: https://waymo.com/blog/2020/04/off-road-but-not-offline--simulation27/.

[68] U.S. Department of Transport. *National Motor Vehicle Crash Causation Survey*. [accessed 23 February 2024]. 2008. url: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811059.

[69] Larissa T. Triess et al. "A Survey on Deep Domain Adaptation for LiDAR Perception". In: *CoRR* abs/2106.02377 (2021). arXiv: 2106.02377. url: https://arxiv.org/abs/2106.02377.

[70] Simon Ulbrich et al. "Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. 2015, pp. 982–988. doi: 10.1109/ITSC.2015.164.

[71] Sourabh Vora et al. "PointPainting: Sequential Fusion for 3D Object Detection". In: *CoRR* abs/1911.10150 (2019). arXiv: 1911.10150. url: http://arxiv.org/abs/1911.10150.

[72] Chunwei Wang et al. "PointAugmenting: Cross-Modal Augmentation for 3D Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 11794–11803.

[73] Yifan Wang et al. "Patch-based Progressive 3D Point Set Upsampling". In: *CoRR* abs/1811.11286 (2018). arXiv: 1811.11286. url: http://arxiv.org/abs/1811.11286.

[74] Martin Weinmann, Boris Jutzi, and Clément Mallet. "Feature relevance assessment for the semantic interpretation of 3D point cloud data". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2013), pp. 313–318. url: https://api.semanticscholar.org/CorpusID:62132969.

[75] Bichen Wu, Alvin Wan, et al. "SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 1887–1893. doi: 10.1109/ICRA.2018.8462926.

[76] Bichen Wu, Xuanyu Zhou, et al. "SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud". In: *CoRR* abs/1809.08495 (2018). arXiv: 1809.08495. url: http://arxiv.org/abs/1809.08495.

[77] Xiaoyang Wu et al. *Point Transformer V3: Simpler, Faster, Stronger*. 2023. arXiv: 2312.10035 [cs.CV].

[78] Aoran Xiao et al. *PolarMix: A General Data Augmentation Technique for LiDAR Point Clouds*. 2022. arXiv: 2208.00223 [cs.CV].

[79] Kai Yuanqing Xiao et al. "Noise or Signal: The Role of Image Backgrounds in Object Recognition". In: *CoRR* abs/2006.09994 (2020). arXiv: 2006.09994. url: https://arxiv.org/abs/2006.09994.

[80] Xiuwei Xu et al. *Back to Reality: Weakly-supervised 3D Object Detection with Shape-guided Label Enhancement*. 2022. arXiv: 2203.05238 [cs.CV].

[81] Yan Yan, Yuxing Mao, and Bo Li. "SECOND: Sparsely Embedded Convolutional Detection". In: *Sensors* 18.10 (2018). issn: 1424-8220. doi: 10.3390/s18103337. url: https://www.mdpi.com/1424-8220/18/10/3337.

[82] Zeyi Yang. *What's next for robotaxis in 2024*. [accessed 23 February 2024]. 2024. url: https://www.technologyreview.com/2024/01/23/1086936/whats-next-for-robotaxis-2024/.

[83] Li Yi, Boqing Gong, and Thomas A. Funkhouser. "Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds". In: *CoRR* abs/2007.08488 (2020). arXiv: 2007.08488. url: https://arxiv.org/abs/2007.08488.

[84] Xiangyu Yue et al. "A LiDAR Point Cloud Generator: from a Virtual World to Autonomous Driving". In: *CoRR* abs/1804.00103 (2018). arXiv: 1804.00103. url: http://arxiv.org/abs/1804.00103.

[85] Sangdoo Yun et al. "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features". In: *CoRR* abs/1905.04899 (2019). arXiv: 1905.04899. url: http://arxiv.org/abs/1905.04899.

[86] Ekim Yurtsever et al. "A Survey of Autonomous Driving: Common Practices and Emerging Technologies". In: *IEEE Access* 8 (2020), pp. 58443–58469. issn: 2169-3536. doi: 10.1109/access.2020.2983149. url: http://dx.doi.org/10.1109/ACCESS.2020.2983149.

[87] Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". In: *CoRR* abs/1710.09412 (2017). arXiv: 1710.09412. url: http://arxiv.org/abs/1710.09412.

[88] Jinlai Zhang et al. "PointCutMix: Regularization Strategy for Point Cloud Classification". In: *CoRR* abs/2101.01461 (2021). arXiv: 2101.01461. url: https://arxiv.org/abs/2101.01461.

[89] Wu Zheng et al. “SE-SSD: Self-Ensembling Single-Stage Object Detector From Point Cloud”. In: *CoRR* abs/2104.09804 (2021). arXiv: 2104.09804. url: https://arxiv.org/abs/2104.09804.

[90] Qian Yi Zhou, Jaesik Park, and Vladlen Koltun. “Open3D: A Modern Library for 3D Data Processing”. In: *arXiv:1801.09847* (2018).